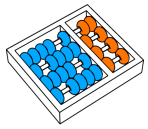
Universidade Estadual de Campinas
Instituto de Computação



Carlos Alberto da Silva

Management of Cloud Computing using security criteria

Gerenciamento de Nuvem Computacional usando critérios de segurança

CAMPINAS
2015

Carlos Alberto da Silva

Management of Cloud Computing using security criteria

Gerenciamento de Nuvem Computacional usando critérios de segurança

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Paulo Lício de Geus

Este exemplar corresponde à versão final da Tese defendida por Carlos Alberto da Silva e orientada pelo Prof. Dr. Paulo Lício de Geus.

CAMPINAS
2015

Agência(s) de fomento e nº(s) de processo(s): FUNDECT, 23/200.308/2009

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Silva, Carlos Alberto da, 1968-
Si38m Management of cloud computing using security criteria / Carlos Alberto da Silva. – Campinas, SP : [s.n.], 2015.

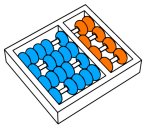
Orientador: Paulo Lício de Geus.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação em nuvem - Medidas de segurança. 2. Computação em nuvem - Gerenciamento de segurança. 3. Métricas de segurança. 4. Sistemas de informação gerencial - Medidas de segurança. 5. Tecnologia da informação - Sistemas de segurança. 6. Sistemas de segurança. I. Geus, Paulo Lício de,1956-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Gerenciamento de nuvem computacional usando critérios de segurança
Palavras-chave em inglês:
Cloud computing - Security measures
Cloud computing - Security management
Security metrics
Management information systems - Security measures
Information technology - Security measures
Security measures
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Paulo Lício de Geus [Orientador]
Edmundo Roberto Mauro Madeira
Diego de Freitas Aranha
Carlos Alberto Maziero
Adriano Mauro Cansian
Data de defesa: 25-09-2015
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação



Carlos Alberto da Silva

Management of Cloud Computing using security criteria

Gerenciamento de Nuvem Computacional usando critérios de segurança

Banca Examinadora:

- Prof. Dr. Paulo Lício de Geus (Orientador)
  Instituto de Computação - UNICAMP
- Prof. Dr. Edmundo Roberto Mauro Madeira
  Instituto de Computação - UNICAMP
- Prof. Dr. Diego de Freitas Aranha
  Instituto de Computação - UNICAMP
- Prof. Dr. Carlos Alberto Maziero
  Universidade Federal do Paraná - UFPR
- Prof. Dr. Adriano Mauro Cansian
  Universidade Estadual de São Paulo - UNESP

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 25 de setembro de 2015

Dedication

To my Family.

Verba volant, scripta manent.

Caio Tito (Senado romano)

Acknowledgements

I would like to thank **CAPES** and **Fundect** (Process #23/200.308/2009) for the financial support.

I would like to thank my advisor, Professor Paulo Lício de Geus, for supporting me during the past years, listening to my frequent complaints and giving me freedom to pursue my goals. His advice and ideas were essential not only to glue the pieces of this thesis together, but to allow me to grow as a scientist and as a better person.

I thank the Brazilian Government, my employer, for investing in me as a researcher.

I also thank my friends, students and research colleagues for all the support, chit-chatting, laughs, coding, paper writing, mainly Anderson Soares Ferreira, André Grégio, Dario Simões Fernandes Filho, Marcelo Palma, Vitor Monte Afonso. If I forgot someone, I am sorry.

I thank the Harada and Resstel families throughout accommodation, entertainment and support.

I finally thank my family, especially my wife Elizangela, my dad (Tiago in memorian), my mom (Maria) and my children: Felipe, Eloá, Maria Eduarda and Fernando. Their support, patience, love and friendship makes my life more meaningful.

Resumo

A nuvem computacional introduziu novas tecnologias e arquiteturas, mudando a computação empresarial. Atualmente, um grande número de organizações optam por utilizar arquiteturas computacionais tradicionais por considerarem esta tecnologia não confiável, devido a problemas não resolvidos relacionados a segurança e privacidade. Em particular, quanto à contratação de um serviço na nuvem, um aspecto importante é a forma como as políticas de segurança serão aplicadas neste ambiente caracterizado pela virtualização e serviços em grande escala de multi-locação. Métricas de segurança podem ser vistas como ferramentas para fornecer informações sobre o estado do ambiente. Com o objetivo de melhorar a segurança na nuvem computacional, este trabalho apresenta uma metodologia para a gestão da nuvem computacional usando a segurança como um critério, através de uma arquitetura para monitoramento da segurança com base em acordos de níveis de serviço de segurança Security-SLA para serviços de IaaS, PaaS e SaaS, que usa métricas de segurança.

Abstract

Cloud Computing has introduced new technology and architectures that changed enterprise computing. Currently, there is a large number of organizations that choose to stick to traditional architectures, since this technology is considered unreliable due to yet unsolved problems related to security and privacy. In particular, when hiring a service in the cloud, an important aspect is how security policies will be applied in this environment characterized by both virtualization and large-scale multi-tenancy service. Security metrics can be seen as tools to provide information about the status of the environment. Aimed at improving security in the Cloud Computing, this work presents a methodology for Cloud Computing management using security as a criterion, across an architecture for security monitoring based on Security-SLA for IaaS, PaaS and SaaS services using security metrics.

List of Figures

List of Tables

The Glossary

CSA	Cloud Security Alliance (organization).
CSP	Cloud Service Provider is a company that offers some component of cloud computing to other businesses or individuals.
CVE	Common Vulnerabilities and Exposures system provides a public known information-security vulnerabilities and exposures.
CVSS	Common Vulnerability Scoring System is a open industry standard for assessing the severity of computer system security vulnerabilities.
ENISA	The European Network and Information Security Agency (organization).
GQM	Goal-Question-Metric methodology.
IaaS	Infrastructure as a Service, service model of NIST.
IDS	Intrusion Detection Systems are softwares that detect an attack on a network or computer system.
IPS	Intrusion Prevention Systems are placed in-line and are able to actively prevent/block intrusions that are detected.
IDPS	Intrusion Detection and Prevention Systems are network security appliances that monitor network or system activities for malicious activity.
ISACA	Information Systems Audit and Control Association (organization).
ISO	International Organisation for Standardization (organization).
NIST	National Institute of Standards and Technology (organization).
QoS	Quality of Service, a defined measure of performance in a system.
PaaS	Platform as a Service, service model of NIST.
SaaS	Software as a Service, service model of NIST.
SLA	Service Level Agreement is a part of a standardized service contract where a service is formally defined.
SLO	Service Level Objectives, which are the actual topics to be measured by the SLA. Each SLO may be composed of one or more QoS metrics.
Security-SLA	Security Service Level Agreement is a part of a standardized security service contract where a service is formally defined.
WSLA	Web Service Level Agreement is a standard for service level agreement compliance monitoring of web services.
VM	Virtual Machine is an operating system OS or application environment that is installed on software which imitates dedicated hardware.
VMI	Virtual Machine Introspection, a technique to monitor internal states and events of a virtual machines without inserting a component inside the virtualized system.
VMM	Virtual Machine Monitor, a “probe” inserted in a layer located between the host and the guest in a virtualization system.

Contents

Chapter 1

Introduction

In general, emphasizing security policies ensures that the business goals at a more abstract or higher level are achieved, and that the controls must be proportional to the value of what one seeks to protect [Landwehr, 2001].

Cloud Computing is a technology that provides users with processing power, data storage and applications over the network, through a model of utility computing whose computing resources and software are offered as services through pay-per-use [Fox et al., 2009]. It provides its customers access to computing resources that would not be available in traditional architectures, due to the technical complexity and high costs involved [Krutz and Vines, 2010]. Just as the use of cloud services is growing among home users, who use such services for personal communication (social networking and e-mail), storage and even home office applications, its adoption by enterprises and governments is also going strong.

Despite the growth of this market and massive investments, there is still a great concern about security in these environments. According to [Foster et al., 2008], the security issues are considered as the main obstacle to the migration of services to cloud environments. The growing concern and dissatisfaction with security in cloud services is the result of a combination of several factors, among which may be cited: i) a lack of knowledge of technical characteristics and risks in cloud environments [Chen et al., 2010, Subashini and Kavitha, 2011]; ii) a lack of well defined interoperability standards in environment [Hoefer and Karagiannis, 2010]; iii) the loss of control over data and applications [Krutz and Vines, 2010]; iv) a history of failures in Computing Clouds that resulted in the unavailability of services, data loss and information leakage [Subashini and Kavitha, 2011]; v) and the lack of guarantees regarding any sort of security levels [ENISA, 2009a].

In order to decide where to invest and how to prioritize investments, the service provider needs to know how well each system is protected, which systems are under higher risks, and how much should be invested to achieve the desired level of protection. This is only possible with a range of standard values that allow a system level setting and comparison with the level of protection for each of the security systems. For example, on a scale of [0-4], (0) would represent a security level corresponding to little reliability and/or presenting a serious security problem, and (4) a security level that is safer and/or does not present any security issue. For relevant information, the service provider must define what should be measured, i.e. what the events that have an impact on security are

and how to extract significant numbers of events.

Driven both by the increasing demand for the use of services in cloud computing and the large number of security issues in these environments, various institutions started working on typifying services and standards specifications for interoperability and security in Cloud Computing. Although such actions are a significant step towards the creation of secure environments, most of these efforts are still at a preliminary stage and will require considerable time to mature and to be adopted, as interests of providers and pressure from the service consumers amount.

1.1 Overview of the Problem

The processes of service provisioning based on Security-SLA and efficient management of resources in an autonomic manner have been identified as major research challenges in Cloud environments.

In today's scenario, it is clear that demand is great not only for standards but also for the specification of mechanisms, tools, recommendations, references, monitoring and security management schemes on Cloud Computing environments. Some of these attempts are:

- The Information Systems Audit and Control Association (ISACA) presents control objectives for cloud [ISACA, 2014a], which are guidelines on cloud governance, relationship between clients and service providers and specific control problems, and also presents [ISACA, 2014b], which is a practical guide to assess cloud risks and help determine the most suitable cloud model to meet the company's needs;

- The Cloud Security Alliance (CSA) presents [CSA, 2014a], which are detailed guidance on security principles and practices for Cloud Computing, and [CSA, 2014b], which is a free public record of security controls provided by various computing services in the cloud;

- The National Institute of Standards and Technology (NIST) presents security guides [Mell and Grance, 2011, Voas et al., 2012, NIST, 2013a, NIST, 2013b, NIST, 2013c], which are recommendations on Cloud Computing;

- The European Network and Information Security Agency (ENISA) presents information security [ENISA, 2009b], describing the risks and benefits of Cloud Computing;

- The International Organization for Standardization (ISO) presents the international security standards for cloud computing, ISO/IEC-27017 [ISO/IEC-27017:2014, 2014] and ISO/IEC-27018 [ISO/IEC-27018:2014, 2014], which are, respectively, extensions to the ISO series ISO/IEC-27000 [ISO/IEC-27000:2009, 2009] and ISO/IEC-27002 [ISO/IEC-27002:2005, 2005];

- The Cloud Standards Customer Council dedicated to accelerating cloud's successful adoption, and present into the standards, security and interoperability issues surrounding the transition to the cloud [Cloud-Council, 2012, Cloud-Council, 2015].

The specification of security controls, aimed to meet the needs of the service user, may be achieved through the use of Security Service Level Agreements (Security-SLA), which are seen as important tools for the security management of Cloud Computing [Righi et al., 2004, de Chaves et al., 2010a, de Chaves et al., 2010b, Jaatun et al., 2012, Nayak et al., 2013]. Nevertheless, one notices in the literature a lack of works actually dealing with specifying and monitoring these agreements. When they do, the solutions describe the representation of such agreements and the adoption of monitoring systems developed for traditional computing architectures. By analyzing these solutions, one can observe that they are not fully prepared for monitoring cloud environments because:

- Existing tools have little or no support for SLA monitoring, and there are mechanisms for management of Security-SLA;
- The information used for monitoring agreements depends on the collection mechanisms that run on the monitored machine. In infrastructure cloud services, virtual machines are controlled by the user, which means that the installation of such mechanisms, in addition to relying on user collaboration, is still subject to incompatibilities caused by differences in operating systems, libraries etc. Moreover, they are sensitive to tampering in cases where the user is malicious;
- Existing tools do not consider fundamental events occurring in cloud environments, such as the creation, modification, migration and stopping of virtual machines.

Among problem examples in cloud computing surrounding the lack of transparency and control over security are: i) Amazon's DynamoDB database issue on 20/Sep/2015 [Amazon, 2015]: Amazon Web Services and some of the big companies relying on its public cloud infrastructure had a rough Sunday morning with Netflix and other client sites reporting problems; ii) Skype was offline on 21/Sep/2015 [Skype, 2015]: bought by Microsoft in 2011, has an estimated 300 million active users a month, with more than 660 million users registered worldwide.

1.2 Objectives

The main goal of this thesis is to present a new management process across a new monitoring methodology that complies with security requirements specified through Security-SLAs. To this end: i) the security policy of the provider is converted (in a useful, organized and understandable manner) to the objectives of the Security-SLA; ii) an agent-based monitoring architecture allows for the collection and validation of security metrics and iii) the information collected is applied in the cloud management process.

1.3 Contributions

Research in cloud security is very broad and plenty of effort has been spent by the security community to address the multiple kinds of threats and vulnerabilities involved with this complex environment.

In this thesis, the focus is on management of Cloud Computing using security criteria and on what can be done with the monitoring information that is collected. Thus, in addition to analyzing the time series of each security metric and establishing baselines on past behavior, it is also presented a process to validate the collected metrics. The main contributions of this thesis are:

- A review of the security monitoring scene in Cloud Computing environment;
- A discussion about the current Service Level Agreements (SLA) in information technology (IT), and parameterization of Security Service Level Agreements (Security-SLA) in Cloud Computing;
- A methodology based on security policies for the management of Cloud Computing using security criteria;
- A security monitoring architecture for management of cloud computing that defines a high level abstraction for information gathering and suggests from where and how security metrics will be collected;
- The definition of a model that allows for the organization and combination of each metric in a coherent hierarchy, and also obtaining security information to Cloud Computing environment;
- A way to perform more transparent, interactive, visual and organized management, in order to provide a sense of security levels in Cloud Computing environments.

More than a closed and final work, the current work is seen as a starting point for organizations to collect and correlate security data, so as to produce indicators that will improve information security knowledge, also allowing one to check their assessment over time. New metrics may and should be added to this model, due to technological changes (that lead to new threats and vulnerabilities or make possible the collection of new kinds of data) or to new systems or components being added to the environment. Evolving systems/components may have different security metrics, depending on factors such as architecture, hardware and software used etc.

Furthermore, the work concerning this thesis is not limited to the contents of this document: two Masters and three undergraduate students have been co-advised. As a result of this effort, articles and technical production have been generated and skills in research leadership have been gained.

1.4 Thesis organization

The remainder of this thesis is organized as follows:

- Chapter 2 introduces the basic concepts of Computing Clouds, their features, service models, the main issues related to security, the need for monitoring and their mechanisms for this environment.

- Chapter 3 discusses the theoretical aspects of unmeasurable qualities and security metrics, so that they can be used with Service Level Agreements (SLA) in general and Security-SLAs in particular. It also discusses reference parameters for Security-SLAs.

- Chapter 4 discusses works related to the main theme of the thesis, i.e. monitoring process and guidelines to monitoring Security-SLAs in Cloud Computing environments.

- Chapter 5 applies the GQM methodology for the management of Cloud Computing using security criteria across the security metrics hierarchy that was conceived in this thesis.

- Chapter 6 evaluates and validates the proposed methodology over several aspects: i) Return On Security Investment (ROSI); ii) parameter management for Security-SLAs; iii) obtaining the Index of Security; iv) and obtaining the Index of Allocation.

- Chapter 7 presents the proposed monitoring architecture, based on Security-SLA, its components and system interaction with the virtual machines running in the cloud environment, its execution flow and architecture validation.

- Chapter 8 concludes with some considerations and summarizes general and specific results obtained. A list of publications and submitted articles is also presented.

It is worth noticing that, since the chapters are based on papers published over a time span, the analyzed set of samples may be different (regarding the set's size and variety).

Chapter 2

Cloud Computing

Cloud Computing is a revolutionary technology model in which computing resources are offered to users as a service, ensuring large processing capacity and flexibility. Despite the great interest in this technology, there is still great misinformation about its characteristics, potential and risks. This chapter presents the basic concepts about Cloud Computing service models and the process of monitoring this environment, also discussing issues related to security monitoring.

2.1 A brief overview

Cloud Computing is associated with a new paradigm for computing supply infrastructure [Vaquero et al., 2008], where processing power and applications are offered as services. Although it is considered a new technology, its origins date back to the 1960s when the computer scientist John McCarthy predicted that computing would be organized as a public utility service [Garfinkel, 1999]. Since then, the intensified use of the Internet and the emergence of new technologies allowed for Computing Clouds to offer a wide range of services, including virtualization of computing resources, software development and end-user applications of various types. This variety contributes to the existence of different definitions of what is Cloud Computing, each focusing on specific characteristics. These technologies are:

- **Autonomic computing**: a comprehensive set of services to help businesses develop in an increasingly autonomic way, i.e. self-managing IT infrastructures;
- **Grid computing**: a collection of computer resources from multiple locations to reach a common goal, as a distributed system;
- **Service Oriented Architectures (SOA)**: a design pattern in which application components provide services to other components via a communication protocol over a network. The principles of service-orientation are independent of any seller, product or technology;
- **Software as a Service (SaaS)**: a software distribution and deployment model in which applications are provided to customers as a service. The applications can run

on the users' computing systems or the provider's Web servers. SaaS provides for efficient patch management and promotes collaboration;

- **Platform Virtualization**: refers to the act of creating a virtual (rather than actual) version of something, including (but not limited to) a virtual computer hardware platform, operating system (OS), storage device or computer network resources;
- **Utility computing**: a service provisioning model in which a service provider makes computing resources and infrastructure management available to the customer as needed and charges them by specific usage rather than a flat rate.

Initially, the cloud settings were described as aspects of "computing as a service" and "distributed computing". Over the years, the definitions have become more complete, progressing to the definition proposed by NIST [Mell and Grance, 2011], which describes aspects such as the sharing of virtualized resources and self-service. According to NIST, cloud computing is defined as follows [Mell and Grance, 2011]:

> "Model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Figure 2.1 presents the NIST cloud definition [Mell and Grance, 2011] that also deals with the following important concepts [NIST, 2013a]: Deployment Models, Service Models, Essential Characteristics, Hosting and Roles. The next sections describe these concepts.

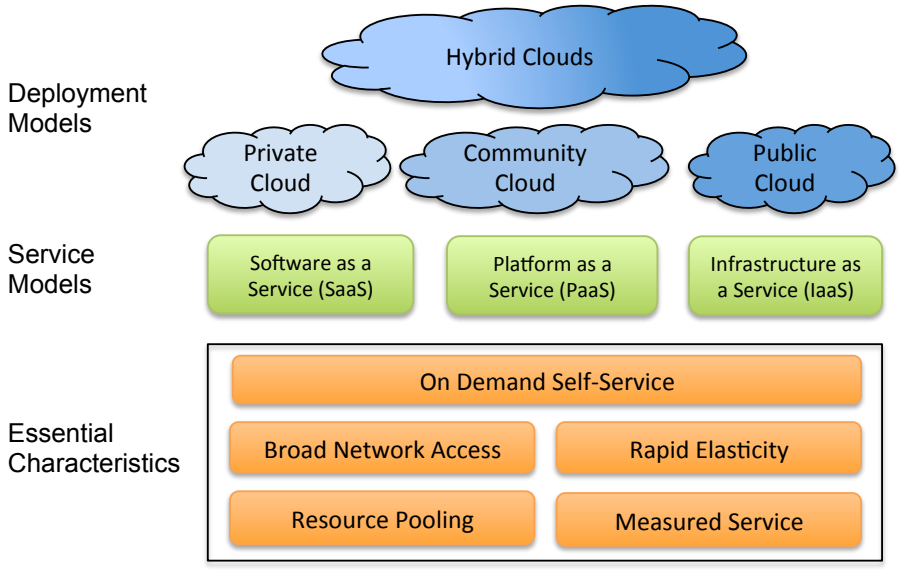


Figure 2.1: NIST Cloud Definition.

2.1.1 Deployment Models

The Deployment Models document classifies clouds according to their way of management and the relationship between service users [NIST, 2013a]. Under this view, clouds may be classified as follows:

- **Private cloud**: the cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers, e.g. business units. It may be owned, managed, and operated by the organization, a third party or some combination of them, and it may exist on or off premises;
- **Community cloud**: the cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns, e.g. mission, security requirements, policy, and compliance considerations. It may be owned, managed and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises;
- **Public cloud**: the cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or governmental organization, or some combination of them. It exists on the premises of the cloud provider;
- **Hybrid cloud**: the cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability, e.g. cloud bursting for load balancing between clouds.

2.1.2 Service Models

According to the service model document [NIST, 2013a], Cloud Computing may be classified as follows:

- **Cloud Software as a Service (SaaS)**: the capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a Web browser, e.g. Web-based email, or a program interface. The consumer does not manage or control the underlying cloud infrastructure, including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings;
- **Cloud Platform as a Service (PaaS)**: the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or consumer-acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations;

- **Cloud Infrastructure as a Service (IaaS)**: the capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications; and possibly limited control of select networking components, e.g. host firewalls.

An important feature of computer service architecture in clouds is the interdependence between different service models, c.f. Figure 2.2. It can be observed that a SaaS cloud service directly depends on the resources provided by the PaaS layer, just as the PaaS layer builds on the features offered by the IaaS layer.

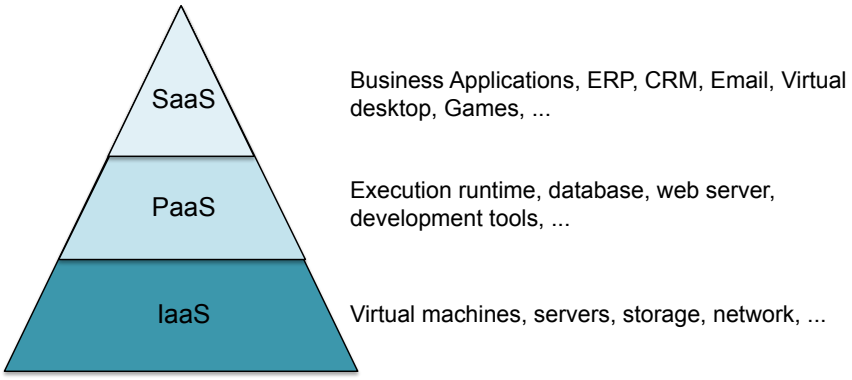


Figure 2.2: Stack of cloud services.

2.1.3 Essential characteristics

The following essential characteristics document must be presented by any cloud service [NIST, 2013a, NIST, 2013b], as described below:

- **On-demand self-service**: a consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider;

- **Broad network access**: capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms, e.g. mobile phones, tablets, laptops, and workstations;

- **Resource pooling:** the provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction, e.g. country, state, or datacenter. Examples of resources include storage, processing, memory, and network bandwidth;

- **Rapid elasticity**: capabilities can be rapidly and elastically provisioned, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time;
- **Measured Service**: cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service, e.g. storage, processing, bandwidth, and active user accounts. Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

2.1.4 Hosting

According to the type of hosting, Cloud Computing maybe be considered internally or externally hosted, depending on customer ownership, control of architectural design and the degree of available customization.

2.1.5 Governance

Figure 2.3 shows the existing governance limits in the traditional architecture and different service models of cloud computing [Mather et al., 2009], which resources are managed by the customer and which are managed by the provider.

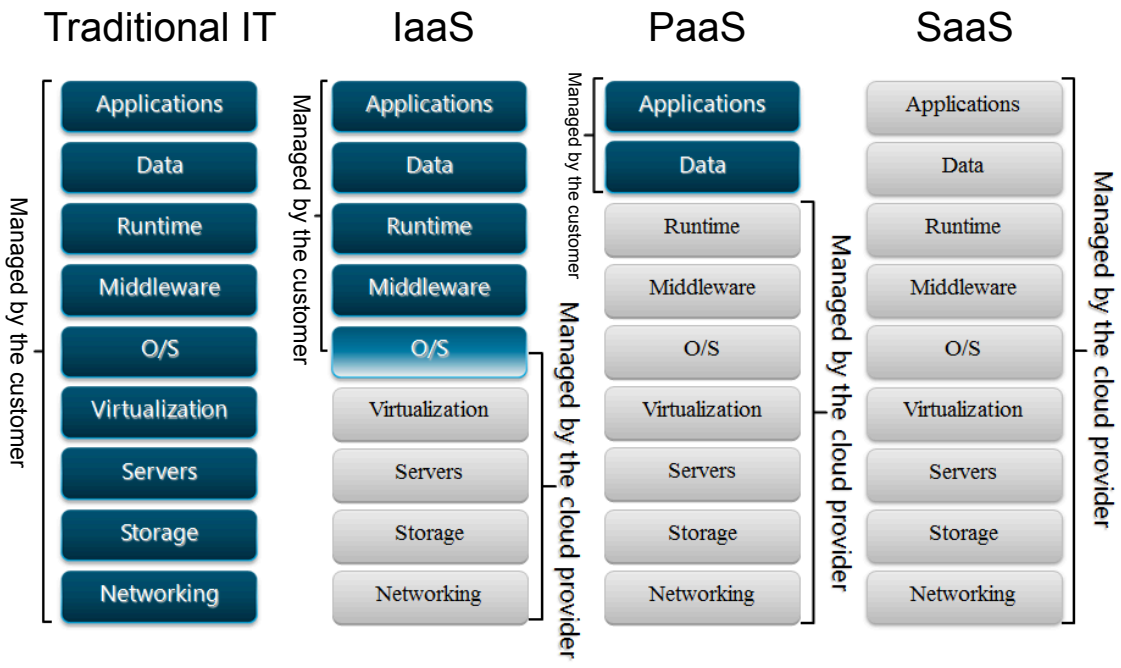


Figure 2.3: NIST Service Offering.

2.1.6 Roles

Multiple roles can be supported by a Cloud developer, many of which can exist within a single provider [NIST, 2013a]: Cloud Auditor, Cloud Service Provider, Cloud Service Carrier, Cloud Service Broker, and Cloud Service Consumer.

In this proposal, it is interested in the Cloud Auditor, who has the responsibility to carry out the monitoring tasks and collection of information (the core set of Security Components), necessary to implement security for the Cloud Computing environment [NIST, 2013a].

2.1.7 Analyzing Cloud Options in Depth

Table 2.1 shows us that different cloud deployment models have varying management, ownership, locations, and access levels.

Table 2.1: Deployment Model's Responsibilities

Model	**Managed by**	**Owned by**	**Location**	**Used by**
Public	External CSP	External CSP	Off-Site	Untrusted
Private	Customer or external CSP	Customer or external CSP	On-site or off-site	Trusted
Hybrid	Customer and external CSP	Customer and external CSP	On-site and off-site	Trusted and untrusted

2.2 Cloud Computing: the need for monitoring

Cloud monitoring is a task of essential importance for both providers and customers. On the one side, it is a key tool for controlling and managing hardware, software infrastructures and services being provided. On the other side, it provides information and indicators about the current state of various issues of the hired services in Cloud Computing environment.

The continuous monitoring of the cloud and Security-SLAs (for example, in terms of availability, duration, violation, penalty, compensation etc.) supplies both the providers and the customers with information such as Quality of Service (QoS), service continuity and application performance offered through the cloud, also allowing to implement mechanisms to prevent or recover violations for both the provider and customers. The monitoring process is clearly instrumental for all the activities covered by the role of Cloud Auditor [NIST, 2013a].

Figure 2.4 present the main aspects of cloud monitoring considered in this work: monitoring and properties, and that will be described in the Sections 2.2.1 and 2.2.2, respectively.

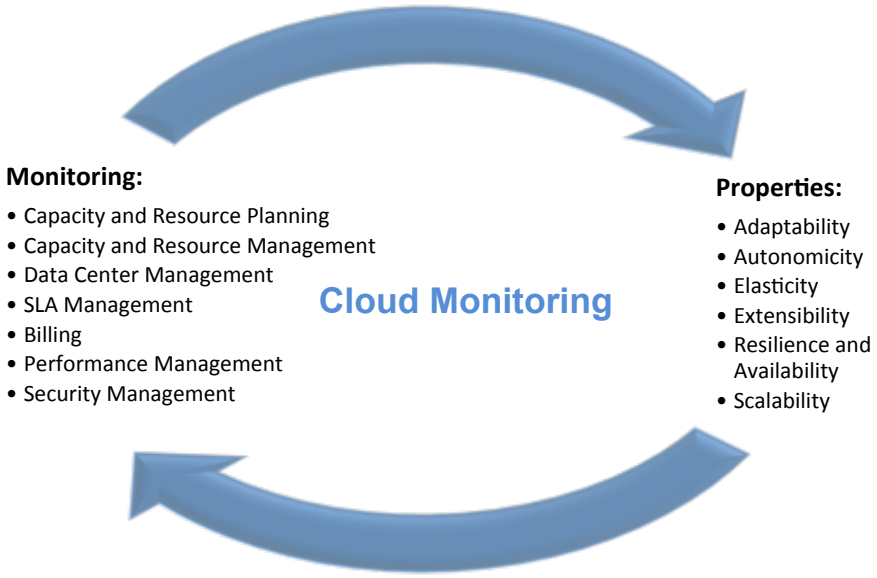


Figure 2.4: Cloud monitoring and properties.

2.2.1 Monitoring

As seen in Figure 2.4, the monitoring process is objective, performing the following management tasks:

- **Capacity and resource planning:** in order to guarantee the performance required by hired services, administrators have to: (i) quantify capacity and resources, e.g. Central Processing Unit (CPU), memory, storage, etc. to be provided. Cloud service providers usually offer guarantees in terms of QoS and thus of resources and capacity for their services as specified in SLAs [Hasselmeyer and d'Heureuse, 2010, Shao and Wang, 2011]. To this end, monitoring becomes essential for cloud service providers to predict and keep track of the evolution of all the parameters involved in the process of QoS assurance [Kubert et al., 2011] in order to properly plan their infrastructure and resources for complying with the SLAs;

- **Capacity and resource management:** a Cloud must have a monitoring system able to accurately capture the state of resources [Viratanapanu et al., 2010]. Virtualization has become a key component to implement Cloud Computing, hiding the high heterogeneity of resources of the physical infrastructure, and infrastructure providers have to manage both physical and virtualized resources [Shao et al., 2010, Ferretti et al., 2010, Lakshmanan et al., 2010, Kubert et al., 2011];

- **Data center management:** cloud services are provided through large-scale data centers integrating monitoring and analytics. Data center management activities imply two fundamental tasks: i) monitoring and collecting desired hardware and software metrics; and ii) data analysis, that processes such metrics for management

actions: . resource provisioning, troubleshooting, or planning future actions to prevent and correct violations [Hu et al., 2011, Buyya et al., 2011a];

- **SLA management:** the flexibility in terms of resource management by Cloud Computing calls for new programming models in which cloud applications can take advantage of this new feature [Rak et al., 2011], whose basic premise is monitoring;

- **Billing:** one of the characteristics of Cloud Computing is to offer "measured services", allowing the customer to pay proportionally to the use of the service, according to the type of service and the price model adopted [Mell and Grance, 2011];

- **Performance management:** with infrastructure maintenance delegated to the providers, the Cloud Computing model is attractive for most customers. Monitoring is then necessary since it may considerably improve the performance of real applications [Fox et al., 2009];

- **Security management:** cloud security is very important for a number of reasons. Security is considered as one of the most significant obstacles to the adoption of Cloud Computing, especially considering certain kinds of critical applications and governmental customers [Chen et al., 2010]. Different works in the literature have provided reviews and recommendations for cloud security [Chen et al., 2010, Spring, 2011a, Spring, 2011b]. To manage security in cloud infrastructures and services, proper monitoring systems are needed.

2.2.2 Properties

In the cloud scenario, a distributed monitoring system is required to have several properties to operate properly, consequently, introduce new security issues. In Figure 2.4, it has described these properties for taxonomy of cloud monitoring considered in this work and in literature:

- **Adaptability:** a monitoring system must adapt itself to varying computation and network loads in order not to be invasive [Fox et al., 2009, Krutz and Vines, 2010, Clayman et al., 2010, Buyya et al., 2011a, Vu et al., 2015];

- **Autonomicity:** an autonomic monitoring system can keep running without intervention and reconfiguration, while hiding intrinsic complexity to providers and customers [Fox et al., 2009, Buyya et al., 2011a, Mian et al., 2013, Vu et al., 2015];

- **Elasticity:** a dynamic monitoring system is elastic if it can manage itself correctly in regards to the virtual resources created and destroyed by expansion and contraction [Fox et al., 2009, Clayman et al., 2010, Krutz and Vines, 2010, Buyya et al., 2011a, Vu et al., 2015]. Demand scalability and support for upsizing or downsizing of the pool of monitored resources, also referred to as dynamism
[Hasselmeyer and d'Heureuse, 2010];

- **Extensibility:** a monitoring system is extensible if it can easily be extended through plug-ins or functional modules [Hasselmeyer and d'Heureuse, 2010] [Krutz and Vines, 2010, Kubert et al., 2011, Vu et al., 2015];

- **Resilience and Availability:** a monitoring system is resilient when the persistence of service delivery can justifiably be trusted when facing changes [Laprie, 2008, Fox et al., 2009, Krutz and Vines, 2010, Buyya et al., 2011a, Vu et al., 2015], and it is available if it provides services according to the system design whenever customers request them [Shirey, 2007, Krutz and Vines, 2010];

- **Scalability:** due to the large number of parameters to be monitored about a huge number of resources, a monitoring system is scalable if it can cope with a large number of probes [Fox et al., 2009, Clayman et al., 2010, Krutz and Vines, 2010, Buyya et al., 2011a, Vu et al., 2015].

2.2.3 Security Monitoring Views

Currently, cloud monitoring can provide information about aspects of system performance, behavior, evolution, security, etc. This information is analyzed and applicable to the appropriate domain or system level: server, infrastructure, platform or application. For example, in an IaaS cloud such as Amazon EC2 [Amazon, 2008], customers can monitor the state of their virtual machine instances in order to get knowledge about system load, memory usage and performance. In the same scenario, the Cloud Service Provider (CSP) would need to monitor all VM instances, continuously making sure SLA requirements are satisfied. The CSP would also require monitoring information at the server level in order to effectively control overall system load, VM allocation and migration, etc. Therefore, the point of view of the entity that obtains the monitoring information (client, management system, CSP, etc.) and its role in the monitoring system determines what kind of information has to be provided.

Figure 2.5 presents different entities that require different security monitoring and have different views of the cloud. From this perspective, two main cloud security-monitoring views can be distinguished:

- Client-side security monitoring view: the cloud is capable of providing a specific set of computing services, expressed as the service provisioning relationship established between customer and CSP (Security-SLA). This monitoring information helps the customer to understand the characteristics of the services received and their use;

- Cloud-service-provider-side security monitoring view: the cloud is viewed as a complex distributed infrastructure, with many hardware and software elements combined together to provide a specific set of services. Security monitoring information produces knowledge for the CSP about the internal functioning of the different cloud elements, their state, performance, etc. This security information serves as an internal status control in order to guarantee the Security-SLAs.

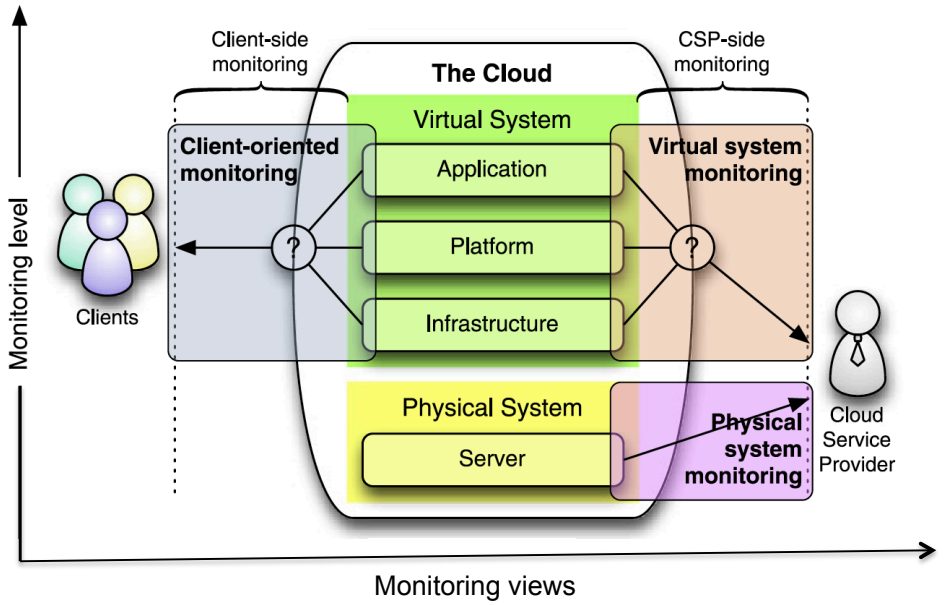


Figure 2.5: Cloud monitoring level views.

These two security views are complementary and address different requirements of cloud security monitoring, creating differentiated views of the system behavior and evolution.

2.3 Security in Cloud Computing

The share of responsibility in traditional architectures, the user has control over all the infrastructure and therefore, hold full responsibility for security. As for the cloud environment, the responsibility of the user decreases according to the service model used. IaaS service users are responsible for the security of the VM and its applications, SaaS services, and all responsibility lies with the service provider.

From the provider perspective, responsibility sharing is also a challenge, as in IaaS services, the user has complete control over the VM. This causes this element to become a weak point within the cloud architecture that can be used to commit itself or other VMs physical machine (node) where VM is executed.

2.3.1 Classification of Risks

In Cloud Computing, the different technologies that integrate to form the cloud architecture (hardware, operating systems, virtualization technologies, etc.) introduce known risks to the technological model. At the same time, their combined use introduces new risks.

ENISA addresses the security risks of Cloud Computing [ENISA, 2009a], and classify them into four categories:

1. **Policy and organizational**: these are risks related to loss of governance, obser-

vance break the organizational policies and regulatory standards, dependence on service providers, etc.;

2. **Technical**: related to questions about the security of the technologies used by the provider, such as network communication, service management etc.;

3. **Legal**: concern the breakdown of data protection due to differences in legislation between countries that host cloud servers;

4. **Others risks not specific to the cloud**: in the course of risk analysis, the following threats which are not specific to Cloud Computing, but should nevertheless be considered carefully when assessing the risk of a typical cloud-based system.

2.3.2 Risk Assessment Process

The risk level is estimated on the basis of the likelihood of an incident scenario, mapped against the estimated negative impact. The likelihood of an incident scenario is given by a threat-exploiting vulnerability with a given likelihood.

In Figure 2.6, the following shows the risk level as a function of the business impact and likelihood of the incident scenario. The resulting risk is measured on a scale of 0 to 8 that can be evaluated against risk acceptance criteria, and estimation of risk levels based on ISO/IEC 27005:2011 [ISO/IEC-27005:2011, 2011]. This risk scale could also be mapped to a simple overall risk rating:

	Likelihood of incident scenario	Very Low (Very Unlikely)	Low (Unlikely)	Medium (Possible)	High (Likely)	Very High (Frequent)
Business Impact	Very Low	0	1	2	3	4
	Low	1	2	3	4	5
	Medium	2	3	4	5	6
	High	3	4	5	6	7
	Very High	4	5	6	7	8

Figure 2.6: Estimation of risk levels based on ISO/IEC 27005:2011.

- Low risk: 0–2
- Medium Risk: 3–5
- High Risk: 6–8

Figure 2.7 shows the distribution of the risk probabilities and impacts. The likelihood of each incident scenario and the business impact was determined in consultation with the expert group contributing to this report, drawing on their collective experience.

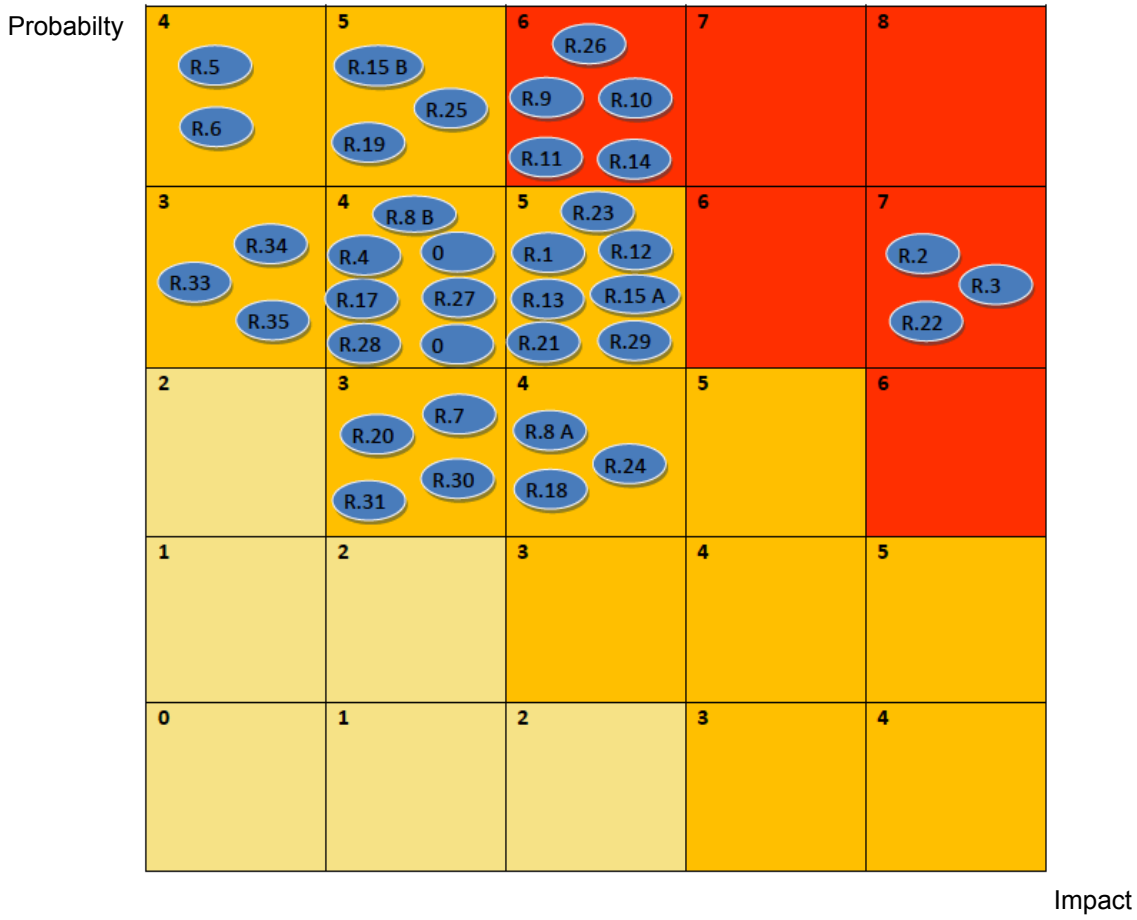


Figure 2.7: Risk distribution.

Table 2.3.2 describes different groups of risks [ENISA, 2009a], where column Topic describes the Id risk, column Class describes the type of risk: (P) for "Policy and Organizational"; (T) for "Technical"; (L) for "Legal"; and (X) for "Risks not specific to the Cloud", column Description presents the risk definition and column Value describes the risk level: High, Medium, Low. More importantly, the impact of these risks and responsibility for its control varies according to the type of service [Vacca, 2013].

Table 2.2: Summary of Risks in Cloud Computing

Topic	Class	Description	Value
R.1	P	**Lock-in**: currently, there are no tools, procedures, standard data formats or service interfaces that could guarantee data and service portability for a customer to migrate from one provider to another.	**High**

Continued on next page

Table 2.2 – *Continued from previous page*

Topic	Class	Description	Value
R.2	P	**Loss of governance**: while using the cloud, the client necessarily concedes control to the CSP on a number of security issues. The loss of control and governance could affect the confidentiality, integrity and availability of data, cause a deterioration of performance and Quality of Service and introduce new compliance challenges.	**High**
R.3	P	**Compliance challenges**: customers make considerable investments in achieving certification, meeting industry standards and regulatory requirements. The risks of a migration to the cloud are: i) the CSP not providing evidence of their own compliance; ii) the CSP not permitting auditing.	**High**
R.4	P	**Loss of business reputation due to co-tenant activities**: as examples, spamming, port scanning or the serving of malicious content from cloud infrastructure can lead to: i) a range of IP addresses being blocked, including the attacker's and other innocent tenants of an infrastructure's; ii) confiscation of resources due to neighbor activities (neighbor subpoenaed).	**Medium**
R.5	P	**Cloud service termination or failure**: in the short or medium term, some Cloud Computing services could be terminated. The impact of this threat for the cloud customer could lead to a loss or deterioration of service delivery performance and Quality of Service, as well as an investment loss.	**Medium**
R.6	P	**Cloud provider acquisition**: could increase the likelihood of a strategic shift and may put non-binding agreements at risk, e.g. software interfaces, security investments and non-contractual security controls. The final impact could be damaging for the organization's reputation, customer or patient trust and employee loyalty and experience.	**Medium**
R.7	P	**Supply chain failure**: a cloud service provider can outsource certain specialized tasks of its "production" chain to third parties. Any interruption or corruption in the chain or a lack of coordination of responsibilities between all the parties involved may lead to: unavailability of services, loss of data confidentiality, integrity and availability, economic and reputational losses due to failure to meet customer demand, violation of SLA, cascading service failure, etc.	**Medium**
R.8	T	**Resource exhaustion (under or over provisioning)**: Inaccurate modeling of customer demands by the cloud provider may lead to service unavailability, access control compromising and economic and reputational losses due to resource exhaustion. The customer takes a level of calculated risk in allocating all resources from a cloud service provider because resources are allocated according to statistical projections.	**Medium**

Continued on next page

Table 2.2 – *Continued from previous page*

Topic	Class	Description	Value
R.9	T	**Isolation failure**: This class of risks includes the failure of mechanisms separating storage, memory, routing, and even reputation between different tenants of the shared infrastructure.	**High**
R.10	T	**Cloud provider malicious insider - abuse of high privilege roles**: the malicious activities of an insider could potentially have an impact on the confidentiality, integrity and availability of all kinds of data and services and therefore indirectly on the organization's reputation, customer trust and the employees' experience. As cloud use increases, employees of cloud providers increasingly become targets of criminal gangs.	**High**
R.11	T	**Management interface compromise (manipulation, availability of infrastructure)**: the customer management interfaces of public cloud providers are Internet accessible and mediate access to larger sets of resources (than traditional hosting providers) and therefore pose an increased risk, especially when combined with remote access and Web browser vulnerabilities.	**Medium**
R.12	T	**Intercepting data in transit**: Cloud Computing, being a distributed architecture, implies more data in transit than traditional infrastructures. Sniffing, spoofing, man-in-the-middle attacks, side channel and replay attacks should be considered as possible threat sources.	**Medium**
R.13	T	**Data leakage on up/download, intra-cloud**: this is the same as the previous risk category, but applies to data transfer between the cloud provider and the cloud customer.	**Medium**
R.14	T	**Insecure or ineffective deletion of data**: whenever a provider is changed, resources are scaled down, physical hardware is reallocated, and data may be available beyond the lifetime specified in the security policy. Where true data wiping is required, special procedures must be followed and this may not be supported by CSP.	**Medium**
R.15	T	**Distributed Denial of Service (DDoS)**: a common method of attack involves saturating the target environment with external communication requests, such that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable. This can result in financial and economic losses.	**Medium**
R.16	T	**Economic Denial of Service (EDoS)**: destroys economic resources; the worst case scenario would be the bankruptcy of the customer or a serious economic impact.	**Medium**

Continued on next page

Table 2.2 – *Continued from previous page*

Topic	Class	Description	Value
R.17	T	**Loss of encryption keys**: this includes disclosure of secret keys or passwords to malicious parties, the loss or corruption of those keys or their unauthorized use for authentication and non-repudiation (digital signature).	**Medium**
R.18	T	**Undertaking malicious probes or scans**: malicious probes or scanning, as well as network mapping, are indirect threats to the assets being considered. They can be used to collect information in the context of a hacking attempt. A possible impact could be a loss of confidentiality, integrity and availability of service and data.	**Medium**
R.19	T	**Compromise service engine**: each cloud architecture relies on a highly specialized platform and the service engine. The service engine sits above the physical hardware resources and manages customer resources at different levels of abstraction. For example, in IaaS clouds this software component can be the hypervisor. Like any other software layer, the service engine code may have vulnerabilities and is prone to attacks or unexpected failure. Cloud providers must set out a clear segregation of responsibilities that articulates the minimum actions customers must undertake.	**Medium**
R.20	T	**Conflicts between customer hardening procedures and cloud environment**: cloud providers must set out a clear segregation of responsibilities that articulates the minimum actions customers must undertake. The failure of customers to properly secure their environments may pose a vulnerability to the cloud platform if the cloud provider has not taken the necessary steps to provide isolation. Cloud providers should further articulate their isolation mechanisms and provide best practice guidelines to assist customers to secure their resources.	**Medium**
R.21	L	**Subpoena and e-discovery**: in the event of the confiscation of physical hardware as a result of subpoena by law-enforcement agencies or civil suits, the centralization of storage as well as shared tenancy of physical hardware means many more clients are at risk of the disclosure of their data to unwanted parties. At the same time, it may become impossible for the agency of a single nation to confiscate a cloud given pending advances around long distance hypervisor migration.	**High**

Continued on next page

Table 2.2 – *Continued from previous page*

Topic	Class	Description	Value
R.22	L	**Risk from changes of jurisdiction**: customer data may be held in multiple jurisdictions, some of which may be high risk or subject to higher restrictions. Certain countries are regarded as high risk because of their unpredictable legal frameworks and disrespect international agreements. On the other hand, other countries can have stricter privacy laws a might require that certain data cannot be stored or tracked.	**High**
R.23	L	**Data protection risks**: it has to be clear that the cloud customer will be the main person responsible for the processing of personal data, even when such processing is carried out by the CSP in its role of external processor. While some CSP, provide information about their data processing and data security activities, others are opaque about these activities and can cause legal problems for the customer. There may also be data security breaches that are not notified to the controller by the CSP. In some cases, customer might be storing illegal or illegally obtained data, which might put the CSP and other customers at risk.	**High**
R.24	L	**Licensing risks**: licensing conditions, such as per-seat agreements, and online licensing checks may become unworkable in a cloud environment. For example, if software is charged on a per instance basis every time a new machine is instantiated then the cloud customer's licensing costs may increase exponentially, even though they are using the same number of machine instances for the same duration.	**Medium**
R.25	X	**Network breaks**: potentially thousands of customers are affected at the same time.	**Medium**
R.26	X	**Network management**: may occur network congestion, miss connection, no optimal use, etc.	**High**
R.27	X	**Modifying network traffic**: may occur network congestion, miss connection, no optimal use, etc.	**Medium**
R.28	X	**Privilege escalation attack**: a type of network intrusion that takes advantage of programming errors or design flaws to grant the attacker elevated access to the network and its associated data and applications.	**Medium**
R.29	X	**Social engineering attacks**: the art of manipulating people into performing actions or divulging confidential information. The access doesn't have mean access to your systems (though it often does), it can simply mean access to private information that can later be used to compromise your systems (impersonation).	**Medium**

Continued on next page

Table 2.2 – *Continued from previous page*

Topic	Class	Description	Value
R.30	X	**Loss or compromise of operational logs**: operational logs can be vulnerable due to lack of policy or poor procedures for log collection. This would also include retention, access management vulnerabilities, user unprovisioning vulnerabilities, lack of forensic readiness, and operating system vulnerabilities.	**Medium**
R.31	X	**Loss or compromise of security logs**: security logs can be vulnerable due to lack of policy or poor procedures for log collection, basically for manipulation of forensic investigation.	**Medium**
R.32	X	**Backup lost, stolen**: the high-impact risk affects company reputation, all backed up data, and service delivery. It also occurs owing to inadequate physical security procedures, access management vulnerabilities, and user unprovisioning vulnerabilities.	**Medium**
R.33	X	**Unauthorized access to premises**: including physical access to machines and other facilities, the probability of malicious actors gaining access to a physical location is very low, but in the event of such an occurrence, the impact to the cloud provider and its customer is very high. It can affect company reputation data hosted on premises.	**Medium**
R.34	X	**Theft of Computer Equipment**: it can affect company reputation and data hosted on premises; the risk is due to inadequate physical security procedures.	**Medium**
R.35	X	**Natural Disasters**: it can have a high impact on the business involved in the event of its occurrence. If business has untested business continuity or disaster recovery plan, its reputation, data, and service delivery can be severely compromised.	**Medium**

2.4 Summary

This chapter presented a brief overview on topics of cloud Computing for the deployment models, service models, essential characteristics, hosting and governance. In more detail, it has been discussed the main activities in cloud environment that have strong benefit from or actual need of monitoring. To contextualize and study cloud monitoring, background and definitions for key concepts have been provided. It was also discussed the different sides on security monitoring in the cloud Computing environment. Finally, the risks involved in the migration process and utilization of services in Cloud Computing have been presented.

Chapter 3

Service Level Agreement

This chapter discusses the importance of service level agreements as a way to specify guarantees over parameters being monitored, their use in information technology and new mechanisms to represent such agreements.

3.1 Unmeasurable Qualities

When trying to formalize risk (R), at times there is a need to determine tangible values for intangible assets. Risk is directly linked to the degree of probability of a threat to occur and to the degree of negative impact of the incident caused by the threat to the organization [NIST, 2013a], while also measuring the implemented protection effectiveness:

$$R = \frac{(GPO \times GIN)}{GEP}$$

Where:

GPO: probability of occurrence of the threat;

GIN: degree of negative impact of the incident caused by the threat to business;

GEP: degree of effectiveness of the implemented protection.

These variables are intangible and unmeasurable. Overall, the qualities specified in an SLA can be classified into measurable and unmeasurable. The measurable qualities are those that can be measured automatically by means of metrics. While the unmeasurable qualities do not allow an automatic measurement, they cannot be evaluated by a method that results in a single value. In these cases, sets of secondary metrics that measure specific aspects of unmeasurable qualities are used.

The following are **measurable qualities** found in IT services [Bianco et al., 2008]:

1. **Accuracy:** the error rate threshold for the service during a specific period of time;
2. **Availability:** probability that the service will be available when needed;
3. **Capacity:** number of concurrent requests that the service supports;

4. **Cost:** cost of service;
5. **Latency:** the maximum time between the arrival of the request and the time to complete the request;
6. **Provisioning time:** time required for the service to become operational;
7. **Reliability of messages:** guaranteed delivery of messages;
8. **Scalability:** ability to increase the number of operations performed successfully in a time frame.

