Fall 12-1-2016

Distributed Multi-component Approach and System for Enhanced Security of Public Infrastructure as a Service (IAAS) Cloud Computing Environments

Jason Nikolai
Dakota State University

DISTRIBUTED MULTI-COMPONENT APPROACH AND SYSTEM FOR ENHANCED SECURITY OF PUBLIC INFRASTRUCTURE AS A SERVICE (IAAS) CLOUD COMPUTING ENVIRONMENTS

A dissertation submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Doctor of Science

in

Information Systems

December, 2016

By
Jason Nikolai

Dissertation Committee:

Dr. Yong Wang
Dr. Jun Liu
Dr. Mark Hawkes
Dr. Deepak Turaga, IBM Research

DISSERTATION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Science in Information Systems degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: _____

Dissertation Title: _____

_____

Dissertation Chair/Co-Chair:_____ Date: _____

Dissertation Chair/Co-Chair:_____ Date: _____

Committee member:_____ Date: _____

Committee member:_____ Date: _____

Committee member:_____ Date: _____

Committee member:_____ Date: _____

DAKOTA STATE UNIVERSITY

DISSERTATION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Science in Information Systems degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: Jason Nikolai

Dissertation Title: Distributed Multi-component Approach and System for Enhanced Security of Public Infrastructure as a Service (IaaS) Cloud Computing Environments

Dissertation Chair: Y Wang Date: 11-16-2016

Committee member: Date: 11-18-2016

Committee member: Date: 11-21-2016

Committee member: Date: 11-22-16

ACKNOWLEDGMENT

It is believed that Helen Keller once said “Alone we can do so little; together we can do so much.” The work presented here supports this premise. This dissertation would not have possible without the support of several people:

- My wife Danielle and children Kyler and Trever: Throughout my doctoral tenure, my family provided support and sacrificed every step of the way alongside me.
- My advisor, Dr. Yong Wang: Yong’s guidance and intellectual discussions were pivotal in my success.
- My parents, Karen and Al: They taught me the work ethic needed to persevere and overcome obstacles in life, including my seven year journey to complete this dissertation.
- My third line manager at International Business Machines (IBM) and IBM Fellow, Nagui Halim: Nagui provided guidance and support at a critical time in the dissertation process which resulted in my overall success.
- My committee: For reviewing my research, providing comments, and for granting final approval for graduation.
- My employer, IBM, and the management team: IBM supported me through this journey and gave me the opportunity to expand my knowledge and a chance to give back to the research community.
- Co-workers at IBM: Several of my co-workers provided insight on ideas, helped me with coding problems, discussed theoretical questions, and tolerated my sporadic schedule.

All of these amazing people contributed to my success. Their contributions will never be forgotten.

ABSTRACT

Cloud computing has become big business with organizations spending millions of dollars creating and deploying cloud solutions. However, adoption of this multi-tenant and dynamic technology has been slowed by security concerns. In this dissertation, to help increase adoption by reducing security risks, we examine three research questions. First, how can we detect attacks on cloud tenant instances without specific knowledge of tenant applications? Second, how can we assist cloud providers with interpretation of the alert output from security controls in an IaaS cloud environment to improve security? And, third, how can we help protect cloud tenants from insider data theft attacks?

To answer these questions, we utilize the design science research methodology to accomplish the objective of creating and demonstrating a new system composed of novel security controls addressing each research question. We posit a system comprised of three security control artifacts to assist cloud providers with improving their overall security posture. Our proposed system consists of three components: A Hypervisor-based Cloud Intrusion Detection System (HCIDS), a Streaming Cloud Intrusion Monitoring and Classification System (SCIMCS), and a system for detecting insider attacks within cloud computing environments.

First, HCIDS utilizes data from hypervisors running on cloud controller nodes to detect and classify abnormal usage. Instantiation and demonstration of the system reveals a 100 percent detection rate for denial of service attacks from and against virtual machines. Second, SCIMCS addresses the problem of information overload from alerts generated by security controls in dynamic multi-tenant cloud environments. Implementation and evaluation of this approach divulges an average message reduction rate of 95.9 percent based on our experimentation. Third, the system for detecting insider data theft examines node system state and anomalies in network bytes transmitted as well as number of active user counts to detect virtual machine and data store theft. This approach demonstrates a 100 percent detection rate for data theft and unapproved logins on cloud nodes.

Each of these components plays a unique role in improving the overall security posture in Infrastructure as a Service (IaaS) Cloud Computing Environments. The combination of each approach makes up an overall system that addresses intrusion detection

while preserving privacy, information overload from a plethora of controls deployed in a defense in depth strategy, and the concern of insider data theft. Furthermore, each component is designed, instantiated, demonstrated and communicated at respected conferences.

Declaration

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

_____

Jason Nikolai

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

On August 25, 2006, Amazon EC2, one of the leading Infrastructure as a Service cloud providers, went into beta (Barr, 2006). Since then, cloud computing has become big business with organizations spending millions of dollars creating cloud solutions. The largest technology companies in the world now provide cloud computing offerings and solutions (Google, 2015; IBM, 2015; Microsoft, 2015). However, cloud computing is not without its challenges. More specifically, since the inception of cloud computing, security and privacy have been an ongoing concern that has slowed adoption of the technology (Kandukuri, Paturi, & Rakshit, 2009; Lori, 2010; Ren, Wang, & Wang, 2012; Takabi, Joshi, & Ahn, 2010). In chapter one, we explore the background of our research problem, provide a concise problem statement, and define the objectives of this dissertation project.

Background of the Problem

In 2014, IDG Enterprise reported the results of a survey on cloud computing which revealed that 45 percent of respondents had cloud projects return to internal information technology teams. Furthermore, 59 percent of the respondents who indicated projects were brought back in house stated security concerns were a contributing factor (IDG, 2014). The number of cloud providers and organizations considering cloud computing as a paradigm for deploying information technology solutions combined with the continuing concern over security makes cloud computing security an interesting and worthwhile research topic to explore.

In order to comprehend the importance of this problem, one first must have an understanding of cloud computing. Cloud computing can be defined as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Armbrust et al., 2009). In a general sense, a cloud computing environment can be defined as public or private

interconnected computers that provide shared computing power without presenting the underlying structure (Biggs & Vidalis, 2009).

A more precise definition of cloud computing is provided by NIST and illustrated in Figure 1. The NIST definition of cloud computing states that the cloud model consists of five essential characteristics, three service models, and four deployment models (Mell & Grance, 2011). The essential characteristics consist of broad network access, rapid elasticity, measured service, and on-demand self-service. The service models include Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (Iaas). The deployment models are made up of Public Cloud, Private Cloud, Hybrid Cloud, and Community Cloud.

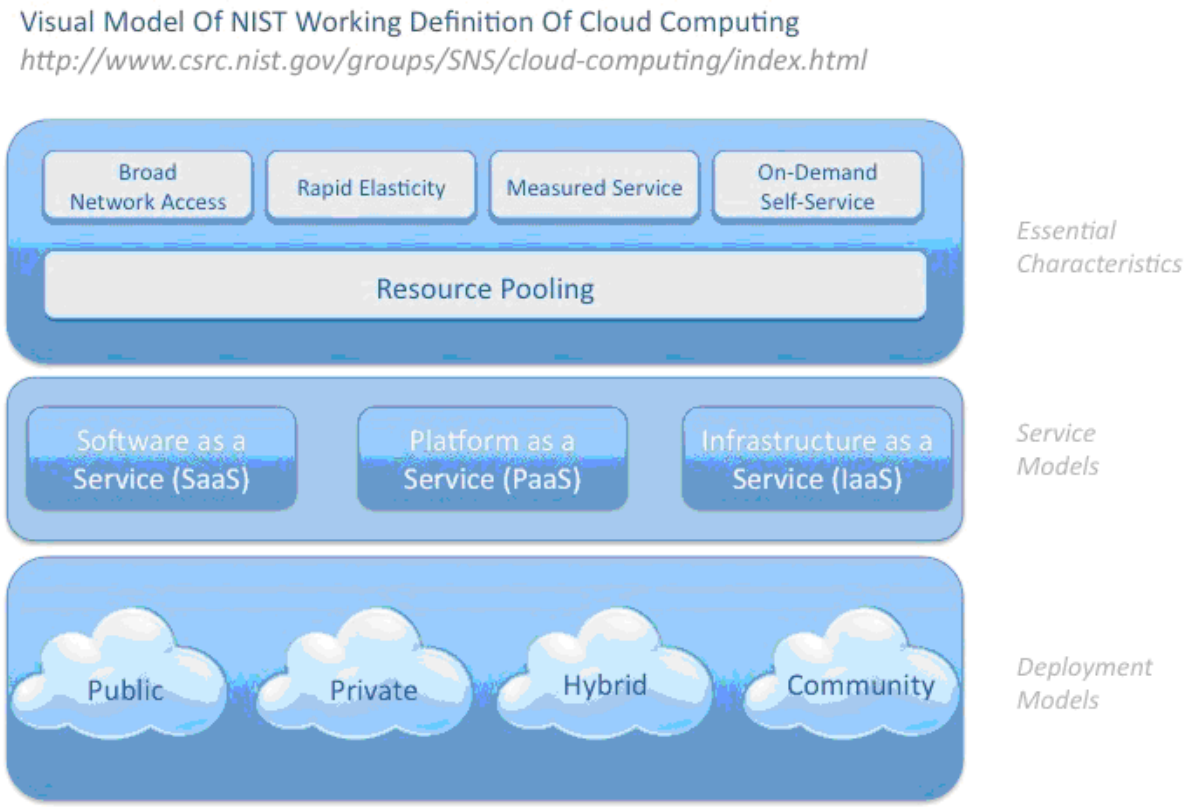

Figure 1. Illustration of NIST Cloud Computing Definition

As stated, cloud computing spans several service and deployment models. However, the essential characteristics are crucial to the definition. Unlike previous IT environments, cloud computing environments provide broad network access, typically from the Internet. In

addition, resources can be provisioned and deprovisioned quickly, charges are generally based on usage, and tenants of these environments perform provisioning and usage activities without extensive interactions with IT departments or administrators. Furthermore, resources are shared, many times between different tenants. The multi-tenant nature and lack of control over the underlying infrastructure open up several security challenges and questions.