Now follows a list of **unmeasurable qualities** [Bianco et al., 2008]:

1. **Interoperability:** ability of intercommunication with other services;
2. **Modificability:** frequency of service changes (interface or implementation);
3. **Security:** ability to resist unauthorized use while providing service to legitimate customers (clients/tenants).

3.2 Metrics

A metric is a standard for measurements, and its value is the result of measuring something. A metric provides a numerical description of a particular feature of the items under investigation. The metric defines both what is being measured (the attribute) and how it is being measured (the unit of measurement)[Herrmann, 2007].

Measurement is the process of collecting metrics and establishing rules for interpretation of the results. Any restrictions or related controls are defined in the measurement process[Payne, 2006].

Each metric can return values such as:

i) number expressing an absolute value of a measured element;

ii) percentage expressing a measured component relative to the total of the elements;

iii) average expressing a mean value of an element relative to a set of elements;

iv) other quantifiable values.

3.3 Security Metrics

Security metrics are a technique which one can monitor and compare the level of security and privacy, the privacy state (status) or the security record of a computing environment. The judicious use of security metrics promotes transparency, decision-making, predictability and proactive planning[Hayden, 2010].

As an example of a security metric specification:

- **Id-Metric**: $\text{Met}_{2.3.18}$

- **Metric Name**: Packet filtering
- **Units**: % (percentage)
- **Formula**: $\mathrm{x} = \frac{\mathrm{Count(Incidents\_Packet\_filtering)}}{\mathrm{Total\_Packet\_filtering}} * 100$
- **Version**: 1.3.6
- **Description**: the Packet filtering Security Metric ($Met_{2.3.18}$) is monitoring unusual activity on the network when packet-filtering devices forward or deny packets based on information in each packet's header, such as IP address or TCP port number. A packet-filtering firewall uses a rule set to determine which traffic should be forwarded and which should be blocked.
- **Place**: Firewall, Router, Proxy, IPS, IDPS, etc.

3.3.1 Time Series Analysis

Time Series Analysis (TSA) can extract long time trends from data, both upwards and downwards, that may also present seasonal behavior.

Time series are formed by sequential observation collected from one or more variables ordered in time. For univariate series observations can be represented in the form (x_1, x_2, .., x_n), where the indices are instants of time $\mathrm{t} \in [1, \mathrm{n}]$ where n is the number of observations. In multivariate series, k represents the number of variables observed at each time t. These sets are represented in the form (x_{1t}, x_{2t}, .., x_{kt}), where $\mathrm{t} \in [1, \mathrm{n}]$.

The modeling of time series allows for understanding their behavior and their properties, which in turn allows the prediction of their behavior over time. This is useful since anticipating the series behavior helps in decision-making processes. The starting point is to perform the series decomposition in order to derive patterns.

The series decomposition will identify which components are working properly in that particular set, besides allowing to obtain indexes and/or equations to forecast future periods of the series.

In this proposal, using the classical model, all time series are seen as composed by four patterns:

(i) Trend (T): describes the behavior of the depicted variable in the time series in the long run. There are three basic goals for such identification: to evaluate its behavior for use in forecasts, to subtract the trend from the series for easy viewing of other components and to identify the level of the series, i.e. the typical value or range of values that the variable may assume;

(ii) Cyclical variations or cycles (C): fluctuations in the variable values longer than a year that are repeated with a certain periodicity;

(iii) Seasonal variations or seasonality (S): fluctuations in the variable values of less than a year, repeated every year, usually depending on the seasons, holidays, festivals and so on; if the data is recorded every year there will be no influence of seasonality in the series;

(iv) Irregular variations (I): unexplained fluctuations, probably the result of random and unexpected events such as natural disasters, terrorist attacks, untimely government decisions etc.

This model has two options to describe the equation:

- The additive model, where the value of the series (Y) is the sum of the component values:

$$Y = T + C + S + R$$

- The multiplicative model, where the value of the series (Y) is the product of the component values:

$$Y = T \cdot C \cdot S \cdot I$$

Obtaining the trend can be done in three ways: through a regression model (linear model such as the - line), by means of moving averages, or through exponential fit, which is nonetheless a moving average.

In this proposal, security metrics will be collected by monitoring agents and treated as time series.

3.3.2 Uncertainty

In metrology, the result of a measurement is not meaningful if a statement of the measurement's uncertainty is not specified. This statement allows users to assess the quality of the measurement results and to build confidence so as to compare results and use them within the range of the measurement uncertainty.

In the context of cloud services, it is critical that the customer of a measured resource be confident about the measurements operated on that resource. These measurements will feed security metrics that could be compared against thresholds to determine the range of acceptable results and then trigger possible reactions.

In this proposal, the security metrics will be validated during the collection process.

3.3.3 Calibration & Measurement Standard

Once new security metrics and units of measurement have been defined for cloud service security properties that can be reusable and comparable, the next step would be the calibration of the measurement systems used for measurement of cloud service security properties against established measurement standards. This would bring better accuracy and consequently better understanding of the security properties behavior involved.

3.4 Service Level Agreement

The specification of guarantee parameters assures that the quality of services is an essential mechanism in environments where outsourcing is used. This Section discusses the importance of service levels as a way to specify such guarantees, its use on information technology and security services, and finally ways of representing such agreements.

3.4.1 Definition

An SLA is part of the contract between the service consumer and service provider and formally defines the level of service. The TeleManagement (TM) Forum's SLA Management Handbook [SLA, 2005] underscores the importance of SLAs for the telecommunications industry:

> "It is the Service Level Agreement (SLA) that defines the availability, reliability and performance quality of delivered telecommunication services and networks to ensure the right information gets to the right person in the right location at the right time, safely and securely. The rapid evolution of the telecommunications market is leading to the introduction of new services and new networking technologies in ever-shorter time scales. SLAs are tools that help support and encourage customers to use these new technologies and services as they provide a commitment from SPs (Service Providers) for specified performance levels [SLA, 2005]."

The Service Level Agreement (SLA) is part of the service contract between provider and customer, and describes the desired Quality of Service (QoS)
[Berberova and Bontchev, 2009]. An SLA alone does not guarantee that the specified qualities are met, but it defines the necessary monitoring mechanisms, points out the responsibilities and defines punishments and compensations if conditions are not met.

In this context, Muller [Muller, 1999] states that such agreements would have a minimal set of information, as described for:

- **Background**: contain a set of information that allows a reader, even non-technical, to understand the current service levels and its guarantees;
- **Parties**: identify the parties to the agreement, including the responsible party within IT, within the business unit and/or application user group;
- **Service**: quantify the volume of the service to be provided by the IT department;
- **Timeliness**: provide a qualitative measure of measure of deadlines for execution of user requests;
- **Availability**: describe the periods where the service will be available to the end users;
- **Limitations**: describe the limits of service provided, while it is capable of peak usage, caused by large demands of use;

- **Compensation**: display the mechanisms used to compensate the user in situations where violations of the contracted service levels occur;
- **Measurements**: describe the process of monitoring service levels, and should also include a brief description of the data collection and extrapolation processes;
- **Renegotiation**: describe how and under what circumstances the SLA can be changed to reflect changes in the environment.

In information technology (IT) services, SLA use takes a different approach than that of telecommunication services. The agreement shall represent both customer's and provider's expectations. As such, obligations may be specified for both parties. The scope of information contained in the agreement is also differentiated. According to Bianco et al. [Bianco et al., 2008], a properly specified IT SLA-based describes each service offered and addresses:

- how delivery of the service at the specified level of quality will become realized;
- the parties involved;
- which metrics will be collected;
- who will collect the metrics and how;
- actions to be taken when the service is not delivered at the specified level of quality and who is responsible for doing them;
- penalties for failure to deliver the service at the specified level of quality;
- how and whether the SLA will evolve as technology changes, e.g. multi-core processors improve the provider's ability to reduce end-to-end latency.

3.4.2 SLA Life Cycle

The SLA is not a static document and its proper use is a result of the implementation of various activities conducted at different stages of his life. According to [SLA, 2005]:

1. **Definition**: this phase is focused on the identification of the service and its features, as well as the definition of quality parameters that will be provided to users;
2. **Negotiation**: in this phase values are defined for the parameters of the service, the cost to the user and penalties if the SLA is violated;
3. **Implementation**: the service is prepared for consumption for the user;
4. **Execution**: the phase of operation and service monitoring. In this phase, the quality parameters specified are evaluated for compliance of the SLA;
5. **Evaluation**: in this phase the provider assesses the quality of the service provided;

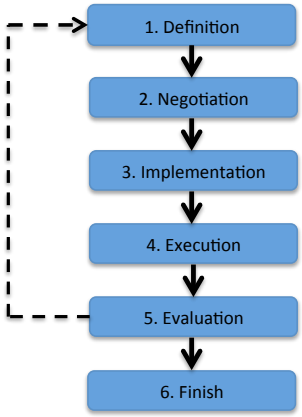


Figure 3.1: Life cycle of an SLA.

6. **Finish**: it matters finalization of the service, whether for reasons of expiration of contract or breach of SLA.

The figure 3.1 shows the life cycle of an SLA, and it consists on six phases:
The renovation process in the SLA occurs from phase 5 (Evaluation) to phase 1 (Definition).

3.4.3 SLA Parameters

In this proposal, the SLA specifies the Service Level Objectives (SLOs), which are the actual topics to be measured by the SLA. Each SLO may be composed of one or more Quality of Service metrics. The combinations of these metrics Quality of Services (QoSs) within a SLO depend greatly on the architecture of the service being provided.

A SLO is defined in terms of achieving a level of service within an agreed set of security metrics for a certain period of time, explaining how and where it should be measured. So SLA is the entire agreement, specifying what the service provided, which is your support, periods, costs and responsibilities of the parties involved, and SLOs are measurable characteristics of the SLA, such as availability, frequency, response time, quality and others.

It is crucial to emphasize that, in the context of an SLA, service level monitoring is as important as their specification. For this purpose, metrics are used to assess compliance with the desired qualities of service. The way these metrics are measured depends on the type of service and quality features that one wants to measure.

3.5 Security-SLA

The increasing use of outsourced IT services causes a growing concern with issues involving privacy and security [Jaatun et al., 2012]. Thus, it is natural for such issues to be addressed in SLAs, which allow the customer to specify security levels that must be guaranteed for the contracted services. However, the specification of SLAs involving security features (Security-SLA) presents challenges that involve the specification of security levels, the representation of these levels and finally monitoring them.

In the literature, the definition of security parameters can be done in two ways: through security policies or from security metrics.

The specification of security settings through policies, as proposed in [Casola et al., 2006], considers the Security-SLA as a set of policies expressed in standard language, e.g. WS-Policy [W3C, 2007b]. Although this approach is able to clearly specify the desired levels of security, the use of policies fail when specifying mechanisms for monitoring, and ignore the representation of various members of SLA information.

This specification from security metrics is a commonly used method that allows definition not only of the security parameters but also of the monitoring process. Unlike the specification of policies, the specification from metrics is based on a set of security metrics that allows checking whether a particular goal (control) is being fulfilled or not.

Table 3.1 presents a group of the metrics about access control with yours respective parameters that must monitored and evaluated by the Security-SLA.

Table 3.1: Parameters of Security-SLA

Class	**Description (requirement)**	**Value**
Access Control	Password size	$>$ 10 characters
	Password hashed	Yes
	Password change	$<$ 45 days
Operational	Mean-Time Between Security Incidents	2 hours
	Mean-Time to Incident Recovery	6 hours
	Mean-Time to Mitigate Vulnerabilities	1 hour

In access control, the parameter to "Password size" defines each password must be at least 10 characters, the parameter "Password encryption" defines that each password must saved in hashed form, and the parameter "Password change" defines that each password must change at most in 45 days. In Operational, the parameter to "Mean-Time Between Security Incidents" defines each security incidents must be at least 2 hours, the parameter "Mean-Time to Incident Recovery" defines that each incident recovery must be at least 6 hours, and the parameter "Mean-Time to Mitigate Vulnerabilities" defines that each mitigate a vulnerabilities must change at most in 1 hour.

The form of representation agreements Security-SLA is an important aspect the monitoring solution, since the entire monitoring process of the agreement is made in an automated fashion. Therefore, the search for a suitable language to the representation of these agreements considered different existing standards, which at first were not focused on the representation of security agreements.

Unlike traditional SLA agreements that have expressed their parameters as numerical values [Muller, 1999], e.g. CPU utilization, network traffic, etc., agreements Security-SLA have a very strong relationship with security policies. Despite this relationship, the representation and evaluation of policies for compliance agreements is unnecessary, since the existing metrics in the agreement should be sufficient to indicate whether a policy is being fulfilled.

3.6 Monitoring Security-SLA

The Cloud Computing model is always strongly based on guaranteeing Quality of Service (QoS), as specified in the Security-SLA, and therefore useful security monitoring will almost always be related to the terms in which the Security-SLAs are specified.

Both physical and virtual systems have to be monitored in order to detect, or even anticipate, system behavior changes that could have an impact on QoS. Therefore, defining the appropriate monitoring security metrics for each type of cloud monitoring is a crucial aspect. There are, however, several issues that need to be considered when defining these security metrics:

(i) In the case of security monitoring, security metrics have to provide information in the same terms as the Security-SLA is specified, i.e. using the same parameters.

(ii) Virtual system monitoring metrics have to be defined in a similar way, but probably including additional internal information. Again, this is strongly dependent on the terms in which Security-SLAs are defined, and therefore, strongly varies from one cloud service to other.

(iii) The case of physical system monitoring are those used in regular distributed systems, such as hardware metrics (CPU, memory, physical storage, network traffic, etc.), operating system metrics (system load, virtual memory, etc.) and so on.

Customers must establish trust relationships with the CSP and understand risk in terms of how the provider implements, deploys, and manages security on their behalf [Services, 2011, Zissis and Lekkas, 2012]. The CSP must address the fundamentals of security and privacy, such as identity management, access control, data control, network access, protected communication, and so on, agreed with the customer by means of the corresponding Security-SLA as part of the service. However, customers can implement their own monitoring infrastructure on top of the service as a part of the client-oriented monitoring to know the service behavior.

3.7 Summary

This chapter presented the main properties about unmeasurable qualities, metrics, security metrics, and how the analysis of temporal series can predict a security metric behavior. It described SLA and Security-SLA, the composition of a Service Level Agreement by Service Level Objectives, and the relationship between an SLO and a security metric. Finally, it presented issues that need to be addressed in the security monitoring scenario of a Security-SLA.

Chapter 4

Related Work

This chapter presents the summary of related work for this proposal, the tools and frameworks used in monitoring cloud environments and a brief description of the guides for monitoring Security-SLAs.

4.1 Summary of Related Work

Security and privacy are among the most discussed topics when researching about information migration from traditional systems to Cloud Computing [Chen et al., 2010]; they have been consistently ranked as one of the top challenges to cloud adoption, but it is not clear which security issues are particular to cloud computing. At this point, however, these issues do not appear to require completely new security controls but instead the creative usage of existing security techniques.

Despite all the cloud advantages, enterprise customers are still reluctant to deploy their business in the cloud. Security and privacy are among the most discussed topics in research about information migration from traditional systems to Cloud Computing [Subashini and Kavitha, 2011]. In this scenario, several groups and organization are interested in developing security solutions and standards for the cloud, for example, the Cloud Security Alliance (CSA) is gathering solution providers, nonprofit organizations and individuals to discuss current and future best practices for information assurance in the cloud [CSA, 2014a, CSA, 2014b].

The initial surveys [Foster et al., 2008] described the intricate relations between the Cloud and Grid Computing paradigms, but Cloud Computing is not a completely new concept [Fox et al., 2009]. It has connections to other relevant technologies such as utility computing, cluster computing and distributed systems in general [Ahmed et al., 2012].

The conceptual, technical, economic and user experience characteristics of Cloud Computing are described in [Vaquero et al., 2008, Liu et al., 2010a, Buyya et al., 2011a], which state more clearly what cloud computing is and so contribute to distinguish it from other research areas. Taxonomies for cloud computing services [Rimal et al., 2009, Hoefer and Karagiannis, 2010] have also been proposed, allowing for the extraction of a consensus definition as well as a minimum definition containing the topic's essential features. Issues of security and privacy are relegated to a background role or their im-

portance is just mentioned in passing, with assumptions that there must be mechanisms to monitor them but no word on how to do it. Nevertheless, research from Dana Petcu [Gonzalez et al., 2012, Petcu, 2014b, Petcu and Craciun, 2014, Petcu, 2014a] intended to serve the design of an SLA-based cloud security monitoring system using taxonomies for the fields of Cloud Computing, monitoring, security and SLAs. In their approach, Security-SLA parameters are composed of system metrics using mapping techniques. However, they consider neither resource metric monitoring nor Security-SLA violation detection.

In the cloud environment, the problem is to find a method of security assessment suitable for services. Henning [Henning, 1999] proposed to evaluate a service against security domains, each of which would be evaluated separately and assigned a level (from 1 to 4). In contrast, this work's essential goal is to propose a new methodology which can be applied to different security requirements to generate security metrics.

In [Krautsevich et al., 2010, Krautsevich et al., 2011] the authors investigated how one could define "more secure" relations and described the basic formal model for a description and analysis of security metrics. In this thesis the basic goal is slightly different; it focus on already proven usage of formal models, where each security metric has some complex range of values to be normalized to the scale [0-4], in a monotonic way.

The goal of SLA Management [SLA, 2005] is to assist two parties, customer and provider, in developing a Service Level Agreement (SLA) by providing a practical view of the fundamental issues. Other researches described methodologies for SLA management [Gonzalez and Helvik, 2012, Wu et al., 2013a, Rak et al., 2013], some other attempted to integrate security policies with SLA or security requirements with Security-SLA using QoS models [de Chaves et al., 2010b, Emeakaroha et al., 2010a, Bernsmed et al., 2011, Zhien and Yiqi, 2012], yet only tackling control mechanisms to achieve SLA compliance over CPU, memory and network bandwidth requirements [Brandic et al., 2009] and self-manageable cloud services [Brandic, 2009]. This thesis was directed to generate security metrics for any infrastructure device or cloud service through Security-SLAs, providing the necessary control mechanisms to deal with these metrics.

Comuzzi et al. [SLA@SOI, 2010] define the process for SLA establishment adopted within the EU SLA@SOI project's framework. The authors propose the architecture for monitoring SLAs considering two requirements introduced by the SLA establishment: the availability of historical data for evaluating SLA offers and the assessment of the capability to monitor the terms in an SLA offer. Hwever, they do not consider monitoring of low-level security metrics and mapping them to high-level SLA parameters for ensuring the Security-SLA goals.

In [ISO/IEC-27000:2009, 2009, ENISA, 2009c, CAMM, 2010, SSAE, 2011, IFAC, 2011, CSA, 2011, ISO, 2011, ISO/IEC-27017:2014, 2014], security models or guidelines were described for security control specifications deployable by the cloud service provider to ensure the service's security. Such solutions are not flexible and scalable, thus hindering new adjustments to the service being monitored. To tackle this issue, this thesis' proposal makes use of the GQM approach [Basili et al., 1994, Basili, 2002].

The literature on traditional Return on Investment (ROI) [Cavusoglu et al., 2004, Keshavarzi et al., 2013] and Return on Security Investment (ROSI) [Stout et al., 2006,

Brocke et al., 2007, Tsalis et al., 2013] describes traditional solutions to compute Total Cost of Ownership (TCO), Net Present Value (NPV) and Internal Rate of Return (IRR). These approaches, however, only considered the immediate costs of contracting and migrating to the cloud and failed to do it for the long-term costs of operating in the cloud. Worse still, they also fail to deal with the hidden costs that could damage the expected return. This thesis' proposal approaches the problem by describing a method to compute ROSI for each security requirement being monitored, so as to offer the client subsidies to help him/her decide on migrating or not to the cloud environment.

Solutions to the problem of adding resource allocation into SLAs [Hu et al., 2009, Younge et al., 2010, Buyya et al., 2011b, Al-Haj et al., 2013, Nakamura et al., 2014] proposed algorithms for IaaS, PaaS and SaaS providers who wanted to minimize infrastructure cost and SLA violations in dynamic, resource-sharing Cloud Computing environments. So, due to the willingness of providers to postpone security issues, one might conclude that previous research work had not contemplated security in the VM migration process. In this thesis' proposal, security was prioritized.

In this scenario, Table 4.1 presents the additional topics on Cloud, security, Security-SLA, allocation of resources and Return On Security Investment that are discussed in this proposal. Column **Topic** presents the related subject, column **Description** presents the related characteristics and column **Reference** presents related articles.

Table 4.1: Summary of related work

Topic	**Description**	**Reference**
Cloud Computing	works proposing surveys and taxonomies of cloud computing services	[Letaifa et al., 2010, Zhang et al., 2010, Mell and Grance, 2011, Atzori et al., 2011, Voas et al., 2012, NIST, 2013b, NIST, 2013c]
Security properties	presents formal specification model, measure the quality of protection (QoP), and rigorous analysis of security metrics	[Foley et al., 2006, Mana and Pujol, 2008, Krautsevich et al., 2010, Buyya et al., 2011a, Cloud-Council, 2012, Vacca, 2013, Cloud-Council, 2015]

Continued on next page

Table 4.1 – *Continued from previous page*

Topic	Description	Reference
Security policies x SLA	methodologies that describe the integration of security policies with SLA/Security-SLA	[Righi et al., 2004, Arshad et al., 2009, Hayden, 2010, de Chaves et al., 2010b, Buyya et al., 2011a, Bernsmed et al., 2011, Cloud-Council, 2012, Zhien and Yiqi, 2012, Rahulamathavan et al., 2014, Cloud-Council, 2015]
SLA manager	methodologies that describe Quality of Service (QoS) model and are expressed using Service Level Agreements (SLAs)	[Brandic et al., 2009, Brandic, 2009, Buyya et al., 2011a, Gonzalez and Helvik, 2012, Wu et al., 2013a, Rak et al., 2013]
Security manager	how to manage security in complex systems such as cloud computing	[Arshad et al., 2009, Halonen and Hatonen, 2010, Krutz and Vines, 2010, Bayuk, 2011, Rebollo et al., 2012, Petcu and Craciun, 2014]
Security metrics	A comparative analysis model and taxonomy of security metrics	[Payne, 2006, Jaquith, 2007, Savola, 2007a, Savola, 2007b, Rochwerger et al., 2009, Savola, 2009, Bruno, 2011, Frenz, 2010, Frenz, 2010, Brotby and Hinson, 2013]
Security metrics x SLA	methodologies used in the specification of metrics for Security-SLA, and monitoring solution	[Henning, 1999, Righi et al., 2004, Emeakaroha et al., 2010a]
Goal-Question-Metric methodology	Goal-Question-Metric (GQM) methodology proposed for empirical measurements in software testing, based on well-defined goals	[Basili et al., 1994, Basili, 2002]
GQM methodology x Security-SLA	GQM model being used together with the COBIT framework to specify metrics for Security-SLA in cloud computing	[Putri and Mganga, 2011, Zhengwei et al., 2013, Al-Hassan, 2013]
GQM x Security-SLA in cloud	GQM model being used to build a metrics hierarchy to generate an index of security in cloud computing	[Silva et al., 2012, Silva and Geus, 2014c, Silva and Geus, 2014a, Silva and Geus, 2014b]

Continued on next page

Table 4.1 – *Continued from previous page*

Topic	**Description**	**Reference**
Security framework or tools	specify (high-level) security controls that should be deployed by the cloud service provider (CSP) to ensure the service is secure	[NIST, 2002, ISO/IEC-27000:2009, 2009, ENISA, 2009c, CAMM, 2010, SSAE, 2011, Dempsey et al., 2011, IFAC, 2011, CSA, 2011, ISO, 2011, ISO/IEC-27017:2014, 2014, Hogben and Dekker, 2012, FedRAMP, 2013]
Return On Security Investment	present solutions to the problem of Return On Security Investment (ROSI)	[Cavusoglu et al., 2004, Stout et al., 2006, Brocke et al., 2007, Keshavarzi et al., 2013, Tsalis et al., 2013, Nakamura et al., 2014]
Allocation of resources x SLA	present solutions to the problem of allocation of computing resources of the clouds based on a number of criteria	[SLA, 2005, Hu et al., 2009, Liu et al., 2010b, Younge et al., 2010, Yazir et al., 2010, Younge et al., 2010, Liu et al., 2010b, Buyya et al., 2011b, Beloglazov et al., 2011, Al-Haj et al., 2013, Nakamura et al., 2014]

4.2 Cloud Monitoring

Table 4.2 describes de current tools and frameworks for security monitoring in the Cloud Computing environments. The column Acronym presents the name of the tools/framework, the column Type presents: (I) executes like Infrastructure-as-a-Service (IaaS), (P) executes like Platform-as-a-Service (PaaS), (S) executes like Software-as-a-Service (SaaS), and (F) executes like Physical system. The column Class presents: (C) Commercial Cloud Monitoring Platforms, (R) Research prototypes cloud monitoring platforms, and (O) Open source cloud monitoring platforms. Finally, the column Model presents: (A) SLA-oriented cloud monitor tools, (T) Cloud security monitor tools, and (S) Services for assessing cloud performance and dependability.

Table 4.2: Summary of current tools and frameworks

Acronym	Type	Class	Model	Short description
Aftersight	I P S	R	T	Analyze the behavior/execution of a VM from non-deterministic events and inputs to a VM [Chow et al., 2008]
Amazon Cloud-Watch	I	C	T	Monitoring service for Amazon Web Service cloud resources and the applications run on AWS [Amazon, 2008]
Aneka management	P	C	A	Platform for managing deployment and execution of applications on clouds based on QoS/SLA [Manjrasoft, 2008]
AzureWatch	I P S F	C	T	cloud-based service dedicated to advanced monitoring, auto-scaling of Azure-based solutions [Paraleap, 2011]
Azure Suite	I P S F	C	T	Monitoring and managing servers and coordinating resources for the applications [Microsoft Windows Azure, 2008]
Boundary	I P S F	C	S	Monitor, manage and optimize the cloud, infrastructure, platform, service, and servers [Boundary, 2010]
CA UIM	I P S	C	A	CA Unified Infrastructure Management, scalable platform SLA for monitoring [CA Tech., 2015]
CASViD	S	R	A	Application level monitoring for SLA violation detection in clouds using SNMP [Emeakaroha et al., 2010b]
Centrify	S	C	S	Identity management and auditing for cloud across Identity-as-a-Service (IDaaS) [Centrify, 2008]
CipherCloud	S	C	T	Monitoring service, protection controls, data loss prevention, malware detection [CipherCloud, 2011]
CloudClimate	I F	C	S	Reporting monitored metrics as measured from different cloud infrastructure [Paessler, 1998]
CloudCmp	I P S	O	S	Benchmark suite for cloud platforms, it runs on a cloud instance as a web service [CloudCmp, 2011]
CloudCompass	I P	O	A	Extension of the SLA WS-Agreement for cloud, and dynamic QoS rules [GRyCAP, 2009]
CloudCruiser	I P S F	C	S	Monitor, manage and optimize the hybrid cloud architecture, compare public/private service [Cloud Cruiser, 2010]
CloudFlare	I P S	C	T	Protects websites, optimize delivery, blocks threats, limit abusive bots and crawlers [CloudFlare, 2009]
CloudFloor	I P S F	C	S	Monitoring service, SaaS-based infrastructure of any provider, critical communications [Everbridge, 2011]

Continued on next page

Table 4.2 – *Continued from previous page*

Acronym	Type	Class	Model	Short description
CloudHarmony	I P S F	C	S	Provides a set of: OS-layer metrics, application-layer benchmarks and user-layer tests [CloudHarmony, 2013]
Cloudkick	I	C	T	Monitor, analyze on the availability/performance of the websites and cloud resources [Rackspace, 2012]
CloudPassage	I P S	C	T	Security monitoring/control platforms, and integration with security tools and systems [CloudPassage, 2011]
CloudSec	I	R	T	Security monitoring for multiple concurrent VMs on a cloud platform in an IaaS setting [Ibrahim et al., 2011]
CloudSleuth	I P	C	S	Web-based cloud performance visualization tool, analysis of IaaS and PaaS providers [Dynatrace, 2008]
CloudStack Zenpack	I F	O	T	Scalable Infrastructure-as-a-Service Cloud Computing platform [CloudStack, 2008], ZenPack Zenoss extension [Zenoss, 2011]
CloudStone	I P S	O	S	CloudStone: multi-platform, multi-language benchmark and measurement tools for Web 2 [Sobel et al., 2008]
CloudWatcher	S	R	T	CloudWatcher: network security monitoring using OpenFlow in dynamic cloud networks [Shin and Gu, 2012]
Cloudyn	I P S	C	S	Monitor, manage and optimize hybrid clouds for performance and cost [Cloudyn, 2012]
Collectl	I P S	O	T	Benchmarking and monitoring a system's: CPU, disk, memory, network, etc. [Collectl, 2007]
DARGOS	I	O	T	Distributed Architecture for Resource manaGement and mOnitoring in cloudS [Corradi et al., 2012]
DocTrackr	I F	C	T	Control extends throughout set user privileges for each person you share a document with [Intralinks, 2007]
FBCrypt	I P S	O	T	Encrypts the I/O between a VNC client and a user VM using the Virtual Machine Monitor [Egawa et al., 2012]
Ganglia	F	O	T	Scalable distributed monitoring system such as clusters and grids [Ganglia Software, 2000]
GMonE	I P S F	R	T	General-purpose cloud monitoring tool for IaaS, PaaS, SaaS and physical system [Montes et al., 2013]

Continued on next page

Table 4.2 – *Continued from previous page*

Acronym	Type	Class	Model	Short description
GridICE	F	R	T	Monitoring service for grid systems and virtual organization-oriented service [Andreozzia et al., 2005]
Groundwork	I P S	O	T	Scalable monitoring a system's: virtualization, network, application, etc. [Groundwork Software, 2011]
HP OpenView	F	C	T	Integrated management of networks and systems for distributed computing environments [Hewlett-Packard, 2005]
Hyperic-HQ	I P S	O	T	It monitors operating systems, middleware and applications (MySQL) [Hyperic-HQ Software, 2010]
HyperWall	I P S	R	T	Scalable monitoring a system's: virtualization, application, server, storage data, etc. [Szefer, 2013]
IBM Tivoli Monitoring	F	C	T	System monitoring manages O. S., databases, servers in distributed/host environments [IBM, 2005]
Intermapper Cloud Monit.	I F	C	T	Network management and monitoring tools across multiple platforms [HelpSystems, 2012]
JasMINe	I P S	O	T	It monitors operating systems, applications, servers, etc. [Jasmine Software, 2010]
K-Tracer	I P S	R	T	Dynamically analyze Windows kernel-level code and extract malicious behaviors from rootkits [Lanzi et al., 2009]
KVMSec	I P S	R	T	A security extension for Linux kernel virtual machines [Lombardi and Pietro, 2009]
Lares	I P S	R	T	Architecture for secure active monitoring using virtualization [Payne et al., 2008]
Lattice	I F	R	T	Framework for monitoring virtual machines executing under hypervisor control [Galis et al., 2010]
Livewire	I P S	R	T	A virtual machine introspection based architecture for intrusion detection [Rosenblum, 2003]
Logic Monitor	I P S F	C	T	Cloud performance monitoring for infrastructure and applications [Logic Monitor, 2009]
LoM2HiS	I	R	A	Bridging the gap between monitored metrics and SLA parameters in cloud environments [Emeakaroha et al., 2010a]
Lycosid	I P S	R	T	Hidden process identification and detection: comparing guest view with a VMM image [Jones et al., 2008]
MARS	I	C	T	Cisco security: Monitoring, Analysis and Response System, designed to monitor threats [Cisco, 2010]
MAVMM	I P S	R	T	Malware analysis of the applications running inside a guest O.S. (VMM) [Nguyen et al., 2009]

Continued on next page

Table 4.2 – *Continued from previous page*

Acronym	Type	Class	Model	Short description
MISURE	I P S	R	T	Monitoring infrastructure is a monitoring-as-a-service for data analysis [Smit et al., 2013]
MonALISA	F	O	T	MONitoring Agents using a Large Integrated Services Architecture [Monalisa, 2005]
Monitis	I P S F	C	T	Cloud Monitoring of the infrastructure, websites, applications [Monitis, 2006]
Nagios	F	O	T	Monitoring and alerting for servers, switches, applications, and services [Nagios, 1996]
New Relic	I P S F	C	A	Monitoring application performance for SLA, availability, scalability, capacity [New Relic, 2008]
NICKLE	I P S	R	T	Based on a scheme memory shadowing to running VM and real-time kernel code authentication [Riley et al., 2008]
Nimbus	I P	O	T	Providing Infrastructure-as-a-Service and platform capabilities to the scientific community [Nimbus Project, 2006]
Okta	I P S F	C	T	Identity management for the Cloud, identity platform for developers, privilege provisioning [Okta, 2009]
OpenNebula Monitoring	I	O	T	Monitoring infrastructure is a monitoring-as-a-service for CPU, memory, etc. [OpenNebula, 2005]
OPNET	I S F	C	T	Monitoring infrastructure, analyze network performance and application performance [Riverbed, 2012]
Overshadow	I P S	R	T	Protects the privacy and integrity of application data in VM [Chen et al., 2008]
PacketTrap	I F	C	T	Virtual infrastructure monitoring, and network traffic and website surfing analysis [Dell, 2012]
PCMons	I P S	O	T	Private Clouds MONitoring Systems, information for monitoring data visualization [Project PCMONS, 2008]
PoKeR	I P S	R	T	Avoiding profiler of kernel rootkits: hooking behavior, kernel modifications, code injection [Riley et al., 2009]
Proofpoint	I P S F	C	S	Security-as-a-service, hybrid email services, combine SaaS with physical or virtual presence [ProofPoint, 2002]
QoS-MONaaS	S	R	A	Quality of Service MONitoring as a Service, a formal SLA and performance indicators violation [Adinolf et al., 2009]
Qualys	I P S F	C	T	Providing Security-as-a-Service, monitoring infrastructure, platform, and service [Qualys, 1999]

Continued on next page

Table 4.2 – *Continued from previous page*

Acronym	Type	Class	Model	Short description
ReVirt	I P S	R	T	Intrusion analysis through virtual-machine logging and replay [Dunlap et al., 2002]
Rkprofiler	I P S	R	T	Sandbox-based malware tracking using QEMU virtualization for Windows [Xuan et al., 2009]
Sandpiper	I P S	O	A	Automates the process of monitoring, reconfiguring VMs, and checking SLAs violations [Sandpiper, 2008]
SecMon	I P S	R	T	A secure introspection framework for hardware c using a VM Monitor for Windows OS [Wu et al., 2013b]
SecVisor	I P S	R	T	A tiny hypervisor to provide lifetime kernel code integrity [Seshadri et al., 2007]
Sensu	I	O	T	Monitoring platforms, independent agents and focused on extensibility and elasticity [Sonian, 2006]
sFlow	I F	C	T	Detecting, diagnosing, fixing network problems, and understanding application P2P, Web [sFlow, 2003]
SIGAR	I P S	O	T	System Information Gatherer and Reporter is a framework/monitoring for O.S. and hardware [Project Sigar, 2006]
SIM	I P S	R	T	Secure In-VM Monitoring, a framework for security monitoring applications [Sharif et al., 2009]
Site24x7	S	C	A	Website monitoring service, checking the availability, performance, SLA, uptime reporting [Site24x7, 2007]
SilverSky	I P S F	C	S	Security-as-a-service platform across network security services and email security services [SilverSky, 2013]
SLA@SOI	I	O	A	SLA management platform for monitoring of distributed systems based on events [SLA@SOI, 2010]
Snorby	I P S	O	T	Application for network/host security monitoring, and Intrusion Detection Systems (IDS) [Snorby, 2011]
SPAE	I	C	T	Network monitoring tool using SNMP for various protocols to cloud resources [Shalb, 2010]
Splunk' Storm	I P S	C	T	Cloud security monitoring for web-sites, applications, servers, networks, mobile, etc. [Splunk, 2005]
Threat Stack	I	C	T	Monitoring: user loggings, network connections, servers, firewall policies, audit alerts [Threat Stack, 2014]
TIMACS	F	R	T	Framework for management of large computing systems, scalable low level system monitoring [TIMACS, 2009]

Continued on next page

Table 4.2 – *Continued from previous page*

Acronym	Type	Class	Model	Short description
TrustVisor	I P S	R	T	Hypervisor protects pieces of application logic to be execute in isolation [McCune et al., 2010]
Up.Time	I P S	C	A	Provides SLA monitoring, and reporting the impact of each infrastructure element on SLA [Up.Time software, 2008]
Vaultive	I P S F	C	T	Encryption-in-use/gateway architecture being processed, searched, sorted while encrypted [Vaultive, 2012]
Vmware Cloud-Status	I P S F	C	T	Infrastructure and O.S. monitoring, application and middleware monitoring [Vmware, 2008]
VMWatcher	I P S	R	T	A malware detection mechanism between observed events at the VMM level and guest O.S. context [Jiang et al., 2007]
White Hat Security	I P S F	C	T	Focused on protecting your website from the ground up, including in the coding process [WhiteHat, 2001]
Zabbix	I P S F	O	T	Monitor performance/availability of servers, WEB applications, databases, networking, SLA [Zabbix, 2001]
Zscaler	I P S F	C	T	Protect from advanced persistent threats by monitoring all the traffic that comes in and out [Zscaler, 2008]

4.3 Guides for Security-SLA monitoring

The list below shows the guides produced as references to decision makers on monitoring the Security Service Level Agreements in Cloud Computing environment:

- In the US, NIST issued: Special Publication SP-800-137 [Dempsey et al., 2011] on Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations. This document is a guideline for organizing internal security processes focused on maintaining on-going awareness of information security, vulnerabilities, and threats. The focus in this document is on the customer, to allow the customer to continuously monitor security properties of the acquired service.

- The Cloud Standards Customer Council describe the Practical Guide to Cloud Service Level Agreements that is a collaborative effort that brings together diverse customer-focused experiences and perspectives into a guide for IT and business who are considering cloud adoption [Cloud-Council, 2012, Cloud-Council, 2015].

- The European Commission's European Cloud Partnership initiative, in European Network and Information Security Agency (ENISA), present a practical guide aimed

at the procurement and governance of cloud services. The main focus is on the public sector, but much of the guide is also applicable to private sector procurement. This guide provides advice on questions to ask about the monitoring of security (including service availability and continuity). The goal is to improve public sector customer understanding of the security of cloud services and the potential indicators and methods which can be used to provide appropriate transparency during service delivery [Hogben and Dekker, 2012].

- The Federal Risk and Authorization Management Program (FedRAMP) [FedRAMP, 2013] is the action for federal government agencies to abide by in the procurement of cloud services. It provides a standardized and centralized approach to security assessment, authorization and continuous monitoring for cloud-based services, and federal security requirements, e.g. Federal Information Security Management Act (FISMA) [NIST, 2002]. The program not only sets security requirements, it also monitors the implementation of security measures, for example, by quarterly periodic vulnerability scan reports with a specific focus on continuous monitoring.

4.4 Summary

This chapter presented a summary on Cloud Computing, Security, Security-SLA and Return On Security Investment, also presenting tools and frameworks used in monitoring cloud environments, and finally a brief description of guides for monitoring Security Service Level Agreements in Cloud Computing environments. To the best of our knowledge, none of the discussed approaches deals with mapping of low-level resource metrics to high-level Security-SLA parameters and Security-SLA violation detection at runtime, which are desirable features for enforcing Security-SLAs in cloud-like environments.

Chapter 5

Methodology Proposal (SMH)

This chapter presents the Security Metrics Hierarchy (SMH) methodology for the management of security for Cloud Computing environments using security metrics.

5.1 Security Metrics Hierarchy

In the 1970s, the Goal-Question-Metric (GQM) method [Basili et al., 1994] was designed to move testing for software defects from the qualitative and subjective state it was currently in to an empirical model, in which defects would be measured against defined goals and objectives that could then be linked to results.

The GQM methodology defines a measurement model on three levels: i) Conceptual level (goal) a goal is defined for an object for a variety of reasons, with respect to various models of quality, from several points of view and relative to a particular environment; ii) Operational level (question) a set of questions is used to define models of the object under study and then attention is focused on that object to characterize the assessment or achievement of a specific goal; iii) Quantitative level (metric) a set of metrics, based on the models, is associated with every question in order to answer it in a measurable way.

In our methodology, the security metrics hierarchy is generated directly from the GQM definition process, during which stage security features are mapped to corresponding security metrics. Table 5.1 shows the relationship (steps) between the GQM methodology, the Security Metrics Hierarchy (SMH) and Portfolio of Metrics.

Table 5.1: Relationship between: GQM, SMH and portfolio of metrics

GQM levels (1st step)	**SMH levels** (2nd step)	**Portfolio of metrics levels** (3rd step)
Conceptual level	Group metric	**Aggregation goal**
Operational level	Metric	**Group objective**
Quantitative level	Submetric	**Service level objective**

For each goal statement identified in the conceptual level, a group metric will be defined. The operational level identifies which objects or activities must be observed or collected to measure the individual components of the goal statement. Lastly, the

quantitative level defines which metrics remains explicitly aligned with the higher level goal statement.

Other Example: Security-related downtime, understanding how long your systems are up and available to users is a common IT metric. Understanding how security impacts availability is also important, particularly when you need to compare security to other IT challenges. Table 5.2 illustrates an example project for measuring security-related downtime using GQM methodology in SMH.

Table 5.2: GQM Project for Security-Related Downtime

GQM levels	**Id**	**Description**	**SMH levels**
Goal	1	The goal of this project is to understand security impacts on system **Statement**: availability by comparing security-related downtime to general, and availability from the perspective of the security team.	Group
Question	1.1	Question How often is the system down due to failure?	Metric
	1.1.1	Metrics in Time Between Failures	Submetric
	1.1.2	Failure Duration	Submetric
	1.1.3	Mean System Availability	Submetric
Question	1.2	Question how often is the system down due to maintenance?	Metric
	1.2.1	Metrics in Time Between Maintenance	Submetric
	1.2.2	Maintenance Duration	Submetric
	1.2.3	Mean System Availability	Submetric
	1.2.4	Metrics how often is Downtime the result of a security event?	Submetric
Question	1.3	Question Number of security events in time period	Metric
	1.3.1	Duration of Event Remediation	Submetric

This scenario demonstrates the importance of the perspective component of the GQM template. For the security team, understanding how much impact on general availability results from security-related issues would be important. But from the perspective of a system user, downtime is downtime. Users usually don't care that they are grounded as a result of a security problem, a misconfiguration, or the fact that Bob accidentally unplugged the wrong box? they just want the system back up.

5.1.1 Modeling Security Metrics Hierarchy

The security metrics hierarchy (Fig. 5.1) is derived from the GQM methodology. Its components are: i) Index of Security (IndSec); ii) Group Metrics (Met_i); iii) Metrics ($Met_{i.j}$); iv) Submetrics ($Met_{i.j.k}$).

In context, the nomenclature for the security metrics is $Met_{i.j.k}$, measured in the Cloud Computing environment for the the virtual machine (v) on host (h) and service (s), for layers IaaS, PaaS and SaaS. The reference i.j.k identifies the location of the metric in the

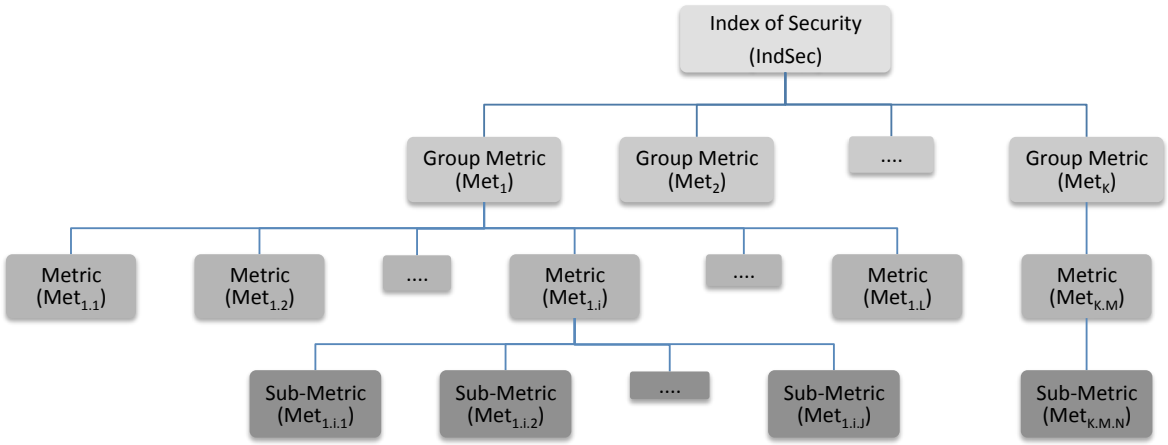


Figure 5.1: Security Metrics Hierarchy (SMH).

hierarchy, where: i refers to the security item, j refers to the group metric, and k refers to the primitive metric.

The Index of Security (IndSec) is defined as the value in a set of security items $(\mathrm{Met}_1, \mathrm{Met}_2, ..., \mathrm{Met}_n)$:

$$\mathrm{IndSec} = \mathrm{f}^*(\mathrm{Met}_1, \mathrm{Met}_2, ..., \mathrm{Met}_n)$$

The value of a metric group (Met_i) is defined as the value from a set of metrics $(\mathrm{Met}_{i.1}, \mathrm{Met}_{i.2}, ..., \mathrm{Met}_{i.m})$:

$$\mathrm{Met}_i = \mathrm{f}^1(\mathrm{Met}_{i.1}, \mathrm{Met}_{i.2}, ..., \mathrm{Met}_{i.m})$$

The value of a metric $(Met_{i.j})$ is defined as the highest value from a set of submetrics $(\mathrm{Met}_{i.j.1}, \mathrm{Met}_{i.j.2}, ..., \mathrm{Met}_{i.j.k})$:

$$\mathrm{Met}_{i.j} = \mathrm{f}^2(\mathrm{Met}_{i.j.1}, \mathrm{Met}_{i.j.2}, ..., \mathrm{Met}_{i.j.k})$$

The submetric represents a subpart of a metric; it is used when a metric can be specialized in several ways, with each one having a different contribution to the overall metric. An example of the portfolio of security metrics is shown in Table A in Appendix A (page 132).

The representation of a security metric in the hierarchy is described as follow: a value $\mathrm{Met}_{i.j.k}$ is a measurement for the for the group metric i, metric j and submetric k, or more vertical levels.

In this proposal, the Security Metrics Hierarchy should separate into two groups of security requirements: Infrastructure and Service. At the first level, which is the junction of Infrastructure and Service values, the function f^* can be rendered as one among several possible functions, like, for instance, Max, Min, AVG etc., where Max, Min and AVG

are, respectively, the largest, smallest and average value assumed by the group of submetrics. At the second level, which is a specific sublevel of Infrastructure or Service, the functions $f^1, f^2, \cdots, f^{n-1}, f^n$ can be calculated using formulas similar to their submetrics for infrastructure or Service.

The customer or provider may build as many levels in the SMH as required and use the functions deemed important to describe the security status. For example, in Infrastructure, the Firewall would be a group of metrics, with each submetric in the group generating information on a specific topic. If the customer wants to know the average value of this group, then she would use average (AVG), or if the customer or provider wants to know the Standard Deviation (SD), then SD should be used and so on.

5.1.2 Normalization of Security Metrics

The normalization process will convert each value of a security metric in SMH to a value in the range of [0-4]. Each expected value falls in the range [0, 1, 2, 3, 4], respectively corresponding to the expected security levels [Critical, High, Medium, Low, None].

The motivation behind value normalization is:

i) to extract a meaning for the values measured by the primitive metrics;

ii) to allow sorting them by their absolute value;

iii) to prevent the value domains of security metrics from having instances that are difficult to be compared with each other, and to simplify the computing model using a method to converge the values of each primitive metric measured to a common scale of values.

On the proposed scale of [0–4], the highest value (0) represents a security level less reliable and/or presenting a serious security problem. The lowest value (4) represents a security level that is safer and/or that does not present any security issue.

Logical metric

A metric of type logic must return a logical value measured from an event, e.g. “Anti-virus installed?”. After the normalization (Tab. 5.3) one gets:

Table 5.3: Normalization of logical metric

Index	Logical value (x)
0	Yes
1	
2	
3	
4	No

The normalization function is described as y = f(x), where x can be a measured logic value Yes or No:

$$y = \begin{cases} 0 & \text{if } x = Yes \\ 4 & \text{if } x = No \end{cases}$$

Depending on the nature of the measured value for the security metric, the normalization formula can be reversed for clarity of the result, example: the metric "Anti-virus configured?", it can take the value "4" to "Yes" and "0" to "No" (inverse of the metric "Anti-virus installed?").

Numerical metric

It is a metric that returns a numerical value representing an event, e.g. "number of security patches installed". After the normalization (Tab. 5.4) one gets:

Table 5.4: Normalization of numerical metric

Index	# **Numeric Metric (x)**
0	$]\ -\infty\ ,\ 0\ ]$
1	$]\ 0\ ,\ a_1\ ]$
2	$]\ a_1\ ,\ b_1\ ]$
3	$]\ b_1\ ,\ c_1\ ]$
4	$]\ c_1\ ,\ +\infty\ [$

With respect to the start and end boundaries of each interval, it has:

$$0 < a_1 \leq b_1 \leq c_1 < +\infty$$

The values a_1, b_1 *and* c_1 must be calculated using historical information about this security metric and the use of the techniques like sensitivity analysis.

The normalization function is described as y = f(x), where x is the value measured by the numerical metric:

$$y = \begin{cases} 0 & \text{if } x \in\ ]\ -\infty\ ,\ 0\ ] \\ 1 & \text{if } x \in\ ]\ 0\ ,\ a_1\ ] \\ 2 & \text{if } x \in\ ]\ a_1\ ,\ b_1\ ] \\ 3 & \text{if } x \in\ ]\ b_1\ ,\ c_1\ ] \\ 4 & \text{if } x \in\ ]\ c_1\ ,\ +\infty\ [\end{cases}$$

Again, depending on the nature of the measured value for the security metric, the normalization formula can be reversed for clarity of the result. Example: the metric "number of security patches installed", can take the value "0" to "$]\ -\infty\ ,\ 0\ ]$" and "4" to "$]\ c_1\ ,\ +\infty\ [$" (inverse of the metric "number of security patches not installed").

Figure 5.2 illustrates some behaviors of security metrics over time to the metrics:

- Met-1 - utilization rate of memory per VM;
- Met-2 - number of threats by VM running;
- Met-3 - time between mitigations for VM's;

- Met-4 - utilization rate of I/O network;
- Met-5 - utilization rate of CPU per VM;
- Met-6 - utilization rate of CPU per VM servers.

And after application of this normalization methodology of metric values for the range of values of [0-4], their behavior becomes predictable (result of the function f*(x)), and facilitates the analysis of this security level as well as the process of identifying the motivation and/or event that triggers changes in scale values.

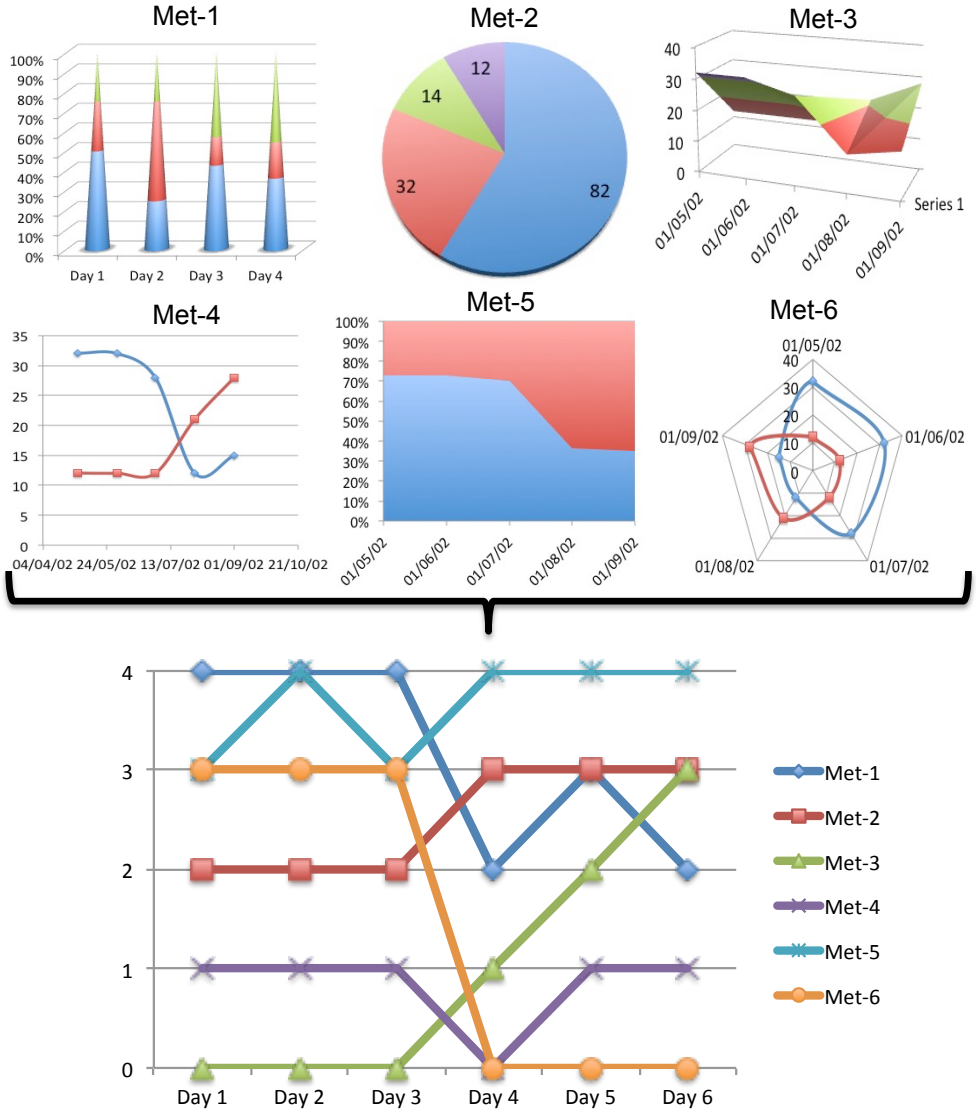


Figure 5.2: Result of normalizing some security metrics.

The next section describes the formalism to define security metrics, and features validating the scale of values.