The focus of our research is on Public Infrastructure as a Service (IaaS) cloud environments. Infrastructure as a Service cloud environments provide cloud tenants with the capability to provision computing resources to run arbitrary software and operating systems. The tenant does not control the underlying infrastructure but does have control over the provisioned instance. Essentially, tenants are given access to computing resources in order to perform computing activities as needed. The public deployment model provides unique challenges over other hosted information technology environments in that the environment is shared possibly among tenants, some of whom may be competitors or bad actors.

To secure these environments, the Cloud Security Alliance, a leading group on cloud computing security, published the security guidance report that recommends a security approach which utilizes a defense in depth (SANS, 2001) strategy where people, process, and layers of technology all play a role in securing a cloud environment (Cloud Security Alliance, 2011). Their approach to securing IaaS cloud environments involves processes, procedures, controls, and audits. No single technology has been shown to address all of the security concerns in cloud environments. Furthermore, managing multiple processes, procedures, and controls in an effective manner is nontrivial with many open research questions.

Statement of the Problem

Our research aims to address security and privacy concerns through the instantiation of a system composed of novel technical cloud security artifacts. More specifically, we examine the problems of detecting bad actors in cloud environments earlier while preserving the privacy of cloud tenants. We posit three design science research questions. First, how can we detect attacks on cloud tenant instances without specific knowledge of tenant applications in order to preserve privacy? Second, how can we assist cloud providers with interpreting the output from security controls in an IaaS cloud environment to improve security? And, third, how can we help protect cloud tenants from insider data theft attacks?

Objectives of the Project

Our research contribution from this work is demonstrated through three security artifacts making up a system to assist cloud providers with securing Infrastructure as a Service (IaaS) cloud computing environments. Although no one technical solution is going to completely remove the security threats and risks within multi-tenant cloud computing environments, additional security artifacts will help to improve the overall security posture of these environments. As the security posture in these environments improves, cloud tenants will gain more confidence in the security of these environments. We believe that the long term end result will be less risk for cloud tenants resulting in increased adoption of IaaS cloud computing.

Summary

In this chapter, we introduced our research topic and provided an introduction to cloud computing and some of the security issues. Furthermore, at a high level, we described the general security challenges facing these environments, stated our research problem, and provided the objectives of our research project. Chapter two provides a more detailed discussion on cloud computing and a literature review of security related challenges associated with these environments.

CHAPTER 2

LITERATURE REVIEW

Cloud computing provides several opportunities for organizations to optimize resources and reduce costs. However, these benefits do not come without challenges and security risks. This chapter is broken into two parts, both based on existent literature. First, we briefly examine the history and provide a detailed definition of cloud computing. Second, we present the findings of our cloud computing security challenges literature review.

History and Definition of Cloud Computing

In a general sense, a cloud computing environment can be defined as public or private interconnected computers that provide shared computing power without presenting the underlying structure (Biggs & Vidalis, 2010). The idea of cloud computing is not new. The original concepts date back to the 1960s (Kaufman, 2009). Cloud computing as it is known today has emerged from the construction of large scale commodity-computer datacenters combined with advances in the World Wide Web and Web 2.0 (Armbrust et al., 2010). Developments in virtualization technology on commodity hardware helped to provide the underlying technology for rapid provisioning and de-provisioning of resources in a cost effective manner. In order to understand cloud computing and associated security issues, one first must understand the different services offered and deployment models. The consensus within literature describes cloud computing as one of following services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

IaaS can be described as offering information technology infrastructure as a service. This is similar to outsourcing the datacenter. However, unlike traditional data center outsourcing, cloud tenants use self-service mechanisms to provision and deprovision resources and may share these environments with other unrelated and unknown tenants. In addition, cloud customers are typically charged for the resources that they use. Examples of IaaS include: Amazon AWS, IBM Softlayer, and Oracle's Cloud Service.

The PaaS model can be described as a shared development environment that is provided as a service. The provider offers blocks of code, or application program interfaces (APIs), which developers use to create web based applications that are stored in the providers cloud environment. Some examples of PaaS are: IBM Bluemix, Force.com, and Google App Engine.

In the SaaS model, end users do not purchase software and store it locally. Rather, software remains in a remote location referred to as a "cloud." The customer pays based upon usage. The backend infrastructure is physically shared among customers, but it is logically divided among end users. Examples of Saas include: Google Docs and Zoho (Almulla & Chan Yeob, 2010).

There are three deployment models in cloud computing: private cloud, public cloud, and hybrid, or mixed, cloud environment. Private cloud is for internal use by an organization. Private clouds reside within an organization's internal data center. Because the organization controls the data center, this cloud model is as secure as the data center. Public clouds are cloud computing environments that are hosted by providers, available to the general public, and are typically based on a pay per use model. In public implementations and offerings, the cloud computing infrastructure is available via the Internet (Katzan, 2010). A mixed cloud environment may be a community cloud or a hybrid cloud deployment. Community cloud is a shared cloud among multiple organizations usually with a shared interest. Hybrid Clouds are a mix of public and private cloud environments. Hybrid clouds generally consist of a private cloud with interfaces to external cloud services. Hybrid clouds provide organizations with cloud computing advantages but with less risk than public clouds (Katzan, 2010; Ramgovind, Eloff, & Smith, 2010).

Cloud computing provides two significant advantages over other technologies: elasticity and cost. First, it provides organizations with flexibility to scale up or scale down their information technology as needed to meet the demands of the organization. From a physical resource perspective, cloud computing allows organizations to provision new resources as needed which relieves some of the planning burden. Cloud users can start using a small environment and provision upward as needed and only pay for what is used (Armbrust, et al., 2010; Doelitzscher, Reich, & Sulistio, 2010). This feature, typically termed elasticity, is one of the primary features that sets cloud computing apart from previous, similar

technologies (Owens, 2010). Second, it provides computing power at an affordable cost that would not otherwise be available to users previous to the inception of cloud computing (Kaufman, 2009).

Some of the key characteristics of cloud computing include: on-demand self-service, broad network access, resource pooling, and measured service. Thanks to the increase in affordable network bandwidth, reliable networks, and the Internet, it is possible for cloud computing providers to offer high quality data and software services that reside in remote data centers (Cong, Qian, Kui, & Wenjing, 2009). Cloud computing can provide cost effective pay-as-you-go information technology environments (Biggs & Vidalis, 2010).

Organizations that leverage cloud computing will likely achieve savings in hardware, software, time provisioning servers, productivity, and system administration. Savings in hardware will occur due to the reduction in the number of servers required to complete work, less data center floor space, and reduction in power consumption. Software costs will likely decrease due to less operation system licenses being needed and reduced software maintenance cost. Server provisioning takes less time with automated provisioning technology and tools. Productivity can be improved by cloud support staff rapidly responding to end user requests. Even system administration costs will likely decrease as less system administrators will be required to manage more systems (Almulla & Chan Yeob, 2010).

The benefits are clear and likely to result in continued adoption of the cloud computing paradigm. This is evident based on a October 30, 2009 report released by Gartner Inc. that stated cloud computing has become a top significant technology issue (Chang-Lung, Uei-Chin, Chang, & Chun-Jung, 2010). However, this paradigm might not be appropriate for all industries (Kaufman, 2009). A survey of federal government agencies funded by the Lockheed Martin Cyber Security Alliance found that 70 percent of those surveyed were concerned about data security, privacy, and integrity in a cloud computing environment (Anonymous, 2010). In another survey of more than 170 businesses, 50 percent of the respondents stated concerns with security issues relating to cloud computing resources (Biggs & Vidalis, 2010). Multiple surveys have shown security to be ranked first as the greatest challenge or issue of cloud computing (Popovic & Hocenski, 2010). Although cloud services can relieve organizations of hosting burdens and reduce costs, a number of security concerns continue to plaque the cloud computing paradigm.

Cloud Computing Security Challenges Literature Review

Our literature review is conducted in two parts. First, we perform a general literature review of cloud computing security issues in this chapter. Second, chapters four, five, and six contain a related works section which presents the results of literature reviews specifically targeting the components of the system posited in this dissertation. The literature review in this chapter follows the methodology outlined in Figure 2.

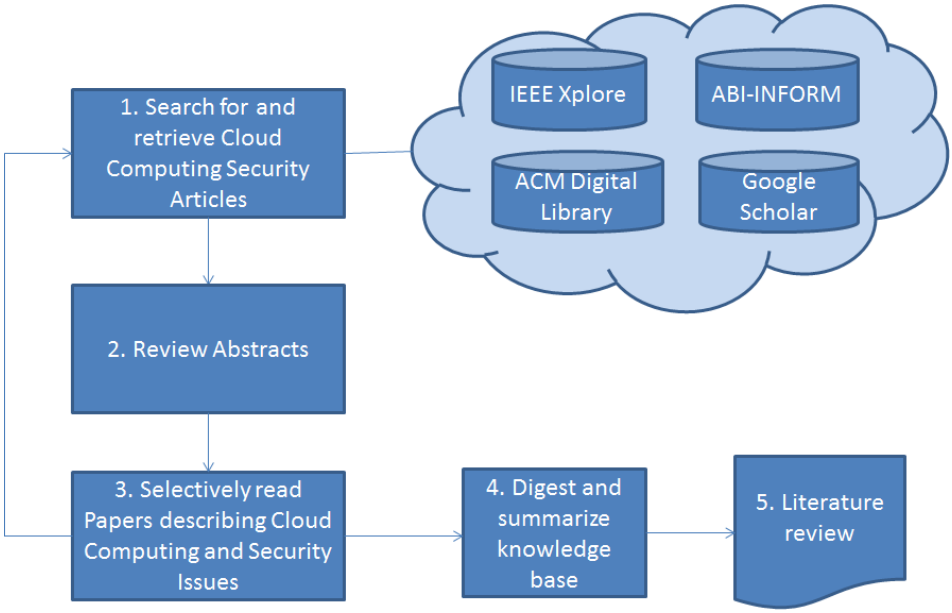

Figure 2. Literature Review Methodology

The literature review revealed 16 general security topics relating to cloud computing security: Auditability, Availability of Data, Data Location, Data Segregation, Data Storage Correctness, Disaster Recovery, Investigative Support, Long Term Viability / Data Lock-in, Performance, Privacy, Regulatory Compliance, Reputation Fate Sharing, Restricted Access, Security Controls, Trust and SLAs, and Trusted Interfaces. We summarize each of these topics below:

Auditability

Cloud computing does not necessarily offer a guarantee for data integrity and availability. It is of critical importance to enable public auditability for cloud data storage so that the tenants have the ability to use a third party auditor for achieving appropriate risk

levels for the privacy protection of data (Armbrust, et al., 2010; Cong, Qian, Kui, & Wenjing, 2010).