5.1.3 Formal Security Metrics Hierarchy

In the measurement theory [Suppes and Zinnes, 1962, Finkelstein and Leaning, 1984], the representation theorem (simplify) was defining like:
Definition 1: Let Q be a set of elements, r and q be its members $(r, q \in Q)$. Let also $\mathbf{R} = \{R_1, ..., R_n\}$ be a set of relations on Q. The tuple $\langle Q, \mathbf{R} \rangle$ is called an empirical relational system. Measurement can be seen as an objective empirical function that assigns a real value to an element $(M : Q \mapsto \mathbb{R})$ and $\mathbf{P} = \{P_1, ..., P_n\}$ is a set of relations on $\mathbb{R}$ (reals), which is in a binary relation with $\mathbf{R}$ (i.e., each R_i corresponds to P_i).
Then:

$$\forall i \; R_i(r, q, ...) \Longleftrightarrow P_i(M(r), M(q), ...)$$

Where the *security measurement function* is representative if one of the following relations holds:

$$\forall r, q \in Q \; (r \sim_S q) \Longleftrightarrow M(r) = M(q) \text{ and}$$
$$((r \succ_S q) \Longleftrightarrow M(r) > M(q) \text{ xor}$$
$$(r \succ_S q) \Longleftrightarrow M(r) < M(q))$$

Where, $r \sim_S q$ means that r is equally secure as q, and $r \succ_S q$ means that r is more secure than q. Consequently, security measurement function shows that *a measurement function must be monotone.*

In [Kramosil and Michalek, 1974], The definition of metrics in the measurement theory was defining like:
Definition 2: *Metric* is a function M on a set Q which determines the distance between two members of the set $(M : Q \times Q \mapsto \mathbb{R})$ and satisfies the properties:

1. $M(q_1, q_2) \geq 0 \quad \forall q_1, q_2 \in Q \; (positivity)$
2. $M(q_1, q_2) = 0 \quad iff \quad q_1 = q_2 \quad \forall q_1, q_2 \in Q \; (identity)$
3. $M(q_1, q_2) = M(q_2, q_1) \quad \forall q_2, q_1 \in Q \; (symmetry)$
4. $M(q_1, q_3) \leq M(q_1, q_2) + M(q_2, q_3) \quad \forall q_1, q_2, q_3 \in Q \; (triangle \; inequality)$

In [Barzilai, 2005, Coatanea et al., 2007], conclude that: the distance the mapping from empirical quantities to real numbers is established.

It defines a formal approach to normalization of the numerical metric like:
Definition 3: Let Q be a set of elements and r and q be its members $(r, q \in Q)$. Let also $\mathbf{R} = \{R_1, ..., R_n\}$ be a set of relations on Q. The tuple $\langle Q, \mathbf{R} \rangle$ is called an empirical relational system. Measurement can be seen as an objective empirical function that assigns a real value to an element $(M : Q \mapsto \mathbb{R})$ and $\mathbf{P} = \{P_1, ..., P_n\}$ is a set of relations on $\mathbb{R}$ (reals) that is in a binary relation with $\mathbf{R}$ (i.e., each R_i corresponds to P_i).
Then:

$$\forall i \; R_i(r, q, ..) \Longleftrightarrow P_i(M(r), M(q), ..)$$

With respect to the start and end boundaries of each interval, it has:

$$0 < \mathrm{a}_1 \leq \mathrm{b}_1 \leq \mathrm{c}_1 < +\infty$$

In the range [0-4] representing the relation monotone, there exists five relations for the values of the intervals:

$$R_5(\ \{\forall\ x \mid\ c_1 < x < +\infty\ \}\ ) \Longleftrightarrow P_5(M(x) = 4)$$
$$R_4(\ \{\forall\ x \mid\ b_1 < x \leq c_1\ \}\ ) \Longleftrightarrow P_4(M(x) = 3)$$
$$R_3(\ \{\forall\ x \mid\ a_1 < x \leq b_1\ \}\ ) \Longleftrightarrow P_3(M(x) = 2)$$
$$R_2(\ \{\forall\ x \mid\ 0 < x \leq a_1\ \}\ ) \Longleftrightarrow P_2(M(x) = 1)$$
$$R_1(\ \{\forall\ x \mid -\infty < x \leq 0\ \}\ ) \Longleftrightarrow P_1(M(x) = 0)$$

And another formal description:

$$< \{c_1 < x < +\infty\ \},\ 4\ >$$
$$< \{b_1 < x \leq c_1\ \},\ 3\ >$$
$$< \{a_1 < x \leq b_1\ \},\ 2\ >$$
$$< \{\ 0 < x \leq a_1\ \},\ 1\ >$$
$$< \{-\infty < x \leq 0\ \},\ 0\ >$$

For each primitive security metric (numerical type) in the hierarchy of security metrics, that converges to values in the range [0-4] representing the relation monotone, using $r \succ_S q$ (means that r is more secure than q):

$$(4 \succ_S 3),\ (4 \succ_S 2),\ (4 \succ_S 1),\ (4 \succ_S 0)$$
$$(3 \succ_S 2),\ (3 \succ_S),\ (3 \succ_S 0)$$
$$(2 \succ_S 1),\ (2 \succ_S 0)$$
$$(1 \succ_S 0)$$

5.1.4 Validation of Security Metrics

To validate the collected security metrics, the monitoring scheme performs two steps:

1. The values measured by the security metrics in the range [0–4] are classified as true-positive (TP), false-positive (FP) true-negative (TN) and false-negative (FN);

2. Validation indicators are calculated for the model:

   - Precision, $\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$, indicates the percentage of events correctly classified as incident among those which were classified as such;
   - Recall, $\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$, indicates the percentage of events properly classified as incidents among all the real events;
   - F-measure, $\mathrm{F} = \frac{2\times\mathrm{P}\times\mathrm{R}}{\mathrm{P+R}}$, is the harmonic mean between precision and recall;
   - Accuracy, $\mathrm{A} = \frac{\mathrm{TP+TN}}{\mathrm{TP+FP+TN+FN}}$, indicates the percentage of correctly classified events.

By analyzing the values of the validation indicators, one can determine the degree of reliability for the collected security metric values.

As an example, monitoring DoS attacks on Web servers (Apache [Apache, 1995] [Coar and Bowen, 2008], Nginx[Nginx, 2003, Sharma, 2015] and Lighttpd[Lighttpd, 2005, Bogus, 2008]) was based on a machine learning mechanism, one trained specifically for the signatures generated by the machine under monitoring to detect anomalies. Two

data sets were generated for each server: the first one, used for model training, was comprised of data gathered over a period of 20 minutes; the second one, containing the data to be evaluated, was collected over a period of 24 hours. In both sets, valid random attacks and traffic with varying rates and durations were generated. Using an environment analysis and mining composite data through the KNIME [KNIME, 2010] and Weka [University of Waikato, 2013] tools, different classification algorithms were tested: k-Nearest Neighbour (KNN), Support Vector Machines (SVM) and Multilayer Perceptron.

The validation process will be illustrated based on of a case study [Ferreira, 2013]. The values in Table 5.5 show the distribution of true-positive values (TP), false-positives (FP), true-negatives (TN) and false-negatives (FN) for detecting attacks using all parameters (column "All") and using only the parameters selected by the genetic algorithm (column "Filtered"). It can be seen that the use of this selector contributed to a significant reduction in the number of false positives.

Table 5.5: Result of the detection of attacks

	TP		**FP**		**TN**		**FN**	
	All	Filtered	All	Filtered	All	Filtered	All	Filtered
Apache	1416	1413	1042	11	14826	15857	0	3
Nginx	1374	1377	282	6	15621	15897	9	6
Lighttpd	938	939	44	48	16301	16297	2	1

It is also observed that the SVM algorithm, using a linear kernel, showed the best results in the event classification process. The evaluation was performed using a parameter selection mechanism, based on a genetic algorithm. With this mechanism it was possible to eliminate unnecessary parameters for building the attack signature and also to reduce the cost of processing the classification model.

Finally, Table 5.6 presents how metrics were evaluated: precision, recall, f-measure and accuracy. One notices that the use of parameter selection (columns labeled "Filtered") sharply increased the classification precision in the Apache and Nginx data sets to almost perfect levels, while the LightTPD data set results remained steady. Similarly, the accuracy of the classification model also approached perfect levels for Apache and Nginx, and also remained steady, close to perfect levels, for LightTPD [Ferreira, 2013].

Table 5.6: Results of the metrics evaluation in percentage

	Precision		**Recall**		**F-Measure**		**Accuracy**	
	All	Filtered	All	Filtered	All	Filtered	All	Filtered
Apache	57.61	99.23	100.00	99.79	73.10	99.51	93.97	99.92
Nginx	83.16	99.59	99.38	99.54	90.50	99.57	98.32	99.93
Lighttpd	95.52	95.12	99.74	99.89	97.59	97.45	99.73	99.72

5.1.5 Security Metrics Behavior

After collection and analysis of packet filtering of security of metric of a firewall (infrastructure), through the in-side type monitoring agent, the prediction of their behavior

through the classic model of time series analysis will be presented the trend of the series of this security metric.

Figure 5.3 presents the packet filtering security of metric was collected and in the period: 01/02/2014 to 06/30/2015, in schedule of eleven consecutive hours.

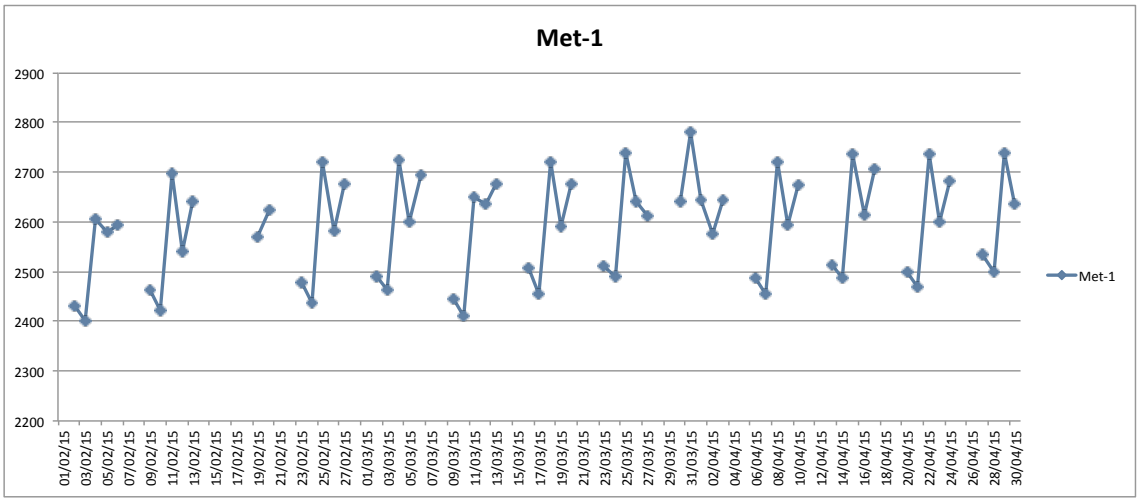


Figure 5.3: Packet filtering collected.

Figure 5.4 presents the metric applying filters in the time series, excluding saturdays, sundays and holidays.

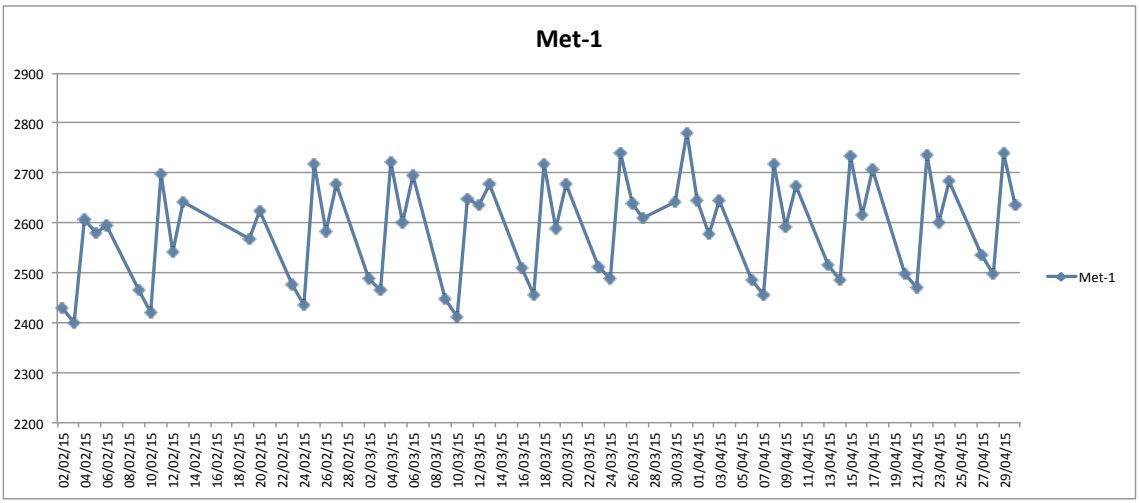


Figure 5.4: Packet filtering filtered.

Figure 5.5 presents the normalized metric to the scale of values [0-4]. The function of normalization can be summarized for f(x), where x account the number of filtered packets, and assumes "2" for values greater/equal to 2500, and assumes "3" to values lower than 2500. The parameters of the normalization function, are set in the mechanism for collecting and normalizing each metric, and then specified as a SLO rule in Security-SLA.

This security metric was measured while running the PostgreSQL [PostgreSQL, 1996] database (service hired), and analyzing the behavior of the application that ran in this database, and growth in the number of filtered pacts, seasonal fluctuations.

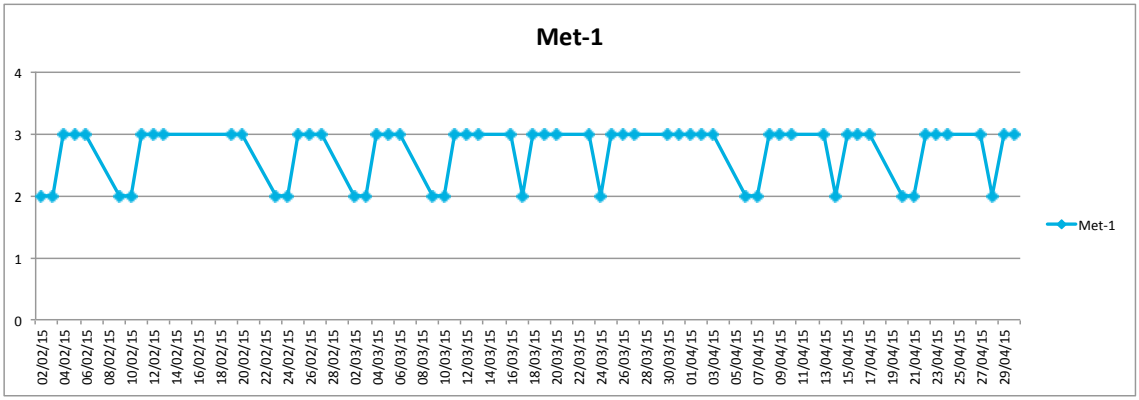


Figure 5.5: Packet filtering normalized.

It uses the method of least squares for the stretch of the coefficients that best fits the data. The difference here is that the independent variable is always the time (measured directly, e.g., years 2014, 2015, or by counting periods of 1, 2, 3).

For this linear case, the straight trend will be:

$$T = a + b \cdot t$$

Where T is the trend value, the value of t is time, b is the slope of the line (if positive indicates increasing trend, if the negative trend is downward) and is the linear coefficient of the straight line. The coefficients of the equations are expressed as follows.

$$a = \frac{\sum_{i=1}^{n} y_i \; - \; b \cdot \sum_{i=1}^{n} t_i}{n} \qquad b = \frac{n \cdot \sum_{i=1}^{n} (t_i \cdot y_i) \; - \; \sum_{i=1}^{n} t_i \cdot \sum_{i=1}^{n} y_i}{n \cdot \sum_{i=1}^{n} (t_i^2) \; - \; (\sum_{i=1}^{n} t_i)^2}$$

Where y_i is any of the registered variable value in the time series, t_i is the period associated with y_i, n is the number of periods of the series. To find the coefficients simply calculate the sums (as in simple linear regression analysis).

Figure 5.6 presents the prediction for the normalized metric to the scale of values [0-4] using the reference one week.

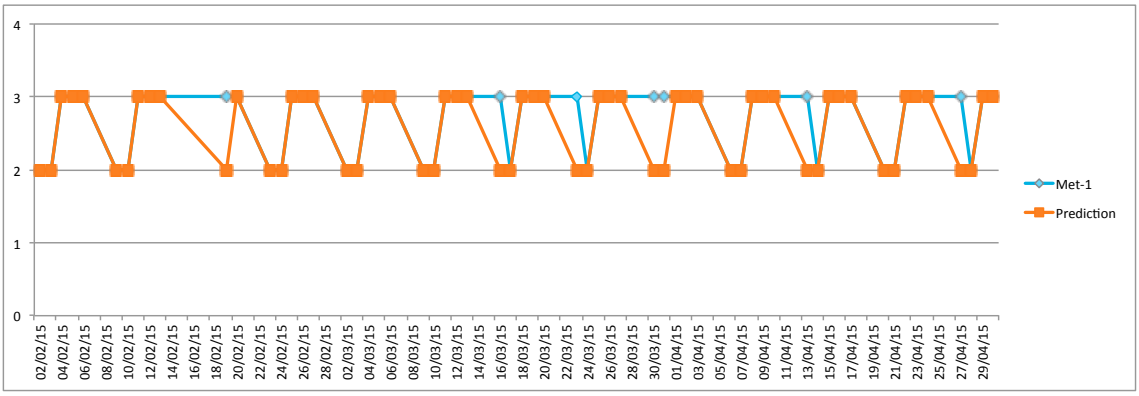


Figure 5.6: Packet filtering prediction.

In the context of this work, it seeks to identify the factors that produce the change of a value in the range of [0-4] for each security metric measured the Cloud Computing environment, summarized as follows:

- **By customer**: when induces or produces a change in the service contract, examples: lightning propaganda, changes in service configuration, and etc.;
- **By provider**: when induces or produces a change in the service contract, examples: changes in the service configuration, introduces a new device on the network, and etc.;
- **By customer and provider**: when induces or produces any change in service contract, examples: the client makes lightning advertising, and provider for meet the new demand on the network, produces changes in the service configuration and the network, and etc.;
- **The system management**: because the demands for high or low: system (provider) produces physical change to meet the service (elasticity), produce the change of the service to another server, events derived from the provider resource shares (network, CPU, memory): the system can change the service to another avoiding server violations of Security-SLA and SLA.

Figure 5.7 presents the case of study to the monitoring of factors that produce the change of a value in the range of [0-4] on the database PostgreSQL [PostgreSQL, 1996], provided service as SaaS, using the reference one trimester with the total of 79 events, summarized as follows:

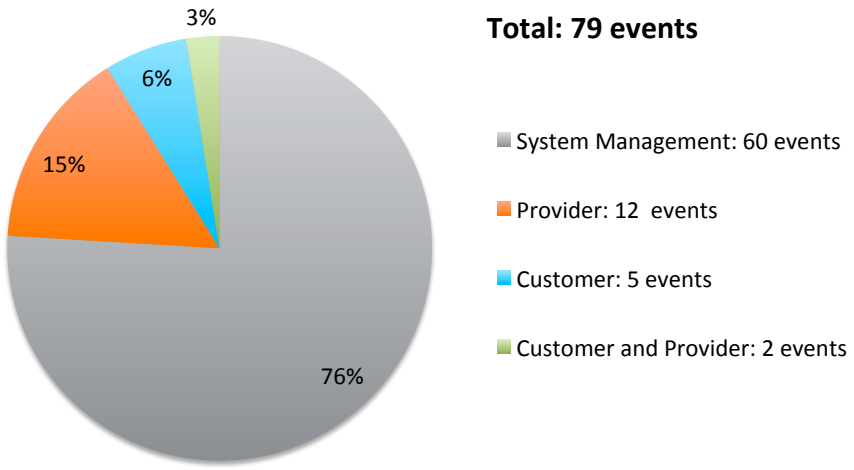


Figure 5.7: Change of a value in the range of [0-4].

These factors can be used to predict the future behavior of security metrics in a cloud environment.

5.2 Application of SMH

In the scenario of on-demand allocation and scalability of the cloud environment, the Security Metrics Hierarchy (SMH) Methodology proposal introduced a flexible, extensible

concept, easy to understand and use. Based on security policies, this SMH is composed of two components: the infrastructure and the services.

The infrastructure includes quantitative and qualitative requirements of hardware and software to manage the cloud infrastructure through monitoring security metrics, regardless of manufacturer, technology, service models etc. As required, this methodology copes well with an increasing or decreasing number of active devices being monitored. For example, in the Infrastructure metric (item Met_1), one may add a group of metrics called *Group_Dell_PowerConnect_Switches*, composed of Dell PowerConnect manageable switches (group metric $\mathrm{Met}_{1.18}$), with 200 units and enumerated as follows:

$$\{\mathrm{Unit\_Dell\_PCon\_001}, \mathrm{Unit\_Dell\_PCon\_002}, \ldots, \mathrm{Unit\_Dell\_PCon\_200}\}$$
$$(\text{metrics: } \{\mathrm{Met}_{1.18.1}, \mathrm{Met}_{1.18.2}, \ldots, \mathrm{Met}_{1.18.200}\})$$

As such, the methodology allows the classification, recording and monitoring of the performance of each switch's individual security requirements planned for it.

In the Services context, regardless of the type of service that was hired (IaaS, PaaS, SaaS or others), services will be measured by security metrics that provide an overall view of security. For example, in the Database service metric (item Met_2), one might add a group of metrics called *Relational_Databases*, composed of relational databases (group metric $\mathrm{Met}_{2.25}$), with five types of relational databases (Oracle, MySQL, PostgreSQL, Sybase and DB2) and enumerated as follows:

$$\{\mathrm{Unit\_Rel\_Database\_1}, \mathrm{Unit\_Rel\_Database\_2}, ..., \mathrm{Unit\_Rel\_Database\_5}\}$$
$$(\text{metrics: } \{\mathrm{Met}_{1.25.1}, \mathrm{Met}_{1.25.2}, ..., \mathrm{Met}_{1.25.5}\})$$

Therefore, the methodology allows the classification, recording and monitoring of the performance of each relational database's individual security requirements planned for it.

5.3 Summary

This chapter presented the Security Metrics Hierarchy (SMH) methodology for the management of security for cloud computing environments using security metrics. It described the normalization process of the security metrics, the formal structure of the Security Metrics Hierarchy, the validation process of the security metrics and presented the security metrics behavior through the classic model of time series analysis. Finally, it showed an example of the SMH's application when dealing with multiple devices in the cloud infrastructure.

Chapter 6

Management of Cloud using security criteria

This chapter uses the methods and techniques presented for the security management of Cloud Computing environments using security metrics [Silva and Geus, 2015, Silva et al., 2012, Silva and Geus, 2014a, Silva and Geus, 2014b, Silva and Geus, 2014c]. The portfolio of security metrics structured in a hierarchy represents the security requirements of the Cloud Computing environment, allowing for measuring the quality of the provided/hired service.

6.1 Return On Security Investment (ROSI)

Potential customers of Cloud Computing perceive a lack of transparency and a relative lack of control when compared to the traditional computing models[Foster et al., 2008, Pearson, 2013]. In the industry these services are referred to as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), respectively.

Such customers (or companies) decide to migrate part of their data, services or infrastructure to a Cloud Computing service provider (CSP) based on the following parameters: expected benefits, adoption costs, performance, flexibility, business opportunities and others [Kantarcioglu et al., 2011, Johnson and Qu, 2012, Martens and Teuteberg, 2012]. In the current literature, it is unclear whether the CSP should provide security services or not, and what these services characteristics and varying levels of protection and costs involved should be.

The decision to migrate refers to a particular document: “Deployment Profile”, which is defined in the context of this article to include four elements that will be evaluated: mobilized assets, types of Cloud service, deployment models and a specific CSP. Due to the varying levels of customer controls for each profile, different security services are offered by the CSP. Increased protection on the side of the CSP should raise rates deployment and maintenance costs, while less protection means more control and client-side costs.

From a customer’s perspective, this proposal presents a new qualitative and quantitative approach to the security requirements in the process of migrating to Cloud Computing, and proposes the use of security metrics to analyze the benefits and costs of security

for a given deployment profile. The ultimate goal is to properly assess whether the decision to migrate assets, e.g. data, services, applications, infrastructure etc. to Cloud Computing is beneficial or not, both economically and security-wise. This means that the customer has to evaluate both the level of security provided by the CSP and the costs that such controls present. It assumes that the CSP is cooperating and willing to reveal its offered security services through a portfolio of security metrics.

6.1.1 Return on Investment (ROI)

Return on Investment (ROI) is one of several financial indicators available to estimate the financial result of the company's investments (in this proposal: a possible client who hires a service from a CSP). This calculation takes into account the cost of an investment and its expected earnings, and provides an estimate of how favorable the investment will be. To calculate the ROI (simple ROI), the cost of an investment should be subtracted from the gain (return) of the investment, and the result divided by the cost of the investment; the result is expressed as a percentage or fee. In most cases, a rate greater than 0 (zero) means that the return is greater than the cost, then the investment can be considered beneficial (how beneficial depends on the objectives of the investment or corporate standards of the company) [ENISA, 2012]:

$$\mathrm{ROI} = \frac{(\mathrm{Gain\_From\_Investment} - \mathrm{Cost\_of\_Investment})}{\mathrm{Cost\_of\_Investment}}$$

Where:

- Gain From Investment: the final value of the benefits;
- Cost of Investment: the initial value of the investment

Such values can be estimated or calculated.

6.1.2 Methodology for ROSI Calculation

There are many possible ways to estimate ROSI in a Cloud Computing environment, and no approach is suitable for all situations due to the measurable and non-measurable qualities. Selecting the best option for a particular case depends on many factors, including what the business drivers for migrating to the Cloud are (revenue growth versus cost savings), the approach to prepare and evaluate the business cases (emphasis on tangible versus intangible QoS), and where the company is in the growth cycle/maturation of business (new business versus mature company).

This approach is outlined in three steps for calculating ROSI; also, the described concepts can also be applied to other scenarios, requiring more or less steps depending on circumstances.

Step 1: Determine costs and benefits of the Cloud

At this stage will be set costs and benefits of the Cloud with the following substeps: i) define high level business requirements (functional); ii) define the service model in the

initial/basic Cloud; iii) make a risk assessment of the initial/basic Cloud model; iv) estimate the cost of migration from the current to the Cloud-based model, and estimate the tangible and intangible benefits; v) consider other Cloud models (private, public, community and hybrid); vi) re-evaluate cost/benefit for the ideal alignment model.

Step 2: Assess costs and current benefits

At this stage will be set the actual costs and benefits of the Cloud, with the following substeps: i) make a current estimate for the business requirements used in step 1, set the current service model to meet the same functional requirements and compliance; ii) make a risk assessment (or review it if one already exists) of the current service model; iii) make an estimate of costs/benefits, and include operating costs/current maintenance (TCO), cost of mitigating risks, and costs/intangible benefits.

Step 3: Make an estimate for ROSI

At this step will be calculated an estimate for ROSI, with the following substeps: i) compare the costs and benefits of the current and alternative option; ii) calculate ROSI and other financial indicators of investment returns; iii) calculate the intangibles.

6.1.3 Calculating ROSI

This section presents the definition of new economic indicators created by this approach to calculate the return on investment in security for the security requirements in a Cloud Computing environment.

ROSI_{a}

The financial indicator ROSI_{a} is the arithmetic difference between the value measured by the CSP for the security requirement "Deployment Profile" and the value expected by the customer, as follows:

$$\text{ROSI}_{\text{a}} = -(\text{Evaluated\_Metric} \; - \; \text{Expected\_Value})$$

Where:

- Evaluated_Metric: the value measured by the CSP for the security metric requirement, between 0 and 4;
- Expected_Value: the value expected by the client for the security requirement, between 0 and 4.

ROSI_{vi}

The financial indicator $ROSI_{vi}$ is obtained by subtracting the expected value from the measured value for the given metric, as informed by the client, relative to the measured value, as follows:

$$\text{ROSI}_{\text{vi}} = \frac{(\text{Evaluated\_Metric} - \text{Expected\_Value})}{\text{Evaluated\_Metric}}$$

Where:

- Evaluated_Metric: the value measured by the CSP for the security metric requirement, between 0 and 4;
- Expected_Value: the value expected by the client for the security requirement, between 0 and 4.

ROSI_{vf}

The financial indicator ROSI_{vf} is obtained by subtracting the expected value from the measured value for the given metric, as informed by the client, relative to the expected value, as follows:

$$\text{ROSI}_{\text{vf}} = \frac{(\text{Evaluated\_Metric} - \text{Expected\_Value})}{\text{Expected\_Value}}$$

Where:

- Evaluated_Metric: the value measured by the CSP for the security metric requirement, between 0 and 4;
- Expected_Value: the value expected by the client for the security requirement, between 0 and 4.

The results for the previous indicators may fall into the following ranges:

- Positive: the CSP is ensuring greater security than the customer expects (beneficial to the client). Example: ROSI_{a} being 2.0 indicates that the Gain from Investment is 200 % above the cost of investment;
- Zero: the CSP has exactly the level of security that the customer wishes/requests;
- Negative: the CSP is presenting less security than the customer expects (prejudicial to the client). Example: ROSI_{a} being -1.0 indicates that the Gain from Investment is 100 % below the cost of investment.

6.1.4 Deployment Profile

The migration decision has to be implemented on the customer side. Essentially, the customer has to answer to the following question: “Are the security controls offered adequate and efficient from a security perspective?”. The answer to this question must affect the decision to migrate to the Cloud Computing [Martens and Teuteberg, 2012].

The logical process of the decision follows four steps:

1. Define a deployment profile. The customer selects the migrated assets, the Cloud type and the deployment model, coupled with a CSP offering such a service.

2. Define a set of controls for the deployment profile. These may be offered by the CSP or implemented by the tenant (due to the Cloud migration).

3. Calculate each security metric to ROSI (beneficial or prejudicial to the client) for the implemented security controls.

4. Evaluate Return on Security Investment (ROSI) for each security metrics combination of the deployment profile.

The last step results in the evaluation of the ROSI of one or more profiles and the customer deciding whether he will migrate or selecting the CSP, the model, and the type of Cloud that is more beneficial.

Figure 6.1 shows the process of creating the "Deployment Profile", where the customer chooses, from the portfolios of Infrastructure and Service security metrics, which security metrics to use and their expected values. Each expected value falls in the range [0, 1, 2, 3, 4], respectively corresponding to the expected security levels [Critical, High, Medium, Low, None].

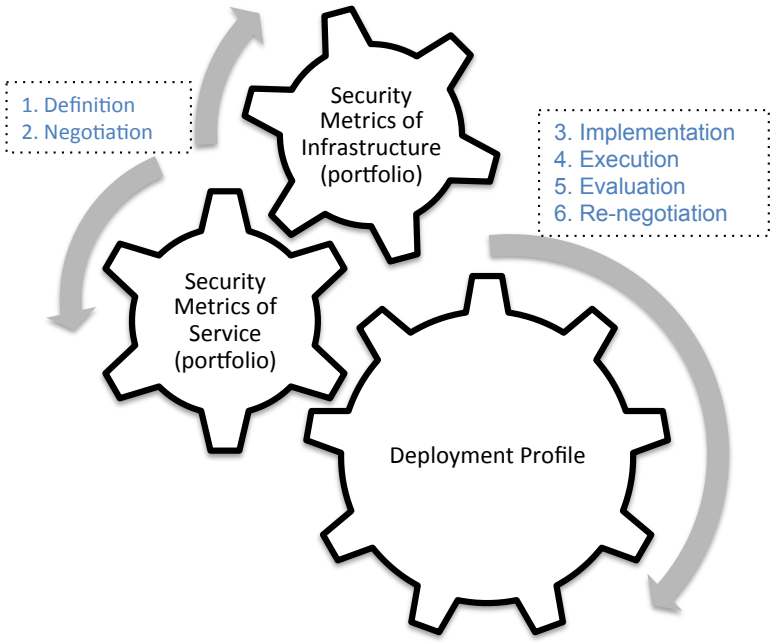


Figure 6.1: Creating the Deployment Profile.

6.1.5 Case Scenario

Let's consider a case scenario of customer John who considers whether to migrate assets (e.g. data, services, etc.) to a Cloud deployment provided by CSP_X. CSP_X provides a private Cloud deployment for Carlos, while the available offered service is Software-as-a-Service (SaaS).

The customer chose in the range [0, 1, 2, 3, 4], respectively corresponding to the expected security levels to some security metrics (portfolio) to security requirements, and

calculate the $ROSI_{a,vi,vf}$. The result of $ROSI_{a,vi,vf}$ should guide decision-making about hiring or not the service SaaS offered by the CSP_X.

Table 6.1 shows the firewall security metrics from the infrastructure portfolio chosen by the customer, where: i) Identification (Id) singles out a metric from the portfolio; ii) Evaluated Metric (EM) is the value that the CSP_X is committed to meet via contract (measured by the CSP_X through a time series analysis); iii) Expected Value (EV) is the value expected by the client for the security requirement.

Table 6.1: Deployment Profile (Portfolio of Infrastructure metrics)

Id	**Description**	**EM**	**EV**	$ROSI_a$	$ROSI_{vi}$	$ROSI_{vf}$
1.	**Firewall**					
1.1	Average time vulnerabilities are patched	4	3	1	0.20	0.25
1.2	Security event records	3	2	1	0.25	0.33
1.3	Application Software Security Threat Level	2	3	-1	-0.33	-0.25
1.4	Backup mechanism	3	2	1	0.25	0.33
1.5	Mean time between failures	4	3	1	0.20	0.25
1.6	Mean time between maintenance	3	4	-1	-0.25	-0.20

Table 6.2 shows the PostgreSQL database security metrics from the service portfolio chosen by the customer.

Table 6.2: Deployment Profile (Portfolio of metrics for a Service)

Id	**Description**	**EM**	**EV**	$ROSI_a$	$ROSI_{vi}$	$ROSI_{vf}$
3.	**PostgreSQL Database**					
3.1	Default TCP port	2	2	0	0	0
3.2	Default user service account	3	2	1	0.25	0.33
3.3	Insecure user account	2	1	1	0.33	0.50
3.4	Mean time to verify latest security patches	3	2	1	0.25	0.33
3.5	SQL injection	4	3	1	0.20	0.25

Figures 6.2 and 6.3 illustrate the behavior of the security metrics that make up the deployment profile values: i) Evaluated Metric (EM) is the value that the CSP_X is committed to comply via contract (blue lines); ii) Expected Value (EV) is the value expected by the customer from the CSP_X (black lines); iii) Average EM (Aveg-EM) is the arithmetic mean value of the metrics measured by the CSP_X (green lines). Figure 6.2 illustrates the firewall security metrics from the infrastructure portfolio and Figure 6.3 the PostgreSQL database security metrics from the service portfolio.

The red dots (expected values) that are above or equal to the blue dots (evaluated metrics) are the requirements that are met by the CSP_X (beneficial to the client). Otherwise, the expected values that are below the evaluated metrics are the CSP_X requirements that do not meet the customer's security level (prejudicial to the client).

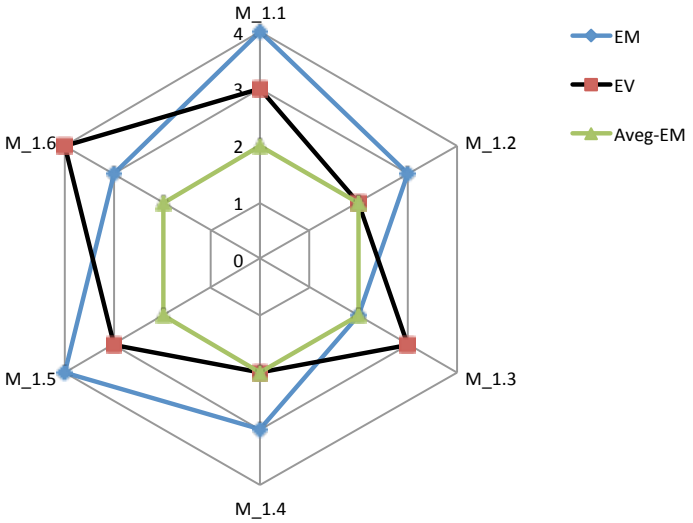


Figure 6.2: Firewall Metrics.

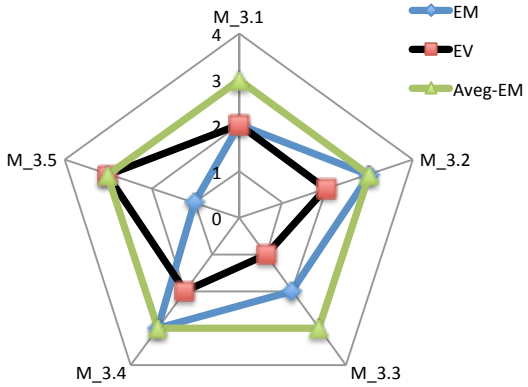


Figure 6.3: PostgreSQL Metrics.

Analyze the results of ROSI

The client should consider the behavior of each chosen security metric, in order to understand and evaluate all aspects involving the Cloud migration decision. For instance, the client may reject some of the included security controls or replace them with equivalent ones, based on their benefit to the security cost or the result for $ROSI_{a,vi,vf}$.

Based on data from Table 6.1, the security metrics $M_{1.3}$ and $M_{1.6}$ indicate negative values for the return on security investment, i.e. these metrics are presenting security levels below what the customer expects (prejudicial to the client).

When analyzing the percentage of security requirements that have negative values for $ROSI_{a,vi,vf}$, it verifies that 18% of them are prejudicial to the client (2 out of 11).

For security requirements that differ from the contract, the customer can choose one of the following:

- Accept the migration to the CSP_X based on the described Deployment Profile and

the results for $ROSI_{a,vi,vf}$;

- Reject the migration to the Cloud due to the results for $ROSI_{a,vi,vf}$ that are not satisfactory;
- Choose another Deployment Profile with new parameters: assets, models, types and controls;
- Choose another CSP_X that satisfies the given parameters.

Thus, the customer can identify which what security requirements are guaranteed by the CSP_X (beneficial to the client), and what not (prejudicial to the client).

Table 6.3 shows the necessary investment for firewall security metrics from the infrastructure portfolio chosen by the customer. The currency used in the example is irrelevant, so it considers the values as plain numbers (e.g. 30), where: Cost is the annual value for: install, configure, training, etc. and Investment is the necessary annual value to improve the security requirement provided by the CSP_X.

Table 6.3: Necessary Investment (Portfolio of Infrastructure metrics)

Id	**Description**	$ROSI_a$	Cost	Investment
1.	**Firewall**			
1.3	Application Software Security Threat Level	-1	8,000	16,000
1.6	Mean time between maintenance	-1	5,000	10,000

The CSP_X can use the $ROSI_a$ values to calculate the investment must do to meet the security requirements detrimental to the client, for example, metrics "1.3" and "1.6" need 100% investment to meet customer needs, i.e., double the amount invested in the process security controls.

6.1.6 Results

This work proposed a new quantitative and qualitative methodology for obtaining the Return On Security Investment for a specific deployment profile through the use of security metrics. The proposal covers the services offered by the Cloud Computing providers from a client security perspective. Furthermore, this approach has the advantage of supporting a hierarchical decomposition and also presents a solution to deal with intangible costs and benefits, thereby allowing for distributed and scalability features. In Appendix A (page 132), this portfolio of security metrics is described in more details.

6.2 Managing Security-SLA

The methodology builds on the concept that the customer, when hiring a service in the Cloud (SaaS, PaaS or IaaS), may choose from a portfolio of security metrics that will be continuously monitored by the environment.

Within this approach, a database holds two classes of security metrics, according to their functionality. The infrastructure device class consists of all the hardware devices and

related Cloud software, such as networking, firewalls, routers, proxies, operating systems, etc. on which monitoring agents will run to generate security metrics. The service class consists of all SaaS, PaaS or IaaS hardware/software components that provide the service contracted by the customer, with the monitoring agents running on those assets generating security metrics about the Virtual Machine (VM).

6.2.1 Automatic Security-SLA

Figure 6.4 represents the proposed lifecycle of Security-SLA management for Cloud Computing environments, which is based on the following phases:

(1) Definition: this phase is focused on the selection of the infrastructure and service security metrics, its features and the definition of quality parameters that will be provided to customers. A database with all the security metrics (portfolio) is offered;

(2) Negotiation: in this phase are defined values for the security metrics parameters (range 0–4), cost to the customer and penalties in case the Security-SLA is violated;

(3) Implementation: the security metrics are prepared according to the available infrastructure devices that will allow for the service execution in the environment;

(4) Execution: it is the phase when monitoring security metrics for the infrastructure devices and service takes place. Specified quality parameters (SLO) are evaluated for compliance with the Security-SLA;

(5) Evaluation: in this phase the provider assesses the security quality provided;

(6) Re-negotiation: deals with the service ending, be it for reasons of contract expiration or for Security-SLA re-negotiation.

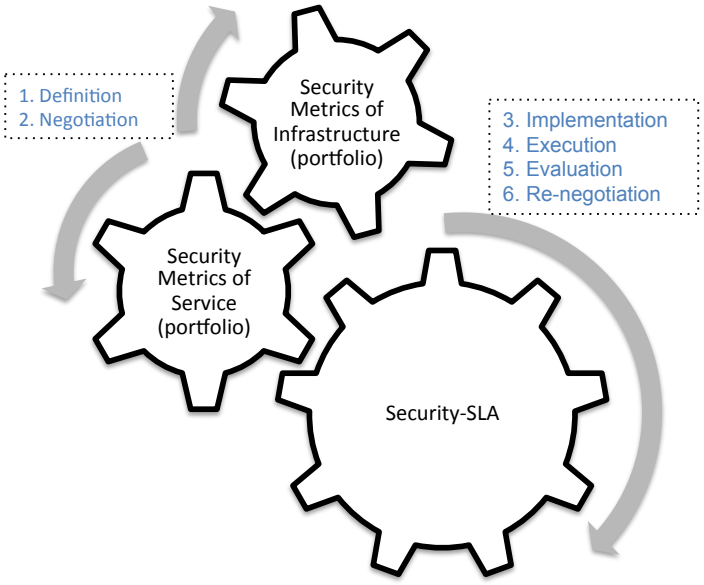


Figure 6.4: Lifecycle of a Security-SLA.

Figure 6.5 presents a simplified overview of this Security-SLA management process for Cloud Computing environments. As today's business systems typically consist of layers of complex systems, user-level Security-SLAs cannot be directly mapped to physical infrastructure. Services can be composed of other more fundamental services, maybe even provided by third parties. Consequently, a gradual mapping of higher-level Security-SLA requirements onto lower levels, and aggregation of lower-level resources to higher-level ones is crucial to allow binding of user-level Security-SLAs to the infrastructure. This vertical flow of information must carefully reflect service interdependencies. In addition to Security-SLAs, vertical information flow also covers monitoring, tracking and accounting data, having to support intermediation and negotiation processes at each layer. The Security-SLA management process may deal with different stakeholders, namely customers, service and infrastructure providers, and also various business steps such as business valuation, contracting and sales. The illustration also shows the role of software providers responsible for creating components with predictable behavior. In this context, one notices the integration of multiple levels, as there are several interested parties (suppliers of software/services/infrastructure and customers), various roles (IT people, experts, customers), various types of services, various aspects of service level (availability, performance etc.), all under the full lifecycle of the Security-SLA (definition, negotiation, implementation, execution, evaluation and re-negotiation).

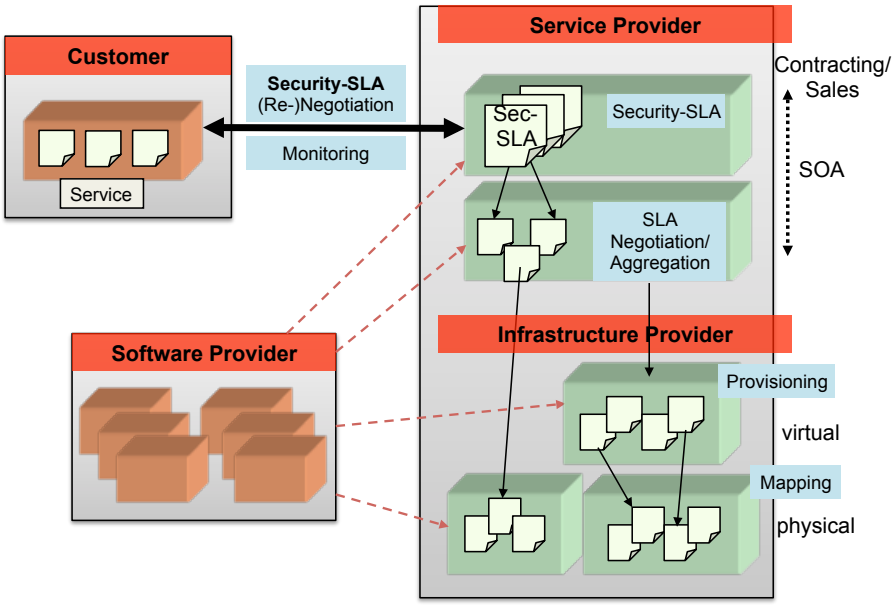


Figure 6.5: Security-SLA management in Cloud Computing.

6.2.2 Monitoring Security-SLA

Monitoring agents are specialized programs responsible for the monitoring process. Each program is tasked with collecting information from existing components in the infrastructure. Such information will be used in the Security-SLA validation.

When an agent runs, it gets the Security-SLA parameters: service that will be monitored, execution parameters and some metric identifiers that will be measured. "Place"

specifies the elements where the measurement is done, like VM, Firewall, IDS/IPS etc. "Procedure" specifies whether the type is black-box or not. "Frequency" of measurement in hours. SLO specifies the contract value in the 0–4 range. Finally, "Incidence" specifies the percentage of samples that stayed above the SLO value.

6.2.3 Case Study

A case study was developed and tested on a Cloud Computing environment based on OpenNebula [OpenNebula, 2014], on a machine with a 2.8 GHz intel i7 quad-core processor and 32 GB of RAM running Gentoo Linux and the KVM hypervisor. The customer chooses mysql enterprise edition as a SaaS service and as an infrastructure service Intrusion Detection and Prevention System (IDPS). During the negotiation, the client specified that monitoring would be performed 10 times (events in between two samples being accumulated to the next sample). The system under test is responsible for human resources management at an University and holds about 400 tables, 200 users and 5 administrators. Table 6.4 describes the security metrics that compose the Security-SLA.

Table 6.4: Security Metrics chosen by the user

Item	Description (metric)	Value of Metric
2	**Infrastructure Cloud Computing**	$\text{Met}_{2} \geq 3$
2.4	**Intrusion Detect and Prevention System**	$\text{Met}_{2.4} \geq 3$
2.4.1	Packet Fragmentation	$\text{Met}_{2.4.1} \geq 4$
2.4.2	Stream Segmentation	$\text{Met}_{2.4.2} \geq 3$
2.4.3	Remote Procedure Call Fragmentation	$\text{Met}_{2.4.3} \geq 3$
2.4.4	Recovery from Abnormal System Shutdown	$\text{Met}_{2.4.4} \geq 4$
2.4.5	Security Events Records	$\text{Met}_{2.4.5} \geq 4$
2.4.6	Evasion Attacks	$\text{Met}_{2.4.6} \geq 3$
9	**SaaS Cloud Computing**	$\text{Met}_{9} \geq 3$
9.1	**Database mysql**	$\text{Met}_{9.1} \geq 2$
9.1.1	Default User Service Account	$\text{Met}_{9.1.1} \geq 2$
9.1.2	Insecure User Account	$\text{Met}_{9.1.2} \geq 2$
9.1.3	Default TCP Port	$\text{Met}_{9.1.3} \geq 2$
9.1.4	SQL Injection	$\text{Met}_{9.1.4} \geq 2$

IDPS Metrics

As an example of the IDPS metric, the following parameters were chosen:

Metric Name: Stream Segmentation

Description: The Stream Segmentation Security Metric ($\text{Met}_{2.4.2}$) is monitoring unusual activity on the network, like the remote host advertising a zero window size, dropped TCP connections and session timeouts. By manipulating the way in which a TCP stream is segmented, it is possible to evade detection by some firewalls and IDPS.

When doing that, an attacker could overwrite a portion of a previous segment in a stream with new data in a subsequent segment. This method could allow the attacker to hide or obfuscate the attack on the network.

Formula: $\text{Met}_{2.4.2} = \text{Count(Incidents)}$

SLO Value: 3

Incidence: 90.00%

Table 6.5 describes the distribution of sample values (0–4 range) for the Stream Segmentation Metric ($\text{Met}_{2.4.2}$), for each monitored incident and their percentage of occurrence.

Table 6.5: Samples of Stream Segmentation Metric

$\text{Met}_{2.4.2}$	**Incidents**	**Percentage**
4	32	0.15%
3	1,505	6.96%
2	15,219	70.37%
1	4,422	20.45%
0	448	2.07%

Based on data from Table 6.5, Figure 6.6 presents the visual result of monitoring the Stream Segmentation Metric ($\text{Met}_{2.4.2}$) during the evaluated time span (1 to 10, i.e. the radii in the illustration). The hired SLO value was ≥ 3, the measured average MA value was 2 in the period, and Incidence total was 7.11% (percentage of incidents of levels 3 and 4: 0.15 + 6.96), however this is in sharp contrast with the contracted value of 90.00%. Therefore, one can conclude that not only there is a problem with the hired security level, but the relation between the measured 7.11% and the expected 90.00% values for the Incidence also suggests that the delivered security is very poor.

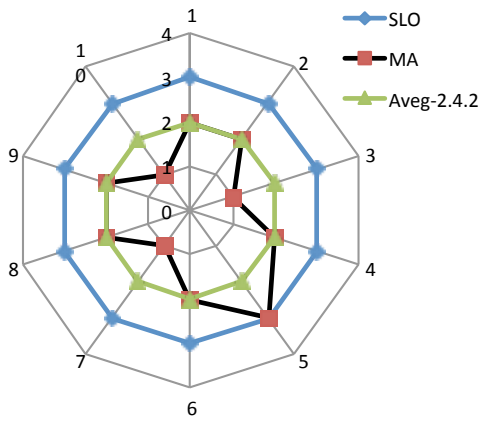


Figure 6.6: Behavior of the security metrics: stream segmentation.

MySQL Metrics

Heuristics. It implemented two heuristic algorithms to compute the incidents over the MySQL service:

a. **Log:** Uses black-box external monitoring, analyzes log files of the database, and identifies and records incidents. Let n be the number of records in the log file, m the number of operations and l the number of permissions. The asymptotic complexity is $O(n \log ml)$

b. **Interface:** Uses black-box internal monitoring, runs a PHP code inside the system interface and each command in the interface, verifies and records the incidents. Let n be the number of commands in the interface, m the number of operations and l the number of permissions. Its asymptotic complexity is O(nml)

As an example of MySQL metric, the following parameters were chosen:

Metric Name: Insecure User Account Security

Description: The Insecure User Account Security Metric ($Met_{9.1.2}$) is monitoring whether the customer used a default user account instead of an administrator account. The transaction log for the database is checked for the combination: source (Administrator or User), type of operation (select, update, drop, alter and create), and permission of the operation.

Formula: $Met_{9.1.2} = Count(Incidents)$

SLO Value: 2

Incidence: 80.00%

Table 6.6 describes the distribution of sample values (0–4 range) for the Insecure User Account Metric ($Met_{9.1.2}$), for each monitored incident and their percentage of occurrence.

Table 6.6: Samples of Insecure User Account Metric

$Met_{9.1.2}$	**Incidents**	**Percentage**
4	5,678	19.84%
3	18,104	63.25%
2	3,624	12.66%
1	1,000	3.49%
0	215	0.75%

Based on data from Table 6.6, Figure 6.7 presents the visual result of monitoring the Insecure User Account Metric ($Met_{9.1.2}$), during the evaluated time span (1 to 10, i.e. the radii in the illustration). The hired SLO value was ≥ 2, the measured average MA value was 3 in the period, and Incidence total was 95.75% (percentage of incidents of levels 2 to 4: 19.84 + 63.25 + 12.66), yielding a value well above the contracted 80.00%.

Therefore, one can conclude that security as contracted has not only been met, but in fact the relative position of the 95.75% figure towards the hardest target of 100% suggests that security was quite good.

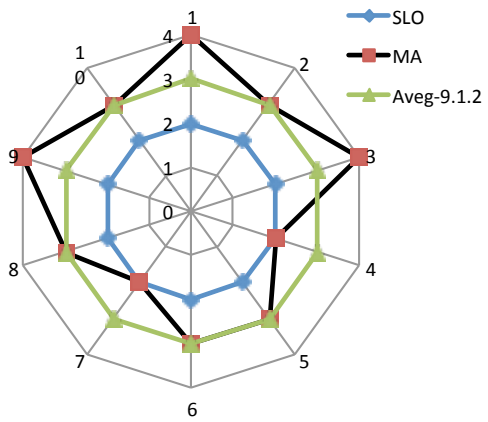


Figure 6.7: Behavior of the security metric "insecure user account".

6.2.4 Results

It presented a substantial contribution to make an automatic way of contracting a Security-SLA using as basis a portfolio of security metrics for the infrastructure and services classes. It also introduced a new model to view information about security through a range of values (0–4) and treated the problem of managing intangible and unmeasurable numbers. Moreover, it proposed a new way of managing security levels (top-down view) that considers values for each security metric with its respective risk, Quality of Service (QoS) and impact. Separating Security-SLA in two reference security value classes allows for an abstracted visualization of security and helps to easily spot which security items present values below the expected values. Thus, the customer may have a more tangible feeling of how the hired service is being protected. In Appendix A (page 132), this portfolio of security metrics is described in more details.

6.3 Obtaining Index of Security (IndSec)

The calculation of the Index of Security based on the values collected and organized in the Security Metrics Hierarchy. The Figure 6.8 describes Security Index composition.

In this model, the entire process monitoring/auditing of the security metrics is divided into two stages:

(a) Static Analysis: in this process, analysis parameters are as follows: (i) Security-SLA defined by the client and provider; (ii) the requirements of Quality of Service (QoS-SLA) derived from the SLA; (iii) and vulnerabilities recorded in the NVD database to the type of service, identifying the risk and impact through the CVSS [NIST, 2014];

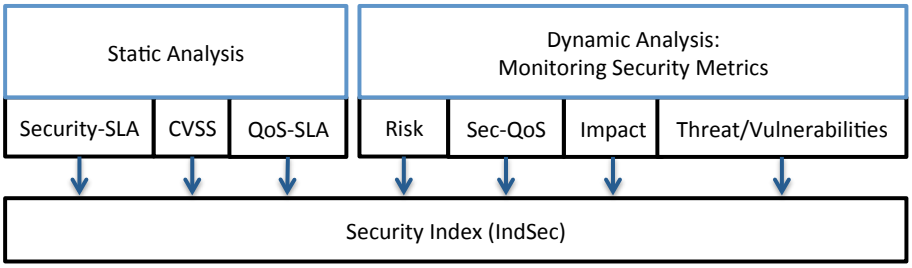


Figure 6.8: Calculating index of security.

(b) Dynamic Analysis: the security requirements defined in the statistical analysis step will be measured by monitoring/auditing of security metrics to the threats and vulnerabilities detected at runtime, and the measured values? to the final degree of impact, risk, quality security service (QoS-Sec-SLA) threats and vulnerabilities detected.

6.3.1 Normalization of Risk and Impact

Based on the National Vulnerability Database (NVD) [NIST, 2014], the portfolio of security metrics can be complemented with risk and impact values to the type of activity being measured (hired service). The NVD provides a score [0-10.0] through the Common Vulnerability Scoring System (CVSS) for the characteristics and impacts of the vulnerabilities of a system.