Availability of Data

The outsourced nature of cloud computing raises concerns of ensuring availability of data. When an organization moves data and applications to a cloud environment, it loses control of the data which makes ensuring data availability difficult (Zhang, Wuwong, Li, & Zhang, 2010). In addition, data availability includes the capability of a cloud provider to move data to alternative environments if one environment becomes compromised (Armbrust, et al., 2010; Ramgovind, et al., 2010). An example of where data availability became an issue in a cloud computing environment is when Liquid Motors lost all of its servers in a data center raid, then lost its suit against the FBI (Neumann, 2009).

Cloud availability threats also include risks from network based attacks, such as Distributed Denial of Service (DDoS) attacks, as well as the cloud service provider's environment setup and competence level (Almulla & Chan Yeob, 2010; Owens, 2010). A proper risk management process is needed to address the risks of moving applications and data into a cloud computing environment to ensure availability (Messmer, 2009).

Data Location

Data location is concerned with the physical location of data and whether a cloud provider will allow a tenant to dictate where data is located (Barnes, 2010; Doelitzscher, et al., 2010; Ramgovind, et al., 2010). There are confidentiality issues pertaining to where data is stored and where the data has travelled. In a cloud environment, data is relocated for optimized storage, but each time when the data is moved, a copy may be retained at the location (Acello, 2010; Almulla & Chan Yeob, 2010).

One example of where location plays a critical role is with Germany's Federal Data Protection Act. This act states that personal data can only be transferred for processing into countries with the same adequate level of privacy protection laws. Whenever personal data is acquired and/or processed by third-party instances, the affected person must be notified according to this act. Users must know the exact location of their data and the cloud provider's court of jurisdiction (Doelitzscher, et al., 2010).

Data Segregation

Data segregation is the assurance that data is separated using trusted encryption techniques and technologies (Barnes, 2010; Ramgovind, et al., 2010). Data isolation is not trivial in a multi-tenant environment. Appropriate tools are required to ensure proper data protection is available in shared environments (Naqvi, Dallons, & Ponsard, 2010). In addition, concerns around virtualization security exist, including threats specific to virtual environments and shared hypervisors. Although it may be cost-effective to use a shared administrative management system to manage multiple customer environments, concerns around data segregation exist (Owens, 2010).

Data Storage Correctness

Data is typically stored in multiple physical locations in a cloud environment and can move to other locations rapidly. Maintaining data integrity in these dynamic environments may present technical issues that increase data integrity risk if not properly managed. Tenants must be assured that the cloud provider can competently manage the complexity of these dynamic environments (Cong, et al., 2009).

Disaster Recovery

Recoverability encompasses the ability to recover data in the cloud environment when an unplanned event occurs. Before moving data or applications to a cloud computing environment, an organization should understand the cloud computing providers plan for recovering from disasters (Barnes, 2010; Ramgovind, et al., 2010).

Investigative Support

Investigative support is the ability of the vendor to provide forensic analysis and investigative support when illegal activities occur (Barnes, 2010; Ramgovind, et al., 2010). If a security breach occurs, gathering evidence from cloud computing environments can be difficult because of the underlying dynamic nature of the environment. Data is typically spread dynamically across multiple hosts and data centers and maintaining chain of custody during investigation can be problematic (Biggs & Vidalis, 2010; Wolthusen, 2009).

Legal and regulatory concerns relating to cloud computing jurisdiction of data that crosses borders are nontrivial. It is not clear if governments have access to cloud data that spans borders (Kaufman, 2009). Cross-border placement of data can result in compliance issues and can hinder cybercrime investigations (Biggs & Vidalis, 2010).

Long Term Viability / Data Lock-in

Long term viability refers to concerns surrounding the ability to retrieve data from a cloud computing environment if the provider no longer meets the needs of the tenant or goes out of business. There is an ongoing concern about proprietary data formats and whether data can be retrieved in the case the tenant wishes to leave the cloud environment (Armbrust, et al., 2010; Barnes, 2010; Ramgovind, et al., 2010).

Performance

A concern exists relating to performance of the cloud environment and data transfer bottlenecks (Armbrust, et al., 2010). Virtualization technologies and shared environments may slow processing capabilities of systems. Furthermore, security controls protecting data flowing over the Internet and being processed as well as stored within a cloud environment may reduce performance. Although benefits can be achieved by leveraging cloud computing for certain applications, there are likely hidden operational and performance costs (Bauer, 2010).

Privacy

When a cloud provider houses large amounts of data, data mining techniques can be used to derive personal information about tenants. For example, the Google Corporation offers multiple cloud services and has access to a plethora of data (Chow et al., 2009). Some have described cloud computing as being similar to a utility. However, unlike a utility, such as electricity, where an attacker is not interested in accessing a specific electron, an attacker may be interested in the data stored and transferred into a cloud computing environment.

In addition, ownership of data should be understood when considering moving sensitive data to cloud environments. Does the cloud provider own the data? Does the organization placing the data into the environment own the data? Or, if the tenant is running

an application in a cloud environment that contains tenant customer data, does the customer with data stored in a data record own the data (Katzan, 2010)? In some cloud computing environments, a fundamental right to privacy is expected. However, techniques such as anonymous authentication make it difficult or impossible to track the real user if a transaction is disputed (Lu, Lin, Liang, & Shen, 2010).

Regulatory Compliance

Regulatory compliance encompasses the issues around meeting the regulatory needs of an organization (Barnes, 2010; Ramgovind, et al., 2010). Careful measures must be in place to comply with government regulations and industry standards, such as FFIEC (Federal Financial Institutions Examination Council), HIPPA (Health Insurance Portability and Accountability Act), and PCI DSS (Payment Card Industry Data Security Standards) (Katzan, 2010). Many of these controls must be incorporated into the environment by the cloud provider and not the tenant. Concerns around controls, such as encryption, must be taken into account when assessing the compliance related risks in these environments. Questions, such as whether the cloud provider has passed a SAS-70 audit, should be asked (Messmer, 2009). Transparency is especially important for regulatory reasons (Chow, et al., 2009).

In addition, from the enforcement side, the definition of what constitutes compliance will not be fully clear until judges and regulators have a better track record and case history to dictate what is expected and reasonable (Ericson, 2009).

Reputation Fate Sharing

Cloud computing is a shared environment. Hence, if not properly managed, one bad actor in the cloud environment may have a negative impact on other tenants. For example, if one tenant is compromised and IP addresses become black listed, other consumers of the cloud may be impacted by no fault of their own (Armbrust, et al., 2010). A tenant running a critical application for customers in a cloud environment may become unavailable. Unfortunately, the customers of that tenant will likely fault the tenant and not the cloud provider.

Restricted Access

Restricted access refers to allowing only those who should be permitted to access data to actually have access to the data. An organization that is placing data in a cloud computing environment must be certain that the cloud provider has competent and honest administrators who have put appropriate access controls in place (Barnes, 2010; Ramgovind, et al., 2010). In addition, proper network security measures are needed to ensure only appropriately authorized users can access systems, applications, and network data in these shared environments (Kaufman, 2009).

Security Controls

Different cloud providers have differing levels of security controls in place. One provider may have a well-established and respected information security program with controls and monitoring in place while another provider may not. It is important to understand the security controls and the security postures of cloud providers before placing trust in their environment (Kaufman, 2009).

A cloud environment is only as secure as its weakest link. The multi-tenant nature of cloud environments make for prime targets of cybercriminals. The movement towards increased hosting of data and applications in the cloud and less reliance on user machines for storing private data is likely to increase the threat of phishing and other attacks targeted at stealing access credentials (Chow, et al., 2009).

Trust and Service Level Agreements (SLAs)

Data that is moved into the cloud is under control of a third party provider. Access to that data is in the hands of that cloud provider. A concise service level agreement between the tenant and cloud provider must be in place in order to reduce risks especially with regard to availability. The cloud provider must be trusted to deliver on the commitments in the SLA (Ramgovind, et al., 2010). Service Level Agreements are a key component to defend against cybercrime and must evolve to counter dynamic attacks from cybercriminals. These agreements must be well written and monitored (Biggs & Vidalis, 2010).

In addition to a standard SLA, a Sec-SLA which is a formally negotiated document that defines security metrics for a cloud computing environment may be considered. However, for a Sec-SLA to provide value, it must be monitored and enforced which requires buy-in from the cloud provider (de Chaves, Westphall, & Lamin, 2010).

Trusted Interfaces

Cloud computing is primarily managed through network connections. Public clouds are managed through the Internet and proper controls are needed to ensure the interface is secure (Chang-Lung, et al., 2010). It is critical that interfaces are secure and encryption protocols, such as TLS, are up to date to ensure known vulnerabilities are not used to exploit weaknesses.

There are a vast number of security-related issues to be considered before an organization moves its application and data into a cloud computing environment. Some of these challenges present interesting research topics to be explored and solved. The cloud computing paradigm introduces new concerns that must be addressed in order to achieve high levels of adoption for all types of applications and data. Depending on an organization's security needs, regulatory compliance requirements, and customer demands, the issues summarized in this chapter should be considered before moving applications and data into a cloud computing infrastructure.

Cloud computing certainly has a number of advantages over self-managed data centers and applications. There is potential for reduced hardware and operating costs along with the ability to scale on demand. However, it is clear, security challenges are prevalent in these environments. Organizations considering moving applications and data into a cloud computing environment must assess the risks and ensure that proper controls and mitigation plans are in place to achieve an acceptable risk level. Although risks can be mitigated, in order for cloud computing to thrive to its full potential, more of the security challenges must be addressed (Lu, et al., 2010).

Summary

In this chapter, we provided a history and definition for cloud computing. In addition, we presented our findings from a general literature review on cloud computing security challenges. The driving force behind our research is to address a subset of these challenges. More specifically, as mentioned in chapter one, our primary research is focused on the instantiation of new security artifacts for cloud environments to improve the overall security posture while preserving tenant privacy with the goal to increase adoption of IaaS cloud computing environments. Chapter three examines the research methodology we follow while conducting our work.