In this proposal, the scores of risk and impact are normalized is a range of values [0-4]. This process is equivalent to the normalization process of the normalization of security metrics to the value scaling [0-4] described in the Section 5.1.2 (page 62).

In Appendix B (page 304), this normalization scene is described in more details.

6.3.2 Function of Time

All values measured in SA and DA analytics converge to values in the range [0-4] representing the level of security monitored and audited, representing the range of security values as [Critical, High, Medium, Low, None]. The index of security (IndSec) is calculated by the average arithmetic of all N elements of analysis (EA_i) that compose the SA and DA analyzes (to f^*):

$$\mathrm{IndSec}^{\mathrm{T}} = \lfloor \frac{\sum_{i=1}^{N} \mathrm{EA_i} * \mathrm{T_i}}{\sum_{i=1}^{N} \mathrm{T_i}} \rfloor$$

where T_i is the amount of time a value (slice-time) in the range [0-4] assumes the total time measured.

6.3.3 Function of Weight, Impact, Risk

By analogy, the calculation of IndSec may be due to the weight (P_i), impact (I_i) and risk (R_i), substituting T_i in function:

$$\text{IndSec}^{W} = \lfloor \frac{\sum_{i=1}^{N} EA_i * W_i}{\sum_{i=1}^{N} W_i} \rfloor, \text{IndSec}^{I} = \lfloor \frac{\sum_{i=1}^{N} EA_i * I_i}{\sum_{i=1}^{N} I_i} \rfloor, \text{IndSec}^{R} = \lfloor \frac{\sum_{i=1}^{N} EA_i * R_i}{\sum_{i=1}^{N} R_i} \rfloor$$

The parameters of the weight (W_i), Impact (I_i) and risk (R_i) define the importance of submetric within the composition of the metric hence the level above (group level).

6.3.4 Case Study

A case study was developed and tested on a Cloud Computing environment based on OpenNebula [OpenNebula, 2014], on a machine with a 2.8 GHz intel i7 quad-core processor and 32 GB of RAM running Gentoo Linux and the KVM hypervisor. The customer chooses PostgreSQL object-relational database system as a SaaS service and as an infrastructure service firewall. During the negotiation, the client specified that monitoring would be performed 10 times (events in between two samples being acumulate to the next sample). The system under test is responsible for academic system at an University and holds about 100 tables, 9000 users and 10 administrators.

As an example of the firewall metric, the following parameters were chosen:

Metric Name: firewall

Description: firewall metric ($Met_{2.3}$) is composed of other submetrics measuring performance, maintenance, threats and vulnerabilities. This case study, $Met_{2.3}$ is calculated by average of $Met_{2.3.i}$ on time, so, dividing the value each submetric by time of incidence T_i.

Formula: $Met_{2.3} = \lfloor \frac{\sum_{i=1}^{N} Met_{2.3.i} * T_i}{\sum_{i=1}^{N} T_i} \rfloor$

SLO Value: 3

Incidence: 70.00%

Based on data from Table 6.7, during the evaluated time span (1 to 10, i.e. the radii in the illustration). The hired SLO value was ≥ 3 (hired), the measured average MA value was 2 in the period (measured), and Incidence total was 72.16% (percentage of incidents of level 2). Therefore, one can conclude that security was bad with 72.16% of the sample below expectations (there is a security problem).

Table 6.7: Measured for Firewall metric (MA)

$Met_{2.3}$	**Incidents**	**Percentage**
4	1,931	9.29 %
3	3,469	16.69 %
2	15,003	72.16 %
1	320	1.54 %
0	67	0.32 %

Figure 6.9 presents the visual result of monitoring the Firewall metric ($Met_{2.3}$).

Table 6.8 illustrates a security metric, specifically the firewall metric $M_{2.3}$ for this case study.

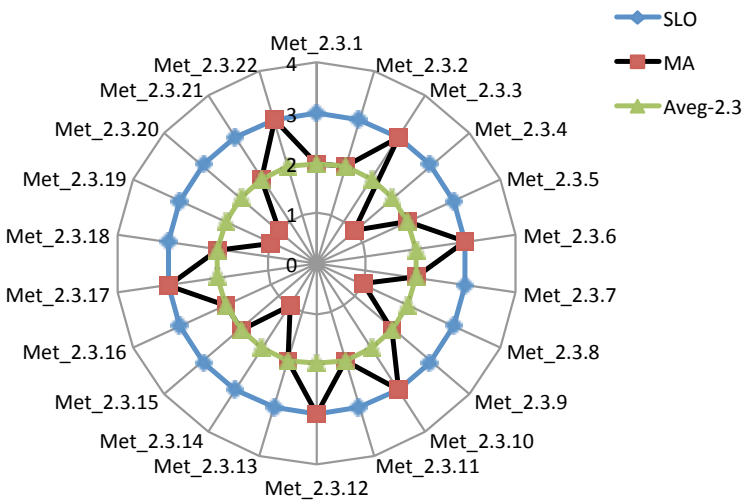


Figure 6.9: Measured for firewall metric (SLO), $\text{Met}_{2.3}$, is 3 and engaged MA is 2.

6.3.5 Results

It proposed a methodology for management of Cloud Computing using security criteria. The index of security (IndSec) transparently conveys the security level measured in the Cloud Computing environment for the various security features modeled in the metrics hierarchy. Moreover, this approach has the advantage of supporting hierarchical decomposition, which allows the model to be more scalable and distributed. In Appendix A (page 132), this portfolio of security metrics is described in more details.

6.4 Obtaining Index of Allocation (IndAlloc)

The Index of Allocation (IndAlloc) is calculated from the index of security, and its value represents the resource allocation percentage that will be supplied to the user:

$$\text{IndAlloc}(\%) = 100 - (20 * (4 - \text{IndSec}))$$

Table 6.9 shows the resource allocation percentage calculated as a function of the index of security.

The next two sections describe two alternative strategies to be chosen by the user for the implementation of the allocation index.

Strategy A: Apply in All

In the Apply in All (AA) strategy, the physical structure of the Cloud Computing is seen as a single logical unit for resource allocation management. Therefore, this results in an allocation index ranging from 20% to 100% of the original allocation factor. Figure 6.10 depicts this strategy.

As an example of this strategy, a client who has a security index that equals 2 will be allocated 60% of his/her resource requests.

Table 6.8: Security-SLA (Security Metrics)

Item	Description (Metric)	Value of metric
2	**Infrastructure Cloud Computing**	$\text{Met}_{2} \geq 3$
2.3	**Firewall**	$\text{Met}_{2.3} \geq 2$
2.3.1	Average number of hours an unauthorized device is found on the network	$\text{Met}_{2.3.1} \geq 3$
2.3.2	Average time vulnerabilities are patched	$\text{Met}_{2.3.2} \geq 4$
2.3.3	Application Software Security Threat Level	$\text{Met}_{2.3.3} \geq 4$
2.3.3	Backup mechanism	$\text{Met}_{2.3.4} \geq 4$
2.3.5	Mean time between failures	$\text{Met}_{2.3.5} \geq 4$
2.3.6	Mean time between maintenance	$\text{Met}_{2.3.6} \geq 4$
2.3.7	Mean time failure duration	$\text{Met}_{2.3.7} \geq 4$
2.3.8	Mean time to incident detection/identification	$\text{Met}_{2.3.8} \geq 3$
2.3.9	Mean time to incident eradication	$\text{Met}_{2.3.9} \geq 3$
2.3.10	Mean time to incident recovery	$\text{Met}_{2.3.10} \geq 3$
2.3.11	Mean time maintenance duration	$\text{Met}_{2.3.11} \geq 4$
2.3.12	Mean time to recovery from abnormal system shut-down	$\text{Met}_{2.3.12} \geq 3$
2.3.13	Mean time system availability	$\text{Met}_{2.3.13} \geq 4$
2.3.14	Mean time system down (failure)	$\text{Met}_{2.3.14} \geq 4$
2.3.15	Mean time system down (maintenance)	$\text{Met}_{2.3.15} \geq 4$
2.3.16	Security events records	$\text{Met}_{2.3.16} \geq 3$
2.3.17	Security rule control	$\text{Met}_{2.3.17} \geq 3$
2.3.18	Packet filtering	$\text{Met}_{2.3.18} \geq 2$
2.3.19	Total number of malicious packets found entering network	$\text{Met}_{2.3.19} \geq 3$
2.3.20	Total number of unauthorized devices found in a given period	$\text{Met}_{2.3.20} \geq 3$
2.3.21	Total number of vulnerabilities found	$\text{Met}_{2.3.21} \geq 3$
2.3.22	Unauthorized device threat eevel	$\text{Met}_{2.3.22} \geq 3$

Table 6.9: Resource allocation using the index of security

Index of Allocation			
$\text{IndSec}^{v,h,s}$	IndAlloc^{v}	**Priority**	**Impact**
4	100 %	maximal	none
3	80 %	high	low
2	60 %	medium	medium
1	40 %	low	high
0	20 %	minimal	critical

Strategy B: Apply in Regions

In the Apply in Regions strategy (AR), the physical structure of the Cloud Computing is divided into five logical regions that represent the calculated security levels (0, 1, 2, 3 and

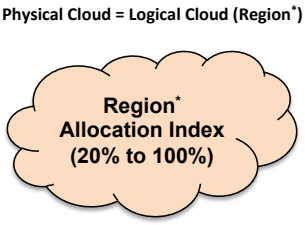


Figure 6.10: Computing the allocation index through Apply in All (AA).

4), where $\text{Region}_i \Leftrightarrow \text{IndSec} = i$. Each region will provide a specific resource percentage, as shown in Table 6.9. Figure 6.11 shows this strategy, which may be exemplified by, say, a client who owns a security index that equals 2, that will be located in Region_2 and allocated 60% of his/her resource requests.

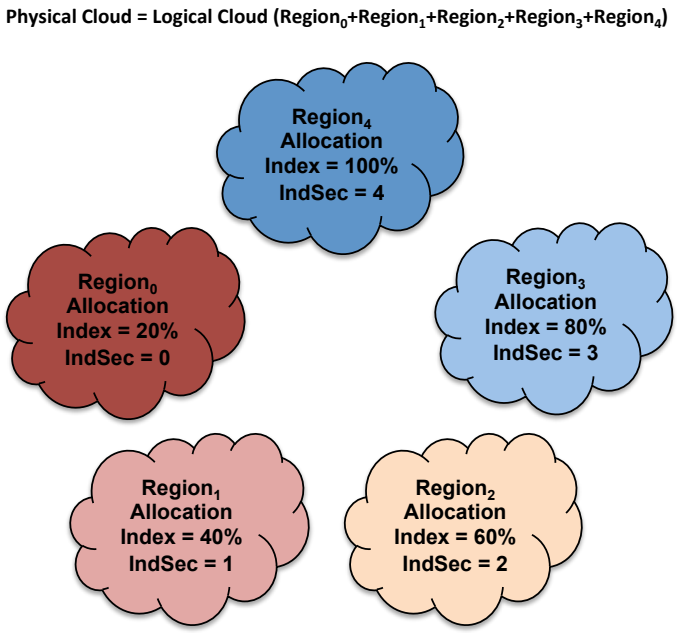


Figure 6.11: Computing the allocation index through Apply in Regions (AR).

Strategy Comparison

Strategy A has a lower cost to create and maintain (single region), therefore presents a lower security level, since all clients continue to share the same infrastructure. A security incident on a client's domain could have consequences on other client's. In contrast, strategy B has a higher creation and maintenance cost but due to its intrinsic confinement guarantees, to a certain degree, a more desirable security level among different client services. Figure 6.12 shows an estimate of this comparison.

6.4.1 Case Study

To validate the proposed cost management in relation to time, a prototype was developed and tested in a small Cloud environment based on the OpenNebula solution

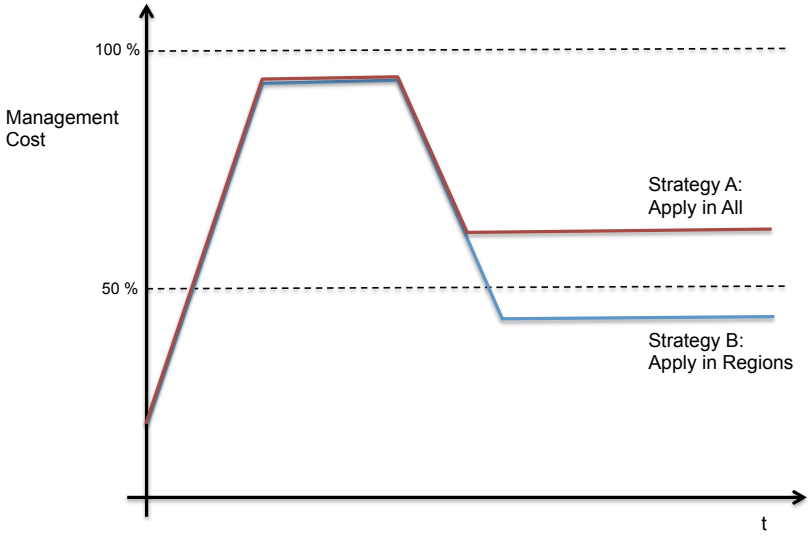


Figure 6.12: Strategy comparison.

[OpenNebula, 2014] composed of 6 machines with 2 intel Xeon Quad core 3.6 GHz and 16 GB of RAM running O.S. Gentoo Linux and KVM hypervisor. With a machine as a server to Infrastructure Manager Server, and the other five for servers (Node-0, .., Node-4) for the execution hired services, connected through the switch, and representing the five levels of the scale of values [0-4] the Figure 6.13.

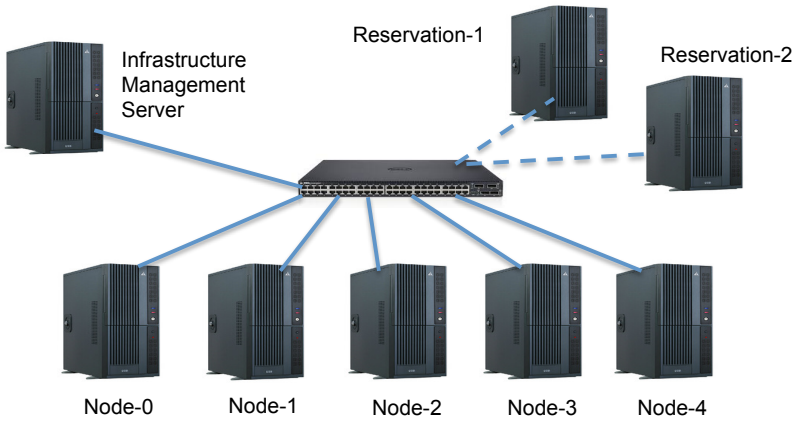


Figure 6.13: Security architecture: IndSec and IndAlloc.

Tests conducted during validation intended to determine the consumption of CPU and memory resources, and the volume of network traffic produced (performance cost) to implement the strategies A (Apply in All) and B (Apply in Regions). For the consumer-side testing of these resources, a number of allocations were performed using different types of services provided by the Cloud and different numbers of virtual machines. Each experiment lasted twelve hours and collected measurements for 5 Nodes, two Reservation Servers and one Infrastructure Management Server.

Figure 6.14 shows the results for Strategy A of this comparison, with the following comments:

- During the time interval from 3 to 7 minutes, there happens the process of assigning

virtual machines among the five servers (nodes) with maximum CPU, memory and network consumption (VMs are loaded in the nodes according to the IndSec value);

- During the time interval from 8 to 16 minutes, there is no physical separation of the nodes, which allows for a VM to be migrated to another node as needed. At this stage, if a node has less than 30% utilization of resources (CPU, memory and network), its VMs are migrated to another node and the node is put to standby;
- From this moment onward, the migration of VMs and deactivation of nodes settles, which causes the increased use of CPU, memory and network resources (leveled above 50%)

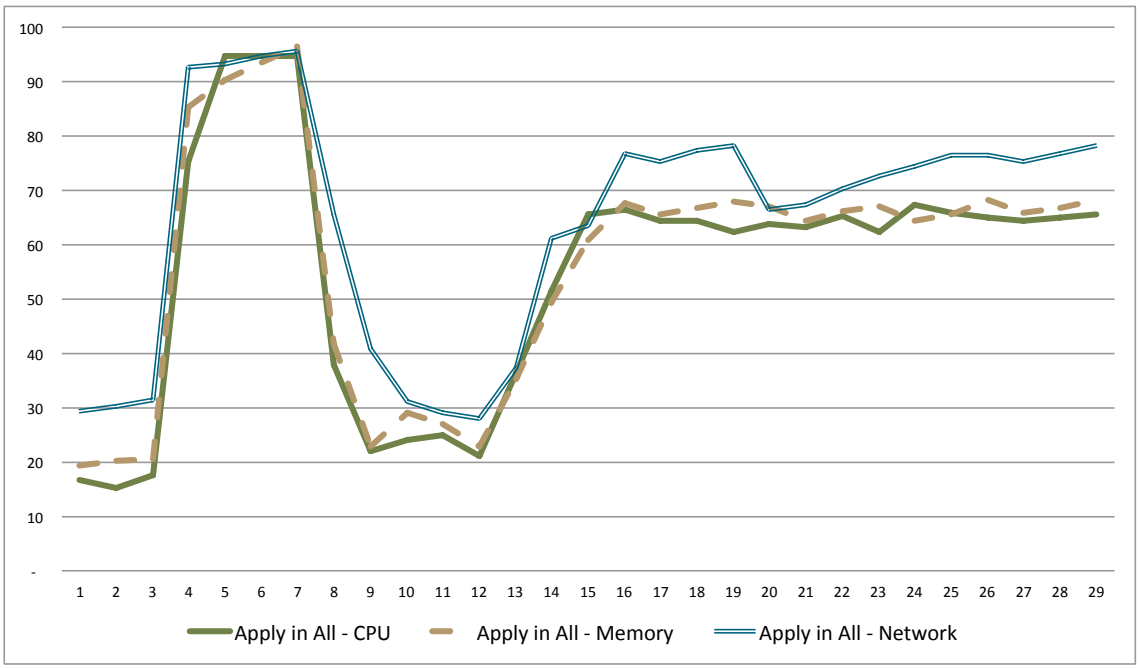


Figure 6.14: Result strategy A: Apply in All.

Notice that the migration of VMs respected only the SLA requirements for CPU, memory and network.

Figure 6.15 shows the results for Strategy B of this comparison, with the following comments:

- During the time interval from 3 to 8 minutes, there happens the process of assigning virtual machines among the five servers (nodes);
- From this moment onward, the normal processing of services in the Cloud environment settles and there is physical separation of the nodes: a VM cannot be migrated to another node. CPU, memory and network resources are being used less than 50%, as in [Beloglazov et al., 2011].

Notice that for each server the VMs were physically separated according to their IndSec, with resource use limitation according to Table 6.9.

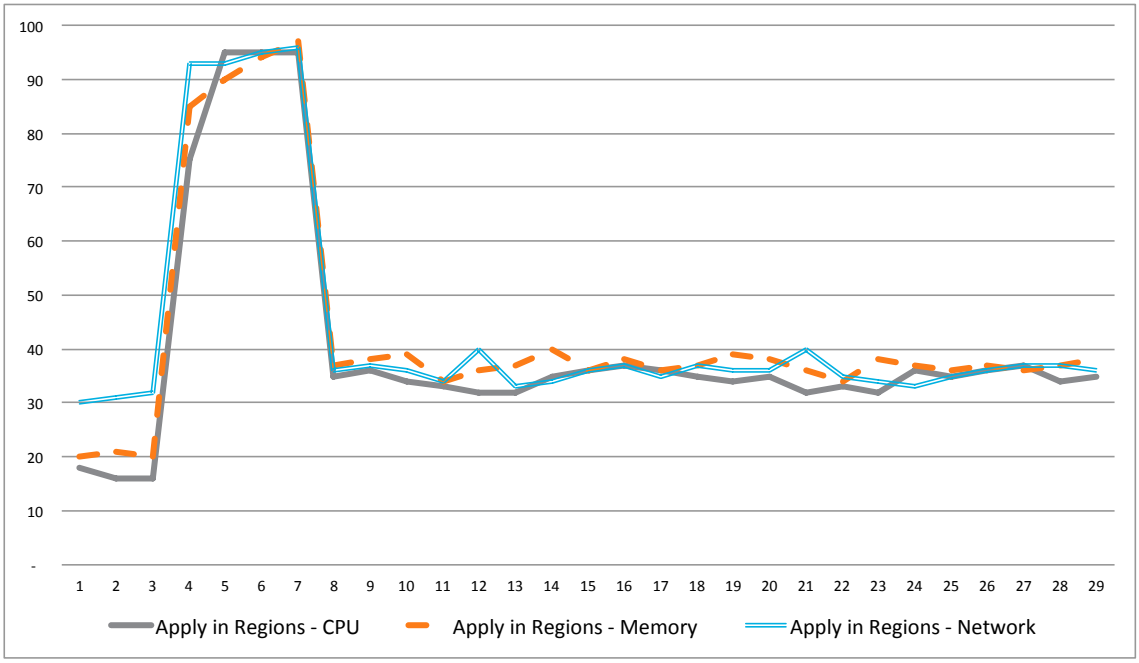


Figure 6.15: Result strategy B: Apply in Regions.

One important note is that the tests conducted were designed to assess the performance cost and, for this reason, the agents were designed to have low impact on resource consumption. In real environments, collecting information may result in CPU-intensive activities, depending on the type of information collected and the collection technique.

The results show that, despite the SOAP standard being considered heavy by making all communication through XML messages, the message envelope sizes do not generate significant traffic. During the experiments it was observed that the highest traffic activity occurs at the time of creating the VMs, when transfer agreements and monitoring agents are required. After this phase, the generated traffic comes from data transfer operations, whose range depends on the existing schedules in the agreement.

Figure 6.16 shows the combined results for Strategies A and B of this comparison.

6.4.2 Implementing Security Allocation (IndAlloc)

To implement IndAlloc at thresholds 20%, 40%, 60%, 80% and 100%, limiting memory and CPU usage requires setting appropriate parameters on the KVM's application profile, together with proper configuration of control groups (cgroups) [Turner et al., 2010]. Each VM in the computing server requires a dedicated control group. Cgroups is a Linux kernel feature that allows hierarchical allocation and management of system resources (for example, CPU, memory and disk I/O) of groups of processes [Taku Izumi, 2012]. In this case, the groups were named Group20, Group40, Group60, Group80 and Group100.

The cgroups memory subsystem isolates the memory behavior of a group of processes (tasks) from the rest of the system. It reports on memory resources used by the processes in a cgroup and sets limits on memory used by those processes. These are KVM's cgroup memory parameters:

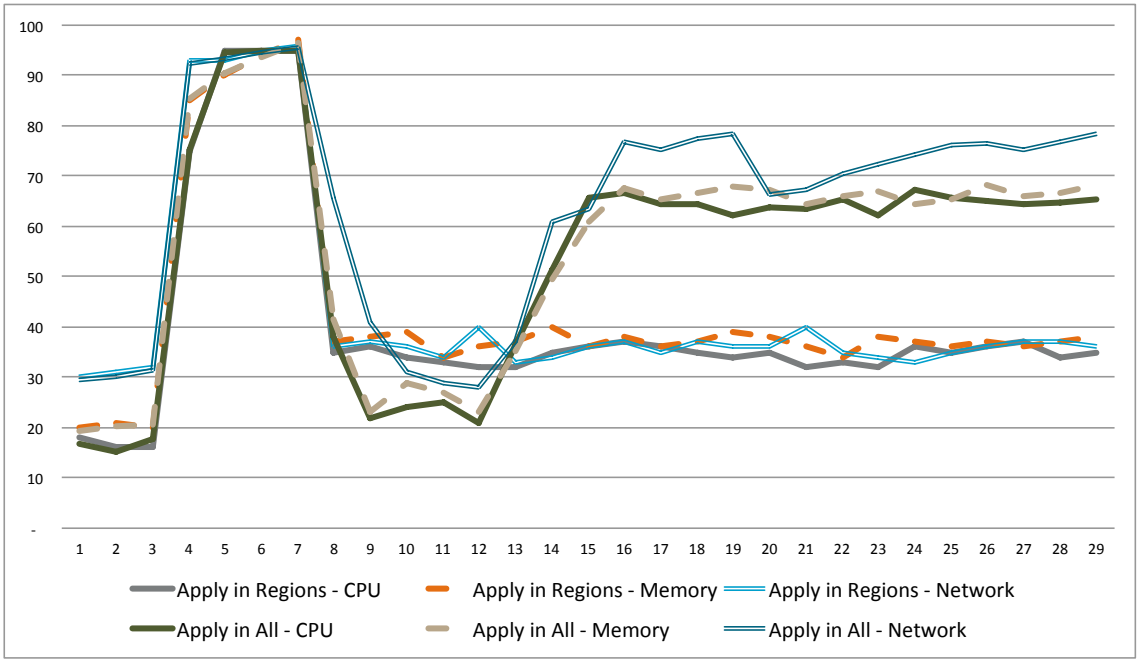


Figure 6.16: Result Strategy A and B.

- `memory.memsw.limit_in_bytes` stores the value of the virtualMemoryLimit parameter, which is configured in the VM application profile;
- `memory.limit_in_bytes` stores the value of virtualMemoryLimit and physicalMemoryLimit parameter, which is configured in the VM (application) profile.

The cgroups CPU subsystem isolates the CPU time consumption of a group of processes from the rest of the system. It reports on CPU usage by the processes in a cgroup and sets limits on CPU consumption by those processes. These are KVM's cgroup CPU parameters:

- `cpu.cfs_period_us` specifies a period of time in microseconds for how regularly a cgroup's access to the CPU resources should be reallocated. The upper limit of the cpu.cfs_period_us parameter is 1 second and the lower limit is 1000 microseconds;
- `cpu.cfs_quota_us` specifies the total amount of time in microseconds for which all tasks in a cgroup can run during one period (as defined by cpu.cfs_period_us). As soon as tasks in a cgroup use up all the time specified by the quota, they are throttled for the remainder of the time specified by the period and not allowed to run until the next period.

Together, the `cpu.cfs_period_us` and `cpu.cfs_quota_us` store the `cpuLimit` parameter value, which is configured within the VM (application) profile. The `cpuLimit` parameter stores the number of cores on which a KVM service (VM) is expected to run. Table 6.10 describes how the `cpuLimit` value translates the `cpu.cfs_period_us` and `cpu.cfs_quota_us` cpu cgroup parameters.

The KVM platform translates the `cpuLimit` value to cpu cgroup parameters using these formulas:

$$\texttt{cpu.cfs\_period\_us} = 100000 \quad (0.1 \text{ second})$$

Table 6.10: Examples for IndSec, IndAlloc and `cpuLimit` value parameters

IndSec	**IndAlloc**	**cpuLimit**	**cpu.cfs_quota_us**	**cpu.cfs_period_us**
0	20%	1	100000	100000
1	40%	2	200000	100000
2	60%	3	300000	100000
3	80%	4	400000	100000
4	100%	5	500000	100000

and

$$\texttt{cpu.cfs\_quota\_us} = m * \texttt{cpu.cfs\_period\_us} \quad ,$$

which in this case is the range of IndSec, [0–4], where m is equal to 1, 2, 3, 4 or 5, respectively.

6.4.3 Results

The results allows for stating that this new model to view security information through a range of values (0–4) copes quite well with the task it proposed to address, dealing with the problem of managing software (provided services) with different security levels within the same environment. The Index of Allocation (IndAlloc) presented two strategies for resource management that addressed scalability and granularity in Cloud Computing, while presenting the customer a more tangible feeling of how the hired service is being protected. In Appendix A (page 132), this portfolio of security metrics is described in more detail.

6.5 Summary

This chapter described a new quantitative and qualitative methodology for obtaining the return on security investment for a specific deployment profile through the use of security metrics. The proposal covers the services offered by the Cloud computing providers from a client security perspective. It described in detail the Security-SLA management and presented case studies to validate it. It also presented the Index of Security and Index of Allocation for cloud computing environments using security metrics.

Chapter 7

Cloud Security Monitoring Architecture

This section presents the solution for monitoring Security-SLAs. The basic features of the architecture will be discussed, its components, functionality and mechanisms for integration with the Cloud architecture. The section also describes the approach for representing security agreements with a view on automated monitoring.

The first version of this architecture was described in [Ferreira, 2013]; since then, new improvements have been incorporated by the research group. The proposed monitoring solution aims to verify compliance with Security-SLAs for virtual machines running on Clouds. To this end, the architecture was designed to ease the integration of the distributed Cloud infrastructure components. The main features of the architecture are:

- **Monitoring Security-SLA-based**: In addition to monitoring security parameters that must be guaranteed by the service, the proposed solution is based on the agreement itself, the metrics information and the intervals between data collection. The solution also guarantees a high level of automation of the agreement monitoring process;

- **Multi-agent system**: collecting information that will be used in the Security-SLA compliance verification is done through specialized programs. Each agent program collects the desired information from existing components in the Cloud infrastructure, supplying a subset of security metrics;

- **Multiplatform architecture**: the proposed solution supports monitoring of different platforms and settings. This enables its use in different implementations of the Cloud infrastructure and different virtualization technologies from the hired services.

To represent a Security-SLA, a hybrid approach was adopted, one in which numerical metrics are used to assess compliance with the agreement. Security policies may be optionally specified for use by security control mechanisms. The language used was Web Service Level Agreement (WSLA) [Ludwig et al., 2003], capable of representing complex metrics through linking with lower-level submetrics, describe schedule time controls for data collection and use conditional expressions and actions for monitoring of the agreement's goals [Ferreira, 2013].

7.1 Components Architecture

The infrastructure of a Cloud is made up of different hardware and software components that operate in coordination in order to provide the service. In Clouds, the main components are: Cloud Nodes, which constitute the computing power offered to customers; the Infrastructure Management Server, which allows management of the nodes, storage systems, network devices (switches, routers, Internet Service Providers etc.) and security (firewall, IDS/IPS etc.).

To interact with the Cloud infrastructure, the monitoring solution consists of different software components that run on Cloud Nodes, Security-SLA Servers and Infrastructure Management Servers. The architecture also defines a Security-SLA Manager component that is responsible for managing all information related to Security-SLA Agreements and the collected monitoring data. Although represented as a single entity, the Security-SLA Manager can be deployed across multiple physical servers, in order to ensure scalability and redundancy for the proposed architecture.

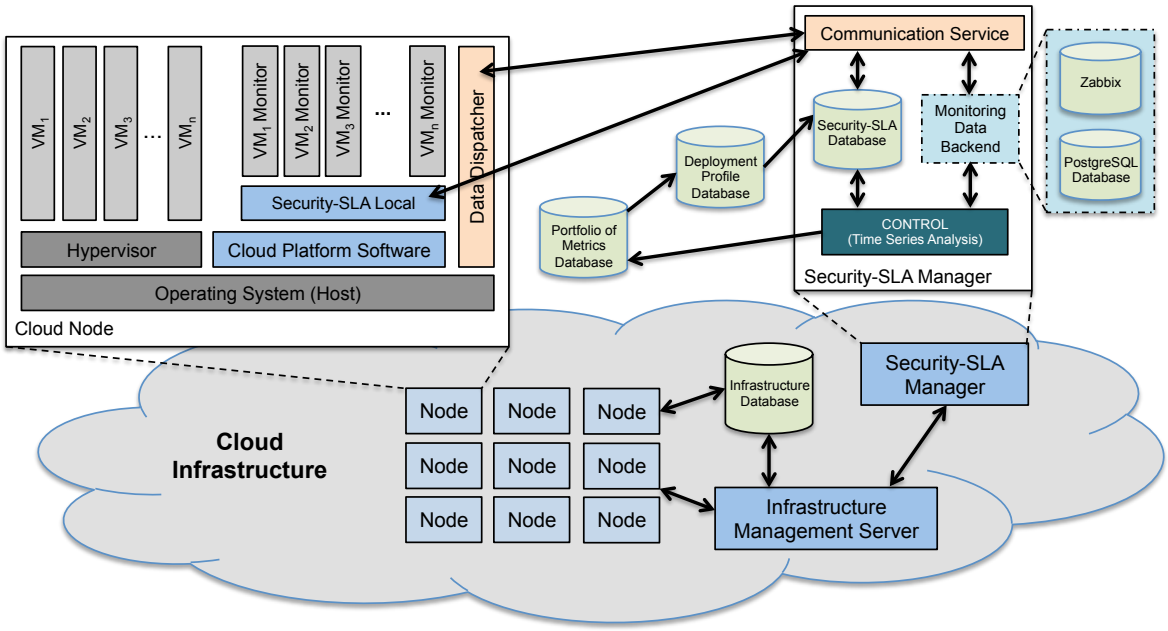


Figure 7.1: Proposed Cloud security architecture.

Figure 7.1 presents the components of the monitoring architecture and the relationship of these components with the Cloud infrastructure:

- **Cloud Node**: presents the basics components of a Cloud computing environment, such as operating system (host), virtual machine (VM) and hypervisor. The components that present new functionality will be described as follows:
  - **Security-SLA Local**: runs on each Cloud Node and Security-SLA Manager. Its function is to prepare the environment and also monitor the implementation of the VM Monitor. For this, the Security-SLA Local communicates with the Infrastructure Manager Server and the Security-SLA Manager, which allows

the retrieval of information about the Security-SLA agreements and monitor the changes to the execution state of the VM;

The communication between the Infrastructure Manager Server and Security-SLA Local is done via hooks. A hook allows actions to be taken when specific events occur in the Cloud Infrastructure. In the case of the monitoring solution, events that are treated are related to creation, initialization, suspension, migration and stopping a VM. In each of these events, a new instance of the Security-SLA Local is started, the actions are performed according to the state of the VM and the Security-SLA Local is finished;

The communication between the Security-SLA Local and the VM Monitor is made through an exclusive communication channel based on sockets. Through this channel, the Security-SLA Local can request initiation or suspension of the monitoring process and also finishing the Security-SLA Local execution, for situations when the VM is finished or migrated to another Cloud Node;

– **VM Monitor**: the Virtual Machine Monitor (VM Monitor) is a multithreaded utility run in the Cloud Node that is responsible for controlling the VM monitoring process. Each VM allocated to run on the Node has an associated VM Monitor, which remains running throughout the VM life-cycle. The VM Monitor is terminated only when the VM finishes its execution or when it is migrated to another Node. While remaining resident, the VM Monitor only collects information during the period in which the VM is running, Thus reducing resources usage on the Node;

  Each VM Monitor Consists of three distinct types of threads: the main, responsible for communication with the Security-SLA Local; the scheduler, responsible for controlling the execution of monitoring tasks; and the threads of tasks that control the agents' execution;

  To coordinate the execution of the agents, the scheduler uses the information (schedule) existing in the Security-SLA itself. When the time comes for the task execution, the scheduler creates the task thread and the file that will be used to store the collected data;

  The format of the file generated during the task's execution follows the extension model of WSLA, described in [Ferreira, 2013]. After running all agents, the task thread is finished and the file containing the collected data is moved to the transfer area (spool), where it will await its processing by the Data Dispatcher;

– **Data Dispatcher**: the process responsible for the transfer of measurement files collected by Monitors VM to the Security-SLA Manager. The mechanism used for transfer is based on a shared spool area which is periodically monitored by the transmitter, looking for new data. As new files are found, the transmitter uses the Data service in this communication layer to send data to the Security-SLA Manager. During the transfer process, transaction control mechanisms are used to ensure integrity at the destination;

  - **Cloud Platform Software**: this software controls large pools of computing, storage, and networking resources throughout a datacenter. Among the open source Cloud solutions are Eucalyptus [Eucalyptus, 2012], OpenNebula [OpenNebula, 2014] and OpenStack [Project, 2008], which also offer integrated capabilities for monitoring. These solutions allow monitoring only of basic information such as CPU load, storage space usage and network traffic. In the proposed architecture, OpenNebula was chosen for convenience purposes: i) private and public resources; ii) lightweight and easy to install, maintain, operate and use; iii) fully open-source and customizable to fit into any data center and policies; iv) private and hybrid clouds and datacenter virtualization.
  - **Agents**: they form the basis of all the monitoring process and are responsible for measuring the entities whose metrics are specified in the Security-SLA agreements. An agent is an executable utility that can be either a binary format or a script to be created in any language that is supported by the Node;

    Communication between an agent and VM Monitor is simple. As an agent runs, it receives the parameters from the VM Monitor: the identifier of the VM that will be monitored, execution parameters and a set of metrics that will be measured. The results are output to the VM Monitor through standard output (stdout). The success or failure in running the agent is reported through the return code from the operating system and possible error messages are sent through the standard error device (stderr);

    During its execution, the agent has a mechanism provided by the architecture that provides facilities for temporary data persistence and obtains additional information, such as the type of Cloud solution used, hypervisor, network interfaces, VM type and paths to important directories.

- **Infrastructure Management Server**: this module is responsible for managing the Cloud Infrastructure and Nodes that compose the Cloud. Its role is to register on the Infrastructure Database data about the infrastructure, classifying the kinds of devices, available communication, schedules and intervals for these communications, as well as the record of available Nodes for use in the Cloud, identifying for each Node the kind of operating system, virtual machine, hypervisor and Cloud Platform Software;

- **Infrastructure Database**: the relational database that is responsible for storing information about the Cloud Infrastructure and Nodes that compose the Cloud. It also records relationships between Services that are available, hired or in execution by customers. Such information is used by the monitoring agents to set the data collection parameters of the security metrics;

- **Security-SLA Manager**: in the Cloud Infrastructure, this module is responsible for the management and control of all information related to Security-SLA agreements and monitoring collected data. Internally, it has the following submodules:

  - **Communication Service**: the component that provides access to information from the Security-SLA Database to the other components of the architecture. To provide a secure, standardized communication mechanism, this component was implemented as a Web Services server that abstracts database entities and provides a set of operations for each entity through standard Simple Object Access Protocol (SOAP) [W3C, 2007a];

    Aimed at integrating the monitoring solution to other existing systems, storage of the collected data is done through modules that allow the use of different data repositories, transparently to the rest of the architecture;

    The Communication Service module receives data collected from the Data Dispatcher module and transfers it to the Backend Monitoring Data module;
  - **Backend Data Monitoring**: this submodule is the integration between the data collected in the monitoring process and the entire Cloud management structure through the Zabbix tool [Zabbix, 2001] and the PostgreSQL database [PostgreSQL, 1996];

    Zabbix is the ultimate enterprise-level software designed for real-time monitoring of a number of metrics, scalable to tens of thousands of servers, virtual machines and network devices. It provides mechanisms for device discovery and monitoring distributed agent-based event alerts;

    The PostgreSQL is a well-known relational database that records data from the monitoring process, notifications and alerts;
  - **Security-SLA Database**: this database records all the parameters of each SLO's Security-SLA metrics and their respective submetrics, originated from the collecting agents;
  - **Control (Time Series Analysis)**: the Control module has the functionality of a Security-SLA administrator and is implemented as a command line utility that allows the monitoring solution managers to supply the managing information to the database. Through it one can manage user information, import Security-SLA agreements, register and maintain metrics and manage information collecting agents. The collected metrics will be analyzed through the Time Series Analysis submodule to predict the behavior of the security metrics and then record information about the Quality of Service in the Portfolio of Metrics Database.

- **Portfolio of Metrics Database**: this database records all the security metrics for the Infrastructure and Service class, their respective submetrics and the Quality of Service class, which will be offered to the customers;

- **Deployment Profile Database**: this database records all the security metrics for the Infrastructure and Service class, their respective submetrics, which will be used to compute the Return On Security Investment.

This architecture does not specify a rigid method for collecting measurements, thus allowing different mechanisms to be used to obtain information.

7.1.1 Agents in IaaS, PaaS and SaaS

Within the context of Cloud Computing, the VM is a black-box inasmuch as the provider does not know what is running inside it, and so no knowledge on the kind of service that is running is available. In this approach, the process of monitoring and/or auditing for black-boxes is divided into three strategies, namely:

(A) **Outside**: it covers all the hardware devices and Cloud entities, such as networking, firewalls, routers, proxies, operating systems etc. The monitoring agents will run on these hardware/software to generate security metrics and transmit the data through the SNMP and HTTP protocols or through direct access to databases or other external services. In summary, this strategy covers the devices that make up the infrastructure of the Cloud computing environment;

(B) **Inside**: For the monitoring agents to run inside the virtual machine (VM), the customer has to permit the installation of the monitoring/auditing code that collects security information;

(C) **Introspection**: for monitoring intrinsic elements of a black-box, Virtual Machine Introspection (VMI) from the LibVMI library is used [Rosenblum, 2003]. Introspection allows access to the VM memory from the hypervisor, enabling information queries about the data structures of the VM's operating system kernel [VMITools, 2013]. Security anomaly detection is performed through machine learning over security metrics, deriving performance signatures for the monitoring system.

The monitoring process depends on the type of service being hired and where threats and vulnerabilities are identified, according to attack profiles. It computes the risk and impact to the Cloud customer. The next section describes the execution flow of the proposed architecture.

7.2 Execution Flow

The execution flow of the monitoring process, shown in Figure 7.2, is integrated to the Inside and Outside strategies as operations performed on VMs by the Cloud management solution.

When a customer performs an operation on a VM through the Control submodule, this transaction is sent to the running Node through the Cloud Management Solution (flows 1 and 2). At this point, the Cloud infrastructure notifies the Security-SLA Local about the event that occurred (flow 3) using a hook mechanism.

According to the type of operation requested by the customer, the Cloud Management Solution may perform different activities in the monitoring process. These activities involve: obtaining information about the Security-SLA and transferring it to the agents in charge of monitoring the VM (flow 4); execution or completion of the VM Monitor (flow 6); sending notifications about VM state changes to the VM Monitor (flow 7); and VM status update to the database from the information received by the VM Monitor (flows 9 and 10).

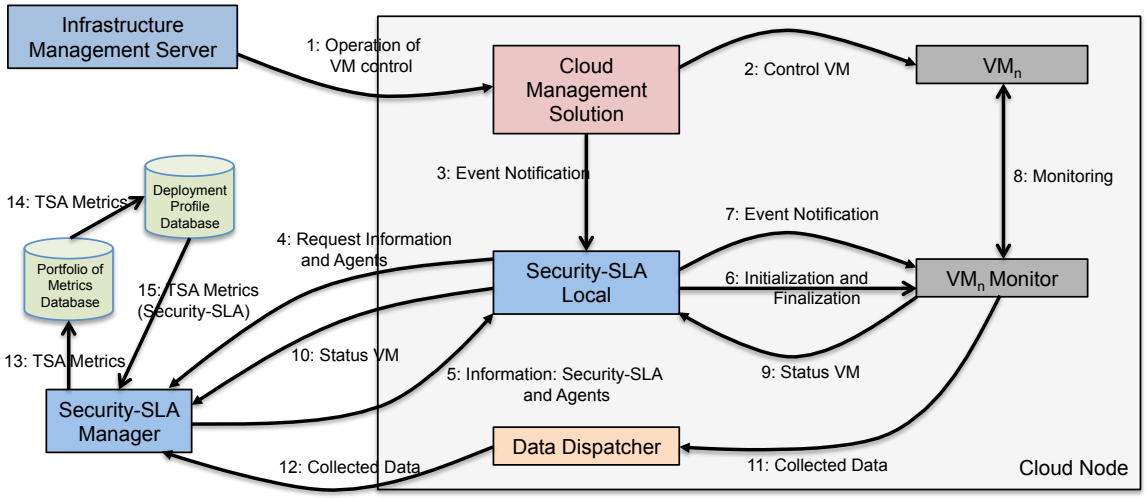


Figure 7.2: Execution flow for the proposed Cloud security architecture.

While running the monitor, the information received from the Cloud Management Solution determines the beginning or end of the data collection cycle. When data collection is active (the VM is running), the monitor performs the monitoring tasks and receives measurements from the agents (flow 8). At the end of each task's execution, the monitor sends the data to the Data Dispatcher (flow 11), which in turn transfers this data for storage and compliance assessing through monitoring data from the Backend Data Monitoring (flow 12).

Based on the data collected and stored in the Monitoring Data Backend, the Time Series Analysis submodule will predict the security metrics' behavior and finally record information about the Quality of Service in the Portfolio of Metrics Database (flow 13). Such information is again transferred to the Deployment Profile Database (flow 14) and that will allow computing the Return On Security Investment for each service provided (flow 15).

7.3 Architecture Validation

To validate the proposed cost management as related to time, a prototype was developed and tested in a small Cloud environment based on the OpenNebula solution [OpenNebula, 2014], composed of 6 machines with 2 Intel quad-core 3.6 GHx Xeon CPUs, with 16 GB of RAM running Gentoo Linux and the KVM hypervisor. Network composed with Gigabit Ethernet switch and cat 6 cabling.

One machine ran as the Infrastructure Manager Server and the other five as computing servers (Node-0, .., Node-4) for execution of the hired services, connected through a switch and representing the five levels of the scale of values [0-4].

Tests conducted during validation intended to determine the CPU consumption and the network traffic produced by the proposed architecture. For CPU utilization tests, a series of experiments were carried out using different monitoring intervals and with different numbers of virtual machines. Each experiment lasted twelve hours and measurements were collected by the Cloud Nodes and Cloud Management Solution.

Tables 7.1 and 7.2 describe the percentage of CPU utilization and network traffic volume produced on a Cloud Node, respectively.

Table 7.1: CPU utilization on a Cloud Node

Collection intervals	**CPU usage**		
	2 VMs	**7 VMs**	**15 VMs**
30 sec	0.1 %	0.2 %	0.3 %
1 min	0.1 %	0.2 %	0.2 %
10 min	0.1 %	0.1 %	0.2 %

In these simulations, the Cloud Node should have at least two VMs, one for monitoring and one for the provided service.

Table 7.2: Network traffic produced on a Cloud Node

Collection intervals	**Network traffic produced**		
	2 VMs	**7 VMs**	**15 VMs**
30 sec	0.1 %	0.2 %	0.3 %
1 min	0.1 %	0.1 %	0.2 %
10 min	0.1 %	0.1 %	0.2 %

Tables 7.3 and 7.4 describe the percentage of CPU utilization and network traffic volume, respectively, produced by the Cloud Management Solution.

Table 7.3: CPU utilization on the Cloud Management Solution

Collection intervals	**Use CPU**		
	1 VM	**7 VMs**	**15 VMs**
30 sec	0.0 %	0.1 %	0.1 %
1 min	0.0 %	0.0 %	0.1 %
10 min	0.0 %	0.0 %	0.0 %

In these simulations, the Cloud Management Solution must have at least one VM for monitoring.

Table 7.4: Network traffic produced on the Cloud Management Solution

Collection Intervals	**Network traffic produced**		
	1 VM	**7 VMs**	**15 VMs**
30 sec	0.0 %	0.1 %	0.1 %
1 min	0.0 %	0.1 %	0.1 %
10 min	0.0 %	0.0 %	0.1 %

The results show that the average CPU utilization by the architecture components represented less than 0.3 % of the total CPU time on the Cloud Node and on the Cloud Management Solution, even when multiple instances of VMs were monitored simultaneously. As expected, the most significant CPU utilization occurred on the Cloud Node, which executes the management tasks of monitoring and data collection through agents.

The network utilization tests measured the traffic generated by the communication between processes running on the Cloud Node (Security-SLA Local and Security-SLA Server) and the Cloud Management Solution. This traffic is strictly composed of SOAP messages representing information request on agreement operations, agent and data transfer, and VM state change notifications.

The results show that the SOAP standard does not generate significant traffic. During the experiments more traffic was observed at the VM's creation moment, when agreement and agent transfers take place. After this phase, the generated traffic comes from data transfer operations (sendData), whose intervals depend on the schedules present in the agreement.

7.4 Summary

This chapter presented the Cloud security monitoring architecture solution proposed in this thesis, based on Security-SLA. Also Discussed was the execution flow, validation of the architecture and its interaction with the Cloud components.

Chapter 8

Conclusions and future work

The infrastructure of a cloud environment is very complex. This complexity translates into more effort needed for monitoring and managing security. The greater scalability and larger size of clouds as compared to traditional service hosting infrastructures involve more complex security monitoring systems, which have therefore to be more scalable, robust and efficient.

The Security Metrics Hierarchy (SMH) Methodology proposal introduced a general, flexible and extensible, yet of simple use and easy to understand way of managing security in the cloud through a unified view of diverse security aspects. It showed that this methodology may be used with a plethora of services. Based on the security policy, SMH is composed of two parts, infrastructure and services. Infrastructure accounts for all kinds of hardware and software available commercially, independent of manufacturer, technology and services model, scaling up to large numbers of active devices being monitored. Services, regardless of the service type (IaaS, PaaS, SaaS or others), contract and model of billing, allows for measuring security in a tangible, more meaningful way.

The Normalization process attempts to convert any security metric to a scale of values [0-4], standardizing metrics and thus allowing analyses and comparisons in regard to those metrics' performance and behavior.

By analyzing the values of the validation indicators, as true-positive (TP), false-positive (FP), true-negative (TN) and false-negative (FN) rates, one can determine the degree of reliability for the collected security metric values by computing precision, recall, f-measure and accuracy, for instance.

The security metrics behavior can be time-series analyzed to produce a series trend, therefore producing behavior baselines for continuous, proper monitoring of security requirements compliance over time to a given service contract.

Security-SLAs are used to guarantee customers a certain level of security quality for their services. In a situation where this level of quality is not met, the provider pays penalties for the breach of contract. In order to save cloud providers from paying penalties and consequently increasing their profit, providers have to monitor the current security status of resources and continuously check whether the established Security-SLAs are being kept or violated. Thus, in order to facilitate appropriate monitoring of automatic Security-SLAs, low-level security metrics to high-level Security-SLA requirements mappings were developed. This in turn enables anticipating future Security-SLA violation threats so as

to react before actual violations occur.

The direct conversion of SMH to Security-SLA, where each security metric is a Service Level Objective (SLO), presents greater transparency and quality in the process. Once established, Security-SLAs should not be violated even in case of system failures, changed environmental conditions or unpredictable events. Thus, self-management of the Infrastructure and Service while minimizing user-interactions with the system represents a new approach to manage cloud computing environments based on security requirements. The customer can monitor what is being offered in the contract, transparently tracking and viewing security requirements of the Security-SLA.

One of the main contributions of this work is to allow the customer to analyze and compare actual security metric values supplied by the Cloud Service Provider to the values expected in the negotiated Security-SLA. The provider, in turn, may identify which of the requirements are turning out to be beneficial or prejudicial to the customer, so now the provider is able to advise the customer about service migrations to the cloud. The ROSI guides the customer to choose one of the options: accept the migration, reject it or choose another Deployment Profile with new parameters for assets, models, types and controls, or even choose another provider that satisfies the given parameters. In addition, other indexes like IndSec (as a function of time, weight, or impact risk) and IndAlloc (based on the SMH) helps managing the cloud computing environment both from a customer's as well as from a provider's perspective, which were evaluated with some study cases.

The Cloud Security Monitoring Architecture presents a new solution for monitoring Security-SLA-based virtualization resources in terms of on-demand service provision and detection of possible Security-SLA violations. These rely on predefined service level objectives and on the use of knowledge databases to predict security metrics in an attempt to prevent such violations. The architecture reports also provide a baseline (taken over appropriate monitoring intervals) for applications, services IaaS, PaaS, SaaS and others, depending on their resource consumption behavior.

8.1 General Results

In this thesis, the following general results were obtained for the proposed model:

- levels of abstraction that allows the process of security management to be monitored abstracted from the hardware, software and services, applicable to IaaS, PaaS, SaaS and other;
- scalability that allows monitoring of all infrastructure and services, with both automatic and on-demand scaling;
- a way of managing complex systems security-wise at a lower cost;
- a way of extending a security monitoring system to cope with new technologies;
- granularity to present information about the cloud computing environment with multiple security level views, up to a synthetic outlook of the cloud, involving all the variables affecting QoS and other requirements;

- smoother transition from security policies at the design level to a management level.

8.2 Specific Results

In this thesis, the following specific results were obtained for the proposed model:

- a new methodology for management of Cloud Computing using security criteria;
- a cloud security monitoring architecture for management based on Security-SLA;
- a model for the visualization of the monitoring scene based on Security-SLA;
- a new approach to calculating the return on security investment from a customer perspective;
- the introduction of a composite index of security (IndSec) to assess the service provided;
- the introduction of an index of allocation (IndAlloc) for the provider, to allow for service isolation and better resource provisioning from the security point of view.

8.3 Future Work

Security metrics are a critical aspect of the selection, operation and use of Cloud services. They allow managers to gain a better understanding of Cloud service properties through consistent, reproducible and repeatable observations. They can also be used for a wide range of objectives, from decision-making to operation.

The proposed architecture may be extended and integrated into other models that address other aspects of the metrics ecosystem, like the context of a given security metric, the observation and measurement results based on a given security metric or the scenarios that make use of security metrics. Some of these aspects are already being explored in a joint research partnership.

More specifically, one finds outlined below some possible development themes derived from this thesis:

- **Automatic normalization**: the architecture currently uses security metrics that can be measured automatically from the environment, but the process still requires experts to set up limiting values for the ranges, which means that our model is highly dependent on human intervention;
- **New formulations for IndSec and IndAlloc**: other approaches based on behavior prediction through the classic model of time series analysis, or else;
- **Prediction based on both low-level and high-level metrics**: identification of patterns resulting from low-level metrics (related to IaaS: CPU, memory, operating system, network etc.) to high-level service metrics from Security-SLA parameters. This process strongly depends on the type of services being provided and the behavior of the security metrics in the long run;

- **Allocation management using security criteria**: is already under development based an Integer Linear Programming (ILP) to schedule incoming requests according to different QoS and security levels. The joint research proposes mapping those levels into the charging models offered by IaaS, PaaS and SaaS providers. It will then be possible to maximize the number of requests properly executed, according to the customer's QoS and security requirements, while at the same time minimizing costs;

- **Green Cloud technologies**: achieving energy and cost efficient monitoring, since the activities involved may be highly demanding in terms of computing and communication resources, network topology (routers, switches etc.), cooling systems and so on;

- **Application of Security Metrics Hierarchy to other technologies**: in the context of mobile communications, trust relationships exist between enterprise management and users, but the enterprise has no control over the mobile device's security configuration, and even the user who owns the device does not have adequate means to control its configuration.

8.4 Publications

The list below includes published results that served as the basis for this text:

1. 2012: Projeto e Gestão de Segurança na Computação na Nuvem. Carlos Alberto da Silva, Anderson Soares Ferreira. Workshop de Tese e Dissertações de 2012 no Instituto de Computação, Institute of Computer Unicamp, Campinas, Brazil. Pages: 65-69. 23 May 2012.

2. 2012: A Methodology for Management of Cloud Computing using Security Criteria. Carlos Alberto da Silva, Anderson Soares Ferreira, and Paulo Lício de Geus. IEEE Latin American Conference on Cloud Computing and Communications (Latin-Cloud'12). Pages: 49-54. 26-27 November 2012, Porto Alegre, Brazil.

3. 2014: Gestão da Segurança para ambiente de Nuvem Computacional. Carlos Alberto da Silva and Paulo Lício de Geus. V Escola Regional de Informática (ERI). Universidade Federal de Mato Grosso do Sul, Campus de Ponta Porã, 27-29 Agust 2014.

4. 2014: Arquitetura de monitoramento para Security-SLA em Nuvem Computacional do tipo SaaS. Carlos Alberto da Silva and Paulo Lício de Geus. XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg'14). 3-6 november 2014, Belo Horizonte (MG), Brazil.

5. 2014: An Approach for Security-SLA in Cloud Computing Environments. Carlos Alberto da Silva and Paulo Lício de Geus. 6th IEEE Latin-American Conference on Communications (LATINCOM'14). Pages 1-6, 5-7 november 2014, Cartagena das Índias, Colômbia.

6. 2015: Return On Security Investment for Cloud Computing: a Customer Perspective. Carlos Alberto da Silva, and Paulo Lício de Geus. ACM 7th International conference on Management of computational and collective Intelligence in Digital EcoSystems (MEDES'15). 25-29 October 2015, Caraguatatuba/São Paulo, Brazil.

8.5 Submitted Articles

Computer security:

1. A New Approach to Management of Cloud Computing using Security Criteria. Journal of IEEE Transactions on Cloud Computing. Submitted in September 2015.

8.6 Future Articles

1. Journal paper about "Allocation management using Security Criteria" in partnership with Cristiano Costa Argemon Vieira, FACOM, UFMS.

2. Journal paper about "A new approach for Green Cloud using Security Criteria" in partnership with Marcos Paulo Moro, UFGD.

Bibliography

[A. Riley et al, 2008] A. Riley et al (2008). ITIL Configuration Management. Technical report. Available in `http://www.itlibrary.org/index.php?page=Configuration_Management`. Access in July 15, 2015.

[Adinolf et al., 2009] Adinolf, O., Cristaldi, R., Coppolino, L., and Romano, L. (2009). Qos-monaas: A portable architecture for qos monitoring in the cloud. *IEEE 8th International Conference on Signal Image Technology and Internet Based Systems (SITIS'12)*, pages 527–532.

[Ahmed et al., 2012] Ahmed, M., Chowdhury, A. S. M. R., Ahmed, M., and Rafee, M. M. H. (2012). An advanced survey on cloud computing and state-of-the-art research issues. *International Journal of Computer Science Issues (IJCSI)*.

[Al-Haj et al., 2013] Al-Haj, S., Al-Shaer, E., and Ramasamy, H. G. V. (2013). Security-aware resource allocation in clouds. *IEEE 10th International Conference on Services Computing (ICSC 2013)*, pages 400–407.

[Al-Hassan, 2013] Al-Hassan, M. N. M. (2013). *Thesis for Master Degree: A Semantic Ontology based Concept for Measuring Security Compliance of Cloud Service Providers*. Faculty of Information Technology, Amman, Jordan.

[Amazon, 2008] Amazon (2008). Amazon CloudWatch. Available in `http://aws.amazon.com/es/cloudwatch/`. Access in April 15, 2015.

[Amazon, 2015] Amazon (2015). Problems with Amazon's DynamoDB database affected Netflix and other web sites on Sunday morning. Available in `http://fortune.com/2015/09/20/amazon-cloud-snafu/`. Access in September 20, 2015.

[Andreozzia et al., 2005] Andreozzia, S., Bortolib, N. D., Fantinelc, S., Ghisellia, A., Rubinia, G. L., Tortoneb, G., and Vistolia, M. C. (2005). Gridice: a monitoring service for grid systems. *Future Generation Computer Systems*, page 559–571.

[Apache, 1995] Apache (1995). The Apache Software Foundation. Available in `https://www.apache.org/`. Acessado em 09 de julho de 2013.

[Arshad et al., 2009] Arshad, J., Townend, P., and Xu, J. (2009). Quantification of security for compute intensive workloads in clouds. *15th International Conference on Parallel and Distributed Systems - IEEE*.

[Atzori et al., 2011] Atzori, L., Granelli, F., and Pescapè, A. (2011). A network-oriented survey and open issues in cloud computing. *Cloud Computing: Methodology, System, and Applications*, pages 91–108.

[Barzilai, 2005] Barzilai, J. (2005). Measurement and preference function modelling. *International Transactions in Operational Research*, 12:173—183.

[Basili, 2002] Basili, V. (2002). *Goal Question Metric (GQM) Approach, Encyclopedia of Software Engineering*. John Wiley & Sons.

[Basili et al., 1994] Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, 2:528–532.

[Bayuk, 2011] Bayuk, J. (2011). Cloud security metrics. *IEEE 6th International Conference on System of Systems Engineering (SoSE'11)*, pages 341–345.

[Beloglazov et al., 2011] Beloglazov, A., Buyya, R., Lee, Y. C., and Zomaya, A. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. 82:47–111.

[Berberova and Bontchev, 2009] Berberova, D. and Bontchev, B. (2009). Design of service level agreements for software services. *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (CompSysTech '09)*.

[Bernsmed et al., 2011] Bernsmed, K., Jaatun, M. G., Meland, P. H., and Undheim, A. (2011). Security slas for federated cloud services. pages 202–209.

[Bianco et al., 2008] Bianco, P., Lewis, G. A., and Merson, P. (2008). Service level agreements in service-oriented architecture environments. Technical Report CMU/SEI-2008-TN-021, Carnegie Mellon University - SEI. Available in `http://www.sei.cmu.edu/reports/08tn021.pdf`. Access in December 15, 2014.

[Bogus, 2008] Bogus, A. (2008). *Lighttpd*. Packt Publishing Ltd. ISBN: 978-1-847192-10-3.

[Boundary, 2010] Boundary (2010). Boundary software. Available in `http://boundary.com/`. Access in July 15, 2015.

[Brandic, 2009] Brandic, I. (2009). Towards self-manageable cloud services. *33rd Annual IEEE International Computer Software and Applications Conference*.

[Brandic et al., 2009] Brandic, I., Music, D., Leitner, P., and Dustdar, S. (2009). Vieslaf framework: Enabling adaptive and versatile sla-management. *Grid Economics and Business Models, Lecture Notes in Computer*, 5745.

[Brocke et al., 2007] Brocke, J., Buddendick, C., and Strauch, G. (2007). Return on security investments - design principles of measurement systems based on capital budgeting. *AMERICAS CONFERENCE ON INFORMATION SYSTEMS (AMCIS'2007)*, pages 21–31.

[Brotby and Hinson, 2013] Brotby, W. K. and Hinson, G. (2013). *PRAGMATIC Security Metrics, Applying Metametrics to Information Security.* CRC Press.

[Bruno, 2011] Bruno, A. M. G. (2011). *Thesis for Master Degree: security metrics to evaluate quality of protection.* IT Depto, Faculty of Sciences, Lisbon University.

[Buyya et al., 2011a] Buyya, R., Broberg, J., and Goscinski, A. (2011a). *Cloud Computing - Principles and Paradigms.* Wiley.

[Buyya et al., 2011b] Buyya, R., Garg, S. K., and Calheiros, R. N. (2011b). Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. *Proceedings of the International Conference on Cloud and Service Computing (CSC 2011).*

[CA Tech., 2015] CA Tech. (2015). CA Unified Infrastructure Management (formerly CA Nimsoft Monitor). Available in `http://www.ca.com/us/opscenter/ca-unified-infrastructure-management.aspx`. Access in April 15, 2015.

[CAMM, 2010] CAMM (2010). Steering committee for the common assurance maturity model. Technical report, Common Assurance Maturity Model (CAMM). Available in `http://common-assurance.com/resources/CAMM-response-to-Cloud-computing.pdf`. Access in July 15, 2015.

[Casola et al., 2006] Casola, V., Mazzeo, A., Mazzocca, N., and Rak, M. (2006). A sla evaluation methodology in service oriented architectures, quality of protection. *Quality of Protection, Advances in Information Security*, 23:119–130.

[Cavusoglu et al., 2006] Cavusoglu, H., Cavusoglu, H., and Zhang, J. (2006). Economics of security patch management. *The Fifth Workshop in the Economics of Information Security (WEIS'2006).*

[Cavusoglu et al., 2004] Cavusoglu, H., Mishra, B., and Raghunathan, S. (2004). A model for evaluating it security investments. *COMMUNICATIONS OF THE ACM*, 47(7):87–92.

[Centrify, 2008] Centrify (2008). Centrify software. Available in `http://www.centrify.com/`. Access in July 15, 2015.

[Chen et al., 2008] Chen, X., Garfinkel, T., Lewis, E. C., Subrahmanyam, P., Waldspurger, C. A., Boneh, D., Dwoskin, J., and Ports, D. R. K. (2008). Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. *ACM SIGOPS Oper. Syst. Rev.*, 43(3):2–13.

[Chen et al., 2010] Chen, Y., Paxson, V., and Katz, R. H. (2010). What's new about cloud computing security? Technical report, Electrical Engineering and Computer Sciences University of California at Berkeley.

[Chew et al., 2008] Chew, E., Swanson, M., Stine, K., Bartol, N., Brown, A., and Robinson, W. (2008). Performance measurement guide for information security. Technical Report SP-800-55-rev1, National Institute of Standards and Technology (NIST). Available in `http://csrc.nist.gov/publications/nistpubs/800-55-Rev1/SP800-55-rev1.pdf`. Access in July 15, 2015.

[Chow et al., 2008] Chow, J., Garfinkel, T., and Chen, P. M. (2008). Decoupling dynamic program analysis from execution in virtual environments. *USENIX 2008 Annual Technical Conference on Annual Technical Conference (ATC'08)*, page 1–14.

[Cichonski et al., 2012] Cichonski, P., Millar, T., Grance, T., and Scarfone, K. (2012). Computer security incident handling guide. Technical Report SP-800-61 Revision 2, National Institute of Standards and Technology (NIST). Available in `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf`. Access in July 15, 2015.

[CICSWG, 2000] CICSWG (2000). Incident Cost Analysis and Modeling Project (ICAMP) Final Report 2. Technical report, Committee on Institutional Cooperation Security Working Group (CICSWG). Available in `https://www.cic.net/docs/default-source/reports/icampreport2.pdf?sfvrsn=0`. Access in July 15, 2015.

[CipherCloud, 2011] CipherCloud (2011). CipherCloud Software. Available in `http://www.ciphercloud.com`. Access in April 15, 2015.

[Cisco, 2010] Cisco (2010). Security Monitoring Analysis Response System. Available in `http://www.cisco.com/c/en/us/products/security/security-monitoring-analysis-response-system/index.html`. Access in April 15, 2015.

[Clayman et al., 2010] Clayman, S., Galis, A., and Mamatas, L. (2010). Monitoring virtual networks with lattice. *IEEE/IFIP - Network Operations and Management Symposium Workshops (NOMS Wksps)*, page 239–246.

[Cloud-Council, 2012] Cloud-Council (2012). Practical guide to cloud service level agreements version 1.0. Technical report, Cloud Standards Customer Council. Available in `http://www.cloud-council.org/2012_Practical_Guide_to_Cloud_SLAs.pdf`. Access in July 15, 2015.

[Cloud-Council, 2015] Cloud-Council (2015). Security for Cloud Computing Ten Steps to Ensure Success Version 2.0. Technical report, Cloud Standards Customer Council. Available in `http://www.cloud-council.org/Security_for_Cloud_Computing_Version_2.pdf`. Access in July 15, 2015.

[Cloud Cruiser, 2010] Cloud Cruiser (2010). Cloud cruiser software. Available in `http://cloudcruiser.com/`. Access in July 15, 2015.

[CloudCmp, 2011] CloudCmp (2011). Project CloudCmp. Available in `https://github.com/angl/cloudcmp`. Access in December 15, 2014.

[CloudFlare, 2009] CloudFlare (2009). CloudFlare Software. Available in `https://www.cloudflare.com`. Access in April 15, 2015.

[CloudHarmony, 2013] CloudHarmony (2013). Cloudharmony. Available in `http://cloudharmony.com/`. Access in July 15, 2015.

[CloudPassage, 2011] CloudPassage (2011). Cloudpassage software. Available in `http://cloudpassage.com`. Access in April 15, 2015.

[CloudStack, 2008] CloudStack (2008). Cloudstack project. Available in `http://www.cloudstack.org/`. Access in July 15, 2015.

[Cloudyn, 2012] Cloudyn (2012). Cloudyn monitor. Available in `http://www.cloudyn.com/`. Access in July 15, 2015.

[Coar and Bowen, 2008] Coar, K. and Bowen, R. (2008). *Apache Cookbook, 2nd Edition*. O'Reilly Media, Inc. ISBN: 978-0-596-52994-9.

[Coatanea et al., 2007] Coatanea, E., Yannou, B., and Honkala, S. (2007). Measurement theory and dimensional analysis: methodological impact on the comparison and evaluation process. *In Proceedings of ASME-07*.

[Collectl, 2007] Collectl (2007). Available in `http://collectl.sourceforge.net/index.html`. Access in December 15, 2014.

[Corradi et al., 2012] Corradi, A., Foschini, L., Povedano-Molina, J., and Lopez-Soler, J. M. (2012). Dds-enabled cloud management support for fast task offloading. *IEEE Symposium on Computers and Communications (ISCC'2012)*, pages 67–74.

[CSA, 2011] CSA (2011). Cloud controls matrix. Technical report, Cloud Security Alliance (CSA). Available in `https://cloudsecurityalliance.org/research/ccm/`. Access in July 15, 2015.

[CSA, 2014a] CSA (2014a). Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. Technical report, Cloud Security Alliance (CSA). Available in `https://cloudsecurityalliance.org/csaguide.pdf`. Access in December 15, 2014.

[CSA, 2014b] CSA (2014b). Security, trust & assurance registry (star). Technical report, Cloud Security Alliance (CSA). Available in `https://cloudsecurityalliance.org/star/`. Access in December 15, 2014.

[de Chaves et al., 2010a] de Chaves, S. A., Westphall, C. B., and Lamin, F. R. (2010a). SLA perspective in security management for cloud computing. *Sixth International Conference on Networking and Services*, pages 212–217.

[de Chaves et al., 2010b] de Chaves, S. A., Westphall, C. B., and Lamin, F. R. (2010b). Sla perspective in security management for cloud computing. *Sixth International Conference on Networking and Services (ICNS'10)*, pages 212–217.

[Dell, 2012] Dell (2012). Dell PacketTrap Remote monitoring and mangagement (RMM). Available in `http://www.packettrap.com`. Access in April 15, 2015.

[Dempsey et al., 2011] Dempsey, K., Chawla, N. S., Johnson, A., Johnston, R., Jones, A. C., Orebaugh, A., Scholl, M., and Stine, K. (2011). Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations. Technical Report SP-800-137, National Institute of Standards and Technology (NIST). Available in `http://csrc.nist.gov/publications/nistpubs/800-137/SP800-137-Final.pdf`. Access in July 15, 2015.

[Dorofee et al., 2007] Dorofee, A., Killcrece, G., Ruefle, R., and Zajicek, M. (2007). Incident management capability metrics version 0.1. Technical report, Software Engineering Institute. TECHNICAL REPORT CMU/SEI-2007-TR-008, ESC-TR-2007-008, Available in `https://resources.sei.cmu.edu/asset_files/TechnicalReport/2007_005_001_14873.pdf`. Access in July 15, 2015.

[Dunlap et al., 2002] Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A., and Chen, P. M. (2002). Revirt: Enabling intrusion analysis through virtual-machine logging and replay. *ACM SIGOPS Operating Systems Review - OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 211–224.

[Dynatrace, 2008] Dynatrace (2008). Dynatrace synthetic monitoring. Available in `http://www.dynatrace.com/en/index.html`. Access in July 15, 2015.

[Egawa et al., 2012] Egawa, T., Nishimura, N., and Kourai, K. (2012). Dependable and secure remote management in iaas clouds. *In Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, pages 411–418.

[Emeakaroha et al., 2010a] Emeakaroha, V. C., Brandic, I., Maurer, M., and Dustdar, S. (2010a). Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. *In International Conference on High Performance Computing and Simulation (HPCS'10)*, pages 48–54.

[Emeakaroha et al., 2010b] Emeakaroha, V. C., Calheiros, R. N., Netto, M. A. S., Brandic, I., and Rose, C. A. F. D. R. A. F. D. (2010b). Desvi: An architecture for detecting sla violations in cloud computing infrastructures. *2nd International ICST Conference on Cloud Computing.*

[ENISA, 2009a] ENISA (2009a). Cloud computing: Benefits, risks and recommendations for information security. Technical report, European Network and Information Security Agency (ENISA). Available in `http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport`. Access in December 15, 2014.

[ENISA, 2009b] ENISA (2009b). Cloud computing: Benefits, risks and recommendations for information security. Technical report, European Network and Information

Security Agency (ENISA). Available in `http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport`. Access in December 15, 2014.

[ENISA, 2009c] ENISA (2009c). Cloud computing information assurance framework. Technical report, European Network and Information Security Agency (ENISA). Available in `http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-information-assurance-framework`. Access in December 15, 2014.

[ENISA, 2012] ENISA (2012). Introduction to return on security investment. Technical report, European Network and Information Security Agency (ENISA). Available in `https://www.enisa.europa.eu/activities/cert/other-work/introduction-to-return-on-security-investment`. Access in December 15, 2014.

[Eucalyptus, 2012] Eucalyptus, I. (2012). Eucalyptus system. Technical report. Available in `http://www.eucalyptus.com/eucalyptus-cloud`. Access in December 15, 2014.

[Everbridge, 2011] Everbridge (2011). Everbridge software. Available in `http://www.everbridge.com/products/everbridge-platform/system-capacity-resiliency-security/`. Access in July 15, 2015.

[FedRAMP, 2013] FedRAMP (2013). Security assessment framework. Technical report, Federal Risk and Authorization Management Program (FedRAMP). Available in `https://www.fedramp.gov/`. Access in July 15, 2015.

[Ferreira, 2013] Ferreira, A. S. (2013). *Uma arquitetura para monitoramento de segurança baseada em acordos de níveis de serviço para nuvens de infraestrutura*. Instituto de Computação, UNICAMP, Brazil. Available in `http://www.bibliotecadigital.unicamp.br/document/?code=000915715`. Access in July 15, 2015.

[Ferretti et al., 2010] Ferretti, S., Ghini, V., Panzieri, F., Pellegrini, M., and Turrini, E. (2010). Qos-aware clouds. *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD '10)*, pages 321–328.

[Finkelstein and Leaning, 1984] Finkelstein, L. and Leaning, M. S. (1984). A review of the fundamental concepts of measurement. *Measurement*, 1(2):25–34.

[Foley et al., 2006] Foley, S. N., Bistarelli, S., O'Sullivan, B., Herbert, J., and Swart, G. (2006). Multilevel security and quality of protection. *QUALITY OF PROTECTION - Security Measurements and Metrics*, 23:93–105.

[Foster et al., 2008] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop (GCE '08)*, pages 1–10.

[Fox et al., 2009] Fox, A., Armbrust, M., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., and Zaharia, M. (2009). Above the

clouds: A Berkeley view of cloud computing. Technical report, Electrical Engineering and Computer Sciences University of California at Berkeley.

[Frenz, 2010] Frenz, S. (2010). Ontology-based generation of it-security metrics. *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10)*, pages 1833–1839.

[Galis et al., 2010] Galis, A., Clayman, S., and Mamatas, L. (2010). Monitoring virtual networks with lattice. *IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Wksps)*, pages 239 – 246.

[Ganglia Software, 2000] Ganglia Software (2000). Ganglia monitoring system. Available in `http://ganglia.sourceforge.net`. Access in December 15, 2014.

[Garfinkel, 1999] Garfinkel, S. L. (1999). *Architects of the Information Society: Thirty-Five Years of the Laboratory for Computer Science at MIT*. The MIT Press.

[Gonzalez and Helvik, 2012] Gonzalez, A. J. and Helvik, B. E. (2012). Guaranteeing sla availability in telecommunications networks. pages 1–6.

[Gonzalez et al., 2012] Gonzalez, N., Miers, C., Redigolo, F., Simplicio, M., Carvalho, T., Naslund, M., and Pourzandi, M. (2012). A quantitative analysis of current security concerns and solutions for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications, SpringerOpen Journal*, 11(1).

[Groundwork Software, 2011] Groundwork Software (2011). Groundwork. Available in `http://www.gwos.com/`. Access in December 15, 2014.

[GRyCAP, 2009] GRyCAP (2009). Grid y computación de altas prestaciones group. Available in `http://www.grycap.upv.es/compaas/index.htm`. Access in December 15, 2014.

[Halonen and Hatonen, 2010] Halonen, P. and Hatonen, K. (2010). Towards holistic security management through coherent measuring. *Proceedings of the Fourth European Conference on Software Architecture (ECSA'10), ACM*, pages 155–161.

[Hasselmeyer and d'Heureuse, 2010] Hasselmeyer, P. and d'Heureuse, N. (2010). Towards holistic multi-tenant monitoring for virtual data centers. *IEEE/ IFIP Network Operations and Management Symposium Workshops (NOMS Wksps)*, page 350–356.

[Hayden, 2010] Hayden, L. (2010). *IT Security Metrics: A Practical Framework for Measuring Security & Protecting Data*. McGraw-Hill Osborne.

[HelpSystems, 2012] HelpSystems (2012). Intermapper cloud monitor. Available in `http://www.helpsystems.com/intermapper/`. Access in April 15, 2015.

[Henning, 1999] Henning, R. R. (1999). Security service level agreements: Quantifiable security for the enterprise? *Proceedings of the 1999 Workshop on New Security Paradigms (NSPW '99)*, pages 54–60.

[Herrmann, 2007] Herrmann, D. S. (2007). *Complete guide to security and privacy metrics*. Auerbach Publications. ISBN: 0-8493-5402-1.

[Hewlett-Packard, 2005] Hewlett-Packard (2005). HP OpenView. Available in `http://www8.hp.com/us/en/software/enterprise-software.html`. Access in April 15, 2015.

[Hoefer and Karagiannis, 2010] Hoefer, C. N. and Karagiannis, G. (2010). Taxonomy of cloud computing services. *Proceedings of the 4th IEEE Workshop on Enabling the Future Service-Oriented Internet (EFSOI'10)*, pages 1345–1350.

[Hogben and Dekker, 2012] Hogben, G. and Dekker, M. (2012). Procure secure: A guide to monitoring of security service levels in cloud contracts. Technical report, European Network and Information Security Agency (ENISA). Available in `http://www.enisa.europa.eu/activities/Resilience-and-CIIP/cloud-computing/procure-secure-a-guide-to-monitoring-of-security-service-levels-in-\cloud-contracts`. Access in December 15, 2014.

[Hu et al., 2011] Hu, L., Wang, C., Schwan, K., Talwar, V., Eisenhauer, G., and Wolf, M. (2011). A flexible architecture integrating monitoring and analytics for managing large-scale data centers. *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC '11)*, pages 141–150.

[Hu et al., 2009] Hu, Y., Wong, J., Iszlai, G., and Litoiu, M. (2009). Resource provisioning for cloud computing. pages 101–111.

[Hyperic-HQ Software, 2010] Hyperic-HQ Software (2010). Hyperic-HQ. Available in `http://sourceforge.net/projects/hyperic-hq/`. Access in December 15, 2014.

[IBM, 2005] IBM (2005). IBM Tivoli Monitoring. Available in `http://www-03.ibm.com/software/products/en/tivomoni/`. Access in April 15, 2015.

[Ibrahim et al., 2011] Ibrahim, A. S., Hamlyn-Harris, J., Grundy, J., and Almorsy, M. (2011). Cloudsec: A security monitoring appliance for virtual machines in the iaas cloud model. *5th International Conference on Network and System Security (NSS'2011)*, page 113–120.

[IFAC, 2011] IFAC (2011). International Standard on Assurance Engagements (ISAE) 3402. Technical report, International Federation of Accountants (IFAC). Available in `http://www.ifac.org/sites/default/files/downloads/b014-2010-iaasb-handbook-isae-3402.pdf`. Access in July 15, 2015.

[Intralinks, 2007] Intralinks (2007). Information rights management. Available in `https://www.intralinks.com/platform-solutions/platform/information-rights-management`. Access in July 15, 2015.

[ISACA, 2014a] ISACA (2014a). It control objectives for cloud computing: Controls and assurance in the cloud. Technical report, Information Systems Audit and Control Association (ISACA). Available in

http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/IT-Control-Objectives-for-Cloud-Computing-Controls-and-\Assurance-in-the-Cloud.aspx. Access in December 15, 2014.

[ISACA, 2014b] ISACA (2014b). Security considerations for cloud computing. Technical report, Information Systems Audit and Control Association (ISACA). Available in http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/Security-Considerations-for-Cloud-Computing.aspx. Access in December 15, 2014.

[ISO, 2011] ISO (2011). Guidelines on information security controls for the use of cloud computing services. Technical report, International Organization for Standardization (ISO). Available in http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43757. Access in July 15, 2015.

[ISO/IEC-12207:2008, 2008] ISO/IEC-12207:2008 (2008). *Information technology — Software life cycle processes and ISO/IEC-15288:2008*. International Organization for Standardization (ISO).

[ISO/IEC-27000:2009, 2009] ISO/IEC-27000:2009 (2009). *Information technology - Security techniques - Information security management systems - Overview and vocabulary*. International Organization for Standardization (ISO).

[ISO/IEC-27002:2005, 2005] ISO/IEC-27002:2005 (2005). *Information technology - Security techniques - Code of practice for information security management*. International Organization for Standardization (ISO).

[ISO/IEC-27005:2011, 2011] ISO/IEC-27005:2011 (2011). *Information technology – Security techniques – Information security risk management*. International Organization for Standardization (ISO).

[ISO/IEC-27017:2014, 2014] ISO/IEC-27017:2014 (2014). *ISO/IEC 27017 — Information technology — Security techniques — Code of practice for information security controls based on ISO/IEC 27002 for cloud services (DRAFT)*. International Organization for Standardization (ISO).

[ISO/IEC-27018:2014, 2014] ISO/IEC-27018:2014 (2014). *Information technology - Security techniques - Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors*. International Organization for Standardization (ISO).

[Jaatun et al., 2012] Jaatun, M. G., Bernsmed, K., and Undheim, A. (2012). Security slas - an idea whose time has come? *Proceedings of the International Conference on Cloud and Service Computing (CSC 2011)*, pages 123–130.

[Jaquith, 2007] Jaquith, A. (2007). *Security Merics: Replacing Fear Uncertainty and Doubt*. Addison Wesley.

[Jasmine Software, 2010] Jasmine Software (2010). Jasmine. Available in `http://repository.ow2.org/nexus/content/repositories/ow2-legacy/org/ow2/jasmine/`. Access in December 15, 2014.

[Jiang et al., 2007] Jiang, X., Wang, X., and Xu, D. (2007). Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pages 128–138.

[Johnson and Qu, 2012] Johnson, B. and Qu, Y. (2012). A holistic model for making cloud migration decision: A consideration of security, architecture and business economics. *10th Int. Symposion on Parallel and Distributed Processing with Applications*, page 435–441.

[Jones et al., 2008] Jones, S. T., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H. (2008). Vmm-based hidden process detection and identification using lycosid. *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 91–100.

[Kantarcioglu et al., 2011] Kantarcioglu, M., Bensoussan, A., and Hoe, S. C. (2011). Impact of security risks on cloud computing adoption. *49th Annual Allerton Conference on Communication, Control, and Computing*, page 670–674.

[Kempter, 2011a] Kempter, S. (2011a). Checklist Request for Change RFC. Technical report. ITIL 2011 Service Transition - Change Management, IT Process Map.

[Kempter, 2011b] Kempter, S. (2011b). Configuration Management Process. Technical report. ITIL 2011 Service Transition - Change Management, IT Process Map.

[Keshavarzi et al., 2013] Keshavarzi, A., Haghighat, A. T., and Bohlouli, M. (2013). Research challenges and prospective business impacts of cloud computing: A survey. *The 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 731–736.

[Killcrece et al., 2003] Killcrece, G., Kossakowski, K.-P., Ruefle, R., and Zajicek, M. (2003). State of the practice of computer security incident response teams (csirts). Technical report, Carnegie-Mellon Software Engineering Institute. Available in `http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6571`. Access in July 15, 2015.

[KNIME, 2010] KNIME (2010). KNIME.COM AG. Available in `http://www.knime.org/knime-desktop`. Access in September 20, 2015.

[Kramosil and Michalek, 1974] Kramosil, I. and Michalek, J. (1974). A review of the fundamental concepts of measurement. *Kybernetica*, 5(11):336—-344.

[Krautsevich et al., 2010] Krautsevich, L., Martinelli, F., and Yautsiukhin, A. (2010). Formal approach to security metrics.: what does "more secure" mean for you? *Proceedings of the Fourth European Conference on Software Architecture (ECSA'10)*, pages 162–169.

[Krautsevich et al., 2011] Krautsevich, L., Martinelli, F., and Yautsiukhin, A. (2011). Formal analysis of security metrics and risk. *Proceedings of the 5th IFIP WG 11.2 international conference on Information security theory and practice: security and privacy of mobile devices in wireless communication (WISTP'11)*, pages 304–319.

[Krutz and Vines, 2010] Krutz, R. L. and Vines, R. D. (2010). *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. John Wiley & Sons, Inc.

[Kubert et al., 2011] Kubert, R., Katsaros, G., and Gallizo, G. (2011). Building a service-oriented monitoring framework with rest and nagios. *Proceedings of the 2011 IEEE International Conference on Services Computing (SCC '11)*, pages 426–431.

[Lakshmanan et al., 2010] Lakshmanan, G. T., Keyser, P., Slominski, A., Curbera, F., and Khalaf, R. (2010). A business centric end-to-end monitoring approach for service composites. *Proceedings of the 2010 IEEE International Conference on Services Computing (SCC '10)*, pages 409–416.

[Landwehr, 2001] Landwehr, C. E. (2001). Computer security. *International Journal of Information Security*, 1:3–13.

[Lanzi et al., 2009] Lanzi, A., Sharif, M., and Lee, W. (2009). K-tracer: A system for extracting kernel malware behavior. *In Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS'09)*.

[Laprie, 2008] Laprie, J.-C. (2008). From dependability to resilience. *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[Letaifa et al., 2010] Letaifa, A. B., Haji, A., Jebalia, M., and Tabbane, S. (2010). State of the art and research challenges of new services architecture technologies: virtualization, soa and cloud computing. *International Journal of Grid and Distributed Computing*.

[Lighttpd, 2005] Lighttpd (2005). The Lighttpd Software. Available in `http://www.lighttpd.net/`. Acessado em 09 de julho de 2013.

[Liu et al., 2010a] Liu, J., Gong, C., Zhang, Q., Chen, H., and Gong, Z. (2010a). The characteristics of cloud computing. *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops (ICPPW '10)*, pages 275–279.

[Liu et al., 2010b] Liu, X., Mei, H., Zhang, Y., and Huang, G. (2010b). Integrating resource consumption and allocation for infrastructure resources on-demand. *IEEE 3rd International Conference on Cloud Computing*.

[Logic Monitor, 2009] Logic Monitor (2009). Logic monitor. Available in `http://www.logicmonitor.com/`. Access in April 15, 2015.

[Lombardi and Pietro, 2009] Lombardi, F. and Pietro, R. D. (2009). Kvmsec: A security extension for linux kernel virtual machines. *Proceedings of the 2009 ACM symposium on Applied Computing (SAC'09)*, page 2029–2034.

[Ludwig et al., 2003] Ludwig, H., Keller, A., Dan, A., King, R. P., and Franck, R. (2003). Web service level agreement (wsla) language specification. technical report, ibm. Technical report. Available in `http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf`. Access in July 15, 2015.

[Mana and Pujol, 2008] Mana, A. and Pujol, G. (2008). Towards formal specification of abstract security properties. *The Third International Conference on Availability, Reliability and Security (ARES 08) - IEEE*, pages 80–87.

[Manjrasoft, 2008] Manjrasoft (2008). Aneka management. Available in `http://www.manjrasoft.com/products.html`. Access in July 15, 2015.

[Martens and Teuteberg, 2012] Martens, B. and Teuteberg, F. (2012). Decision-making in cloud computing environments: A cost and risk based approach. *Information Systems Frontiers*, 14(4):871—-893.

[Mather et al., 2009] Mather, T., Kumaraswamy, S., and Latif, S. (2009). *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Series. O'Reilly Media.

[McCune et al., 2010] McCune, J. M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V., and Perrig, A. (2010). Trustvisor: Efficient tcb reduction and attestation. *IEEE Symposium on Security and Privacy (SSP'10)*, page 143–158.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. Technical Report SP-800-145, National Institute of Standards and Technology (NIST). Available in `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`. Access in July 15, 2015.

[Mian et al., 2013] Mian, R., Martin, P., and Vazquez-Poletti, J. L. (2013). Provisioning data analytic workloads in a cloud. *Future Gener. Comput. Syst.*, pages 1452–1458.

[Microsoft Windows Azure, 2008] Microsoft Windows Azure (2008). Microsoft windows azure suite. Available in `http://www.windowsazure.com`. Access in July 15, 2015.

[Monalisa, 2005] Monalisa (2005). Monalisa Project. Technical report. Available in `http://monalisa.caltech.edu/monalisa.htm`. Access in April 15, 2015.

[Monitis, 2006] Monitis (2006). Monitis. Available in `http://portal.monitis.com/`. Access in July 15, 2015.

[Montes et al., 2013] Montes, J., Sánchez, A., Memishi, B., Pérez, M. S., and Antoniu, G. (2013). Gmone: A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29:2026–2040.

[Muller, 1999] Muller, N. J. (1999). Managing service level agreements. *International Journal of Network Management*, 9:155–166.

[Nagios, 1996] Nagios (1996). Nagios Software. Technical report. Available in `http://www.nagios.org`. Access in April 15, 2015.

[Nakamura et al., 2014] Nakamura, L. H. V., Julio C. Estrella, R. H. C. S., Santana, M. J., and da Silva Dias, A. (2014). Providing iaas resources automatically through prediction and monitoring approaches. pages 1–7.

[Nayak et al., 2013] Nayak, D., Butt, M. A., Zaman, M., and Themazi, D. A. (2013). Empowering cloud security through sla. *Journal of Global Research in Computer Science*, pages 30–33.

[New Relic, 2008] New Relic (2008). New relic software. Available in `http://newrelic.com/`. Access in July 15, 2015.

[Nginx, 2003] Nginx (2003). The Nginx Software. Available in `http://nginx.org/`. Acessado em 09 de julho de 2013.

[Nguyen et al., 2009] Nguyen, A. M., Schear, N., Jung, H., Godiyal, A., King, S. T., and Nguyen, H. D. (2009). Mavmm: Lightweight and purpose built vmm for malware analysis. pages 441–450.

[Nimbus Project, 2006] Nimbus Project (2006). Nimbus project. Available in `http://www.nimbusproject.org`. Access in July 15, 2015.

[NIST, 2002] NIST (2002). Federal Information Security Management Act (FISMA). Technical report, National Institute of Standards and Technology (NIST). Available in `http://www.nist.gov/itl/csd/soi/fisma.cfm`. Access in July 15, 2015.

[NIST, 2013a] NIST (2013a). Nist cloud computing security reference architecture. Technical Report SP-500-299, National Institute of Standards and Technology (NIST). Available in `collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/CloudSecurity/NIST_security_Reference_Architecture_2013.05.15_v1.0.pdf`. Access in July 15, 2015.

[NIST, 2013b] NIST (2013b). Nist cloud computing standards roadmap working group. Technical Report SP-500-291 Version-2 FINAL, National Institute of Standards and Technology (NIST). Available in `http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf`. Access in July 15, 2015.

[NIST, 2013c] NIST (2013c). Nist guide to enterprise patch management technologies. Technical Report SP-800-40r3, National Institute of Standards and Technology (NIST). Available in `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf`. Access in July 15, 2015.

[NIST, 2013d] NIST (2013d). Nist security and privacy controls for federal information systems and organizations. Technical Report SP-800-53r4, National Institute of Standards and Technology (NIST). Available in `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf`. Access in July 15, 2015.

[NIST, 2014] NIST (2014). Common vulnerability scoring system (cvss). Technical report, National Institute of Standards and Technology (NIST). Available in `http://nvd.nist.gov/cvss.cfm`. Access in December 15, 2014.

[Okta, 2009] Okta (2009). Okta software. Available in `https://www.okta.com/`. Access in July 15, 2015.

[OpenNebula, 2005] OpenNebula (2005). OpenNebula Monitoring System. Available in `http://archives.opennebula.org/documentation:archives:rel3.4:img`. Access in April 15, 2015.

[OpenNebula, 2014] OpenNebula (2014). OpenNebula Project. Available in `http://www.opennebula.org/`. Access in December 15, 2014.

[Paessler, 1998] Paessler (1998). Prtg network monitor. Available in `http://www.paessler.com/cloud_computing_monitoring`. Access in July 15, 2015.

[Paraleap, 2011] Paraleap (2011). Paraleap technologies. Available in `http://www.paraleap.com/azurewatch`. Access in July 15, 2015.

[Payne et al., 2008] Payne, B., Carbone, M., Sharif, M., and Wenke, L. (2008). Lares: An architecture for secure active monitoring using virtualization. *IEEE Symposium on Security and Privacy (SP'08)*, pages 233 – 247.

[Payne, 2006] Payne, S. C. (2006). A guide to security metrics. Technical report, SANS Institute. Available in `http://www.sans.org/reading_room/whitepapers/auditing/guide-security-metrics_55`. Access in December 15, 2014.

[Pearson, 2013] Pearson, S. (2013). Toward accountability in the cloud. *Jornal IEEE Cloud Computing - Especial Edition: Securing the Cloud*, 1(1):6–10.

[Petcu, 2014a] Petcu, D. (2014a). Sla-based cloud security monitoring: Challenges, barriers, models and methods. *Euro-Par 2014: Parallel Processing Workshops, Lecture Notes in Computer Science*, 8805:359–370.

[Petcu, 2014b] Petcu, D. (2014b). A taxonomy for sla-based monitoring of cloud security. *38th Annual International Computers, Software and Applications Conference*, page 640–641.

[Petcu and Craciun, 2014] Petcu, D. and Craciun, C. (2014). Towards a security sla-based cloud monitoring service. *4th International Conference on Cloud Computing and Services Science (CLOSER'2014)*, page 598–603.

[PostgreSQL, 1996] PostgreSQL (1996). PostgreSQL Software. Available in `http://www.postgresql.org/`. Access in July 15, 2015.

[Project, 2008] Project, O. (2008). Openstack cloud software. Technical report. Available in `http://www.openstack.org`. Access in December 15, 2014.

[Project PCMONS, 2008] Project PCMONS (2008). Private Clouds MONitoring Systems. Technical report. Available in `https://code.google.com/p/pcmons/`. Access in April 15, 2015.

[Project Sigar, 2006] Project Sigar (2006). Hyperic SIGAR. Available in `http://sourceforge.net/projects/sigar/`. Access in April 15, 2015.

[ProofPoint, 2002] ProofPoint (2002). Proofpoint software. Available in `https://www.proofpoint.com/`. Access in July 15, 2015.

[Putri and Mganga, 2011] Putri, N. R. and Mganga, M. C. (2011). *Thesis for Master Degree: Enhancing Information Security in Cloud Computing Services using SLA Based Metrics*. School of Computing - Blekinge Institute of Technology.

[Qualys, 1999] Qualys (1999). Qualys cloud platform. Available in `https://www.qualys.com/`. Access in July 15, 2015.

[Rackspace, 2012] Rackspace (2012). Rackspace Cloud Monitoring. Available in `http://www.rackspace.com/cloud/monitoring/`. Access in July 15, 2015.

[Rahulamathavan et al., 2014] Rahulamathavan, Y., Pawar, P. S., Burnap, P., Rajarajan, M., Rana, O. F., and Spanoudakis, G. (2014). Analysing security requirements in cloud-based service level agreements. *Proceedings of the 7th International Conference on Security of Information and Networks (SIN'14)*, page 73.

[Rak et al., 2013] Rak, M., Suri, N., Luna, J., Petcu, D., Casola, V., and Villano, U. (2013). Security as a service using an sla-based approach via specs. *IEEE International Conference on Cloud Computing Technology and Science*, pages 1–6.

[Rak et al., 2011] Rak, M., Venticinque, S., Máhr, T., Echevarria, G., and Esnal, G. (2011). Cloud application monitoring: The mosaic approach. *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CLOUDCOM '11)*, pages 758–763.

[Rebollo et al., 2012] Rebollo, O., Mellado, D., and Fernández-Medina, E. (2012). A systematic review of information security governance frameworks in the cloud computing environment. *Journal of Universal Computer Science*, 18:798–815.

[Righi et al., 2004] Righi, R. R., Pellissari, F. R., and Westphall, C. B. (2004). Sec-sla: Specification and validation of metrics for security oriented service level agreements. *IV Workshop in Computing Systems Security*.

[Riley et al., 2008] Riley, R., Jiang, X., and Xu, D. (2008). Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. 5230:1–20.

[Riley et al., 2009] Riley, R., Jiang, X., and Xu, D. (2009). Multi-aspect profiling of kernel rootkit behavior. *Proceedings of the 4th ACM European Conference on Computer Systems (EuroSys'09)*, pages 47–60.

[Rimal et al., 2009] Rimal, B. P., Choi, E., and Lumb, I. (2009). A taxonomy and survey of cloud computing systems. *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC (NCM '09)*, pages 44–51.

[Riverbed, 2012] Riverbed (2012). Network performance management (old opnet). Available in `http://www.riverbed.com/products/performance-management-control/network-performance-management/`. Access in April 15, 2015.

[Rochwerger et al., 2009] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Cáceres, J., Ben-Yehuda, M., Emmerich, W., and Galán, F. (2009). The reservoir model and architecture for open federated cloud computing. *IBM J. Res. Dev.*, pages 535–545.

[Rosenblum, 2003] Rosenblum, T. G. M. (2003). A virtual machine introspection based architecture for intrusion detection. *IEEE Symposium on Security and Privacy (SP'08)*, page 191–206.

[Sandpiper, 2008] Sandpiper (2008). Project sandpiper. Technical report. Available in `http://ass.cs.umass.edu/projects/virtualization/sandpiper/`. Access in April 15, 2015.

[Savola, 2007a] Savola, R. M. (2007a). Towards a security metrics taxonomy for the information and communication technology industry. *International Conference on Software Engineering Advances (ICSEA) - IEEE*, page 60.

[Savola, 2007b] Savola, R. M. (2007b). Towards a taxonomy for information security metrics. *Proceeding QoP '07 Proceedings of the 2007 ACM workshop on Quality of protection - ACM*, pages 28–30.

[Savola, 2009] Savola, R. M. (2009). A security metrics taxonomization model for software-intensive systems. *Journal of Information Processing Systems*, 5:197.

[Services, 2011] Services, I. G. T. (2011). Security and high availability in cloud computing environments. Technical report, IBM.

[Seshadri et al., 2007] Seshadri, A., Luk, M., Qu, N., and Perrig, A. (2007). Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses. *Proceedings of 21st ACM SIGOPS symposium on Operating systems principles (SOSP'07)*, page 335–350.

[sFlow, 2003] sFlow (2003). sflow software. Available in `http://www.sflow.org`. Access in April 15, 2015.

[Shalb, 2010] Shalb (2010). Project spae. Available in `http://shalb.com/en/spae/spae_features/`. Access in April 15, 2015.

[Shao and Wang, 2011] Shao, J. and Wang, Q. (2011). A performance guarantee approach for cloud applications based on monitoring. *Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops (COMPSACW '11)*, pages 25–30.

[Shao et al., 2010] Shao, J., Wei, H., Wang, Q., and Mei, H. (2010). A runtime model based monitoring approach for cloud. *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD '10)*, pages 313–320.

[Sharif et al., 2009] Sharif, M. I., Lee, W., Cui, W., and Lanzi, A. (2009). Secure in-vm monitoring using hardware virtualization. pages 477–487.

[Sharma, 2015] Sharma, R. (2015). *NGINX High Performance*. Packt Publishing Ltd. ISBN: 978-1-78528-183-9.

[Shin and Gu, 2012] Shin, S. and Gu, G. (2012). Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). *20th IEEE International Conference on Network Protocols (ICNP'12)*, page 1–6.

[Shirey, 2007] Shirey, R. W. (2007). Internet security glossary. Technical report, Internet Engineering Task Force RFC 4949 Informational.

[Silva et al., 2012] Silva, C. A., Ferreira, A. S., and Geus, P. L. (2012). A methodology for management of cloud computing using security criteria. *Proceedings of the IEEE Latin American Conference on Cloud Computing and Communications (LatinCloud'12)*, pages 49–54.

[Silva and Geus, 2014a] Silva, C. A. and Geus, P. L. (2014a). An approach for security-sla in cloud computing environments. *6th IEEE Latin-American Conference on Communications (LATINCOM'2014)*, pages 1–6.

[Silva and Geus, 2014b] Silva, C. A. and Geus, P. L. (2014b). Arquitetura de monitoramento para security-sla em nuvem computacional do tipo saas. *XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg'2014)*, pages 310–313.

[Silva and Geus, 2014c] Silva, C. A. and Geus, P. L. (2014c). Gestao da segurança para ambiente de nuvem computacional. *Escola Regional de Informática (ERI'2014)*.

[Silva and Geus, 2015] Silva, C. A. and Geus, P. L. (2015). Return on security investment for cloud computing: a customer perspective. *ACM 7th International Conference on Management of computational and collective IntElligence in Digital EcoSystems (MEDES'15)*.

[SilverSky, 2013] SilverSky (2013). Bae systems. Available in `https://www.silversky.com/`. Access in July 15, 2015.

[Site24x7, 2007] Site24x7 (2007). Site24x7 Software. Available in `http://www.site24x7.com`. Access in April 15, 2015.

[Skype, 2015] Skype (2015). Skype is STILL offline. Available in `http://www.dailymail.co.uk/sciencetech/article-3243151/Skype-status-problem-preventing-users-making-calls.html`. Access in September 21, 2015.

[SLA, 2005] SLA (2005). SLA Management Handbook. Technical report, TeleManagement Forum. available in `http://www.tmforum.org/Guidebooks/GB917-SLAManagement/30753/article.html`. Access in December 15, 2014.

[SLA@SOI, 2010] SLA@SOI (2010). Project sla@soi. Available in `http://sourceforge.net/projects/sla-at-soi/`. Access in April 15, 2015.

[Smit et al., 2013] Smit, M., Simmons, B., and Litoiu, M. (2013). Distributed, application-level monitoring for heterogeneous clouds using stream processing. *Future Generation Computer Systems*, 29:2103–2114.

[Snorby, 2011] Snorby (2011). Project snorby. Available in `https://www.snorby.org`. Access in April 15, 2015.

[Sobel et al., 2008] Sobel, W., Subramanyam, S., Sucharitakul, A., Nguyen, J., Wong, H., Klepchukov, A., Patil, S., Fox, A., and Patterson, D. (2008). Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0.

[Sonian, 2006] Sonian (2006). Cloud monitoring sensu. Available in `http://sonian.com/about/sensu/`. Access in July 15, 2015.

[Splunk, 2005] Splunk (2005). Splunk Software. Available in `http://www.splunk.com`. Access in April 15, 2015.

[Spring, 2011a] Spring, J. (2011a). Monitoring cloud computing by layer, part 1. *IEEE Security and Privacy*, pages 66–68.

[Spring, 2011b] Spring, J. (2011b). Monitoring cloud computing by layer, part 2. *IEEE Security and Privacy*, pages 52–55.

[SSAE, 2011] SSAE (2011). Ssae 16 auditing standard. Technical report. Available in `http://www.ssae-16.com/`. Access in July 15, 2015.

[Stout et al., 2006] Stout, B., Sonnenreich, W., and Albanese, J. (2006). Return on security investment (rosi) – a practical quantitative model. *Journal of Research and Practice in Information Technology*, 38(1):55–66.

[Subashini and Kavitha, 2011] Subashini, S. P. and Kavitha, V. R. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, pages 1–11.

[Suppes and Zinnes, 1962] Suppes, P. and Zinnes, J. L. (1962). Basic measurement theory. Technical report, Technical Report 45 of Institute for mathematical studies in the social science, Stanford, USA. Technical Report 45.

[Szefer, 2013] Szefer, J. M. (2013). Architectures for secure cloud computing servers (phd thesis). Technical report, University of Princeton.

[Taku Izumi, 2012] Taku Izumi (2012). Evaluation of CPU cgroup - The Linux Foundation. Available in `https://events.linuxfoundation.org/images/stories/pdf/lcjp2012_izumi.pdf`. Access in September 20, 2015.

[Taylor, 1997] Taylor, J. R. (1997). *An introduction to Error Analysis - the study of uncertainties in physical measurements*. University Science Books.

[Team, 2015] Team, V. B. R. (2015). 2015 data breach investigations report. Technical report. Available in `http://www.verizonenterprise.com/DBIR/2015/`. Access in July 15, 2015.

[Threat Stack, 2014] Threat Stack (2014). Threat Stack Software. Available in `https://www.threatstack.com`. Access in April 15, 2015.

[TIMACS, 2009] TIMACS (2009). TIMACS Software. Available in `http://www.timacs.de`. Access in April 15, 2015.

[Tsalis et al., 2013] Tsalis, N., Theoharidou, M., and Gritzalis, D. (2013). Return on security investment for cloud platforms. *IEEE International Conference on Cloud Computing Technology and Science*, pages 132–137.

[Turner et al., 2010] Turner, P., Rao, B. B., and Rao, N. (2010). Cpu bandwidth control for cfs. pages 245–254.

[University of Waikato, 2013] University of Waikato (2013). Weka 3: Data Mining Software in Java. Available in `http://www.cs.waikato.ac.nz/ml/weka/`. Access in September 20, 2015.

[Up.Time software, 2008] Up.Time software (2008). Up.time software. Available in `http://www.uptimesoftware.com/cloud-monitoring.php`. Access in July 15, 2015.

[Vacca, 2013] Vacca, J. R. (2013). *Computer and Information Security Handbook, 2nd Edition*. Morgan Kaufmann Publishers.

[Vaquero et al., 2008] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*.

[Vaultive, 2012] Vaultive (2012). Vaultive company. Available in `http://vaultive.com/solution/vaultive-encryption-platform/`. Access in July 15, 2015.

[Viratanapanu et al., 2010] Viratanapanu, A., Hamid, A. K. A., Kawahara, Y., and Asami, T. (2010). On demand fine grain resource monitoring system for server consolidation. *In: Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services (ITU-T), IEEE.*, pages 1–8.

[VMITools, 2013] VMITools (2013). Virtual machine introspection tools. Available in `https://code.google.com/p/vmitools/`. Access in December 15, 2014.

[Vmware, 2008] Vmware (2008). Vmware cloudstatus. Available in `http://www.vmware.com/products/vrealize-hyperic/`. Access in July 15, 2015.

[Voas et al., 2012] Voas, J., Grance, T., Patt-Corner, R., and Badger, L. (2012). Cloud computing synopsis and recommendations. Technical Report SP-800-146, National Institute of Standards and Technology (NIST). Available in `http://www.nist.gov/manuscript-publication-search.cfm?pub_id=911075`. Access in July 15, 2015.

[Vu et al., 2015] Vu, Q. H., Ardagna, C. A., Asal, R., Damiani, E., and Ardagna, C. A. (2015). From security to assurance in the cloud: A survey. 48.

[W3C, 2007a] W3C (2007a). Simple Object Acess Prototol (SOAP) Part 1: Messaging Framework (2nd Edition). Available in `http://www.w3.org/TR/soap12-part1/`. Access in July 15, 2015.

[W3C, 2007b] W3C (2007b). Web Services Policy 1.5 - Framework, World Wide Web Consortium.

[West-Brown et al., 2003] West-Brown, M. J., Stikvoort, D., Kossakowski, K.-P., Killcrece, G., Ruefle, R., and Zajicek, M. (2003). Handbook for computer security incident response teams (csirts), 2nd edition, cmu/sei-2003-hb-002. Technical report, Carnegie-Mellon Software Engineering Institute.

[WhiteHat, 2001] WhiteHat (2001). Whitehat security. Available in `https://www.whitehatsec.com/index.html`. Access in July 15, 2015.

[Wu et al., 2013a] Wu, L., Garg, S. K., Buyya, R., Chen, C., and Versteeg, S. (2013a). Automated sla negotiation framework for cloud computing (ccgrid'13). pages 235–244.

[Wu et al., 2013b] Wu, X., Gao, Y., Tian, X., Song, Y., Guo, B., Feng, B., and Sun, Y. (2013b). Secmon: A secure introspec- tion framework for hardware virtualization. *IEEE 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'13).*

[Xuan et al., 2009] Xuan, C., Copeland, J., and Beyah, R. (2009). Toward revealing kernel malware behavior in virtual execution environments. 5758:304–325.

[Yazir et al., 2010] Yazir, Y. O., Matthews, C., Farahbod, R., Nevilley, S., Guitouni, A., Ganti, S., and Coady, Y. (2010). Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. *3rd International Conference on Cloud Computing - IEEE*, pages 91–98.

[Younge et al., 2010] Younge, A. J., von Laszewski, G., Wang, L., Lopez-Alarcon, S., and Carithers, W. (2010). Efficient resource management for cloud computing environments. *IEEE*.

[Zabbix, 2001] Zabbix (2001). Zabbix LLC Software. Available in `http://www.zabbix.com`. Access in July 15, 2015.

[Zenoss, 2011] Zenoss (2011). Zenpack. Available in `https://github.com/zenoss/ZenPacks.zenoss.CloudStack`. Access in July 15, 2015.

[Zhang et al., 2010] Zhang, S., Zhang, S., Chen, X., and Huo, X. (2010). Cloud computing research and development trend. *Proceedings of the 2010 Second International Conference on Future Networks (ICFN '10)*, pages 93–97.

[Zhengwei et al., 2013] Zhengwei, V., Ran, D., Zhigang, L., Xihong, W., and Baoxu, L. (2013). A meta-synthesis approach for cloud service provider selection based on secsla. *IEEE 2013 Fifth International Conference on Computational and Information Sciences (ICCIS'2013)*, pages 1356–1360.

[Zhien and Yiqi, 2012] Zhien, G. and Yiqi, D. (2012). Security slas for ims-based cloud services. *Proceedings of the 2012 Seventh ChinaGrid Annual Conference (CHINAGRID'12)*, pages 57–60.

[Zissis and Lekkas, 2012] Zissis, D. and Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation Computer Systems*, 28:583–592.

[Zscaler, 2008] Zscaler (2008). Zscaler security. Available in `http://www.zscaler.com/`. Access in July 15, 2015.

Appendix A

Portfolio of the Security Metrics

This Appendix presents the portfolio of the security Metrics, the list of metrics presented do not intend to be exhaustive nor suitable to all systems, since as stated before that would be an impossible mission in the context of this work. Table A describes each metric of the security of metric hierarchy and its respective page (full specification) this proposal. The class column specifies the type of security metric: (I) for Infrastructure, (S) for hired Service and (B) for both previous types.