CHAPTER 3

RESEARCH METHODOLOGY

The information systems discipline studies people, organizations, and technology (Hevner, March, Park, & Ram, 2004). Generally, there are two paradigms used to conduct this research: behavioral science and design science. Behavioral science is descriptive in nature and attempts to explain phenomena related to information technology. On the other hand, design science is prescriptive by definition and aims to improve the performance and the results gained from using information technology (March & Smith, 1995). Both approaches are used by researchers in the information systems discipline and are complimentary in nature. In this chapter, we briefly examine the design science research methodology. Then, we discuss the design science research approach that we follow while conducting our research.

Design Science Research

The goal of design science research is to create a means to achieve or better achieve human goals. This differentiates the methodology from other methodologies, especially those used in the natural sciences which attempt to explain natural phenomenon but not create them. Design research may be evaluated and improved upon through research activities where the researcher executes a build and evaluate process. The build and evaluate research steps may be followed by theorizing and justification steps which are the activities similar to those found in the natural sciences. First, during the build phase, the artifact is constructed. Second, the evaluate step consists of evaluating the artifact's performance against a set of criteria. Third, the theorizing phase attempts to explain the interactions with the artifact and the environments as well as its characteristics. If theories are posited, they must be justified with evidence in order to test the theory (March & Smith, 1995).

According to March and Smith, the outputs from design science include four categories, which are also termed artifacts. They include constructs, models, methods, and implementations. Constructs define the language of the concepts in the domain. Models can

conceptualize constructs in order to describe tasks, circumstances, or artifacts. Methods can be described as the steps for accomplishing activities. And, finally, implementations are the instantiated product (March & Smith, 1995).

Hevner, et al. posit seven guidelines for design research (2004). These guidelines can be used by researchers to develop quality design research which contributes artifacts to the information systems (IS) knowledge base. Unlike the natural sciences which examine, understand, and predict phenomena that occur in nature, design science is concerned with the study of the artificial and manmade objects. Hevner, et al. suggest that there are five outputs from design research: constructs, models, methods, instantiations, and better theories. Furthermore, the general methodology of design research consists of five process steps: awareness of problem, suggestion, development, evaluation, and conclusion (Vaishnavi & Kuechler, 2004/5).

Design research is a critical component for the IS discipline for the simple fact that the entire discipline of information systems studies artifacts that have been designed by humans to accomplish the goals of humans (March & Smith, 1995). Some argue that the relevance of information systems is directly related to how research can be applied to design. Figure 3, below, depicts an information systems research framework posited by Hevner, et al.

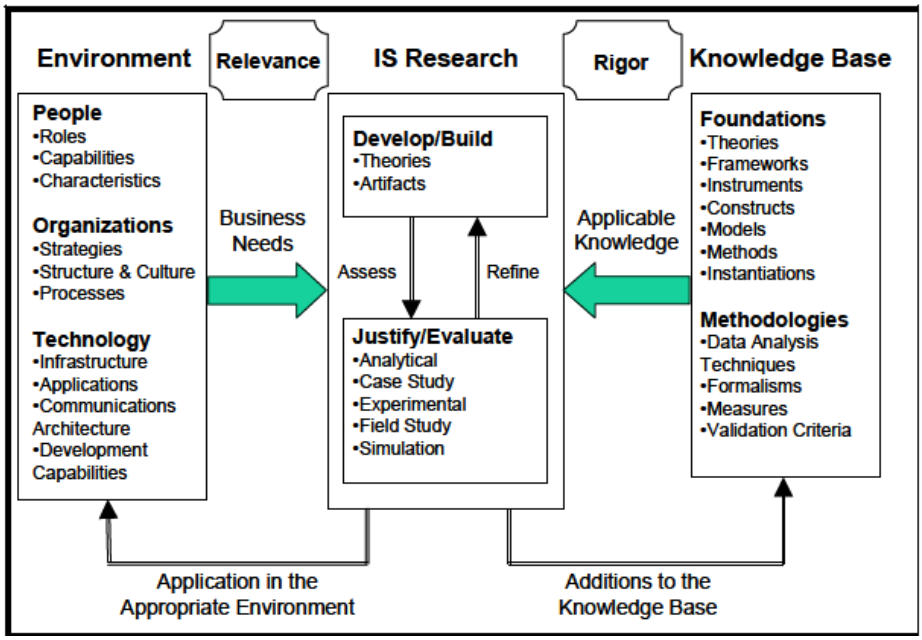

Figure 3. Information Systems Research Framework (Hevner, et al., 2004)

Figure 3 illustrates how business needs and applicable knowledge results in new theories and artifacts. From a design research perspective, new artifacts are developed, refined and then both pragmatic and scholarly contributions are made resulting in both relevant and rigorous research.

The goal of information systems research is to explore the intersection of people, organizations, and technology (Silver, Markus, & Beath, 1995). While the behavioral science aspect of information systems is reactive, design research is proactive and attempts to bring into being new artifacts to solve problems that have utility (Hevner, et al., 2004). Design science research extends the state-of-the-art in the information systems domain by expanding the boundaries of known applications of information technology and by exploring problems that may have not been thought but could be approached using technology (March & Storey, 2008).