Table A.1: Portfolio of the Security Metrics

Id	Description	Class	Page
1.	**Security Policy**		
1.1	**Current Level of Enforcement of the Security Policy**		
1.1.1	Number of reported security policy violations in the previous 12 months (NRSPV)	B	137
1.1.2	Number of enforcement actions taken against policy violations in the previous 12 months (NESTSPV)	B	138
1.2	**Current Structure of the Security Policy**		
1.2.1	Number of documents that make up the corporate security policy (NDMUSP)	B	139
1.2.2	Format(s) of security policy documents (FSPD)	B	140
1.2.3	Location(s) of security policy documents (LSPD)	B	141
1.2.4	Types of policy acknowledgement mechanisms (TPAM)	B	142
1.2.5	Length of time since the last security policy review by management (LTSLSPRBM)	B	143
1.3	**Employees Read and Understand the Security Policy**		
1.3.1	Ratio of employee job descriptions that specify responsibility for following the security policy (REJDTSRFSP)	B	144
1.3.2	Number of security policy awareness or training activities conducted in the previous 12 months (NSPATACP)	B	145
1.3.3	Ratio of employees who have formally acknowledged the security policy in the previous 12 months (REWHFASP)	B	146

Continued on next page

Table A.1 – *Continued from previous page*

Id	Description	Class	Page
1.3.4	Results of a user survey asking how familiar users are with the policy is judged to be (HFUAP)	B	147
1.3.5	Results of a user survey asking how appropriate and usable the policy is judged to be (HAUP)	B	148
1.4	**Enforcement of the Security Policy Increasing**		
1.4.1	Increase in security policy enforcement actions over baseline (ISPEAOB)	B	149
1.4.2	Increase in awareness of corporate security policy (IACSP)	B	150
1.4.3	Increase in efficiency of the security policy process (IESPP)	B	151
1.4.4	Improved response from surveyed users on policy familiarity and usability (IRSUPFU)	B	152
2	**Security-Related Downtime**		
2.1	**System down (failure)**		
2.1.1	Mean Time To Failures (MTTF)	B	153
2.1.2	Mean Time Between Failures (MTBF)	B	154
2.1.3	Mean Time Between Recovery (MTTRc)	B	155
2.1.4	Mean Time Between Repare (MTTRp)	B	156
2.1.5	Mean To System Availability (MTSA)	B	157
2.2	**System down (maintenance)**		
2.2.1	Duration of the Preventive Maintenance (DPM)	B	158
2.2.2	Duration of the Corrective Maintenance (DCM)	B	159
2.2.3	Mean Time Between Maintenance (MTBM)	B	160
2.3	**Downtime resulting from a security event**		
2.3.1	Number of Security Events in a time period, duration of event remediation (NSEER)	B	161
3	**Vulnerability Policies**		
3.1	**Vulnerability Management**		
3.1.1	Total number of registered hosts (TRH)	B	162
3.1.2	Total number of unregistered hosts (TUH)	B	163
3.1.3	Total number of registered hosts vulnerable (TRHV)	B	164
3.1.4	Total number of unregistered hosts vulnerable (TUHV)	B	165
3.1.5	Percentage of registered hosts vulnerable (PRHV)	B	166
3.1.6	Percentage of unregistered hosts vulnerable (PUHV)	B	167
3.2	**Mitigate Vulnerabilities**		
3.2.1	Mean Cost To Mitigate Vulnerabilities (MCTMV)	B	168
3.2.2	Mean Time To Mitigate Vulnerabilities (MTTMV)	B	170
3.2.3	Number of Known Vulnerability Instances (NKVI)	B	172
3.2.4	Percent of Systems Without Known Severe Vulnerabilities (PSWKSV)	B	174
3.2.5	Vulnerability Scan Coverage (VSC)	B	176
3.3	**Internal Vulnerability Assessment**		
3.3.1	Vulnerabilities on the internal nodes (VIN)	B	178

Continued on next page

Table A.1 – *Continued from previous page*

Id	Description	Class	Page
3.3.2	Security vulnerability counts for assessed internal nodes (SVCAIN)	B	180
3.3.3	Ratios of vulnerabilities by type, OS, owner, and so on (RVT)	B	182
3.3.4	Severe vulnerabilities found on the internal nodes (SVFIN)	B	184
3.3.5	CVSS scores for all identified vulnerabilities present on internal nodes (SIVPIN)	B	186
4	**Incident Policies**	B	
4.1	**Incident Management**	B	
4.1.1	Cost of Incidents (COI)	B	188
4.1.2	Incident Handled (IH)	B	190
4.1.3	Incident Rate (IR)	B	192
4.1.4	Mean Cost of Incidents (MCOI)	B	194
4.1.5	Mean Incident Recovery Cost (MIRC)	B	196
4.1.6	Mean Time Between Security Incidents (MTBSI)	B	198
4.1.7	Mean Time from Discovery to Containment (MTDC)	B	199
4.1.8	Mean Time To Incident Discovery (MTTID)	B	201
4.1.9	Mean Time To Incident Recovery (MTTIR)	B	203
4.1.10	Number of Incidents (NI)	B	205
4.1.11	Percentage of Incidents Detected by Internal Controls (PIDIC)	B	207
4.1.12	Time Per Incident (TPI)	B	209
5	**Patch Policies**		
5.1	**Patch Management**		
5.1.1	Configuration Management Coverage (CMC)	B	211
5.1.2	Mean Cost to Patch (MCTP)	B	213
5.1.3	Mean Time To Complete Changes (MTTCC)	B	215
5.1.4	Mean Time to Deploy Critical Patches (MTDCP)	B	217
5.1.5	Mean Time To Patch (MTTC)	B	219
5.1.6	Patch Management Coverage (PMC)	B	221
5.1.7	Patch Policy Compliance (PPC)	B	223
5.2	**Change Management**		
5.2.1	Percent of Changes with Security Exceptions (PCSE)	B	225
5.2.2	Percent of Changes with Security Review (PCSR)	B	227
6	**Infrastructure Security Policies**		
6.1	**Firewall Management**		
6.1.1	Number of Packet Filtering (NPFi)	I	229
6.1.2	Number of Security Rule Control (NSRC)	I	231
6.1.3	Number of Traffic Flow Statistics (NTFS)	I	233
6.1.4	Number of Prevention of DoS (NPD)	I	235
6.1.5	Number of Shutdown (NS)	I	237
6.1.6	Mean Time To Recovery From Shutdown Firewall (MTTRFSF)	I	239
6.2	**Intrusion Detect and Prevention System (IDPS)**		
6.2.1	Number of Packet Fragmentation (NPFr)	I	241
6.2.2	Number of Stream Segmentation (NSS)	I	243

Continued on next page

Table A.1 – *Continued from previous page*

Id	Description	Class	Page
6.2.3	Number of Remote Procedure Call Fragmentation (NRPCF)	I	245
6.2.4	Number of Recovery from Abnormal System Shutdown (NRASS)	I	247
6.2.5	Number of Security Events Records (NSER)	I	249
6.2.6	**Number of Evasion Attacks (NEA)**	I	251
6.2.6.1	Number of URL Obfuscation (NUO)	I	253
6.2.6.2	Number of SMB & NetBIOS Evasions (NSNE)	I	255
6.2.6.3	Number of HTML Obfuscation (NHO)	I	257
6.2.6.4	Number of Payload Encoding (NPE)	I	259
6.2.6.5	Number of FTP Evasion (NFE)	I	261
6.2.6.6	Number of Layered Evasion (NLE)	I	263
6.2.7	Mean Time To Recovery From Shutdown IDPS (MTTRFSI)	I	265
7	**Application Security Policies**		
7.1	**Application Security Management**		
7.1.1	Percentage of Critical Applications (PCA)	S	267
7.1.2	Risk Assessment Coverage (RAC)	S	268
7.1.3	Security Testing Coverage (STC)	S	269
7.1.4	Number of Applications/Service/VM (NASV)	S	271
7.2	**Security Budget Management**		
7.2.1	Information Security Budget as % of IT Budget (SBPITB)	S	273
7.2.2	Information Security Budget Allocation (SBBA)	S	275
7.3	**Application Security**		
7.3.1	Current Anti-Malware Coverage for the Application (CAMCA)	S	277
7.3.2	Number of Anti-Malware (NAM)	S	279
7.4	**Backup Management**		
7.4.1	Mean Time Between Backup Process (MTBBP)	S	281
7.4.2	Number of Backup Processes that have Failed (NBPF)	S	283
7.4.3	Percent of Backup Processes that have Failed (PBPF)	S	285
7.5	**Database - MySql or PostgreSQL**		
7.5.1	Number of Default User Service Account (NDUSA)	S	286
7.5.2	Number of Insecure User Account (NIUA)	S	288
7.5.3	Number of Default TCP Port (NDTP)	S	290
7.5.4	Number of SQL Injection (NSI)	S	292
8	**Prevention initiative by assessing the incidents of data loss**		
8.1	**Sensitive or controlled data leave through e-mail**	S	
8.1.1	E-mail-based data loss events, overall and by data type; ratio of data loss events of all types between corporate divisions (EBDLE)	S	294
8.2	**Differences in the type of data lost**	S	
8.2.1	Chi-square test for types of data loss by corporate division (CTDL)	S	296
9	**Distribution of perimeter security events**		

Continued on next page

Table A.1 – *Continued from previous page*

Id	Description	Class	Page
9.1	**Breakdown of perimeter-related security events**	S	
9.1.1	Perimeter security events by datacenter (PSEBD)	S	298
9.1.2	Ratio of perimeter security events between datacenters (RPSEBD)	S	300
9.2	**Difference threats**	S	
9.2.1	Analysis of variance between reported datacenter perimeter event data (AVBRDPED)	S	302

Id: 1.1.1 - Metric Name: Number of Reported Security Policy Violations in the previous 12 months (NRSPV)

- **Objective**: measures the organization's relative exposure to number of reported security policy violations in the previous 12 months.
- **Description**: measures the number of reported security policy violations in the previous period.
- **Question**: what is the number of reported security policy violations in the previous 12 months?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: NRSPV is calculated by counting those numbers of reported security policy violations in the previous period: 12 month's.

$$\text{NRSPV} = \text{Count(objects)}$$

- **Units**: numbers of objects
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NRSPV values should trend higher over time. It would be ideal to have known to the number of reported security policy violations in the previous 12 months.
- **Sources**: security management systems will provide information on which systems were identified with security policy violations.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: $\text{Count(objects)} < 0$.
- **References**: [Hayden, 2010].

Id: 1.1.2 - Metric Name: Number of Enforcement Actions Taken Against Policy Violations in the previous 12 months (NESTSPV)

- **Objective**: measures the organization's relative exposure to number of enforcement actions taken against policy violations in the previous 12 months (NESTSPV).
- **Description**: measures the number of enforcement actions taken against policy violations in the previous 12 months.
- **Question**: what is the number of enforcement actions taken against policy violations in the previous 12 months?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: NESTSPV is calculated by counting those numbers of enforcement actions taken against policy violations in the previous 12 month's.

$$\text{NESTSPV} = \text{Count(objects)}$$

- **Units**: numbers of objects
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NESTSPV values should trend higher over time. It would be ideal to have known to the numbers of enforcement actions taken against policy violations in the previous 12 months.
- **Sources**: security management systems will provide information on which systems were identified with actions taken against policy violations.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(objects) < 0.
- **References**: [Hayden, 2010].

Id: 1.2.1 - Metric Name: Number of Documents that Make Up the corporate Security Policy (NDMUSP)

- **Objective**: measures the organization's relative exposure to number of documents that make up the corporate security policy (NDMUSP).
- **Description**: measures the number of documents that make up the corporate security policy.
- **Question**: what is the number of documents that make up the corporate security policy?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: NDMUSP is calculated by counting those documents that make up the corporate security policy.

$$\text{NESTSPV} = \text{Count(objects)}$$

- **Units**: numbers of objects.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NDMUSP values should trend higher over time. It would be ideal to have known to the numbers of enforcement actions taken against policy violations in the previous 12 months.
- **Sources**: security management systems will provide information on which systems were identified that make up the corporate security policy.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(objects) < 0.
- **References**: [Hayden, 2010].

Id: 1.2.2 - Metric Name: Format(s) of Security Policy Documents (FSPD)

- **Objective**: measures the organization's relative exposure to number of format(s) of security policy documents (FSPD).
- **Description**: measures the number of format(s) of security policy documents.
- **Question**: what is the number of format(s) of security policy documents?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: FSPD is calculated by counting number of format(s) of security policy documents.

$$\text{NESTSPV} = \text{Count(objects)}$$

- **Units**: numbers of formats.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: FSPD values should trend higher over time. It would be ideal to have known to the number of format(s) of security policy documents.
- **Sources**: security management systems will provide information on which systems were identified with format(s) of security policy documents.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: $\text{Count(objects)} < 0$.
- **References**: [Hayden, 2010].

Id: 1.2.3 - Metric Name: Location(s) of security policy documents (LSPD)

- **Objective**: measures the organization's relative exposure to number of location(s) of security policy documents (content management system, static web page, three-ring binder).
- **Description**: measures the number of location(s) of security policy documents.
- **Question**: what is the number of location(s) of security policy documents?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: LSPD is calculated by counting number of location(s) of security policy documents.

$$\text{LSPD} = \text{Count(objects)}$$

- **Units**: numbers of formats.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: LSPD values should trend higher over time. It would be ideal to have known to the number of location(s) of security policy documents.
- **Sources**: security management systems will provide information on which systems were identified with location(s) of security policy documents.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(objects) < 0.
- **References**: [Hayden, 2010].

Id: 1.2.4 - Metric Name: Types of Policy Acknowledgement Mechanisms (TPAM)

- **Objective**: measures the organization's relative exposure to number of types of policy acknowledgement mechanisms (e-mail notification of users, electronic acknowledgement of policy access or review, hard copy signoff sheet).
- **Description**: measures the number of types of policy acknowledgement mechanisms.
- **Question**: what is the number of types of policy acknowledgement mechanisms?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: TPAM is calculated by counting number of types of policy acknowledgement mechanisms.

$$\text{TPAM} = \text{Count(objects)}$$

- **Units**: numbers of formats.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: TPAM values should trend higher over time. It would be ideal to have known to the number of types of policy acknowledgement mechanisms.
- **Sources**: security management systems will provide information on which systems were identified with types of policy acknowledgement mechanisms.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(objects) < 0.
- **References**: [Hayden, 2010].

Id: 1.2.5 - Metric Name: Length of Time Since the Last Security Policy Review By Management (LTSLSPRBM)

- **Objective**: measures the organization's relative exposure to length of time since the last security policy review by management.
- **Description**: measures the length of time since the last security policy review by management.
- **Question**: what is the Length of time since the last security policy review by management?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: LSPD is calculated by the subtracting the last Date of Review from the Date of Review.

$$\text{LTSLSPRBM} = (\text{Date\_of\_Review}_{\text{n}} - \text{Date\_of\_Review}_{\text{n}-1})$$

- **Units**: numbers of hours, days, months, years.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: LSPD values should trend lower over time. It would be ideal to have known to the length of time since the last security policy review by management.
- **Sources**: security management systems will provide information on which systems were identified the length of time since the last security policy review by management.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(objects) < 0.
- **References**: [Hayden, 2010].

Id: 1.3.1 - Metric Name: Ratio of Employee Job Descriptions That Specify Responsibility for Following the Security Policy (REJDTSRFSP)

- **Objective**: measures the organization's relative exposure to ratio of employee job descriptions that specify responsibility for following the security policy.
- **Description**: measures the ratio of employee job descriptions that specify responsibility for following the security policy.
- **Question**: what is the ratio of employee job descriptions that specify responsibility for following the security policy?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: REJDTSRFSP is calculated by dividing the number of employee job descriptions that specify responsibility for following the security policy by Jobs Total.

$$\text{REJDTSRFSP} = \frac{\text{Count(Jobs\_Descriptions)}}{\text{Count(Jobs\_Total)}} * 100$$

- **Units**: ratio of employee job descriptions that specify responsibility for following the security policy.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: REJDTSRFSP values should trend higher over time. It would be ideal to have known to the ratio of employee job descriptions that specify responsibility for following the security policy.
- **Sources**: security management systems will provide information on which systems were identified the ratio of employee job descriptions that specify responsibility for following the security policy.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(objects) < 0.
- **References**: [Silva et al., 2012].

Id: 1.3.2 - Metric Name: Number of Security Policy Awareness or Training Activities Conducted in the Previous 12 months (NSPATACP)

- **Objective**: measures the organization's relative exposure to number of security policy awareness or training activities conducted in the previous period.
- **Description**: measures the number of security policy awareness or training activities conducted in the previous period.
- **Question**: what is the number of security policy awareness or training activities conducted in the previous 12 months?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: NSPATACP is calculated by counting number of security policy awareness or training activities conducted in the previous period.

$$\text{NSPATACP} = \text{Count}(\text{Activities})$$

- **Units**: number of security activities.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSPATACP values should trend higher over time. It would be ideal to have known to the number of security policy awareness or training activities conducted in the previous period.
- **Sources**: security management systems will provide information on which systems were identified the number of security policy awareness or training activities conducted in the previous 12 months.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: $\text{Count}(\text{objects}) < 0$.
- **References**: [Hayden, 2010].

Id: 1.3.3 - Metric Name: Ratio of Employees Who Have Formally Acknowledged the Security Policy in the previous 12 months (REWHFASP)

- **Objective**: measures the organization's relative exposure to ratio of employees who have formally acknowledged the security policy in the previous period.
- **Description**: measures the ratio of employees who have formally acknowledged the security policy in the previous period.
- **Question**: what is the ratio of employees who have formally acknowledged the security policy in the previous 12 months?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: REWHFASP is calculated by dividing the number of employees acknowledged the security policy in the previous 12 months by Employees Total.

$$\text{REWHFASP} = \frac{\text{Count(Employees\_Acknowledged)}}{\text{Count(Employees\_Total)}} * 100$$

- **Units**: ratio of employees who have formally acknowledged the security policy in the previous period.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: REWHFASP values should trend higher over time. It would be ideal to have known to the number of security policy awareness or training activities conducted in the previous period.
- **Sources**: security management systems will provide information on which systems were identified the ratio of employees who have formally acknowledged the security policy in the previous 12 months.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(Employees_Acknowledged) ≤ 0.
- **References**: [Hayden, 2010].

Id: 1.3.4 - Metric Name: Results of a user survey asking How Familiar Users Are with the Policy is judged to be (HFUAP)

- **Objective**: measures the organization's relative exposure to results of a user survey asking how familiar users are with the policy is judged to be.
- **Description**: measures the results of a user survey asking how familiar users are with the policy is judged to be.
- **Question**: what is the results of a user survey asking how familiar users are with the policy is judged to be?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: HFUAP is calculated by dividing the number of employees survey asking how familiar users are with the policy is judged to be by Employees Total.

$$\text{REWHFASP} = \frac{\text{Count(Employees\_Acknowledged)}}{\text{Count(Employees\_Total)}} * 100$$

- **Units**: results of how familiar users are with the policy is judged to be.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: HFUAP values should trend higher over time. It would be ideal to have known to the number of user survey asking how familiar users are with the policy is judged to be.
- **Sources**: security management systems will provide information on which systems were identified the results of a user survey asking how familiar users are with the policy is judged to be
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(Employees_Acknowledged) ≤ 0.
- **References**: [Silva et al., 2012].

Id: 1.3.5 - Metric Name: Results of a user survey asking How Appropriate and Usable the Policy is judged to be (HAUP)

- **Objective**: measures the organization's relative exposure to results of a user survey asking how appropriate and usable the policy is judged to be.
- **Description**: measures the results of a user survey asking how appropriate and usable the policy is judged to be.
- **Question**: what is the results of a user survey asking how appropriate and usable the policy is judged to be?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: HAUP is calculated by dividing the number of employees survey asking how appropriate and usable the policy is judged to be by Employees Total.

$$\text{REWHFASP} = \frac{\text{Count(Employees\_Acknowledged)}}{\text{Count(Employees\_Total)}} * 100$$

- **Units**: ratio of how appropriate and usable the policy is judged to be.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: HAUP values should trend higher over time. It would be ideal to have known to the results of a user survey asking how appropriate and usable the policy is judged to be.
- **Sources**: security management systems will provide information on which systems were identified the results of a user survey asking how appropriate and usable the policy is judged to be.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(Employees_Acknowledged) ≤ 0.
- **References**: [Silva et al., 2012].

Id: 1.4.1 - Metric Name: Increase in Security Policy Enforcement Actions Over Baseline (ISPEAOB)

- **Objective**: measures the organization's relative exposure to increase in security policy enforcement actions over baseline (expressed as either a raw count or a percentage, as appropriate).
- **Description**: measures the results of increase in security policy enforcement actions over baseline.
- **Question**: what is the results of increase in security policy enforcement actions over baseline?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: ISPEAOB is calculated by adding the difference of the number of security policy enforcement actions and baseline to the n security requirements of the baseline.

$$\text{ISPEAOB} = \sum_{i=1}^{N}(\text{Actions}_i - \text{BaseLine}_i)$$

- **Units**: number of the security policy enforcement actions over baseline.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: ISPEAOB values should trend higher over time. It would be ideal to have known to the increase in security policy enforcement actions over baseline.
- **Sources**: security management systems will provide information on which systems were identified the increase in security policy enforcement actions over baseline.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: $\text{BaseLine}_i = 0$.
- **References**: [Hayden, 2010].

Id: 1.4.2 - Metric Name: Increase in awareness of corporate security policy (IACSP)

- **Objective**: measures the organization's relative exposure to increase in awareness of corporate security policy (number of awareness activities, number of user acknowledgements of the policy).
- **Description**: measures the results of increase in awareness of corporate security policy.
- **Question**: what is the results of increase in awareness of corporate security policy?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: IACSP is calculated by counting the to increase in awareness of corporate security policy.

$$\text{IACSP} = \text{Count(Actions)}$$

- **Units**: number of the increase in awareness of corporate security policy.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: IACSP values should trend higher over time. It would be ideal to have known to the increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the increase in awareness of corporate security policy.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: Count(Actions) ≤ 0.
- **References**: [Hayden, 2010].

Id: 1.4.3 - Metric Name: Increase in efficiency of the security policy process (IESPP)

- **Objective**: measures the organization's relative exposure to increase in efficiency of the security policy process (increased policy reviews, reduction in the number of locations).
- **Description**: measures the results of increase in awareness of corporate security policy.
- **Question**: what is the results of increase in awareness of corporate security policy?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: IESPP is calculated by counting the policy reviews.

$$\text{IESPP} = \text{Count}(\text{Reviews})$$

- **Units**: number of the policy reviews.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: IESPP values should trend higher over time. It would be ideal to have known to the increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the increase in efficiency of the security policy process.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators.
- **Limitations**: it can not be: $\text{Count}(\text{Actions}) \leq 0$.
- **References**: [Hayden, 2010].

Id: 1.4.4 - Metric Name: Improved response from surveyed users on policy familiarity and usability (IRSUPFU)

- **Objective**: measures the organization's relative exposure to increase in awareness of corporate security policy (number of awareness activities, number of user acknowledgements of the policy).
- **Description**: measures the results of increase in awareness of corporate security policy.
- **Question**: what is the results of increase in awareness of corporate security policy?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: IRSUPFU is calculated by counting the number of awareness activities.

$$\text{IRSUPFU} = \text{Count}(\text{Activities})$$

- **Units**: number of the awareness activities.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: IRSUPFU values should trend higher over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified that improved response from surveyed users on policy familiarity and usability.
- **Usage**: it is important in the decision-making process of the end user, but it is more important for industries and integrators. Without the proper data, a manufacturer's piece of equipment would be immediately disqualified.
- **Limitations**: it can not be: $\text{Count}(\text{Actions}) \leq 0$.
- **References**: [Hayden, 2010].

Id: 2.1.1 - Metric Name: Mean Time To Failure (MTTF)

- **Objective**: the goal of this project is to understand security impacts on system.
- **Description**: measures the availability by comparing security-related downtime to general availability from the perspective of the security team.
- **Question**: How often is the system down due to failure?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: FD is calculated by the sum of the operational periods divided by the number of observed failures. The formula is:

$$\text{MTTF} = \frac{\sum(\text{Time\_of\_Down\_time}_{\text{n}} - \text{Time\_of\_Up\_Time}_{\text{n}-1})}{\text{Count(Failures)}}$$

- **Units**: hours per failures.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: FD values should trend higher over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of equipment would be immediately disqualified.
- **Usage**: Mean Time To Failure (MTTF) is a reliability term used to provide the amount of failures per million hours for a product. This is the most common inquiry about a product's life span, and is important in the decision-making process of the end user. MTBF is more important for industries and integrators than for consumers. Most consumers are price driven and will not take MTBF into consideration, nor is the data often readily available. On the other hand, when equipment such as media converters or switches must be installed into mission critical applications, MTBF becomes very important. In addition, MTTF may be an expected line item in an RFQ (Request For Quote). Without the proper data, a manufacturer's piece of equipment would be immediately disqualified.
- **Limitations**: it can not be:
  $\text{Time\_of\_Down\_time}_{\text{n}} = \text{Time\_of\_Up\_Time}_{\text{n}-1}$.
- **References**: [Hayden, 2010].

Id: 2.1.2 - Metric Name: Mean Time Between Failures (MTBF)

- **Objective**: the goal of this project is to understand security impacts on system.
- **Description**: measures the availability by comparing security-related downtime to general availability from the perspective of the security team.
- **Question**: How often is the system down due to failure?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: MTBF is calculated by the sum of the operational periods divided by the number of observed failures. The formula is:

$$\text{MTBF} = \frac{\sum(\text{Date\_of\_Down\_time} - \text{Date\_of\_Up\_Time})}{\text{Count(Failures)}}$$

- **Units**: hours per failures.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTBF values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: Mean Time Between Failure (MTBF) is a reliability term used to provide the amount of failures per million hours for a product. This is the most common inquiry about a product's life span, and is important in the decision-making process of the end user. MTBF is more important for industries and integrators than for consumers. Most consumers are price driven and will not take MTBF into consideration, nor is the data often readily available. On the other hand, when equipment such as media converters or switches must be installed into mission critical applications, MTBF becomes very important. In addition, MTBF may be an expected line item in an RFQ (Request For Quote). Without the proper data, a manufacturer's piece of equipment would be immediately disqualified.
- **Limitations**: it can not be:
  Date_of_Down_time = Date_of_Up_Time.
- **References**: [Hayden, 2010].

Id: 2.1.3 - Metric Name: Mean Time To Recovery (MTTRc)

- **Objective**: is the average time that a device will take to recover from any failure. Up to whole systems which have to be repaired or replaced.
- **Description**: measures the availability by comparing security-related downtime to general availability from the perspective of the security team.
- **Question**: How often is the system down between recoveries?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: MTTR is calculated by the sum of the operational periods divided by the number of observed failures. The formula is:

$$\text{MTTRc} = \frac{\sum(\text{Date\_of\_Discovered\_Failure} - \text{Date\_of\_Return\_Operation})}{\text{Count(Failures)}}$$

- **Units**: hours per recovery.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTTR values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: Mean Time To Recovery (MTTRc) is the time needed to repair a failed hardware module. In an operational system, repair generally means replacing a failed hardware part. Thus, hardware MTTRc could be viewed as mean time to replace a failed hardware module. Taking too long to repair a product drives up the cost of the installation in the long run, due to down time until the new part arrives and the possible window of time required to schedule the installation. To avoid MTTRc, many companies purchase spare products so that a replacement can be installed quickly. Generally, however, customers will inquire about the turn-around time of repairing a product, and indirectly, that can fall into the MTTRc category.
- **Limitations**: it can not be:
  Date_of_Discovered_Failure = Date_of_Return_Operation.
- **References**: [Hayden, 2010].

Id: 2.1.4 - Metric Name: Mean Time To Repair (MTTRp)

- **Objective**: is a basic measure of the maintainability of repairable items. It represents the average time required to repair a failed component or device.
- **Description**: measures the availability by comparing security-related downtime to general availability from the perspective of the security team.
- **Question**: How often is the system down between repair?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: MTTRp is calculated by the sum of the operational periods divided by the number of observed failures. The formula is:

$$\text{MTTRp} = \frac{\sum(\text{Date\_of\_Return\_Repair} - \text{Date\_of\_Begin\_Operation})}{\text{Count(Failures)}}$$

- **Units**: hours per repair.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTTRp values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: Mean Time To Repair (MTTRp) is the time needed to repair a failed hardware module. In an operational system, repair generally means replacing a failed hardware part. Thus, hardware MTTRp could be viewed as mean time to replace a failed hardware module. Taking too long to repair a product drives up the cost of the installation in the long run, due to down time until the new part arrives and the possible window of time required to schedule the installation. To avoid MTTRp, many companies purchase spare products so that a replacement can be installed quickly. Generally, however, customers will inquire about the turn-around time of repairing a product, and indirectly, that can fall into the MTTRp category.
- **Limitations**: it can not be:
  Date_of_Return_Repair = Date_of_Begin_Operation.
- **References**: [Hayden, 2010].

Id: 2.1.5 - Metric Name: Mean To System Availability (MTSA)

- **Objective**: The goal of this project is to understand security impacts on system.
- **Description**: measures the availability by comparing security-related downtime to general availability from the perspective of the security team.
- **Question**: What is system availability?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: Once MTBF and MTTRp are known, the availability of the component can be calculated using the following formula:

$$\text{MTSA} = \frac{\text{MTBF}}{(\text{MTBF} + \text{MTTRp})} * 100$$

- **Units**: percent system availability.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTSA values should trend higher over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: Availability is the probability that a system will work as required when required during the period of a mission.
- **Limitations**: it can not be: MTBF = 0.
- **References**: [Hayden, 2010].

Id: 2.2.1 - Metric Name: Duration of the Preventive Maintenance (DPM)

- **Objective**: the goal of this project is to understand duration of the preventive maintenance on system.
- **Description**: measures the duration of the preventive maintenance from the perspective of the security team.
- **Question**: How often is the system down due to preventive maintenance?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: DPM is calculated by the sum of the operational periods divided by the number of observed failures. The formula is:

$$\text{DPM} = \frac{\sum(\text{Time\_of\_End\_Maintenance} - \text{Time\_of\_Begin\_Maintenance})}{\text{Count(Maintenances)}}$$

- **Units**: hours per maintenances.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: DPM values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: The primary goal of maintenance is to avoid or mitigate the consequences of failure of equipment. This may be by preventing the failure before it actually occurs which Planned Maintenance and Condition Based Maintenance help to achieve. It is designed to preserve and restore equipment reliability by replacing worn components before they actually fail. Preventive maintenance activities include partial or complete overhauls at specified periods, oil changes, lubrication, minor adjustments, and so on. In addition, workers can record equipment deterioration so they know to replace or repair worn parts before they cause system failure.
- **Limitations**: it can not be:
  Time_of_End_Maintenance = Time_of_Begin_Maintenance.
- **References**: [Hayden, 2010].

Id: 2.2.2 - Metric Name: Duration of the Corrective Maintenance (DCM)

- **Objective**: the goal of this project is to understand duration of the corrective maintenance on system.
- **Description**: measures the duration of the preventive maintenance from the perspective of the security team.
- **Question**: How often is the system down due to corrective maintenance?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: DCM is calculated by the sum of the operational periods divided by the number of observed maintenances. The formula is:

$$\mathrm{DCM} = \frac{\sum(\mathrm{Time\_of\_End\_Maintenance} - \mathrm{Time\_of\_Begin\_Maintenance})}{\mathrm{Count(Maintenances)}}$$

- **Units**: hours per maintenances.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: DCM values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: The primary goal of corrective maintenance is to detect, isolate, and rectify a fault so that the failed equipment, machine, or system can be restored to its normal operable state.
- **Limitations**: it can not be:
  Time_of_End_Maintenance = Time_of_Begin_Maintenance.
- **References**: [Hayden, 2010].

Id: 2.2.2 - Metric Name: Mean Time Between Maintenance (MTBM)

- **Objective**: the goal of this project is to understand duration of the corrective and preventive maintenances on system.
- **Description**: measures the duration of the corrective and preventive maintenances from the perspective of the security team.
- **Question**: How often is the system down due to maintenance?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: MTBM is calculated by the sum of the operational periods divided by the number of observed maintenances. The formula is:

$$\text{MTBM} = \frac{\sum(\text{Time\_of\_End\_Maintenance} - \text{Time\_of\_Begin\_Maintenance})}{\text{Count(Maintenances)}}$$

- **Units**: hours per maintenances.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTBM values should trend higher over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: The primary goal of corrective and preventive maintenance is to detect, isolate, and rectify a fault so that the failed equipment, machine, or system can be restored to its normal operable state.
- **Limitations**: it can not be:
  Time_of_End_Maintenance = Time_of_Begin_Maintenance.
- **References**: [Hayden, 2010].

Id: 2.3.1 - Metric Name: Number of Security Events in a time period, duration of event remediation (NSEER)

- **Objective**: measures the organization's relative exposure to number of security events in a time period, duration of event remediation.
- **Description**: measures the results to number of security events in a time period, duration of event remediation.
- **Question**: what is the number of security events in a time period, duration of event remediation?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: NSEER is calculated by counting the number of security events in a time period, duration of event remediation.

$$\text{NSEER} = \text{Count}(\text{Activities})$$

- **Units**: number of security events.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSEER values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: $\text{Count}(\text{Activities}) \leq 0$.
- **References**: [Silva et al., 2012].

Id: 3.1.1 - Metric Name: Total number of registered hosts (TRH)

- **Objective**: measures the organization's relative exposure to number of registered hosts in a time period.
- **Description**: measures the results to number of registered hosts in a time period.
- **Question**: what is the number of registered hosts in a time period?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: TRH is calculated by counting the number of registered hosts in a time period.

$$\text{TRH} = \text{Count}(\text{Registered\_hosts})$$

- **Units**: number of registered hosts.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: TRH values should trend higher over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: $\text{Count}(\text{Registered\_hosts}) \leq 0$.
- **References**: [Hayden, 2010].

Id: 3.1.2 - Metric Name: Total number of unregistered hosts (TUH)

- **Objective**: measures the organization's relative exposure to number of unregistered hosts in a time period.
- **Description**: measures the results to number of unregistered hosts in a time period.
- **Question**: what is the number of unregistered hosts in a time period?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: TUH is calculated by counting the number of unregistered hosts in a time period.

$$\text{TUH} = \text{Count}(\text{Unregistered\_hosts})$$

- **Units**: number of unregistered hosts.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: TUH values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: $\text{Count}(\text{Unregistered\_hosts}) \leq 0$.
- **References**: [Hayden, 2010].

Id: 3.1.3 - Metric Name: Total number of registered hosts vulnerable (TRHV)

- **Objective**: measures the organization's relative exposure to number of registered hosts vulnerable in a time period.
- **Description**: measures the results to number of registered hosts vulnerable in a time period.
- **Question**: what is the number of registered hosts vulnerable in a time period?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: TRHV is calculated by counting the number of registered hosts vulnerable in a time period.

$$\text{TRHV} = \text{Count}(\text{Registered\_hosts\_vulnerable})$$

- **Units**: number of registered hosts vulnerable.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: TRHV values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: $\text{Count}(\text{Registered\_hosts\_vulnerable}) \leq 0$.
- **References**: [Hayden, 2010].

Id: 3.1.4 - Metric Name: Total number of unregistered hosts vulnerable (TUHV)

- **Objective**: measures the organization's relative exposure to number of unregistered hosts vulnerable in a time period.
- **Description**: measures the results to number of unregistered hosts vulnerable in a time period.
- **Question**: what is the number of unregistered hosts vulnerable in a time period?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: TUHV is calculated by counting the number of unregistered hosts vulnerable in a time period.

$$\text{TUHV} = \text{Count}(\text{Unregistered\_hosts\_vulnerable})$$

- **Units**: number of unregistered hosts vulnerable.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: TUHV values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: $\text{Count}(\text{Unregistered\_hosts\_vulnerable}) \leq 0$.
- **References**: [Hayden, 2010].

Id: 3.1.5 - Metric Name: Percentage of registered hosts vulnerable (PRHV)

- **Objective**: measures the organization's relative exposure to percentage of registered hosts vulnerable in a time period.
- **Description**: measures the results to percentage of registered hosts vulnerable in a time period.
- **Question**: what is the percentage of registered hosts vulnerable in a time period?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: PRHV is calculated by the sum of the registered hosts vulnerable divided by the number of Hosts. The formula is:

$$\text{PRHV} = \frac{\text{Count(Registered\_hosts\_vulnerable)}}{\text{Count(Hosts)}} * 100$$

- **Units**: percentage of registered hosts vulnerable.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: PRHV values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: Count(Registered_hosts_vulnerable) ≤ 0.
- **References**: [Hayden, 2010].

Id: 3.1.6 - Metric Name: Percentage of unregistered hosts vulnerable (PUHV)

- **Objective**: measures the organization's relative exposure to percentage of unregistered hosts vulnerable in a time period.
- **Description**: measures the results to percentage of unregistered hosts vulnerable in a time period.
- **Question**: what is the percentage of unregistered hosts vulnerable in a time period?
- **Answer**: a positive integer value that is greater than or equal to zero.
- **Formula**: PUHV is calculated by the sum of the unregistered hosts vulnerable divided by the number of Hosts. The formula is:

$$\text{PUHV} = \frac{\text{Count(Unregistered\_hosts\_vulnerable)}}{\text{Count(Hosts)}} * 100$$

- **Units**: percentage of unregistered hosts vulnerable.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: PUHV values should trend lower over time. It would be ideal to have known to increase in awareness of corporate security policy.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: security management systems will provide information on which systems were identified the security impacts on system. Without the proper data, a manufacturer's piece of software/hardware would be immediately disqualified.
- **Limitations**: it can not be: Count(Unregistered_hosts_vulnerable) ≤ 0.
- **References**: [Hayden, 2010].

Id: 3.2.1 - Metric Name: Mean Cost To Mitigate Vulnerabilities (MCTMV)

- **Objective**: this defines a metric for measuring the mean effort required to mitigate an identified vulnerability that can be remedied. The metric is expressed in the context of a vulnerability management process, with the assumption that the organizations is scanning for known vulnerabilities, a formal system (i.e. change management and electronic ticketing system) is used to track activities to mitigate known vulnerabilities, and there is a known remedy for the vulnerability. The metric is useful where a single vulnerability or remedy (no matter how many systems are affected) is expressed as a single change ticket or as one change ticket per affected network node.

- **Description**: The goal of this metric is to understand the effort required for vulnerability remediation activities. Risk management decisions can take into account the efficiency of vulnerability remediation and make more informed decisions around vulnerability policies, SLAs, and resource allocation in the IT environment.

- **Question**: what is the average (mean) cost to the organization to mitigate an identified vulnerability during the given period?

- **Answer**: A positive integer value that is greater than or equal to zero. A value of "0.0" indicates there were no measured costs to the organization.

- **Formula**: this metric is calculated by summing the total cost to mitigate each vulnerability and dividing it by the total number of mitigated vulnerabilities. This count should also be done for each severity value (Low, Medium, and High):

$$\text{MCTMV} = \frac{\sum(\text{Person\_Hours\_to\_Mitigate} * \text{Hourly\_Rate}) + \text{Other\_Mitigation\_Costs}}{\text{Count}(\text{Mitigated\_Vulnerabilities})}$$

- **Units**: hours per vulnerability

- **Frequency**: monthly, annually.

- **Targets**: ideally, all vulnerabilities would be remedied by a automated vulnerability remediation system, and the mean cost to remediate would be zero. In practice a target can be set based on the expected loss budget determined by risk assessments processes.

- **Sources**: vulnerability tracking systems will provide vulnerability data. Cost data can come from management estimates, ticket tracking systems, and capital and services budgets.

- **Usage**: Mean-Time To Mitigate Vulnerabilities is a type of vulnerability management metric and relies on the common definition of "vulnerability" as defined in the Glossary. Due to the number of vulnerabilities and exposures found by most scanning tools, this metric should generally be calculated for "High" and "Medium" severity vulnerabilities. Combined with the number of identified vulnerabilities this metric can provide visibility into the total cost and effort required to remediate and

manage the known vulnerabilities in the organization. Optimal conditions would reflect a low value in the metric. The lower the value the more efficient and cheaply the organization is able to mitigate identified vulnerabilities. There may be a direct correlation between the number of un-mitigated vulnerabilities and the number of security incidents. Since vulnerabilities may not be addressed due to cost concerns, an organization with a lower average remediation cost may be able to mitigate more vulnerabilities.

- **Limitations**: Note that this assumes:
  - Effort is tracked for vulnerability remediation;
  - Tickets are closed when the change is known to have mitigated the vulnerability;
  - Vulnerabilities can be tracked between scans on a vulnerability instance per-host basis;
  - It is not including in-progress tickets, vulnerabilities that have not been mitigated, or vulnerabilities that do not have a resolution.
- **References**: [ISO/IEC-27002:2005, 2005, NIST, 2013c].

Id: 3.2.2 - Metric Name: Mean Time To Mitigate Vulnerabilities (MTTMV)

- **Objective**: Mean-Time To Mitigate Vulnerabilities (MTTMV) measures the average amount of time required to mitigate an identified vulnerability. This metric indicates the performance of the organization in reacting to vulnerabilities identified in the environment. It only measures the time average times for explicitly mitigated vulnerabilities, and not mean time to mitigate any vulnerability, or account for vulnerabilities that no longer appear in scanning activities.

- **Description**: Mean-Time To Mitigate Vulnerabilities measures the average time taken to mitigate vulnerabilities identified in an organization's technologies. The vulnerability management processes consists of the identification and remediation of known vulnerabilities in an organization's environment. This metric is an indicator of the performance of the organization in addressing identified vulnerabilities. The less time required to mitigate a vulnerability the more likely an organization can react effectively to reduce the risk of exploitation of vulnerabilities. It is important to not that only data from vulnerabilities explicitly mitigated are is included in this metric result. The metric result is the mean time to mitigate vulnerabilities that are actively addressed during the metric time period, and not a mean time to mitigate based on the time for all known vulnerabilities to be mitigated.

- **Question**: how long does it take the organization to mitigate a vulnerability?

- **Answer**: a positive floating-point value that is greater than or equal to zero. A value of "0" indicates that vulnerabilities were instantaneously mitigated.

- **Formula**: Mean-Time To Mitigate Vulnerabilities is calculated by determining the number of hours between the date of detection and the Date of Mitigation for each identified vulnerability instance in the current scope, for example, by time period, severity or business unit. These results are then averaged across the number of mitigated vulnerabilities in the current scope:

$$\text{MTTMV} = \frac{\sum(\text{Date\_of\_Mitigation} - \text{Date\_of\_Detection})}{\text{Count}(\text{Mitigated\_Vulnerabilities})}$$

- **Units**: hours per vulnerability

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTTMV values should trend lower over time. Lower levels of MTTMV are preferred. Most organizations put mitigation plans through test and approval cycles prior to implementation. Generally, the target time for MTTMV will be a function of the severity of the vulnerability and business criticality of the technology. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time to Mitigate Vulnerabilities exists.

- **Sources**: vulnerability management systems will provide information on which systems were identified with severe vulnerabilities.

- **Usage**: Mean-Time To Mitigate Vulnerabilities is a type of vulnerability management metric and relies on the common definition of "vulnerability" as defined in the Glossary. Due to the number of vulnerabilities and exposures found by most scanning tools, this metric should generally be calculated for "High" and "Medium" severity vulnerabilities. Combined with the number of identified vulnerabilities this metric can provide visibility into the time and effort required to manage the known vulnerabilities in the organization. Optimal conditions would reflect a low value in the metric. The lower the value the more quickly the organization is able to react to and mitigate identified vulnerabilities. Since many attacks are designed to exploit known vulnerabilities there may be a direct correlation between a lower time to mitigate vulnerabilities and the number of security incidents. MTTV can be calculated over time, typically per-month. To gain insight into the relative performance and risk , this metric can be calculated for vulnerabilities with differing severity levels, as well as calculated for cross-sections of the organization such as individual business units or geographies.

- **Limitations**: only data from mitigated vulnerabilities are included in this calculation. Therefore it is an indicator of the organization's ability to mitigate vulnerabilities as they are identified, but not necessarily a true representation of the average time taken to mitigate all vulnerabilities that may exist in the organization's environment. Other indicators of the scale of scope of unmitigated vulnerabilities should also be used to assess the performance of the vulnerability management function. Mitigation effort can vary depending on the scope and depth of the mitigation solution, modification of firewall rules or other changes to the environment may be less effort than directly addressing vulnerabilities in an application's code. It is possible that the vulnerabilities that are easier to mitigate are the ones completed in the metric scope, and the remaining vulnerabilities represent the most challenging to mitigate. Therefore the metric result could be biased low compared the to mean time to mitigate remaining known vulnerabilities.

- **References**: [ISO/IEC-27002:2005, 2005, NIST, 2013c].

Id: 3.2.3 - Metric Name: Number of Known Vulnerability Instances (NKVI)

- **Objective**: Number of Known Vulnerability Instances (NKVI) measures the total number of instances of known vulnerabilities within an organization among scanned assets based on the scanning process at a point in time.
- **Description**: Number of Known Vulnerability Instances (NKVI) measures the number of known vulnerabilities that have been found on organization's systems during the vulnerability identification process.
- **Question**: how many open vulnerability instances were found during the scanning process?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates that no instances of known vulnerabilities were found.
- **Formula**: this metric is calculated by counting the number of open vulnerability instances identified. This count should also be done for each severity value (Low, Medium, and High):

$$\text{NKVI} = \text{Count}(\text{Vulnerability.status} = \text{Open})$$

- **Units**: number of vulnerabilities
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NKVI values should trend lower over time. In the ideal case, there would be no known vulnerability instances on any technologies in the organization. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Number of Known Vulnerability Instances exists.
- **Sources**: vulnerability management systems will provide information on which systems were identified with severe vulnerabilities.
- **Usage**: by understanding the number of instances of known exploitable vulnerabilities, the organization can assess relative risk levels across the organization of time, estimate and management remediation efforts, and correlate and predict the volume of security incidents. The vulnerability scanning process can consist of a number of vulnerability scanning activities occurring over a set time period in cases where multiple scans are necessary to cover all of an organization's technologies or potential vulnerability types. This metric should be used in conjunction with other vulnerability metrics to provide context around the magnitude of known vulnerabilities in an organization. Since other metrics are expressed as ratios, this metric quantifies the volume of known vulnerabilities the organization is managing. Combined with the mean time to mitigate vulnerabilities this metric can provide visibility into the time and effort required to manage the known vulnerabilities in the organization. When comparing performance over time and between organizations, this metric can be normalized across the total number of systems. This and additional vulnerability metrics are an area noted for further development by the CIS metrics community.

- **Limitations**: the vulnerability scans may not be comprehensive, instead only attempting to identify a subset of potential vulnerabilities. Different scanning sessions and products can be checking for different numbers and types of vulnerabilities, some may consist of thousands of checks for vulnerabilities, while other products or sessions may only check for hundreds of known vulnerabilities. The scope of the scanning effort may not be complete and may also not be representative of the organizations overall systems. Those systems out of scope may potentially be areas of risk. In some cases key servers or production systems may be excluded from scanning activities. This metric only reports on known vulnerabilities. This does not mean that there are no "unknown" vulnerabilities. Severe vulnerabilities that the organization is unaware of can exist, and potentially be exploited, for years before any public disclosure may occur. When reporting a total number of vulnerabilities, severe vulnerabilities are considered equal to informational vulnerabilities. Reporting this metric by the dimension of Vulnerability Severity will provide more actionable information.

- **References**: [ISO/IEC-27002:2005, 2005, NIST, 2013c].

Id: 3.2.4 - Metric Name: Percent of Systems Without Known Severe Vulnerabilities (PSWKSV)

- **Objective**: Percent of Systems Without Known Severe Vulnerabilities (PSWKSV) measures the organization's relative exposure to known severe vulnerabilities. The metric evaluates the percentage of systems scanned that do not have any known high severity vulnerabilities.

- **Description**: Percent of Systems Without Known Severe Vulnerabilities (PSWKSV) measures the percentage of systems that when checked were not found to have any known high severity vulnerabilities during a vulnerability scan. Vulnerabilities are defined as "High" severity if they have a CVSS base score of [7.0-10.0]. Since vulnerability management involves both the identification of new severe vulnerabilities and the remediation of known severe vulnerabilities, the percentage of systems without known severe vulnerabilities will vary over time. Organizations can use this metric to gauge their relative level of exposure to exploits and serves as a potential indicator of expected levels of security incidents (and therefore impacts on the organization). This severity threshold is important, as there are numerous informational, local, and exposure vulnerabilities that can be detected that are not necessarily material to the organization's risk profile. Managers generally will want to reduce the level of noise to focus on the greater risks first. This metric can also be calculated for subsets of systems, such as by asset criticality of business unit.

- **Question**: of the systems scanned, what percentage does not have any known severe vulnerabilities?

- **Answer**: a positive integer value that is greater than or equal to zero. A value of "100%" indicates that none of the organization's systems have any known high severity vulnerabilities.

- **Formula**: Percent of Systems Without Known Severe Vulnerabilities is calculated by counting those systems that have no open high severity level vulnerabilities (Vulnerability Status != "Open" & CVSS Base Score >= 7.0). This result is then divided by the total number of systems in the scanning scope.

$$\text{PSWKSV} = \frac{\text{Count(Systems\_Without\_Known\_Severe\_Vulnerabilities)}}{\text{Count(Scanned\_Systems)}} * 100$$

- **Units**: percentage of systems

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: PSWKSV values should trend lower over time. It would be ideal to have no known severe vulnerabilities on systems; therefore, an ideal target value would be 100%. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Percent of Systems Without Known Severe Vulnerabilities exists.

- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.

- **Usage**: Percent of Systems Without Known Severe Vulnerabilities is a type of vulnerability management metric and relies on the common definition of "vulnerability" as defined in the Glossary. Due to the number of vulnerabilities and exposures found by most scanning tools, this metric should be calculated for "High" severity vulnerabilities. Optimal conditions would reflect a high value in the metric. A value of 100% would indicate that none of the organizations systems are known to possess severe vulnerabilities. The lower the value, the greater the risk that systems are exploited. Since many attacks are designed to exploit known severe vulnerabilities there may be a direct correlation between a higher percentage of vulnerable systems and the number of security incidents. Percent of Systems Without Known Severe Vulnerabilities can be calculated over time, typically per-week or permonth. To gain insight into the relative performance and risk to one business unit over another, the metric may also be calculated for cross-sections of the organization such as individual business units or geographies.

- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops may be intermittently present for network scans. Systems not scanned, even if they possess severe vulnerabilities will not be included in this metric result. In addition, scanning activities can vary in depth, completeness, and capabilities. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.

- **References**: [Cichonski et al., 2012].

Id: 3.2.5 - Metric Name: Vulnerability Scan Coverage (VSC)

- **Objective**: Vulnerability Scan Coverage (VSC) indicates the scope of the organization's vulnerability identification process. Scanning of systems known to be under the organization's control provides the organization the ability to identify open known vulnerabilities on their systems. Percentage of systems covered allows the organization to become aware of areas of exposure and proactively remediate vulnerabilities before they are exploited.

- **Description**: Vulnerability Scanning Coverage (VSC) measures the percentage of the organization's systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.

- **Question**: what percentage of the organization's total systems has been checked for known vulnerabilities?

- **Answer**: positive integer value that is greater than or equal to zero but less than or equal to 100%. A value of? 100%? indicates that all systems are covered by the vulnerability scanning process.

- **Formula**: Vulnerability Scanning Coverage is calculated by dividing the total number of systems scanned by the total number of systems within the metric scope such as the entire organization:

$$\text{VSC} = \frac{\text{Count(Scanned\_Systems)}}{\text{Count(All\_Systems\_Within\_Organization)}} * 100$$

- **Units**: percentage of systems

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: VSC values should trend higher over time. Higher values are obviously better as it means more systems have been checked for vulnerabilities. A value of 100% means that all the systems are checked in vulnerability scans. For technical and operational reasons, this number will likely be below the theoretical maximum.

- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.

- **Usage**: this metric provides information about how much of the organization's environment is checked for known vulnerabilities. Organizations can use this metric to evaluate their risk position in terms of concentrations of unknown vulnerability states of systems. In combination with other vulnerability metrics, it provides insight on the organization's exposure to known vulnerabilities. The results of the coverage metric indicate the:

  - Scope of the vulnerability scanning activities;
  - Applicability of other metric results across the organization;

  - Relative amount of information known about the organization's vulnerability.

- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops and guest systems may be intermittently present for network scans, resulting in variability of metric results. In addition, scanning activities can vary in depth, completeness, and capability. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.

- **References**: [ISO/IEC-27002:2005, 2005, NIST, 2013c].

Id: 3.3.1 - Metric Name: Vulnerabilities on the Internal Nodes (VIN)

- **Objective**: Vulnerabilities on the Internal Nodes indicates the scope of the organization's vulnerability identification process. Scanning of systems known to be under the organization's control provides the organization the ability to identify open known vulnerabilities on their systems.
- **Description**: Vulnerabilities on the Internal Nodes measures the percentage of the organization's systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.
- **Question**: how many are vulnerabilities on the Internal Nodes?
- **Answer**: A positive integer value that is greater than or equal to zero. A value of "0" indicates that no vulnerabilities on the Internal Nodes were found.
- **Formula**: Vulnerabilities on the Internal Nodes is calculated by counting the number of open vulnerability instances on the Internal Nodes identified:

$$\text{VIN} = \frac{\text{Count(Vulnerability.status=Open)}}{\text{Internal\_Nodes}} * 100$$

- **Units**: number of vulnerabilities
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: VSC values should trend higher over time. Higher values are obviously better as it means more systems have been checked for vulnerabilities. A value of 100% means that all the systems are checked in vulnerability scans. For technical and operational reasons, this number will likely be below the theoretical maximum.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: this metric provides information about how much of the organization's environment is checked for known vulnerabilities. Organizations can use this metric to evaluate their risk position in terms of concentrations of unknown vulnerability states of systems. In combination with other vulnerability metrics, it provides insight on the organization's exposure to known vulnerabilities. The results of the coverage metric indicate the:
  - Scope of the vulnerability scanning activities;
  - Applicability of other metric results across the organization;
  - Relative amount of information known about the organization's vulnerability.
- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops and guest systems may be intermittently present for network scans, resulting in variability of

metric results. In addition, scanning activities can vary in depth, completeness, and capability. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.

- **References**: [ISO/IEC-27002:2005, 2005, NIST, 2013c].

Id: 3.3.2 - Metric Name: Security vulnerability counts for assessed internal nodes (SVCAIN)

- **Objective**: Security vulnerability counts for assessed internal nodes indicates the scope of the organization's vulnerability identification process. Scanning of systems known to be under the organization's control provides the organization the ability to identify open known vulnerabilities on their systems.
- **Description**: Security vulnerability counts for assessed internal nodes from scanning measures the percentage of the organization's systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.
- **Question**: how many are Security vulnerability counts for assessed internal nodes from scanning?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates that no Security vulnerability counts for assessed internal nodes from scanning were found.
- **Formula**: Security vulnerability counts for assessed internal nodes is calculated by counting the number of open vulnerability instances on the internal nodes identified:

$$\text{SVCAIN} = \frac{\text{Count(Vulnerability.status=Open)}}{\text{Internal\_Nodes}} * 100$$

- **Units**: number of vulnerabilities
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: SVCAIN values should trend higher over time. Higher values are obviously better as it means more systems have been checked for vulnerabilities. A value of 100% means that all the systems are checked in vulnerability scans. For technical and operational reasons, this number will likely be below the theoretical maximum.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: this metric provides information about how much of the organization's environment is checked for known vulnerabilities. Organizations can use this metric to evaluate their risk position in terms of concentrations of unknown vulnerability states of systems. In combination with other vulnerability metrics, it provides insight on the organization's exposure to known vulnerabilities. The results of the coverage metric indicate the:
  - Scope of the vulnerability scanning activities;
  - Applicability of other metric results across the organization;
  - Relative amount of information known about the organization's vulnerability.

- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops and guest systems may be intermittently present for network scans, resulting in variability of metric results. In addition, scanning activities can vary in depth, completeness, and capability. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.
- **References**: [Hayden, 2010].

Id: 3.3.3 - Metric Name: Ratios of vulnerabilities by type, OS, owner, and so on (RVT)

- **Objective**: the goal of this project is to assess the remediation priorities for internal servers by identifying the presence of vulnerabilities on internal servers from the perspective of the server administrators.
- **Description**: ratios of vulnerabilities by type from scanning measures the percentage of the organization's systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.
- **Question**: how vulnerable are the internal servers?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates that no security vulnerability counts for assessed internal nodes from scanning were found.
- **Formula**: ratios of vulnerabilities by type is calculated by dividing the number of open vulnerability instances on the Internal Node by the total number of vulnerabilities identified:

$$\text{RVT} = \frac{\text{Count(Vulnerability.status=Open)}}{\text{Count(Vulnerabilities)}} * 100$$

- **Units**: ratios of vulnerabilities by type
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: RVT values should trend higher over time. Higher values are obviously better as it means more systems have been checked for vulnerabilities. A value of 100% means that all the systems are checked in vulnerability scans. For technical and operational reasons, this number will likely be below the theoretical maximum.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: this metric provides information about how much of the organization's environment is checked for known vulnerabilities. Organizations can use this metric to evaluate their risk position in terms of concentrations of unknown vulnerability states of systems. In combination with other vulnerability metrics, it provides insight on the organization's exposure to known vulnerabilities. The results of the coverage metric indicate the:
  - Scope of the vulnerability scanning activities;
  - Applicability of other metric results across the organization;
  - Relative amount of information known about the organization's vulnerability.

- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops and guest systems may be intermittently present for network scans, resulting in variability of metric results. In addition, scanning activities can vary in depth, completeness, and capability. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.
- **References**: [Hayden, 2010].

Id: 3.3.4 - Metric Name: Severe vulnerabilities found on the internal nodes (SVFIN)

- **Objective**: the goal of this project is to assess the remediation priorities for internal servers by identifying the presence and severity of vulnerabilities on internal servers from the perspective of the server administrators.
- **Description**: ratios of vulnerabilities by type from scanning measures the percentage of the organization's systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.
- **Question**: how severe vulnerable are the internal servers?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates that no severe security vulnerability counts for assessed internal nodes from scanning were found.
- **Formula**: ratios of vulnerabilities by type is calculated by dividing the number of open severe vulnerability instances on the Internal Node by the total number of vulnerabilities identified:

$$\text{SVFIN} = \frac{\text{Count(Severe\_\_Vulnerability.status=Open)}}{\text{Count(Vulnerabilities)}} * 100$$

- **Units**: ratios of severe vulnerabilities by type
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: SVFIN values should trend higher over time. Higher values are obviously better as it means more systems have been checked for vulnerabilities. A value of 100% means that all the systems are checked in vulnerability scans. For technical and operational reasons, this number will likely be below the theoretical maximum.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: this metric provides information about how much of the organization's environment is checked for known vulnerabilities. Organizations can use this metric to evaluate their risk position in terms of concentrations of unknown vulnerability states of systems. In combination with other vulnerability metrics, it provides insight on the organization's exposure to known vulnerabilities. The results of the coverage metric indicate the:
  - Scope of the vulnerability scanning activities;
  - Applicability of other metric results across the organization;
  - Relative amount of information known about the organization's vulnerability.

- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops and guest systems may be intermittently present for network scans, resulting in variability of metric results. In addition, scanning activities can vary in depth, completeness, and capability. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.
- **References**: [Silva et al., 2012].

Id: 3.3.5 - Metric Name: CVSS scores for all identified vulnerabilities present on internal nodes (SIVPIN)

- **Objective**: the goal of this project is to assess the remediation priorities for internal servers by identifying the presence and severity of vulnerabilities on internal servers from the perspective of the server administrators.
- **Description**: CVSS scores for all identified vulnerabilities present on internal nodes from scanning measures the percentage of the organization's systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.
- **Question**: how CVSS scores for all identified vulnerabilities present on internal nodes?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates that no CVSS scores for assessed internal nodes from scanning were found.
- **Formula**: CVSS scores for all identified vulnerabilities present on internal nodes is calculated by dividing the number of open CVSS scores vulnerability instances on the Internal Node by the total number of vulnerabilities identified:

$$\text{SIVPIN} = \frac{\text{Count(Severe\_\_Vulnerability.status=Open)}}{\text{Count(Vulnerabilities)}} * 100$$

- **Units**: ratios of severe vulnerabilities by type
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: SIVPIN values should trend higher over time. Higher values are obviously better as it means more systems have been checked for vulnerabilities. A value of 100% means that all the systems are checked in vulnerability scans. For technical and operational reasons, this number will likely be below the theoretical maximum.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: this metric provides information about how much of the organization's environment is checked for known vulnerabilities. Organizations can use this metric to evaluate their risk position in terms of concentrations of unknown vulnerability states of systems. In combination with other vulnerability metrics, it provides insight on the organization's exposure to known vulnerabilities. The results of the coverage metric indicate the:
  - Scope of the vulnerability scanning activities;
  - Applicability of other metric results across the organization;
  - Relative amount of information known about the organization's vulnerability.

- **Limitations**: due to technical or operational incompatibility certain systems may be excluded from scanning activities while other systems such as laptops and guest systems may be intermittently present for network scans, resulting in variability of metric results. In addition, scanning activities can vary in depth, completeness, and capability. This metric assumes that systems scanned for vulnerabilities are systems known to and under full management by the organization. These systems do not include partial or unknown systems. Future risk metrics may account for these to provide a clearer view of all system ranges.

- **References**: [Hayden, 2010].

Id: 4.1.1 - Metric Name: Cost of Incidents (COI)

- **Objective**: Organizations need to understand the impact of security incidents. Impact can take many forms from negative publicity to money directly stolen. Monetary costs provide a set of units that can be directly compared across impact of the incidents and across organizations. In order to make effective risk management decisions, the impact of incidents needs to be measured and considered. Understanding of the costs experienced by the organization can be used to improve security process effectiveness and efficiency.

- **Description**: Cost of Incidents (COI) measures the total cost to the organization from security incidents occurring during the metric time period. Total costs from security incidents consists of the following costs:

  - **Direct Loss (DL)**: value of IP, customer lists, trade secrets; or other assets that are destroyed;
  - **Cost of Business System Downtime (COBSD)**: cost of refunds for failed transactions; and cost of lost business directly attributable to the incident;
  - **Cost of Containment (COC)**: efforts and cost, and consulting services;
  - **Cost of Recovery (COR)**: cost of incident investigation and analysis; effort required to repair and replace systems; replacement cost of systems consulting services for repair or investigation; and additional costs not covered by an insurance policy.
  - **Cost of Restitution (COR)**: penalties and other funds paid out due to breaches of contacts or SLAs resulting from the incident; cost of services provided to customers as a direct result of the incident (e.g. ID Theft Insurance); public relations costs; cost of disclosures and notifications; and legal costs, fines, and settlements.

- **Question**: what is the total cost to the organization from security incidents during the given period?

- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0.0" indicates there were no measured costs to the organization.

- **Formula**: cost of Incidents (COI) is calculated by summing all costs associated with security incidents during the time period:

$$\text{COI} = \sum(\text{DL} + \text{COBSD} + \text{COC} + \text{COR} + \text{COR})$$

- **Units**: R$ per incident

- **Frequency**: monthly, annually.

- **Targets**: Ideally there would be no security incidents with material impacts on the organization, and the metric value would be zero. In practice a target can be set based on the expected loss budget determined by risk assessments processes.

- **Sources**: Incident tracking systems will provide incident data. Cost data can come from both management estimates, ticket tracking systems and capital and services budgets.

- **Usage**: Cost of Incidents (COI) represents the overall known outcome of security systems, processes, and policies. The lower the COI, the less the organization is impacted by security incidents. Optimal conditions would reflect a low value of COI. Costs experienced by organizations may vary as a result of the threat environment, controls in place, and resiliency of the organization. Over time as processes and controls become more effectiveness, COI should be reduced.

- **Limitations**: some incidents such as exposure of business strategy via social engineering may not have a direct incident costs. Significant harm, bad press, or competitive disadvantage may still be experienced for which it is not practical to assign a cost; some new controls may have significant costs and/or address recovery from multiple incidents; this metric relies on the common definition of "security incident" as defined in Terms and Definitions; this metric relies on an organization being able to produce costs or cost estimates related to security incidents.

- **Dimensions**: This metric may include additional dimensions for grouping and aggregation purposes. These dimensions should be applied or tagged at the level of the underlying incident record as described in Security Incident Metrics. For example: (i) priority dimension allows COI to be computed for high, medium, or low severity incidents; (ii) classifications for characterizing types of incidents, such as denial of service, theft of information, etc.; (iii) affected Organization for identifying the affected part of the organization.

- **References**: [CICSWG, 2000, Dorofee et al., 2007, Cichonski et al., 2012].

Id: 4.1.2 - Metric Name: Incident Handled (IH)

- **Objective**: Incident Handled (IH) indicates the number of detected security incidents the organization has handled during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Incident Handled (IH) measures the number of security incidents for a given time period.
- **Question**: what is the number of security incidents that handled during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: to calculate Incident Handled (IH), the number of security incidents are counted across a scope of incidents, for example a given time period, category or business unit:

$$\text{IH} = \text{Count}(\text{Incidents})$$

- **Units**: incidents per period; for example, incidents per week or incidents per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: for example, a growing number of incidents performed by insiders could prompt stronger policy provisions concerning background investigations for personnel and misuse of computing resources and stronger security controls on internal networks, e.g. deploying intrusion detection software to more internal networks and hosts.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of incidents Handled is a type of security incident metric and relies on the common definition of "security incident". Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents Handled might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Incident Handled metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment.

To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or geographies.

- **Limitations**: a security program may or may not have direct control over the number of incidents handled that occur within their environment. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur.
- **References**: [Cichonski et al., 2012, Killcrece et al., 2003].

Id: 4.1.3 - Metric Name: Incident Rate (IR)

- **Objective**: Incident Rate (IR) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Incident Rate (IR) measures the number of security incidents for a given time period.
- **Question**: what is the number of security incidents that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Incident Rate (IR) is calculated by dividing the number of current incident in period by the total number of incidents:

$$\text{IR} = \frac{\text{Count(Incidents\_Period)}}{\text{Count(Incidents)}} * 100$$

- **Units**: incidents per period; for example, incidents per week or incidents per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: IR values should trend lower over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for Incident Rate exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of incidents is a type of security incident metric and relies on the common definition of "security incident". Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Incident Rate metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or geographies.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID.
- **References**: [Killcrece et al., 2003, Cichonski et al., 2012].

Id: 4.1.4 - Metric Name: Mean Cost of Incidents (MCOI)

- **Objective**: Organizations need to understand the impact of security incidents. Impact can take many forms from negative publicity to money directly stolen. Monetary costs provide a set of units that can be directly compared across impact of the incidents and across organizations. In order to make effective risk management decisions, the impact of incidents needs to be measured and considered. Understanding of the costs experienced by the organization can be used to improve security process effectiveness and efficiency.

- **Description**: Mean Cost of Incidents (MCOI) measures the mean cost to the organization from security incidents identified relative to the number of incidents that occurred during the metric time period. Total costs from security incidents consists of the following costs:

  - **Direct Loss (DL)**: value of IP, customer lists, trade secrets; or other assets that are destroyed;
  - **Cost of Business System Downtime (CoBSD)**: cost of refunds for failed transactions; and cost of lost business directly attributable to the incident;
  - **Cost of Containment (CoC)**: efforts and cost, and consulting services;
  - **Cost of Recovery (CoRy)**: cost of incident investigation and analysis; effort required to repair and replace systems; replacement cost of systems consulting services for repair or investigation; and additional costs not covered by an insurance policy.
  - **Cost of Restitution (CoRn)**: penalties and other funds paid out due to breaches of contacts or SLAs resulting from the incident; cost of services provided to customers as a direct result of the incident (e.g. ID Theft Insurance); public relations costs; cost of disclosures and notifications; and legal costs, fines, and settlements.

- **Question**: what is the average (mean) cost to the organization from security incidents during the given period?

- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0.0" indicates there were no measured costs to the organization.

- **Formula**: Mean Cost of Incidents (MCOI) is calculated by summing all costs associated with security incidents by the number of security incidents that occurred during the time period:

$$\text{MCOI} = \frac{\sum(\text{DL}+\text{CoBSD}+\text{CoC}+\text{CoRy}+\text{CoRn})}{\text{Count(Incidents)}}$$

- **Units**: R$ per incident

- **Frequency**: monthly, annually.

- **Targets**: ideally there would be no security incidents with material impacts on the organization, and the metric value would be zero. In practice a target can be set based on the expected loss budget determined by risk assessments processes.

- **Sources**: incident tracking systems will provide incident data. Cost data can come from both management estimates, ticket tracking systems and capital and services budgets.

- **Usage**: Mean Cost of Incidents (MCOI) represents the average impact of a security incident on the organization. This impact is the average known outcome resulting from the interaction of the threat environment with the security systems, processes, and policies or the organization. The lower the MCOI, the less the organization is impacted by security incidents on average. Optimal conditions would reflect a low value of MCOI. Costs experienced by organizations can vary as a result of the threat environment, systems and processes in place, and resiliency of the organization. Over time, the effectiveness of changes to an organization's security activities should result in a reduction in the Mean Cost of Incidents. MCOI should provide a management indicator of the ability of the organization to alter the known impact expected from security incidents.

- **Limitations**: some incidents such as exposure of business strategy via social engineering may not have a direct incident costs. Significant harm, bad press, or competitive disadvantage may still be experienced for which it is not practical to assign a cost; some new controls may have significant costs and/or address recovery from multiple incidents; this metric relies on the common definition of "security incident" as defined in Terms and Definitions; this metric relies on an organization being able to produce costs or cost estimates related to security incidents.

- **Dimensions**: this metric may include additional dimensions for grouping and aggregation purposes. These dimensions should be applied or tagged at the level of the underlying incident record as described in Security Incident Metrics. For example: (i) priority dimension allows MCOI to be computed for high, medium, or low severity incidents; (ii) classifications for characterizing types of incidents, such as denial of service, theft of information, etc.; (iii) affected Organization for identifying the affected part of the organization.

- **References**: [Killcrece et al., 2003, West-Brown et al., 2003, Dorofee et al., 2007].

Id: 4.1.5 - Metric Name: Mean Incident Recovery Cost (MIRC)

- **Objective**: Mean Incident Recovery Cost measures the total costs directly associated with the operational recovery from an incident. While the impact of similar incidents may vary across organizations, the technical recovery should be comparable on a per-system basis across firms.
- **Description**: Mean Incident Recovery Cost (MIRC) measures the cost of returning business systems to their pre-incident condition. The following costs may be taken into consideration:
  - Cost to repair and/or replace systems;
  - Opportunity cost of staff implementing incident handling plan;
  - Cost to hire external technical consultants to help recover from the incident;
  - Cost to installation new controls or procurement of new resources that directly addresses the re-occurrence of the incident, e.g. installation of AV software;
  - Legal and regulatory liabilities resulting from the incident.
- **Question**: what is the average (mean) cost of recovery from a security incidents during the given period?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0.0" indicates there were no measured costs to the organization.
- **Formula**: Mean Incident Recovery Cost (MIRC) is calculated by summing all costs associated with recovering from security incidents by the number of security incidents that occurred during the time period:

$$\text{MIRC} = \frac{(\sum(\text{Cost\_Recovery})}{\text{Count(Incidents)}}$$

- **Units**: R$ per incident
- **Frequency**: monthly, annually.
- **Targets**: Ideally, recovery from security incidents would have no material impacts on the organization, and the metric value would be zero. In practice a target can be set based on the expected loss budget determined by risk assessments processes, and planned incident recovery resources.
- **Sources**: Incident tracking systems will provide incident data. Cost data can come from management estimates, ticket tracking systems and capital and services budgets.
- **Usage**: Mean Incident Recovery Cost (MIRC) represents the average cost the organization incurs while recovering from a security incident. This cost is correlated to the capabilities and resiliency of the systems, processes, and policies. The lower the MIRC, the less the organization is impacted by security incidents on average, and

the greater the general resiliency of the organization's systems. Optimal conditions would reflect a low value of MIRC. Costs experienced by organizations can vary as a result of the threat environment, systems, and processes in place. Over time, the effectiveness of changes to an organization's security activities should result in a reduction in the Mean Incidents Recovery Cost. MIRC should provide a management indicator of the expected ability of the organization's resiliency and ability to recover from security incidents.

- **Limitations**: some incidents, such as theft via social engineering, may not have a direct recovery costs as there may not be a clear end point or may be the result of several related incidents; establishment of new controls or procurement of new resources may have significant costs; this metric is dependent upon when during the incident management process cost information is collected. Depending if information is collected during the occurrence of the incident or following the incident may influence the metric outcome.

- **Dimensions**: this metric may include additional dimensions for grouping and aggregation purposes. These dimensions should be applied or tagged at the level of the underlying incident record as described in Security Incident Metrics. For example: (i) priority dimension allows MIRC to be computed for high, medium, or low severity incidents; (ii) classifications for characterizing types of incidents, such as denial of service, theft of information, etc.; (iii) affected Organization for identifying the affected part of the organization.

- **References**: [Killcrece et al., 2003, Dorofee et al., 2007, Cichonski et al., 2012]

Id: 4.1.6 - Metric Name: Mean Time Between Security Incidents (MTBSI)

- **Objective**: Mean Time Between Security Incidents (MTBSI) identifies the relative levels of security incident activity.
- **Description**: Mean Time Between Security Incidents (MTBSI) calculates the average time, in days, between security incidents. This metric is analogous to the Mean Time Between Failure (MTBF) metric found in break-fix processes for Cloud Computing environment.
- **Question**: for all security incidents that occurred within a given time period, what is the average (mean) number of days between incidents?
- **Answer**: a floating-point value that is greater than or equal to zero. A value of "0" indicates instantaneous occurrence of security incidents.
- **Formula**: for each record, the mean time between incidents is calculated by dividing the number of hours between the time on the Date of Occurrence for the current incident from the time on the Date of Occurrence of the previous incident by the total number of incidents prior to the current incident:

$$\text{MTBSI} = \frac{\sum_{i=1}^{N}(\text{Date\_of\_Occurence}[\text{Incident}_n] - \text{Date\_of\_Occurence}[\text{Incident}_{n-1}])}{\text{Count(Incidents)}}$$

- **Units**: hours per incident interval
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTBSI values should trend higher over. The value of "0" indicates hypothetical instantaneous occurrence between security incidents. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time Between Security Incidents exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: this metric provides an indication of activity within the environment. A higher value for this metric might indicate a less-active landscape. However, an inactive landscape might be caused by a lack of reporting or a lack of detection of incidents.
- **Limitations**: the date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred. This date may be subject to revision as more information becomes known about a particular incident.
- **References**: [Killcrece et al., 2003, Cichonski et al., 2012].

Id: 4.1.7 - Metric Name: Mean Time from Discovery to Containment (MTDC)

- **Objective**: Mean Time from Discovery to Containment (MTDC) characterizes the effectiveness of containing a security incident as measured by the average elapsed time between when the incident has been discovered and when the incident has been contained.
- **Description**: Mean Time from Discovery to Containment (MTDC) measures the effectiveness of the organization to identify and contain security incidents. The sooner the organization can contain an incident, the less damage it is likely to incur. This calculation can be averaged across a time period, type of incident, business unit, or severity.
- **Question**: what is the average (mean) number of hours from when an incident has been detected to when it has been contained?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates instantaneous containment.
- **Formula**: for each incident contained in the metric time period, the mean time from discovery to containment is calculated dividing the difference in hours between the Date of Containment from the Date of Discovery for each incident by the total number of incidents contained in the metric time period:

$$\text{MTDC} = \frac{\sum(\text{Date\_of\_Containment} - \text{Date\_of\_Discovery})}{\text{Count(Incidents)}}$$

- **Units**: hours per incident
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTDC values should trend lower over time. The value of "0" indicates hypothetical instantaneous containment. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time from Discovery to Containment exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: MTDC is a type of security incident metric and relies on the common definition of "security incidents" as defined in Glossary. An incident is determined to be "contained" when the immediate effect of the incident has been mitigated. For example, a DDOS attack has been throttled or unauthorized external access to a system has been blocked, but the system has not yet been fully recovered or business operations are not restored to pre-incident levels. Optimal conditions

would reflect a low value in the MTDC. A low MTDC value indicates a healthier security posture as malicious activity will have less time to cause harm. Given the modern threat landscape and the ability for malicious code to link to other modules once entrenched, there may be a direct correlation between a lower MTDC and a lower incident cost.

- **Limitations**: This metric measures incident containment capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others, and would thus be a more attractive target for attackers, whose attack vectors and capabilities will vary. As such, MTDCs may not be directly comparable between organizations. In addition, the ability to calculate meaningful MTDCs assumes that incidents are detected. A lack of participation by the system owners could skew these metrics. A lower rate of participation in the reporting of security incidents can increase the accuracy of these metrics. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred no later than given the best available information. This date may be subject to revision and more information becomes known about a particular incident. Incidents can vary in size and scope. This could result in a variety of containment times that, depending on its distribution, may not provide meaningful comparisons between organizations when mean values are used.

- **References**: [Killcrece et al., 2003, Cichonski et al., 2012, Team, 2015].

Id: 4.1.8 - Metric Name: Mean Time To Incident Discovery (MTTID)

- **Objective**: Mean-Time-To-Incident-Discovery (MTTID) characterizes the efficiency of detecting incidents, by measuring the average elapsed time between the initial occurrence of an incident and its subsequent discovery. The MTTID metric also serves as a leading indicator of resilience in organization defenses because it measures detection of attacks from known vectors and unknown ones.

- **Description**: Mean-Time-To-Incident-Discovery (MTTID) measures the effectiveness of the organization in detecting security incidents. Generally, the faster an organization can detects an incident, the less damage it is likely to incur. MTTID is the average amount of time, in hours, that elapsed between the Date of Occurrence and the Date of Discovery for a given set of incidents. The calculation can be averaged across a time period, type of incident, business unit, or severity.

- **Question**: what is the average (mean) number of hours between the occurrence of a security incident and its discovery?

- **Answer**: a positive decimal value that is greater than or equal to zero. A value of "0" indicates hypothetical instant detection.

- **Formula**: for each record, the time-to-discovery metric is calculated by dividing the subtracting the Date of Occurrence from the Date of Discovery by the total number of incidents prior to the current incident. These metrics are then averaged across a scope of incidents, for example by time, category or business unit:

$$\text{MTTID} = \frac{\sum(\text{Date\_of\_Discovery} - \text{Date\_of\_Occurrence})}{\text{Count(Incidents)}}$$

- **Units**: hours per incident

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTTID values should trend lower over time. The value of "0 hours" indicates hypothetical instant detection times. There is evidence the metric result may be in a range from weeks to months (2015 Verizon Data Breach Report). Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for MTTIDs exist.

- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.

- **Usage**: Mean-Time-To-Incident-Discovery is a type of security incident metric, and relies on the common definition of "security incident" as defined in Terms in Definitions. Optimal conditions would reflect a low value in the MTTID. The lower the value of MTTID, the healthier the security posture is. The lower the MTTID,

the more time malicious activity is likely to have occurred within the environment prior to containment and recovery activities. Given the current threat landscape and the ability for malicious code to link to other modules once entrenched, there may be a direct correlation between a lower MTTID and a lowergher level-of-effort value (or cost) of the incident. MTTIDs are calculated across a range of incidents over time, typically per-week or per-month. To gain insight into the relative performance of one business unit over another, MTTIDs may also be calculated for cross-sections of the organization, such as individual business units or geographies.

- **Limitations**: this metric measures incident detection capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others, and would thus be a more attractive target for attackers, whose attack vectors and capabilities will vary. As such, MTTIDs may not be directly comparable between organizations. In addition, the ability to calculate meaningful MTTIDs assumes that incidents are, in fact, detected and reported. A lack of participation by the system owners could cause a skew to appear in these metrics. A lower rate of participation in the reporting of security incidents can increase the accuracy of these metrics. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred no later than given the best available information. This date may be subject to revision and more information becomes known about a particular incident. Mean values may not provide a useful representation of the time to detect incidents if distribution of data exhibits significantly bi-modal or multi-model. In such cases additional dimensions and results for each of the major modes will provide more representative results.

- **References**: [Killcrece et al., 2003, Cichonski et al., 2012, Team, 2015].

Id: 4.1.9 - Metric Name: Mean Time To Incident Recovery (MTIR)

- **Objective**: Mean Time to Incident Recovery (MTIR) characterizes the ability of the organization to return to a normal state of operations. This is measured by the average elapse time between when the incident occurred to when the organization recovered from the incident.
- **Description**: Mean Time to Incident Recovery (MTIR) measures the effectiveness of the organization to recovery from security incidents. The sooner the organization can recover from a security incident, the less impact the incident will have on the overall organization. This calculation can be averaged across a time period, type of incident, business unit, or severity.
- **Question**: what is the average (mean) number of hours from when an incident occurs to recovery?
- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0" indicates instantaneous recovery.
- **Formula**: Mean time-to-incident recovery is calculated by dividing the difference between the Date of Occurrence and the Date of Recovery for each incident recovered in the metric time period, by the total number of incidents recovered in the metric time period

$$\text{MTTIR} = \frac{\sum(\text{Date\_of\_Recovery} - \text{Date\_of\_Occurrence})}{\text{Count(Incidents)}}$$

- **Units**: hours per incident
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTIR values should trend lower over time. There is evidence the metric result will be in a range from days to weeks (2015 Verizon Data Breach Report). The value of "0" indicates hypothetical instantaneous recovery. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time to Incident Recovery exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: MTIR is a type of security incident metric and relies on the common definition of "security incidents" as defined in Glossary. Optimal conditions would reflect a low value in the MTIR. A low MTIR value indicates a healthier security posture as the organization quickly recovered from the incident. Given the impact that an incident can have on an organization's business processes, there may be a direct correlation between a lower MTIR and a lower incident cost.

- **Limitations**: this metric measures incident recovery capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities vary. MTIRs may not be directly comparable between organizations. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred. This date may be subject to revision and more information becomes known about a particular incident.

- **References**: [Killcrece et al., 2003, Cichonski et al., 2012, Team, 2015].

Id: 4.1.10 - Metric Name: Number of Incidents (NI)

- **Objective**: Number of Incidents indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Incidents measures the number of security incidents for a given time period.
- **Question**: what is the number of security incidents that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: to calculate Number of Incidents (NI), the number of security incidents are counted across a scope of incidents, for example a given time period, category or business unit:

$$\text{NI} = \text{Count}(\text{Incidents})$$

- **Units**: incidents per period; for example, incidents per week or incidents per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NI values should trend lower over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for Incident Rate exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Incidents is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment.

To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Killcrece et al., 2003, Cichonski et al., 2012].

Id: 4.1.11 - Metric Name: Percentage of Incidents Detected by Internal Controls (PIDIC)

- **Objective**: Percentage of Incidents Detected by Internal Controls (PIDIC) indicates the effectiveness of the security monitoring program.
- **Description**: Percentage of Incidents Detected by Internal Controls (PIDIC) calculates the ratio of the incidents detected by standard security controls and the total number of incidents identified.
- **Question**: of all security incidents identified during the time period, what percent were detected by internal controls?
- **Answer**: positive floating point value between zero and 100. A value of "0" indicates that no security incidents were detected by internal controls and a value of "100" indicates that all security incidents were detected by internal controls.
- **Formula**: percentage of Incidents Detected by Internal Controls (PIDIC) is calculated by dividing the number of security incidents for which the Detected by Internal Controls field is equal to "true" by the total number of all known security incidents:

$$\text{PIDIC} = \frac{\text{Count(Incident\_Detected\_By\_Internal\_Controls = TRUE)}}{\text{Count(Incidents)}} * 100$$

- **Units**: percentage of incidents
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: PIDIC values should trend higher over time. The value of? 100%? indicates hypothetical perfect internal controls since no incidents were detected by outside parties. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Percentage of Incidents Detected by Internal Controls exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: this metric measures the effectiveness of a security monitoring program by determining which incidents were detected by the organization's own internal activities (e.g. intrusion detecti on systems, log reviews, employee observations) instead of an outside source, such as a business partner or agency. A low value can be due to poor visibility in the environment, ineffective processes for discovering incidents, ineffective alert signatures and other factors. Organizations should report on this metric over time to show improvement of the monitoring program.

- **Limitations**: An organization may not have direct control over the percentage of incidents that are detected by their security program. For instance, if all the incidents that occur are due to zero-day or previously unidentified vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations could increase the number of incidents that are detected by the organization.
- **References**: [Killcrece et al., 2003, Cichonski et al., 2012, Team, 2015].

Id: 4.1.12 - Metric Name: Time Per Incident (TPI)

- **Objective**: Time Per Incident (TPI) indicates the time of detected security incidents the organization has handled during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities. Time total period in the steps of the detected security incidents: (i) Incident Definition and Examples; (ii) Preparation, Detection, and Analysis; (iii) Containment, Eradication, and Recovery; (iv) Checklist for Handling Multiple Component Incidents; (v) Recommendations; (vi) Post-Incident Activity.
- **Description**: Time Per Incident (TPI) measures the time of security incidents for a given time period.
- **Question**: what is the time of security incidents that handled during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Time Per Incident (TPI) is calculated by dividing the difference between the Date of Finished and the Date of Occurrence for each incident recovered in the metric time period, by the total number of incidents recovered in the metric time period:

$$\text{TPI} = \frac{\sum(\text{Date\_of\_Finished} - \text{Date\_of\_Occurrence})}{\text{Count(Incidents)}}$$

- **Units**: time per incident; for example, hours, days, months.
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: for example, a growing number of incidents performed by insiders could prompt stronger policy provisions concerning background investigations for personnel and misuse of computing resources and stronger security controls on internal networks (e.g., deploying intrusion detection software to more internal networks and hosts).
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Time Per Incident is a type of security incident metric, and relies on the common definition of "security incident", and theses are calculated across a range of incidents over time, typically per-week or per-month. To gain insight into the relative performance of one business unit over another, TPI may also be calculated for cross-sections of the organization, such as individual business units or geographies.

- **Limitations**: this metric measures time per incident capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities vary. TPI may not be directly comparable between organizations. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred. This date may be subject to revision and more information becomes known about a particular incident.

- **References**: [Killcrece et al., 2003, Cichonski et al., 2012].

Id: 5.1.1 - Metric Name: Configuration Management Coverage (CMC)

- **Objective**: The goal of this metric is to provide an indicator of the scope of configuration management control systems and monitoring. Accurate and timely detection of configuration changes, as well as the ability to assess the state of the current configuration through regular processes or automated means provides organizations with improved visibility into their security posture. If 100% of systems are under configuration monitoring than the organization is relatively less exposed to exploits and to unknown threats resulting from un-approved, untested, or unknown configuration states.

- **Description**: this metric attempts to answer the question? Are system under configuration management control? This question presumes the organization has a configuration management system to test and monitor the configuration states of systems. The percentage of total computer systems in an organization that are under the scope of a configuration monitoring/management system. Scope of configuration monitoring is a binary evaluation: a given system is either part of a system that can assess and report it's configuration state or it is not. Configuration state can be evaluated by automated methods, manual inspection, or audit, or some combination. The computer system population base is the total number of computer systems with approved configuration standards. This may be all systems or only a subset (i.e. only desktops, or only servers, etc.) Organizations that do not have approved standards for their computer systems should report "N/A" rather than a numeric value (0% or 100%).

  - In Scope: examples of percentage of systems under configuration management may include: (i) configuration of servers; (ii) configuration of workstations/laptops; (iii) configuration of hand-held devices; (iv) configuration of other supported computer systems covered by the organizations configuration policy.
  - Out of Scope: examples of computer system configurations that are not in scope include: (i) temporary guest systems (contractors, vendors); (ii) lab/test systems performing to or in support of a specific nonproduction project; (iii) networking systems (routers, switches, access points); (iv) storage systems (i.e. network accessible storage)

- **Question**: what percentage of the organizations systems are under configuration management?

- **Answer**: a positive integer value between zero and 100 inclusive, expressed as a percentage. A value of? 100%? indicates that all technologies are in configuration management system scope.

- **Formula**: Configuration Management Coverage (CMC) is calculated by determining the number of in-scope systems within configuration management scope and then averaging this across the total number of in-scope systems:

$$\text{CMC} = \frac{\sum(\text{InScope\_Systems\_Under\_Configuration\_Management})}{\text{Count}(\text{InScope\_Systems})} * 100$$

- **Units**: percentage of Systems
- **Frequency**: monthly, annually.
- **Targets**: the expected trend for this metric over time is to remain stable or increase towards 100
- **Sources**: configuration management and asset management systems will provide coverage.
- **Usage**: the Configuration Management Coverage metric provides information about well the organization ensures the integrity of their network. Organizations can use this metric to evaluate their risk position in terms of concentrations of inconsistent state of systems. The results of the coverage metric indicate the: (i) scope of the configuration scanning activities; (ii) applicability of other metric results across the organization; (iii) relative amount of information known about the organization's configuration.
- **Limitations**: the organization's critical systems (e.g. production servers) maybe out of scope of the configuration management system by design, for performance or network architecture reasons.
- **References**: [ISO/IEC-12207:2008, 2008, Chew et al., 2008, NIST, 2013d].

Id: 5.1.2 - Metric Name: Mean Cost to Patch (MCTP)

- **Objective**: this defines a metric for measuring the mean effort required to deploy a patch into an environment. The metric is expressed in the context of a patch management process, with the assumption that the organizations has a formal system (i.e. patch management and electronic ticketing system) used to track activities to deploy patches. The metric is useful where a single patch deployment (no matter how many systems are affected) is expressed as a single change ticket or as one change ticket per affected network node. This data can also be recorded as monthly totals where per-patch level granularity is not possible.

- **Description**: The goal of this metric is to understand the effort required for patch management activities. Risk management decisions can take into account the efficiency of patch deployment to make more informed decisions around patch compliance policies, Service Level Agreements, and resource allocation in the IT environment.

- **Question**: what is the average (mean) cost to the organization to deploy a patch during the given period?

- **Answer**: a positive integer value that is greater than or equal to zero. A value of "0.0" indicates there were no measured costs to the organization.

- **Formula**: Mean Cost To Patch is calculated by determining the total cost to deploy patches. These results are then averaged across the number of patches deployed in the current scope:

$$\text{MCTP} = \frac{\sum(\text{Patch\_Cost} + \text{Other\_Patch\_Cost})}{\text{Count(Deployed\_Patches)}}$$

  Patch Cost may be a determined aggregate cost or is calculated based upon the amount of effort put into the patching process as calculated by: $Patch\_Effort * Hourly\_Rate$. Other Patch Costs may include:

  - purchases of additional equipment;
  - purchases of new software versions;
  - cost associated with mitigation of a specific vulnerability;
  - cost associated with vendor waiting to release patch until its next release cycle for identified vulnerabilities;
  - cost associated with delaying updates until next update cycle;
  - cost associated with identifying missing patches;
  - cost associated with downtime during testing and installation of missing patches.

  The cost of patch management systems should not be included in this cost. If a one-time cost is associated with multiple vulnerabilities the cost should be distributed evenly across the relevant vulnerabilities.

- **Units**: R$ per patch
- **Frequency**: monthly, annually.
- **Targets**: ideally, all patches would be deployed by an automated system, and the mean cost to patch would approach zero (given patch testing costs, etc.).
- **Sources**: patch management and IT support tracking systems will provide patch deployment data. Cost data can come from management estimates, ticket tracking systems, and services budgets.
- **Usage**: keeping systems fully patched should reduce risk and result in lower incidents costs to the organization. Organizations generally have to balance the desire to patch systems with the cost and effort of preparing, testing, and deploying patches in their environment. Mean Cost To Patch allows the organization understand the cost of patch deployment, manage trends, and perform cost-benefit analysis patch updates, comparing the cost to the organization to the costs of any increases in security incidents.
- **Limitations**: note that this assumes: (i) effort is tracked for vulnerability remediation; (ii) tickets are closed when the change is known to have mitigated the vulnerability; (iii) vulnerabilities can be tracked between scans on a vulnerability instance per-host basis; (iv) it is not including in-progress tickets, vulnerabilities that have not been mitigated, or vulnerabilities that do not have a resolution.
- **References**: [Cavusoglu et al., 2006, NIST, 2013c].

Id: 5.1.3 - Metric Name: Mean Time To Complete Changes (MTTCC)

- **Objective**: the goal of this metric is to provide managers with information on the average time it takes for a configuration change request to be completed.
- **Description**: the average time it takes to complete a configuration change request.
- **Question**: what is the mean time to complete a change request?
- **Answer**: a positive integer value that is greater than zero. A value of "0" indicates that the organization immediately implements changes.
- **Formula**: the mean time to complete a change request is calculated by taking the difference between the date the request was submitted and the date the change was completed for each change completed within the time period of the metric. This number is then divided by the total number of changes completed during the metric's time period:

$$\text{MTTCC} = \frac{\sum(\text{Completion\_Date} - \text{Submission\_Date})}{\text{Count}(\text{Completed\_Changes})}$$

- **Units**: days per configuration change request
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: MTCC values should generally trend lower over time provided operational system uptime is very high. This number will depend on the organization's business, structure, and use of IT. While a lower value indicates greater effectiveness at managing the IT environment, this should be examined in combination with the use of approval and change review controls. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time to Complete Changes exists.
- **Sources**: configuration management and IT support tracking systems will provide configuration change data.
- **Usage**: managers can use this metric to understand their ability to react to changing needs i n their environment. The faster the approval cycle, the shorter the response time will be. The exact value that reflects a healthy environment will be subjective for the type of company. However, values should be similar for companies of the same size and business focus. By focusing on high-value applications or urgent change requests they can improve their understanding of risk management capabilities. It is useful to pair this metric with data on the absolute number of changes in order to understand the effectiveness of the change management capabilities of the organization.
- **Limitations**: (i) only completed changes: this metric only calculates the result for changes that have been completed during the time period. Changes that have not

occurred will not influence the metric results until they are completed, perhaps several reporting periods later. This may over - report performance while the changes are not completed and under-report performance after the changes has been completed; (ii) scheduled changes: changes that have been submitted with a scheduled change date may result in metric values that do not provide material information. The time taken for the change request to be approved and any delays due to the work queue volumes should be considered, but not time a change request is not being processed in some manner; (iii) variations in the scale of changes: all changes are weighted equally for this metric regardless of the level of effort required or priority of the request and are not taken into account by the current metric definition. Organizations wanting increased precision could group results by categories of change size (e.g. Large, Medium, Small) or normalize based on level of effort.

- **References**: [A. Riley et al, 2008, Kempter, 2011a, Kempter, 2011b].

Id: 5.1.4 - Metric Name: Mean Time to Deploy Critical Patches (MTDCP)

- **Objective**: Mean Time to Deploy Critical Patches (MTDCP) characterizes effectiveness of the patch management process by measuring the average time taken from notification of critical patch release to installation in the organization. This metric serves as an indicator of the organization's exposure to severe vulnerabilities by measuring the time taken to address systems in known states of high vulnerability for which security patches are available. This is a partial indicator as vulnerabilities may have no patches available or occur for other reasons such as system configurations.

- **Description**: Mean Time to Patch Deploy Patches (MTPCP) measures the average time taken to deploy a critical patch to the organization's technologies. The sooner critical patches can be deployed, the lower the mean time to patch and the less time the organization spends with systems in a state known to be vulnerable. In order for managers to better understand the exposure of their organization to vulnerabilities, Mean Time to Deploy Critical Patches should be calculated for the scope of patches with Patch Criticality levels of? Critical?. This metric result, reported separately provides more insight than a result blending all patch criticality levels as seen in the Mean Time to Patch metric.

- **Question**: how many days does it take the organization to deploy critical patches into the environment?

- **Answer**: a positive floating-point value that is greater than or equal to zero. A value of "0" indicates that critical patches were theoretically instantaneously deployed.

- **Formula**: Mean Time to Deploy Critical Patches is calculated by determining the number of hours between the Date of Notification and the Date of Installation for each critical patch completed in the current scope, for example by time period or business unit. These results are then averaged across the number of completed critical patches in the current scope:

$$\text{MTDCP} = \frac{\sum(\text{Date\_of\_Installation} - \text{Date\_of\_Notification})}{\text{Count}(\text{Completed\_Critical\_Patches})}$$

- **Units**: hours per patch

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTDCP values should trend lower over time. Most organizations put critical patches through test and approval cycles prior to deployment. Generally, the target time for Mean Time to Deploy Critical Patches is within several hours to days. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time to Deploy Critical Patches exists.MTDCP values should trend lower over time. Most organizations put critical patches through test and approval cycles prior to deployment. Generally, the target time for Mean Time to Deploy Critical Patches is within several hours to days. Because of the

lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time to Deploy Critical Patches exists.

- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.

- **Usage**: Mean Time to Deploy Critical Patches is a type of patch management metric, and relies on the common definition of "patch" as defined in Glossary. Given that many known severe vulnerabilities result from missing critical patches, there may be a direct correlation between lower MTDCP and lower levels of Security Incidents. MTDCP can be calculated over time, typically per-week or per-month. To gain insight into the relative performance and risk to one business unit over another, MTDCP can be compared against MTTP by cross-sections of the organization such as individual business units or geographies.

- **Limitations**: (i) critical Technologies: this metric assumes that the critical technologies are known and recorded. If the critical technologies are unknown, this metric cannot be accurately measured. As new technologies are added their criticality needs to be determined and, if appropriate, included in this metric; (ii) vendor Reliance. This metric is reliant upon the vendor's ability to notify organization of updates and vulnerabilities that need patching. If the vendor does not provide a program for notifying their customers then the technology, if critical, will always be a black mark on this metric; (iii) criticality Ranking: this metric is highly dependent upon the ranking of critical technologies by the organization. If this ranking is abused then the metric will become unreliable; (iv) patches in Progress. This metric calculation does not account for patch installations that are incomplete or on-going during the time period measured. It is not clear how this will bias the results, although potentially an extended patch deployment will not appear in the results for some time.

- **References**: [NIST, 2013c].

Id: 5.1.5 - Metric Name: Mean Time To Patch (MTTC)

- **Objective**: Mean Time to Patch (MTTP) characterizes the effectiveness of the patch management process by measuring the average time taken from date of patch release to installation in the organization for patches deployed during the metric time period. This metric serves as an indicator of the organization's overall level of exposure to vulnerabilities by measuring the time the organization takes to address systems known to be in vulnerable states that can be remediated by security patches. This is a partial indicator as vulnerabilities may have no patches available or occur for other reasons such as system configurations.

- **Description**: Mean Time To Patch (MTTP) measures the average time taken to deploy a patch to the organization's technologies. The more quickly patches can be deployed, the lower the mean time to patch and the less time the organization spends with systems in a state known to be vulnerable.

- **Question**: how long does it take the organization to deploy patches into the environment?

- **Answer**: a positive floating-point value that is greater than or equal to zero. A value of "0" indicates that patches were theoretically instantaneously deployed.

- **Formula**: Mean Time To Patch is calculated by determining the number of hours between the Date of Availability and the Date of Installation for each patch completed in the current scope, for example by time period, criticality or business unit. These results are then averaged across the number of completed patches in the current scope:

$$\text{MTTP} = \frac{\sum(\text{Date\_of\_Installation} - \text{Date\_of\_Availability})}{\text{Count}(\text{Completed\_Patches})}$$

- **Units**: hours per patch

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTTP values should trend lower over time. Most organizations put patches through test and approval cycles prior to deployment. Generally, the target time for MTTP will be a function of the criticality of the patch and business criticality of the technology. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Mean Time to Patch exists.

- **Sources**: Patch management and IT support tracking systems will provide patch deployment data.

- **Usage**: Mean Time to Patch is a type of patch management metric, and relies on the common definition of "patch" as defined in Glossary. Given that many known vulnerabilities result from missing patches, there may be a direct correlation between lower MTTP and lower levels of Security Incidents. MTTP can be calculated

over time, typically per-week or per-month. To gain insight into the relative performance and risk to one business unit over another, MTTP may also be calculated for different patch criticalities and cross-sections of the organization, such as individual business units or geographies.

- **Limitations**: (i) Critical Technologies: this metric assumes that the critical technologies are known and recorded. If the critical technologies are unknown, this metric cannot be accurately measured. As new technologies are added their criticality needs to be determined and, if appropriate, included in this metric; (ii) Vendor Reliance: this metric is reliant upon the vendor's ability to notify organization of updates and vulnerabilities that need patching. If the vendor does not provide a program for notifying their customers then the technology, if critical, will always be a black mark on this metric; (iii) Criticality Ranking. This metric is highly dependent upon the ranking of critical technologies by the organization. If this ranking is abused then the metric will become unreliable. (iv) Patches in-Progress. This metric calculation does not account for patch installations that are incomplete or on-going during the time period measured. It is not clear how this will bias the results, although potentially an extended patch deployment will not appear in the results for some time.

- **References**: [NIST, 2013c].

Id: 5.1.6 - Metric Name: Patch Management Coverage (PMC)

- **Objective**: Patch Management Coverage (PMC) characterizes the efficiency of the patch management process by measuring the percentage of total technologies that are managed in a regular or automated patch management process. This metric also serves as an indicator of the ease with which security-related changes can be pushed into the organization's environment when needed.

- **Description**: Patch Management Coverage (PMC) measures the relative amount of an organization's systems that are managed under a patch management process such as an automated patch management system. Since patching is a regular and recurring process in an organization, the higher the percentage of technologies managed under such a system the timelier and more effectively patches are deployed to reduce the number and duration of exposed vulnerabilities.

- **Question**: what percentage of the organization's technology instances are not part of the patching process and represent potential residual risks for vulnerabilities?

- **Answer**: a positive integer value that is greater than or equal to zero. A value of? 100%? indicates that all technologies are under management.

- **Formula**: Patch Management Coverage is calculated by dividing the number of the technology instances under patch management by the total number of all technology instances within the organization. This metric can be calculated for subsets of technologies such as by asset criticality or business unit.

$$\text{PMC} = \frac{\text{Count(Technology\_Instances\_Under\_Patch\_Management)}}{\text{Count(Technology\_Instances)}} * 100$$

- **Units**: percentage of technology instances

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: PMC values should trend higher over time. Given the difficulties in manually managing systems at scale, having technologies under patch management systems is preferred. An ideal result would be 100% of technologies. However, given incompatibilities across technologies and systems this is unlikely to be attainable. Higher values would generally result in more efficient use of security resources. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for PMC exists.

- **Sources**: patch management and IT support tracking systems will provide patch deployment data.

- **Usage**: Patch Management Coverage is a type of patch management metric and relies on the common definition of "patch" as defined in Glossary. Optimal conditions would reflect a high value in the metric. A value of 100% would indicate that every technology in the environment falls under the patch management system. The lower the value, the greater the degree of? ad-hoc? and manual patch deployment and

the longer and less effective it will be. Given that many known vulnerabilities result from missing patches, there may be a direct correlation between a higher level of Patch Management coverage and the number of known vulnerabilities in an environment. Patch Management Coverage can be calculated over time, typically per-week or per-month. To gain insight into the relative performance and risk to one business unit over another, Coverage may also be calculated for cross-sections of the organization, such as individual business units or geographies.

- **Limitations**: not all technologies within an organization may be capable of being under a patch management system, for technical or performance reasons, so the results and interpretation of this metric will depend on the specifics of an organizations infrastructure.
- **References**: [NIST, 2013c].

Id: 5.1.7 - Metric Name: Patch Policy Compliance (PPC)

- **Objective**: Patch Policy Compliance (PPC) indicates the scope of the organization's patch level for supported technologies as compared to their documented patch policy. While specific patch policies may vary within and across organizations, performance versus stated patch state objectives can be compared as a percentage of compliant systems.

- **Description**: Patch Policy Compliance (PPC) measures an organization's patch level for supported technologies as compared to their documented patch policy. "Policy" refers to the patching policy of the organization, more specifically, which patches are required for what type of computer systems at any given time. This policy might be as simple as install the latest patches from system vendors? or may be more complex to account for the criticality of the patch or system. "Patched to policy" reflects an organization's risk/reward decisions regarding patch management. It is not meant to imply that all vendor patches are immediately installed when they are distributed.

- **Question**: what percentage of the organization's technologies is not in compliance with current patch policy?

- **Answer**: a positive integer value between zero and 100 inclusive. A value of "100%" indicates that all technologies are in compliance to the patch policy.

- **Formula**: Patch Policy Compliance (PPC) is calculated by dividing the sum of the technologies currently compliant by the sum of all technologies under patch management (where the current patch state is known). This metric can be calculated for subsets of technologies such as by technology value or business unit:

$$\text{PPC} = \frac{\text{Count(Compliant\_Instances)}}{\text{Count(Technology\_Instances)}} * 100$$

- **Units**: percentage of technology instances

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: PPC values should trend higher over time. An ideal result would be 100% of technologies. The expected trend for this metric over time is to remain stable or increase towards 100%. There will be variations when new patches are released for large number of technologies (such as a common operating system) that could cause this value to vary significantly. Measurement of this metric should take such events into consideration. Higher values would generally result in less exposure to known security issues. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Patch Policy Compliance exists.

- **Sources**: Patch management and IT support tracking systems will provide patch deployment data. Audit reports will provide compliance status.

- **Usage**: Patch Management Coverage is a type of patch management metric and relies on the common definition of "patch" as defined in Glossary. Patch Policy Compliance can be calculated over time typically per-week or per-month. To gain insight into the relative risk to one business unit over another, Compliance may also be calculated for cross-sections of the organization, such as individual business units or geographies or technology values and types.

- **Limitations**: this metric is highly dependent upon the current set of patch policy requirements. When patches are released that affect large numbers of technologies (such as common operating systems), this number can vary greatly with time if the lack of new patches makes a system non-compliant.

- **References**: [NIST, 2013c].

Id: 5.2.1 - Metric Name: Percent of Changes with Security Exceptions (PCSE)

- **Objective**: the goal of this metric is to provide managers with information about the potential risks to their environment resulting from configuration or system changes exempt from the organization's security policy.
- **Description**: this metric indicates the percentage of configuration or system changes that received an exception to existing security policy.
- **Question**: what percentage of changes received security exceptions?
- **Answer**: a positive integer value between zero and one, reported as a percentage. A value of? 100%? indicates that all changes are exceptions.
- **Formula**: this Percentage of Security Exception (PCSE) metrics are calculated by counting the number of completed configuration changes that received security exceptions during the metric time period divided by the total number of configuration changes completed during the metric time period:

$$\text{PCSE} = \frac{\text{Count(Completed\_Changes\_with\_Security\_Exceptions)}}{\text{Count(Completed\_Changes)}} * 100$$

- **Units**: percentage of configuration changes
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: PCSE values should trend lower over time. Generally speaking, exceptions made to security policies increase the complexity and difficulty of managing the security of the organization. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Percent of Changes with Security Exceptions exists.
- **Sources**: configuration management and IT support tracking systems will provide configuration change data.
- **Usage**: manager can use this metric to understand their exposure in terms of the percentage of change exceptions to their security policy. While exceptions may be granted based on negligible risk or additional controls, it is possible that accumulated change exceptions could degrade their security posture.By focusing on exceptions granted to changes to high-value applications and technologies, or key business units, managers can focus their attention and resources and increase their understanding of the degree to which security risks may be introduced to these systems.
- **Limitations**: (i) only completed changes: this metric is only calculating the results for changes that have been completed during the time period. Changes in-progress will not be included in this metric if they have not been completed in the metric time period; (ii) variations in the scale of changes. All changes are weighted equally for this metric and do not take into account the amount of effort required. For a

better understanding of the scale of exceptions, organizations should group results by categories of change size (Large, Medium, Small) or normalize based on scale of the change; (iii) dependency on security reviews: security exceptions may only have been granted for systems that received security reviews. Changes implemented without security reviews may include unknown and untracked exceptions to security policies.

- **References**: [A. Riley et al, 2008, Kempter, 2011a, Kempter, 2011b].

Id: 5.2.2 - Metric Name: Percent of Changes with Security Review (PCSR)

- **Objective**: the goal of this metric is to provide managers with information about the amount of changes and system churn in their environment that have unknown impact on their security state.
- **Description**: this metric indicates the percentage of configuration or system changes that were reviewed for security impacts before the change was implemented.
- **Question**: what percentage of changes received security reviews?
- **Answer**: a positive integer value between zero and one hundred that represents a percentage. A value of? 100%? indicates that all changes received security reviews during the metric time period.
- **Formula**: the Percent of Changes with Security Review (PCSR) metric is calculated by counting the number of completed configuration changes that had a security review during the metric in time period divided by the total number of configuration changes completed during the metric time period.

$$\text{PCSR} = \frac{\text{Count(Completed\_Changes\_with\_Security\_Reviews)}}{\text{Count(Completed\_Changes)}} * 100$$

- **Units**: percentage of configuration changes
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: PCSR values should trend higher over time. Generally speaking, change management processes should contain review and approval steps that identify potential business and security risks. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Percent of Changes with Security Review exists.
- **Sources**: configuration management and IT support tracking systems will provide configuration change data.
- **Usage**: managers can use this metric to understand the degree to which changes with unknown security impacts are occurring in their environment. The metric results indicate the amount of churn that has a known impact on the intended security model of the organization. As changes with unknown security implications accumulate, it would be expected that the security model of these systems would degrade. By focusing on changes to high-value applications and technologies or key business units, managers can understand the degree to which security risks may be introduced to these systems.
- **Limitations**: (i) only completed changes: this metric is only calculating the results for changes that have been completed during the time period. Changes in security review policies may not be included in this metric if the changes have not been completed in the metric time period; (ii) variations in the scale of changes. All

changes are weighted equally for this metric regardless of the level of effort required or priority of the request and are not taken into account by the current metric definition. Organizations wanting increased precision could group results by categories of change size (e.g. Large, Medium, Small) or normalize based on level of effort.

- **References**: [A. Riley et al, 2008, Kempter, 2011a, Kempter, 2011b].

Id: 6.1.1 - Metric Name: Number of Packet Filtering (NPFi)

- **Objective**: Number of Packet Filtering indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Packet Filtering measures the number of security incidents for a given time period.
- **Question**: what is the number of packet filtering that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of packet filtering is calculated by counting the number of packets that are filtered by rules of filtering, for example a given time period, category or block specific packets, and permit specific packets:

$$\text{NPFi} = \text{Count}(\text{Filtering\_Packets})$$

- **Units**: number of packet per period; for example, number of packet per week or number of packet per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NPFi values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for packet filtering exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Packet Filtering is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected,

so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.1.2 - Metric Name: Number of Security Rule Control (NSRC)

- **Objective**: Number of Security Rule Control indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Security Rule Control measures the number of security incidents for a given time period.
- **Question**: what is the number of security rule control that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Security Rule Control is calculated by counting the number of rule control, for example a given time period, category type:

$$\text{NSRC} = \text{Count}(\text{Security\_Rules\_Control})$$

- **Units**: number of security rule control per period; for example, number of security rule control per week or number of security rule control per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSRC values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for packet filtering exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Security Rule Control is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in

the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.1.3 - Metric Name: Number of Traffic Flow Statistics (NTFS)

- **Objective**: Number of Traffic Flow Statistics indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Traffic Flow Statistics measures the number of security incidents for a given time period.
- **Question**: what is the number of traffic flow statistics that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Traffic Flow Statistics is calculated by counting the number of rule control, for example a given time period, category type:

$$\text{NTFS} = \text{Count}(\text{Number\_Traffic\_Flow\_Statistics})$$

- **Units**: number of traffic flow statistics per period; for example, number of traffic flow statistics per week or number of traffic flow statistics per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NTFS values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for packet filtering exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Traffic Flow Statistics is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in

the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.1.4 - Metric Name: Number of Prevention of DoS (NPD)

- **Objective**: Number of Prevention of DoS indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Prevention of DoS measures the number of security incidents for a given time period, and security functions should continue to work after attacks are terminated.
- **Question**: what is the number of prevention of DoS that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Prevention of DoS is calculated by counting the number of attacks prevented to type DoS, for example a given time period, category type:

$$\text{NPD} = \text{Count}(\text{Number\_Prevention\_DoS})$$

- **Units**: number of prevention of DoS per period; for example, number of prevention of DoS per week or number of prevention of DoS per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NPD values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for packet filtering exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Prevention of DoS is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected,

so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.1.5 - Metric Name: Number of Shutdown (NS)

- **Objective**: Number of Shutdown indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Shutdown measures the number of security incidents for a given time period.
- **Question**: what is the number of shutdown that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Shutdown is calculated by counting the number of shutdown, for example a given time period, category type:

$$\text{NS} = \text{Count}(\text{Number\_Shutdown})$$

- **Units**: number of shutdown per period; for example, number of shutdown per week or number of shutdown per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NS values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of shutdown exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Shutdown is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the

number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.1.6 - Metric Name: Mean Time To Recovery From Shutdown Firewall (MTTRFSF)

- **Objective**: Mean Time To Recovery From Shutdown Firewall (MTTRFSF) characterizes the effectiveness of the patch management process by measuring the average time taken from date of patch release to installation in the organization for patches deployed during the metric time period. This metric serves as an indicator of the organization's overall level of exposure to vulnerabilities by measuring the time the organization takes to address systems known to be in vulnerable states that can be remediated by security patches. This is a partial indicator as vulnerabilities may have no patches available or occur for other reasons such as system configurations.

- **Description**: Mean Time To Recovery From Shutdown measures the average time taken to deploy a patch to the organization's technologies. The more quickly patches can be deployed, the lower the mean time to patch and the less time the organization spends with systems in a state known to be vulnerable.

- **Question**: for all security incidents that occurred within a given time period, what is the average (mean) number of hours to recovery from shutdown firewall?

- **Answer**: a positive floating-point value that is greater than or equal to zero. A value of "0" indicates that patches were theoretically instantaneously deployed.

- **Formula**: Mean Time To Recovery From Shutdown is calculated by dividing the difference between the Date of Occurrence and the Date of Recovery From Shutdown Firewall for each incident recovered in the metric time period, by the total number of incidents recovered in the metric time period:

$$\text{MTTRFSF} = \frac{\sum(\text{Date\_of\_Recovery} - \text{Date\_of\_Occurrence})}{\text{Count(Incidents)}}$$

- **Units**: hours per recovery.

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTTRFSF values should trend lower over time. The value of "0" indicates hypothetical instantaneous recovery. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for mean time to recovery from shutdown exists.

- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.

- **Usage**: MTIR is a type of security incident metric and relies on the common definition of "security incidents" as defined in Glossary. Optimal conditions would reflect a low value in the MTIR. A low MTIR value indicates a healthier security

posture as the organization quickly recovered from the incident. Given the impact that an incident can have on an organization's business processes, there may be a direct correlation between a lower MTIR and a lower incident cost.

- **Limitations**: this metric measures incident recovery capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities vary. MTIRs may not be directly comparable between organizations. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred. This date may be subject to revision and more information becomes known about a particular incident.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.1 - Metric Name: Number of Packet Fragmentation (NPFr)

- **Objective**: Number of Packet Fragmentation (NPFr) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Packet Fragmentation measures the number of security incidents for a given time period.
- **Question**: what is the number of packet fragmentation that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Packet Fragmentation is calculated by counting the number of packet fragmentation, for example a given time period, category type:

$$\text{NPFr} = \text{Count}(\text{Number\_Packet\_Fragmentation})$$

- **Units**: number of packet fragmentation per period; for example, number of packet fragmentation per week or number of packet fragmentation per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NPFr values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for packet fragmentation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Packet Fragmentation is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in

the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.2 - Metric Name: Number of Stream Segmentation (NSS)

- **Objective**: Number of Stream Segmentation (NSS) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Stream Segmentation measures the number of security incidents for a given time period.
- **Question**: what is the number of stream segmentation that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Stream Segmentation is calculated by counting the number of stream segmentation, for example a given time period, category type:

$$\text{NSS} = \text{Count}(\text{Number\_Stream\_Segmentation})$$

- **Units**: number of stream segmentation per period; for example, number of stream segmentation per week or number of stream segmentation per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSS values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for stream segmentation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Stream Segmentation is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in

the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.3 - Metric Name: Number of Remote Procedure Call Fragmentation (NRPCF)

- **Objective**: Number of Remote Procedure Call Fragmentation (NRPCF) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Remote Procedure Call Fragmentation measures the number of security incidents for a given time period.
- **Question**: what is the number of remote procedure call fragmentation that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Remote Procedure Call Fragmentation is calculated by counting the number of remote procedure call fragmentation, for example a given time period, category type:

$$\text{NRPCF} = \text{Count}(\text{Number\_Remote\_Procedure\_Call\_Fragmentation})$$

- **Units**: number of remote procedure call fragmentation per period; for example, number of remote procedure call fragmentation per week or number of remote procedure call fragmentation per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NRPCF values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for remote procedure call fragmentation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Remote Procedure Call Fragmentation is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of

security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.4 - Metric Name: Number of Recovery from Abnormal System Shutdown (NRASS)

- **Objective**: Number of Recovery from Abnormal System Shutdown (NRASS) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Recovery from Abnormal System Shutdown measures the number of security incidents for a given time period.
- **Question**: what is the number of recovery from abnormal system shutdown that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Recovery from Abnormal System Shutdown is calculated by counting the number of abnormal system shutdown, for example a given time period, category type:

$$\text{NRASS} = \text{Count}(\text{Number\_Abnormal\_System\_Shutdown})$$

- **Units**: number of recovery from abnormal system shutdown per period; for example, number of recovery from abnormal system shutdown per week or number of recovery from abnormal system shutdown per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NRASS values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for remote procedure call fragmentation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of recovery from abnormal system shutdown is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak

capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.5 - Metric Name: Number of Security Events Records (NSER)

- **Objective**: Number of Security Events Records (NSER) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Security Events Records measures the number of security incidents for a given time period.
- **Question**: what is the number of security events records that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Security Events Records is calculated by counting the number of security events, for example a given time period, category type:

$$\text{NSER} = \text{Count}(\text{Number\_Security\_Events\_Records})$$

- **Units**: number of security events records per period; for example, number of security events recordsn per week or number of security events records per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSER values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for remote procedure call fragmentation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of security events records is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents

are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6 - Metric Name: Number of Evasion Attacks (NEA)

- **Objective**: Number of Evasion Attacks (NEA) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Evasion Attacks measures the number of security incidents for a given time period.
- **Question**: what is the number of evasion attacks that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Evasion Attacks is calculated by counting the number of abnormal system shutdown, for example a given time period, category type:

$$\text{NEA} = \text{Count}(\text{Number\_Evasion\_Attacks})$$

- **Units**: number of evasion attacks per period; for example, number of evasion attacks per week or number of evasion attacks per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NEA values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of remote procedure call fragmentation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of evasion attacks is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected,

so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6.1 - Metric Name: Number of URL Obfuscation (NUO)

- **Objective**: Number of URL Obfuscation (NUO) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of URL Obfuscation measures the number of security incidents for a given time period.
- **Question**: what is the number of URL obfuscation that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of URL Obfuscation is calculated by counting the number of URL obfuscation, for example a given time period, category type:

$$\text{NUO} = \text{Count}(\text{Number\_URL\_Obfuscation})$$

- **Units**: number of URL obfuscation per period; for example, number of URL obfuscation per week or number of URL obfuscation per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NUO values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of URL obfuscation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of URL Obfuscation is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected,

so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6.2 - Metric Name: Number of SMB & NetBIOS Evasions (NSNE)

- **Objective**: Number of SMB & NetBIOS Evasions (NSNE) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of SMB & NetBIOS Evasions measures the number of security incidents for a given time period.
- **Question**: what is the number of SMB & NetBIOS evasions that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of SMB & NetBIOS Evasions is calculated by counting the number of SMB & NetBIOS evasionson, for example a given time period, category type:

$$\text{NSNE} = \text{Count}(\text{Number\_SMB\_NetBIOS\_Evasions})$$

- **Units**: number of SMB & NetBIOS evasions per period; for example, number of SMB & NetBIOS evasions per week or number of SMB & NetBIOS evasions per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSNE values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of SMB & NetBIOS evasions exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of SMB & NetBIOS Evasions is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls.

Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6.3 - Metric Name: Number of HTML Obfuscation (NHO)

- **Objective**: Number of HTML Obfuscation (NHO) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of HTML Obfuscation measures the number of security incidents for a given time period.
- **Question**: what is the number of HTML obfuscation that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of HTML Obfuscation is calculated by counting the number of HTML obfuscation, for example a given time period, category type:

$$\text{NHO} = \text{Count}(\text{Number\_HTML\_Obfuscation})$$

- **Units**: number of HTML obfuscation per period; for example, number of HTML obfuscation per week or number of HTML obfuscation per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NHO values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of HTML obfuscation exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of HTML Obfuscation is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents

are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6.4 - Metric Name: Number of Payload Encoding (NPE)

- **Objective**: Number of Payload Encoding (NPE) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Payload Encoding measures the number of security incidents for a given time period.
- **Question**: what is the number of payload encoding that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Payload Encoding is calculated by counting the number of HTML obfuscation, for example a given time period, category type:

$$\text{NPE} = \text{Count}(\text{Number\_Payload\_Encoding})$$

- **Units**: number of payload encoding per period; for example, number of payload encoding per week or number of payload encoding per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NPE values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of payload encoding exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Payload Encoding is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected,

so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6.5 - Metric Name: Number of FTP Evasion (NFE)

- **Objective**: Number of FTP Evasion (NFE) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of FTP Evasion measures the number of security incidents for a given time period.
- **Question**: what is the number of FTP evasion that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of FTP Evasion is calculated by counting the number of FTP evasion, for example a given time period, category type:

$$\text{NFE} = \text{Count}(\text{Number\_FTP\_Evasion})$$

- **Units**: number of FTP evasion per period; for example, number of FTP evasion per week or number of FTP evasion per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NFE values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of FTP evasion exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of FTP Evasion is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment.

To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.6.6 - Metric Name: Number of Layered Evasion (NLE)

- **Objective**: Number of Layered Evasion (NLE) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Layered Evasion measures the number of security incidents for a given time period.
- **Question**: what is the number of layered evasionis that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of Layered Evasionis calculated by counting the number of FTP evasion, for example a given time period, category type:

$$\text{NLE} = \text{Count}(\text{Number\_Layered\_Evasion})$$

- **Units**: number of layered evasionis per period; for example, number of layered evasionis per week or number of layered evasionis per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NLE values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of layered evasionis exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: Number of Layered Evasion is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected,

so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 6.2.7 - Metric Name: Mean Time To Recovery From Shutdown IDPS (MTTRFSI)

- **Objective**: Mean Time To Recovery From Shutdown IDPS (MTTRFSI) characterizes the effectiveness of the patch management process by measuring the average time taken from date of patch release to installation in the organization for patches deployed during the metric time period. This metric serves as an indicator of the organization's overall level of exposure to vulnerabilities by measuring the time the organization takes to address systems known to be in vulnerable states that can be remediated by security patches. This is a partial indicator as vulnerabilities may have no patches available or occur for other reasons such as system configurations.

- **Description**: Mean Time To Recovery From Shutdown IDPS measures the average time taken to deploy a patch to the organization's technologies. The more quickly patches can be deployed, the lower the mean time to patch and the less time the organization spends with systems in a state known to be vulnerable.

- **Question**: for all security incidents that occurred within a given time period, what is the average (mean) number of hours to recovery from shutdown IDPS?

- **Answer**: a positive floating-point value that is greater than or equal to zero. A value of "0" indicates that patches were theoretically instantaneously deployed.

- **Formula**: Mean Time To Recovery From Shutdown IDPS is calculated by dividing the difference between the Date of Occurrence and the Date of Recovery From Shutdown IDPS for each incident recovered in the metric time period, by the total number of incidents recovered in the metric time period:

$$\text{MTTRFSI} = \frac{\sum(\text{Date\_of\_Recovery} - \text{Date\_of\_Occurrence})}{\text{Count(Incidents)}}$$

- **Units**: hours per recovery.

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTTRFSI values should trend lower over time. The value of "0" indicates hypothetical instantaneous recovery. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for mean time to recovery from shutdown exists.

- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.

- **Usage**: MTTRFSI is a type of security incident metric and relies on the common definition of "security incidents" as defined in Glossary. Optimal conditions would reflect a low value in the MTIR. A low MTIR value indicates a healthier security

posture as the organization quickly recovered from the incident. Given the impact that an incident can have on an organization's business processes, there may be a direct correlation between a lower MTIR and a lower incident cost.

- **Limitations**: this metric measures incident recovery capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities vary. MTIRs may not be directly comparable between organizations. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred. This date may be subject to revision and more information becomes known about a particular incident.

- **References**: [Silva and Geus, 2014a].

Id: 7.1.1 - Metric Name: Percentage of Critical Applications (PCA)

- **Objective**: this metric tracks the percentage of applications that are critical to the business.
- **Description**: the percentage of critical applications measures the percent of applications that are critical to the organization's business processes as defined by the application's value rating.
- **Question**: what percentage of the organization's applications is of critical value?
- **Answer**: a positive integer value that is equal to or greater than zero and less than or equal to one hundred, reported as a percentage. A value of? 100%? indicates that all applications are critical.
- **Formula**: the Percentage of Critical Applications (PCA) metric is calculated by dividing the number of applications that have high value to the organization by the total number of applications in the organization:

$$\text{PCA} = \frac{\text{Count(Critical\_Applications)}}{\text{Count(Applications)}} * 100$$

- **Units**: percentage of applications
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: because of the lack of experiential data from the field, no consensus on goal values for the percentage of critical applications. The result will depend on the organization's business and use of IT.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: managers can use this metric to gain a better understanding of the quantity of applications that are critical to their organization. This metric provides a reference to the scale of the organization's use of applications and assists managers with better understanding of the scope and scale of their application security risk.
- **Limitations**: (i) variations in application scope. Different organizations might count as a "single" application a system that another organization may consider several distinct applications, resulting in significantly different numbers of applications between organizations; (ii) variations in application scale: applications within or across organizations might be significantly different in size, so the level of effort required to assess, test or fix vulnerabilities may vary between applications.
- **References**: [Silva and Geus, 2014c].

Id: 7.1.2 - Metric Name: Risk Assessment Coverage (RAC)

- **Objective**: this metric reports the percentage of applications that have been subjected to risk assessments.
- **Description**: risk assessment coverage indicates the percentage of business applications that have been subject to a risk assessment at any time.
- **Question**: what percentage of applications have been the subjected to risk assessments?
- **Answer**: a positive value between zero and one hundred, reported as a percentage. A value of? 100%? would indicate that all applications have had risk assessments.
- **Formula**: the metric is calculated by dividing the number of applications that have been subject to any risk assessments by the total number of applications in the organization:

$$\text{RAC} = \frac{\text{Count(Applications\_Undergone\_Risk\_Assessment)}}{\text{Count(Applications)}} * 100$$

- **Units**: percentage of applications
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: RAC values should trend higher over time. A higher result would indicate that more applications have been examined for risks. Most security process frameworks suggest or require risk assessments for applications deployed in production environments. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Risk Assessment Coverage exists.
- **Sources**: The data source for this metric is a risk assessment tracking system.
- **Usage**: managers can use this metric to evaluate their risk posture in terms of applications that have undergone a risk assessment. A better understanding of the quantity of applications that have not been exposed to a risk assessment allows the organization to evaluate their level of unknown risk associated with these applications.
- **Limitations**: (i) variations in application scope.: different organizations might count as a "single" application a system that another organization may consider several distinct applications, resulting in significantly different numbers of applications between organizations; (ii) variations in application scale: applications within or across organizations might be significantly different in size, so the level of effort required to assess, test or fix vulnerabilities may vary between applications; (iii) depth of Risk assessments: risk assessments can vary in depth due to the methodology used, the amount of time spent, and the quality of the assessment team; (iv) stage when Assessed: risk assessments can occur at varying times in an application's development cycle that may influence the assessment.
- **References**: [Silva and Geus, 2014a].

Id: 7.1.3 - Metric Name: Security Testing Coverage (STC)

- **Objective**: this metric indicates the percentage of the organization's applications have been tested for security risks.
- **Description**: this metric tracks the percentage of applications in the organization that have been subjected to security testing. Testing can consists of manual or automated white and/or black-box testing and generally is performed on systems post-deployment (although they could be in pre-production testing). Studies have shown that there is material differences in the number and type of application weaknesses found. As a result, testing coverage should be measured separately from risk assessment coverage.
- **Question**: what percentage of applications has been subjected to security testing?
- **Answer**: a positive value between zero and one hundred, reported as a percentage. A value of? 100%? would indicate that all applications have had security testing.
- **Formula**: this metric is calculated by dividing the number of applications that have had post-deployment security testing by the total number of deployed applications in the organization:

$$\text{STC} = \frac{\text{Count(Applications\_Undergone\_Security\_Testing)}}{\text{Count(Deployed\_Applications)}} * 100$$

- **Units**: percentage of applications
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: STC values should trend higher over time. Generally, the higher the value and the greater the testing scope, the more vulnerabilities in the organization's application set will be identified. A value of 100% indicates that every application has been subject to post-deployment testing. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Security Testing Coverage exists.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: managers can use this metric to evaluate the degree to which applications have been tested for weaknesses during the post-development phase (dimensions could be used to expand this metric to cover various stages of the development lifecycle). Quantifying the applications not subjected to security testing allows the organization to evaluate their application risk.
- **Automation**: the ability to automate source data collection for this metric is medium. While the results of security testing are often maintained in a tracking system, these systems are generally maintained manually. Once the initial dataset has been collected, use of the dataset can be automated for metric calculation purposes.

- **Limitations**: (i) variations in application scope. Different organizations might count as a "single" application a system that another organization may consider several distinct applications, resulting in significantly different numbers of applications between organizations; (ii) variations in application scale: applications within or across organizations might be significantly different in size, so the level of effort required to assess, test or fix vulnerabilities may vary between applications; (iii) depth of Risk assessments: risk assessments can vary in depth due to the methodology used, the amount of time spent, and the quality of the assessment team.
- **References**: [Silva and Geus, 2014a].

Id: 7.1.4 - Metric Name: Number of Applications/Service/VM (NASV)

- **Objective**: this metric indicates the number of the organization's applications have been tested for security risks.
- **Description**: this metric tracks the number of applications in the organization that have been subjected to security testing. Testing can consists of manual or automated white and/or black-box testing and generally is performed on systems post-deployment (although they could be in pre-production testing). Studies have shown that there is material differences in the number and type of application weaknesses found. As a result, testing coverage should be measured separately from risk assessment coverage.
- **Question**: what number of applications has been subjected to security testing?
- **Answer**: a positive value or zero, expressed as a number.
- **Formula**: this metric is calculated by the number of applications that have had post-deployment security testing in the organization:

$$\text{NASV} = \text{Count}(\text{Applications\_Undergone\_Security\_Testing})$$

- **Units**: number of applications
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NASV values should trend higher over time. Generally, the higher the value and the greater the testing scope, the more vulnerabilities in the organization's application set will be identified. A value of 100% indicates that every application has been subject to post-deployment testing. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for Security Testing Coverage exists.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: managers can use this metric to evaluate the degree to which applications have been tested for weaknesses during the post-development phase (dimensions could be used to expand this metric to cover various stages of the development lifecycle). Quantifying the applications not subjected to security testing allows the organization to evaluate their application risk.
- **Automation**: the ability to automate source data collection for this metric is medium. While the results of security testing are often maintained in a tracking system, these systems are generally maintained manually. Once the initial dataset has been collected, use of the dataset can be automated for metric calculation purposes.

- **Limitations**: (i) variations in application scope. Different organizations might count as a "single" application a system that another organization may consider several distinct applications, resulting in significantly different numbers of applications between organizations; (ii) variations in application scale: applications within or across organizations might be significantly different in size, so the level of effort required to assess, test or fix vulnerabilities may vary between applications; (iii) depth of Risk assessments: risk assessments can vary in depth due to the methodology used, the amount of time spent, and the quality of the assessment team.
- **References**: [Silva and Geus, 2014a].

Id: 7.2.1 - Metric Name: Information Security Budget as % of IT Budget (SBPITB)

- **Objective**: Organizations are seeking to understand if their security spending is reasonable for the level of security performance and in-line with other organizations. This metric presents the IT security budget as a percentage of organizations overall IT budget, tracking the relative cost of security compared to IT operations. This result can also be used to benchmark spending against other organizations.
- **Description**: security budget as a percentage of IT Budget tracks the percentage of IT spending on security activities and systems. For the purposes of this metric, it is assumed that Information Security is included in the IT budget.
- **Question**: what percentage of the IT Budget is allocated to information security?
- **Answer**: a positive value equal to or between 0 and 1, expressed as a percentage. A value of? 100%? indicates that the entire Information Technology budget is dedicated to information security.
- **Formula**: the total budget allocated for security activities and systems for the metric time period is divided by the total information security budget.

$$\text{SBPITB} = \frac{\text{Security\_Budget}}{\text{IT\_Budget}} * 100$$

- **Units**: percentage of IT Budget
- **Frequency**: monthly, annually depending on budget cycle
- **Targets**: because of the lack of experiential data from the field, no strong consensus on the range of acceptable goal values for security spending exists In general, this value should be comparable with peer organizations with similar IT profiles and security activities.
- **Sources**: financial management systems and/or annual budgets.
- **Usage**: examining and tracking the percentage of the IT budget allocated to security allows an organization to compare the costs of securing their infrastructure between an organization's divisions, against other organizations, as well as to observe changes over time. These results will also provide a foundation for the optimization of security spending through comparison of spending with the outcomes of other metrics such as numbers of incidents, time to detection, time to patch, etc. The percentage of budget allocated to security should be calculated over time, typically per - quarter or per-year. To gain insight into the relative performance of one business unit over another, this result may also be calculated for cross-sections of the organization, such as individual business units or geographies where they have discrete budgets.

- **Limitations**: (i) different threat profiles across organizations: while there is systemic risk to common viruses and attacks, there is also firm specific risk based on the companies? specific activities that may require higher or lower level of security spending relative to peer organizations; (ii) different IT profiles across organizations: although in theory all organizations will make market-efficient use of IT, legacy systems and specific implementations will impact the costs of otherwise-similar IT operations as well as the costs of similar levels of security performance; (iii) differences in accounting: different organizations may account for both IT and security spending in different ways that make it hard to compare this value across organizations. Some may leverage IT resources for security purposes that make it hard to account for such partial FTEs without significant activity-based costing exercises; others may have lump-sum outsourced IT contracts without specific information on security spending.

- **References**: [Chew et al., 2008].

Id: 7.2.2 - Metric Name: Information Security Budget Allocation (SBBA)

- **Objective**: Organizations are seeking to understand if their security spending is reasonable for the level of security performance and in-line with other organizations. This metric presents the IT security budget as a percentage of organizations overall IT budget, tracking the relative cost of security compared to IT operations. This result can also be used to benchmark spending against other organizations.
- **Description**: security budget Allocation as a number of IT Budget tracks the number of IT spending on security activities and systems. For the purposes of this metric, it is assumed that Information Security is included in the IT budget.
- **Question**: what number of the IT Budget is allocated to information security?
- **Answer**: a positive value or zero, expressed as a number.
- **Formula**: the total budget allocated for security activities and systems for the metric time period is divided by the total information security budget.

$$\text{SBBA} = \text{Security\_Budget}$$

- **Units**: financial IT Budget
- **Frequency**: monthly, annually depending on budget cycle
- **Targets**: because of the lack of experiential data from the field, no strong consensus on the range of acceptable goal values for security spending exists In general, this value should be comparable with peer organizations with similar IT profiles and security activities.
- **Sources**: financial management systems and/or annual budgets.
- **Usage**: examining and tracking the percentage of the IT budget allocated to security allows an organization to compare the costs of securing their infrastructure between an organization's divisions, against other organizations, as well as to observe changes over time. These results will also provide a foundation for the optimization of security spending through comparison of spending with the outcomes of other metrics such as numbers of incidents, time to detection, time to patch, etc. The percentage of budget allocated to security should be calculated over time, typically per - quarter or per-year. To gain insight into the relative performance of one business unit over another, this result may also be calculated for cross-sections of the organization, such as individual business units or geographies where they have discrete budgets.
- **Limitations**: (i) different threat profiles across organizations: while there is systemic risk to common viruses and attacks, there is also firm specific risk based on the companies? specific activities that may require higher or lower level of security spending relative to peer organizations; (ii) different IT profiles across organizations:

although in theory all organizations will make market-efficient use of IT, legacy systems and specific implementations will impact the costs of otherwise-similar IT operations as well as the costs of similar levels of security performance; (iii) differences in accounting: different organizations may account for both IT and security spending in different ways that make it hard to compare this value across organizations. Some may leverage IT resources for security purposes that make it hard to account for such partial FTEs without significant activity-based costing exercises; others may have lump-sum outsourced IT contracts without specific information on security spending.

- **References**: [Chew et al., 2008].

Id: 7.3.1 - Metric Name: Current Anti-Malware Coverage for the Application (CAMCA)

- **Objective**: the goal of this metric is to provide an indicator of the effectiveness of an organization's antimalware management. If 100% of systems have current anti-malware detection engines and signatures, then those systems are relatively more secure. If this metric is less than 100%, then those systems are relatively more exposed to viruses and other malware. The expected trend for this metric over time is to remain stable or increase towards 100%.

- **Description**: this metric attempts to answer the question "Do it has acceptable levels of anti-malware coverage?" This question presumes the organization has defined what is an acceptable level of compliance, which may be less than 100% to account for ongoing changes in the operational environments. Malware includes computer viruses, worms, trojan horses, most rootkits, spyware, dishonest adware, crimeware and other malicious and unwanted software. The percentage of total computer systems in an organization that have current, up-to-date anti-virus (or anti-malware) software and definition files. "Current" is a binary evaluation: a given system is either configured with both up-to-date detection engines and signatures or it is not. Compliance can be evaluated by automated methods, manual inspection, audit, or some combination. Current coverage of a system is defined as a the most recent version of the engine, and a signature file that is no more than 14 days older than the most recent signature file released.

  - In Scope: examples of systems under considerations for this metric include: servers, workstations/laptops, hand-held devices, and other supported computer systems.
  - Out of Scope: examples of systems that are not under consideration for this metric include: (i) temporary guest systems (contractors, vendors); (ii) lab/test systems performing to or in support of a specific nonproduction project; (iii) networking systems (routers, switches, access points); (iv) storage systems (i.e. network accessible storage).

- **Question**: what percentage of the organizations systems have current anti-malware protection?

- **Answer**: a positive integer value between zero and 100 inclusive, expressed as a percentage. A value of "100%" indicates that all technologies have current anti-malware coverage.

- **Formula**: Current Anti-Malware Coverage (CAMC) is calculated by determining the number of in-scope systems with current coverage and then averaging this across the total number of in-scope systems:

$$\text{CAMCA} = \frac{\text{In\_\_Scope\_\_Systems\_\_with\_\_current\_\_Anti\_\_Malware)}}{\text{In\_\_Scope\_\_Systems}} * 100$$

- **Units**: percentage of systems

- **Frequency**: monthly, annually.
- **Targets**: the expected trend for this metric over time is to remain stable or increase towards 100%.
- **Sources**: configuration management and Anti-malware systems (locally or centrally managed).
- **Usage**: Current Anti-Malware Coverage for the Application (CAMCA) represents the overall compliance to anti-malware policies. The higher the CAMC the greater the number of systems in the organization are running anti-malware with recent signature files, the less likely it is that existing known malware will infect or spread across the organizations systems, or fail to be detected in a timely manner.
- **Limitations**: (i) systems critical to the organization (e.g. production servers) maybe out of scope of the anti-malware management system by design, for performance, or network architecture reasons; (ii) variation in type of anti-malware such as inbound email scanning vs. resident process scanning may be material. The completeness of signature files and frequency of updates may also vary. (iii) the time window defined as current may not be adequate if malware has its impact on the organization before signature files are developed, or before the current window has expired.
- **References**: [NIST, 2013d].

Id: 7.3.2 - Metric Name: Number of Anti-Malware (NAM)

- **Objective**: Name: Number of Anti-Malware (NAM) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: Number of Anti-Malware measures the number of security incidents for a given time period.
- **Question**: what is the number of anti-malware that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of anti-malware is calculated by counting the number of anti-malware, for example a given time period, category type:

$$\text{NAM} = \text{Count}(\text{Number\_Anti} - \text{Malware})$$

- **Units**: number of anti-malware per period; for example, number of anti-malware per week or number of anti-malware per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NAM values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of anti-malware exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of anti-malware is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment.

To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 7.4.1 - Metric Name: Mean Time Between Backup Process (MTBBP)

- **Objective**: Mean Time Between Backup Process (MTBBP) characterizes the effectiveness of the patch management process by measuring the average time taken from date of patch release to installation in the organization for patches deployed during the metric time period. This metric serves as an indicator of the organization's overall level of exposure to vulnerabilities by measuring the time the organization takes to address systems known to be in vulnerable states that can be remediated by security patches. This is a partial indicator as vulnerabilities may have no patches available or occur for other reasons such as system configurations.

- **Description**: Mean Time Between Backup Process measures the average time taken to deploy a patch to the organization's technologies. The more quickly patches can be deployed, the lower the mean time to patch and the less time the organization spends with systems in a state known to be vulnerable.

- **Question**: for all security incidents that occurred within a given time period, what is the average (mean) number of hours between backup process?

- **Answer**: a positive floating-point value that is greater than or equal to zero. A value of "0" indicates that patches were theoretically instantaneously deployed.

- **Formula**: Mean Time Between Backup Process is calculated by dividing the difference between the Date of Occurrence and the Date of End From Shutdown IDPS for each incident recovered in the metric time period, by the total number of incidents recovered in the metric time period:

$$\text{MTBBP} = \frac{\sum(\text{Date\_of\_End} - \text{Date\_of\_Occurrence})}{\text{Count(Incidents)}}$$

- **Units**: hours per backup process.

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: MTBBP values should trend lower over time. The value of "0" indicates hypothetical instantaneous recovery. Because of the lack of experiential data from the field, no consensus on the range of acceptable goal values for mean time between backup process exists.

- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.

- **Usage**: MTBBP is a type of security incident metric and relies on the common definition of "security incidents" as defined in Glossary. Optimal conditions would reflect a low value in the MTIR. A low MTIR value indicates a healthier security posture as the organization quickly recovered from the incident. Given the impact

that an incident can have on an organization's business processes, there may be a direct correlation between a lower MTIR and a lower incident cost.

- **Limitations**: this metric measures incident recovery capabilities of an organization. As such, the importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities vary. MTIRs may not be directly comparable between organizations. The date of occurrence of an incident may be hard to determine precisely. The date of occurrence field should be the date that the incident could have occurred. This date may be subject to revision and more information becomes known about a particular incident.

- **References**: [Silva and Geus, 2014a].

Id: 7.4.2 - Metric Name: Number of Backup Processes that have Failed (NBPF)

- **Objective**: Name: number of Backup Processes that have Failed (NBPF) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of Backup Processes that have Failed measures the number of security incidents for a given time period.
- **Question**: what is the number of failed backup processes that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of Backup Processes that have Failed is calculated by counting the number of anti-malware, for example a given time period, category type:

$$\text{NBPF} = \text{Count}(\text{Number\_Backup\_Processes\_Failed})$$

- **Units**: number of backup processes that have failed per period; for example, number of backup processes that have failed per week or number of backup processes that have failed per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NBPF values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of failed backup processes exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of backup processes that have failed is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents

could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 7.4.3 - Metric Name: Percent of backup processes that have failed (PBPF)

- **Objective**: this metric tracks the percent of failed backup processes that are critical to the business.
- **Description**: the percent of failed backup processes measures the percent of applications that are critical to the organization's business processes as defined by the application's value rating.
- **Question**: what percent of backup processes that have failed?
- **Answer**: a positive integer value that is equal to or greater than zero and less than or equal to one hundred, reported as a percentage. A value of? 100%? indicates that all backup processes are critical.
- **Formula**: the percent of failed backup processes metric is calculated by dividing the number of failed backup processes by the total number of backup processes in the organization:

$$\text{PBPF} = \frac{\text{Count(Failed\_Backup\_Processes)}}{\text{Count(Total\_Backup\_Processes)}} * 100$$

- **Units**: percentage of failed backup processes
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: because of the lack of experiential data from the field, no consensus on goal values for the percentage of critical applications. The result will depend on the organization's business and use of IT.
- **Sources**: security management systems will provide information on which systems were identified the security impacts on system.
- **Usage**: managers can use this metric to gain a better understanding of the quantity of applications that are critical to their organization. This metric provides a reference to the scale of the organization's use of applications and assists managers with better understanding of the scope and scale of their application security risk.
- **Limitations**: (i) variations in application scope. Different organizations might count as a "single" application a system that another organization may consider several distinct applications, resulting in significantly different numbers of applications between organizations; (ii) variations in application scale: applications within or across organizations might be significantly different in size, so the level of effort required to assess, test or fix vulnerabilities may vary between applications.
- **References**: [Silva and Geus, 2014a].

Id: 7.5.1 - Metric Name: Number of Default User Service Account (NDUSA)

- **Objective**: Name: number of Default User Service Account (NDUSA) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of Default User Service Account measures the number of security incidents for a given time period.
- **Question**: what number of default user service account that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of Default User Service Account is calculated by counting the number of default user service account, for example a given time period, category type:

$$\text{NBPF} = \text{Count}(\text{Number\_Default\_User\_Service\_Account})$$

- **Units**: number of default user service account per period; for example, number of default user service account per week or number of default user service account per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NBPF values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of default user service account exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls.

Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 7.5.2 - Metric Name: Number of Insecure User Account (NIUA)

- **Objective**: Name: number of Insecure User Account (NIUA) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of Insecure User Account measures the number of security incidents for a given time period.
- **Question**: what number of insecure user account that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of Insecure User Account is calculated by counting the number of insecure user account, for example a given time period, category type:

$$\text{NIUA} = \text{Count}(\text{Number\_Insecure\_User\_Account})$$

- **Units**: number of default user service account per period; for example, number of default user service account per week or number of default user service account per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NIUA values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of default user service account exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents

metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 7.5.3 - Metric Name: Number of Default TCP Port (NDTP)

- **Objective**: number of Default TCP Port (NDTP) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of Default TCP Port measures the number of security incidents for a given time period.
- **Question**: what number of default TCP port that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of Default TCP Port is calculated by counting the number of default TCP port, for example a given time period, category type:

$$\text{NDTP} = \text{Count}(\text{Number\_Default\_TCP\_Port})$$

- **Units**: number of default TCP port per period; for example, number of default TCP port per week or number of default TCP port per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NDTP values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of default TCP port exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default TCP port is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment.

To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 7.5.4 - Metric Name: Number of SQL Injection (NSI)

- **Objective**: Name: number of SQL Injection (NSI) indicates the number of detected security incidents the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of threats, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of SQL Injection measures the number of security incidents for a given time period.
- **Question**: what number of SQL injection that occurred during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Number of SQL Injection is calculated by counting the number of SQL injection, for example a given time period, category type:

$$\text{NSI} = \text{Count}(\text{Number\_SQL\_Injection})$$

- **Units**: number of SQL injection per period; for example, number of default user service account per week or number of default user service account per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: NSI values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of SQL injection exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit

over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 8.1.1 - Metric Name: E-mail-based data loss events, overall and by data type; ratio of data loss events of all types between corporate divisions (EBDLE)

- **Objective**: Name: e-mail-based data loss events, overall and by data type; ratio of data loss events of all types between corporate divisions (EBDLE) indicates the number of E-mail-based data loss events the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of data loss events, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of E-mail-based data loss events for a given time period.
- **Question**: what number of E-mail-based data loss events during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of events is calculated by counting the number of E-mail-based data loss events, for example a given time period, category type:

$$\text{EBDLE} = \text{Count}(\text{Number\_Events})$$

- **Units**: number of E-mail-based data loss events per period; for example, number of default user service account per week or number of default user service account per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: EBDLE values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of E-mail-based data loss events exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls.

Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 8.2.1 - Metric Name: Chi-square test for types of data loss by corporate division (CTDL)

- **Objective**: Name: chi-square test for types of data loss by corporate division (CTDL) indicates the number of Chi-square test for types of data loss the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of data loss events, effectiveness of security controls, or incident detection capabilities.

- **Description**: number of Chi-square test for types of data loss for a given time period ($\tilde{\chi}^2$) in [Taylor, 1997].

- **Question**: what number of Chi-square test for types of data loss during the time period?

- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.

- **Formula**: number of events is calculated by counting the number of Chi-square test for types of data loss, for example a given time period, category type:

$$\mathrm{CTDL} = \frac{1}{\mathrm{d}} \sum_{1}^{\mathrm{N}} \frac{(\mathrm{O_k} - \mathrm{E_k})^2}{\mathrm{E_k}}$$

  where:
  - $\mathrm{O_k}$ is event of types of data loss (Occurrence);
  - $\mathrm{E_k}$ is event of types of data loss but false-positive;
  - d is number of degress of freedom (this case, $\mathrm{d} = \mathrm{n} - 1$).

- **Units**: number of E-mail-based data loss events per period; for example, number of default user service account per week or number of default user service account per month

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: CTDL values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of E-mail-based data loss events exists.

- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.

- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 9.1.1 - Metric Name: Perimeter security events by datacenter (PSEBD)

- **Objective**: Name: perimeter security events by datacenter (PSEBD) indicates the number of perimeter security events by datacenter the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of data loss events, effectiveness of security controls, or incident detection capabilities.
- **Description**: number of perimeter security events by datacenter for a given time period.
- **Question**: what number of perimeter security events by datacenter during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: number of security events is calculated by counting the number of perimeter security events by datacenter, for example a given time period, category type:

$$\text{PSEBD} = \text{Counter}(\text{Number\_Security\_Events})$$

- **Units**: number of perimeter security events by datacenter per period; for example, number of default user service account per week or number of default user service account per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: PSEBD values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of E-mail-based data loss events exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls.

Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 9.1.2 - Metric Name: Ratio of perimeter security events between datacenters (RPSEBD)

- **Objective**: Name: ratio of perimeter security events between datacenters (RPSEBD) indicates the ratio of perimeter security events between datacenters the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of data loss events, effectiveness of security controls, or incident detection capabilities.
- **Description**: ratio of perimeter security events between datacenters for a given time period.
- **Question**: what ratio of perimeter security events between datacenters during the time period?
- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.
- **Formula**: Ratio of perimeter security events between datacenters is calculated by counting the number of security events by datacenter X, for example a given time period, category type:

$$\text{RPSEBD} = \frac{\text{Counter(Number\_Security\_Events)}}{\text{Counter(Number\_Security\_Events\_between\_datacenters)}} * 100$$

- **Units**: number of perimeter security events by datacenter per period; for example, number of default user service account per week or number of default user service account per month
- **Frequency**: weekly, monthly, quarterly, annually.
- **Targets**: RPSEBD values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of E-mail-based data loss events exists.
- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.
- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection.

However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Id: 9.2.1 - Metric Name: Analysis of variance between reported datacenter perimeter event data (AVBRDPED)

- **Objective**: Name: analysis of variance between reported datacenter perimeter event data (AVBRDPED) indicates the ratio of variance between reported datacenter perimeter event data the organization has experienced during the metric time period. In combination with other metrics, this can indicate the level of data loss events, effectiveness of security controls, or incident detection capabilities.

- **Description**: variance between reported datacenter perimeter event data for a given time period (S^2).

- **Question**: what variance between reported datacenter perimeter event data during the time period?

- **Answer**: a non-negative integer value. A value of "0" indicates that no security incidents were identified.

- **Formula**: variance between reported datacenter perimeter event data is calculated by counting the number of security events by datacenter X, for example a given time period, category type:

$$\text{AVBRDPED} = \frac{1}{n-1} \sum_{1}^{n} (x_k - \bar{x})^2$$

  where:
  - n is the sample size;
  - x_k is reported datacenter perimeter event data during the time period;
  - $\bar{x}$ is the sample mean.

- **Units**: variance between reported datacenter perimeter event data per period; for example, number of default user service account per week or number of default user service account per month

- **Frequency**: weekly, monthly, quarterly, annually.

- **Targets**: AVBRDPED values should trend higher over time? assuming perfect detection capabilities. The value of "0" indicates hypothetical perfect security since there were no security incidents. Because of the lack of experiential data from the field, no consensus on range of acceptable goal values for number of E-mail-based data loss events exists.

- **Sources**: since humans determine when an incident occurs, when the incident is contained, and when the incident is resolved, the primary data sources for this metric are manual inputs as defined in Security Incident Metrics: Data Attributes. However, these incidents may be reported by operational security systems, such as anti-malware software, security incident and event management (SIEM) systems, and host logs.

- **Usage**: number of default user service account is a type of security incident metric and relies on the common definition of "security incident" as defined in Glossary. Optimal conditions would reflect a low number of incidents. The lower the number of incidents, the healthier the security posture would be assuming perfect detection. However, a low number of incidents might also indicate a weak capability to detect incidents. This metric can also indicate the effectiveness of security controls. Assuming similar threat levels and detection capabilities, fewer incidents could indicate better performance of one set of security controls. The Number of Incidents metric is calculated over time, typically per-week or per-month. Not all incidents are easily detected, so the trend of incidents can be useful for indicating patterns in the environment. To gain insight into the relative performance of one business unit over another, the number of incidents may also be calculated for cross-sections of the organization such as individual business units or locations.

- **Limitations**: a security program may or may not have direct control over the number of incidents that occur within their environment. For instance, if all the incidents that occur are due to zero-day or previously unidentified attack vectors then there are not many options left to improve posture. However, this metric could be used to show that improving countermeasures and processes within operations to reduce the number of incidents that occur. Thus, Number of Incidents must be considered in the context of other metrics, such as MTTID. The definition of "Incident" may not be consistently applied across organizations. For meaningful comparisons, similar definitions are necessary. The importance of this metric will vary between organizations. Some organizations have much lower profiles than others and would be a more attractive target for attackers whose attack vectors and capabilities will vary.

- **References**: [Silva and Geus, 2014a].

Appendix B

Common Vulnerability Scoring System

This Appendix presents of normalization scene of risk and impact, it presents method of calculating a metric based on a number of factors like the vulnerabilities present in the system, vulnerability history of the services and impact onto hired service. It measures the existing vulnerability by combining the severity scores of the vulnerabilities present in the system. It uses the National Vulnerability Database (NVD) [NIST, 2014], provided by NIST, to find the vulnerability history of the services running on the system, and from the frequency and severity of the past vulnerabilities.

B.1 Definition

The Common Vulnerability Scoring System (CVSS) [NIST, 2014] provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. Its quantitative model ensures repeatable accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the scores. Thus, CVSS is well suited as a standard measurement system for industries, organizations, and governments that need accurate and consistent vulnerability impact scores. Two common uses of CVSS are prioritization of vulnerability remediation activities and in calculating the severity of vulnerabilities discovered on one's systems. The National Vulnerability Database (NVD) provides CVSS scores for almost all known vulnerabilities.

B.2 Normalization NVD-CVSS (risk and impact)

The example of the normalization of NVD to calculate the risk and impact of security metric Packet Filtering:

- Vulnerability Summary for: CVE-2002-0515
- Original release date: 08/12/2002
- Last revised: 09/05/2008
- Source: US-CERT/NIST

- Overview: IPFilter 3.4.25 and earlier sets a different TTL when a port is being filtered than when it is not being filtered, which allows remote attackers to identify filtered ports by comparing TTLs.
- Impact:
  - CVSS Severity (version 2.0):
  - CVSS v2 Base Score: 5.0 (MEDIUM) (AV:N/AC:L/Au:N/C:P/I:N/A:N) (legend)
  - Impact Subscore: 2.9
  - Exploitability Subscore: 10.0
  - CVSS Version 2 Metrics:
  - Access Vector: Network exploitable
  - Access Complexity: Low
  - Authentication: Not required to exploit
  - Impact Type: Allows unauthorized disclosure of information

The Figure B.1 shows the calculation of overall of this metric. The impact value 2.9, will be normalized to scale values of [0-4] to represent the impact of this metric in the Cloud Computing environment.

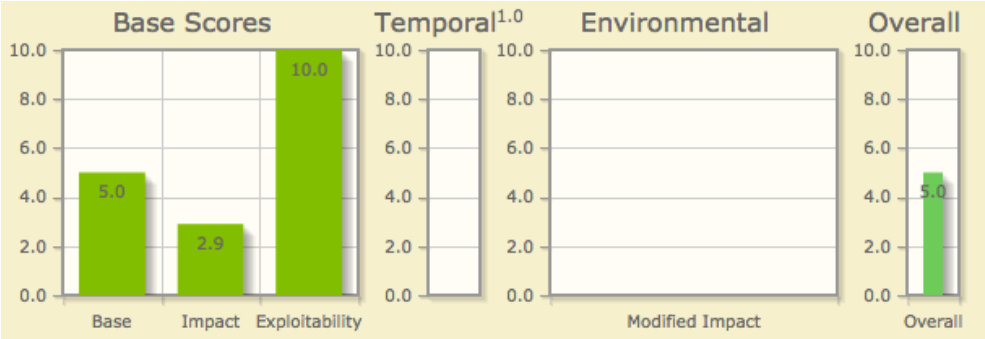


Figure B.1: Calculation of risk and impact of the Packet filtering metric.

Incomplete Data, with some vulnerabilities, all of the information needed to create CVSS scores may not be available. This typically happens when a vendor announces a vulnerability but declines to provide certain details. In such situations, NVD analysts assign CVSS scores using a worst case approach. Thus, if a vendor provides no details about a vulnerability, NVD will score that vulnerability as a 10.0 (the highest rating).

This normalization process is equivalent to the normalization process of the security of metrics to the scale of values [0-4] described in the Section 5.1.2 (page 62).

B.3 Collaboration with Industry

NVD staff are willing to work with the security community on CVSS impact scoring. If you wish to contribute additional information or corrections regarding the NVD CVSS

impact scores, please send email to nvd@nist.gov. It actively works with users that provide us feedback.

B.4 CVSS Calculator Technical Design

This section explains the CVSS Calculator's implementation. This may be useful if you wish to implement your own CVSS calculator based on FIRST's code. Each file is listed with an explanation of how it may be useful in your CVSS calculator implementation.

B.4.1 CVSS.calculateCVSSFromMetrics

Takes Base, Temporal and Environmental metric values as individual parameters and returns: scores for each, severity ratings for each, and a complete Vector String. The input parameters are:

AttackVector, AttackComplexity, PrivilegesRequired, UserInteraction, Scope, Confidentiality, Integrity, Availability, Exploitability, RemediationLevel, ReportConfidence, ConfidentialityRequirement, IntegrityRequirement, AvailabilityRequirement, ModifiedAttackVector, ModifiedAttackComplexity, ModifiedPrivilegesRequired, ModifiedUserInteraction, ModifiedScope, ModifiedConfidentiality, ModifiedIntegrity, ModifiedAvailability

Assuming this to be the case, the following properties are also defined:

baseMetricScore, baseSeverity, temporalMetricScore, temporalSeverity, environmentalMetricScore, environmentalSeverity, vectorString

Each "Score" property contains a number representing the score, each "Severity" property contains a string with the associated severity rating, and the "vectorString" property is a complete Vector String.

An example of a call to this function is:

Algorithm 1: CVSS.calculateCVSSFromMetrics

```
Data: 'N','L','N','R','C','L','L','N'
Result: calculateCVSSFromMetrics('N','L','N','R','C','L','L','N')
var result;
if (output.success === true) then
    result = "Base score is " + output.baseMetricScore + ". " +
      "Base severity is " + output.baseSeverity + ". " +
      "Temporal score is " + output.temporalMetricScore + ". " +
      "Temporal severity is " + output.temporalSeverity + ". " +
      "Environmental score is " + output.environmentalMetricScore + ". " +
      "Environmental severity is " + output.environmentalSeverity + ". " +
      "Vector string is " + output.vectorString + ". "  ;
else
    result = "An error occurred. The error type is " + errorType +
      " and the metrics with errors are " + errorMetrics + ".";
alert (result);
```

This displays an alert box with the contents:

```
Base score is 6.1.
Base severity is Medium.
Temporal score is 6.1.
Temporal severity is Medium.
Environmental score is 6.1.
Environmental severity is Medium.
Vector string is CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N.
```

Refer to the source code for more details on how errors are returned.

B.4.2 CVSS.calculateCVSSFromVector

This is similar to the previous function except that it takes a Vector String as input. Outputs are the same, except that additional error types are defined to handle problems in the format of the Vector String.

Algorithm 2: CVSS.calculateCVSSFromVector

```
Data: "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N/RL:O/CR:L"
Result: calculateCVSSFromVector("CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H
  /I:N/A:N/RL:O/CR:L")
var result;
if output.success === true then
    result = "Base score is " + output.baseMetricScore + ". " +
      "Base severity is " + output.baseSeverity + ". " +
      "Temporal score is " + output.temporalMetricScore + ". " +
      "Temporal severity is " + output.temporalSeverity + ". " +
      "Environmental score is " + output.environmentalMetricScore + ". " +
      "Environmental severity is " + output.environmentalSeverity + ". " +
      "Vector string is " + output.vectorString + ". ";
else
    result = "An error occurred. The error type is " + output.errorType +
      " and the metrics with errors are " + output.errorMetrics + ".";
alert (result);
```

This displays an alert box with the contents:

```
Base score is 8.6.
Base severity is High.
Temporal score is 8.2.
Temporal severity is High.
Environmental score is 6.0.
Environmental severity is Medium.
Vector string is
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N/RL:O/CR:L.
```

B.4.3 CVSS.severityRating

Takes a CVSS score as input and returns the severity rating name associated with that score. An example of a call to this function is:

Algorithm 3: CVSS.severityRating

```
Data: "4.8"
Result: severityRating(4.8)
var result;
if (typeof rating === 'string') then
 |  result = "Returned severity rating is " + rating;
else
 |  if (typeof rating === 'undefined') then
 |   |  result = "The input is not within the range of any defined severity rating.";
 |  else
 |   |_ result = "The input is not recognized as a number.";
 |_
alert (result);
```

This displays an alert box with the contents: returned severity rating is Medium.

B.4.4 CVSS.generateXMLFromMetrics

This is a rudimentary function to demonstrate how an XML representation of a given set of metric values can be generated. The inputs and errors are the same as for the CVSS.calculateCVSSFromMetrics function. The output is a string containing an XML representation of the metric values passed. If no error occurs, the string will be available in the xmlString property of the returned object.

An example of a call to this function is:

Algorithm 4: CVSS.generateXMLFromMetrics

```
Data: 'N','L','N','R','C','L','L','N',undefined,'W'
Result: CVSS.generateXMLFromMetrics('N','L','N','R','C','L','L','N',undefined,'W')
var result;
if (output.success === true) then
    result = output.xmlString;
else
    result = "An error occurred. The error type is " + errorType +
    " and the metrics with errors are " + errorMetrics + "." ;
alert (result);
```

This displays an alert box whose contents begin with:

```
< ?xml version="1.0" encoding="UTF-8"? >
< cvssv3.0 xmlns="https://www.first.org/cvss/cvss-v3.0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.first.org/cvss/cvss-v3.0.xsd
https://www.first.org/cvss/cvss-v3.0.xsd" >

< base_metrics >
< attack-vector > NETWORK < /attack-vector >
< attack-complexity > LOW < /attack-complexity >
< privileges-required > NONE < /privileges-required >
?
```

Refer to the source code for more details on how errors are returned.

B.4.5 CVSS.generateXMLFromVector

This is a rudimentary function to demonstrate how an XML representation of a given Vector String can be generated. It is similar to the previous function except that it takes a Vector String as input. Outputs are the same, except that additional error types are defined to handle problems in the format of the Vector String.

An example of a call to this function is:

Algorithm 5: CVSS.generateXMLFromVector

```
Data: 'CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N/RL:W'
Result: CVSS.generateXMLFromVector('CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/
  C:L/I:L/A:N/RL:W')
var result;
if (output.success === true) then
 |  result = output.xmlString;
else
 |  result = "An error occurred. The error type is " + errorType +
 |  " and the metrics with errors are " + errorMetrics + ".";
alert (result);
```

This displays an alert box whose contents begin with:

```
< ?xml version="1.0" encoding="UTF-8"? >
< cvssv3.0 xmlns="https://www.first.org/cvss/cvss-v3.0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.first.org/cvss/cvss-v3.0.xsd
https://www.first.org/cvss/cvss-v3.0.xsd" >

< base_metrics >
< attack-vector> NETWORK < /attack-vector >
< attack-complexity> LOW < /attack-complexity >
< privileges-required> NONE < /privileges-required >
?
```

Refer to the source code for more details on how errors are returned.

B.4.6 XML Schema Definition

It is sometimes useful to represent the CVSS metric values and scores for a vulnerability in XML format, e.g. to transfer CVSS data between systems.