Our Approach to Design Science Research

The aim of our research is to bring new relevant cloud computing security artifacts into being. In order to ensure research rigor and differentiate our work from design by production, we follow the Peffers, et al. design science research methodology process model shown in Figure 4 (2007).

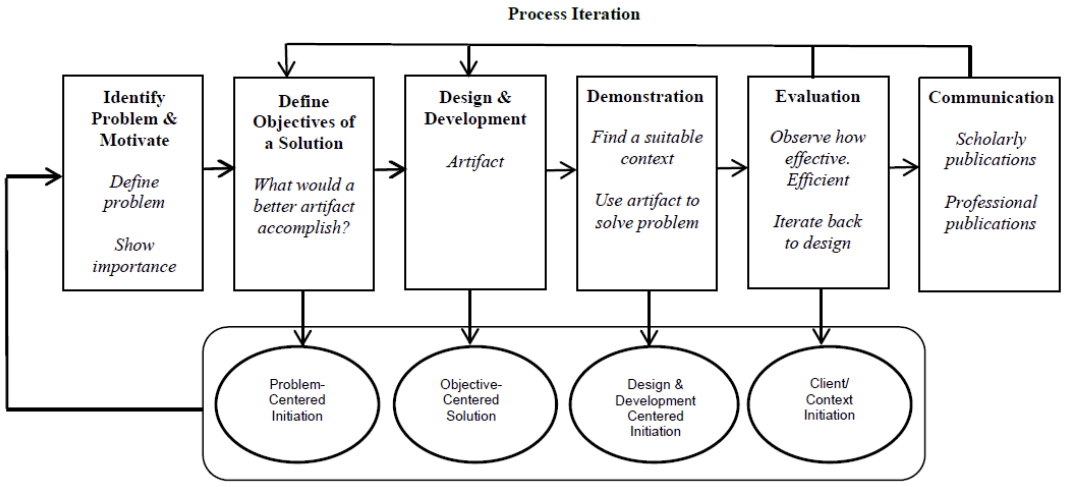

Figure 4. Design Science Research Methodology Process Model

Our approach begins by identifying the problem and motivation behind the problem by conducting a general literature review of open problems in IaaS cloud computing security. Based on the literature review, we find gaps in security controls used within cloud computing environments. First, our work identifies specific security control gaps within Infrastructure as a Service (IaaS) Public Cloud Computing environments which are used to develop the motivation behind our work. The output of the literature review leads us down three areas of research: a new security control to detect attacks at the hypervisor level, a novel system for managing and acting upon security control alerts, and an approach and artifact for detecting insider data theft. Second, we define the objectives of our multi-part solution. Third, we design and develop artifacts that address the objectives of our research. Fourth, we demonstrate the effectiveness of our artifacts. Fifth and sixth, we evaluate the effectiveness of our artifacts and system through experimentation and communicate our work to professional and scholarly communities through publications. Each of these phases are discussed in more detail in chapter four.

Summary

In this chapter, we discussed the design science research methodology and our approach to applying this methodology to our work. Chapter four provides an overview of our system which is composed of the artifacts derived from our research. Furthermore, a mapping of each component to the Peffers, et al. Design Science Research Methodology is discussed along with how our work generally fits within this methodology.

CHAPTER 4

RESEARCH COMPONENTS AND SYSTEM

Following the design science research methodology, we contribute to the information systems and computing knowledge base by bringing new artifacts into being. These components are derived following the Peffers, et al. Design Science Research Methodology as mentioned in chapter three. In order to better understand the relevance of our work, it is important to view each research artifact as a component in a larger system. In this chapter, first, we introduce each of the research components. Second, we discuss how the components fit into an overall system. And, third, we elaborate on the methodology used to construct the artifacts and provide a mapping to information systems design science research concepts.

Research Components

The aim of our research is to address three security challenges in Infrastructure as a Service cloud computing environments. First, we derive a new approach for detecting potential bad actors in the cloud environment by analyzing anomalies in hypervisor performance data. This approach provides a level of privacy for the cloud tenants while offering an additional security control to improve bad actor detection and thwart cloud attacks. Second, we research and develop a methodology and system for reducing the number of alert messages from security controls in a cloud environment. A defense in depth approach is recommended for securing IT infrastructures. However, managing security controls across a vast number of systems can be a challenging task and ineffective if the appropriate alerts are not properly acted upon. Third, our research examines the problem of data theft and insider attacks in cloud environments. Cloud providers generally house multiple tenants who may place sensitive assets into these environments. It is crucial that the providers not only operate and manage controls to prevent outside attacks, but also attacks from within. In cloud environments, insider attacks may occur from actors employed by the cloud provider as well as ill-intentioned tenants wishing to steal data or other assets.

IaaS Cloud Security System

The three components of our research make up an overall system for enhancing the security of IaaS cloud computing environments, and include a Hypervisor-based Cloud Intrusion Detection System (HCIDS), a Streaming Intrusion Monitoring and Classification System for IaaS Cloud (SCIMCS), and a System for Detecting Malicious Insider Data Theft in IaaS Cloud Environments. Figure 5, below, illustrates how the individual components of our research work together to make up our distributed multi-component approach and system for enhanced security of public infrastructure as a service (IaaS) cloud computing environments.

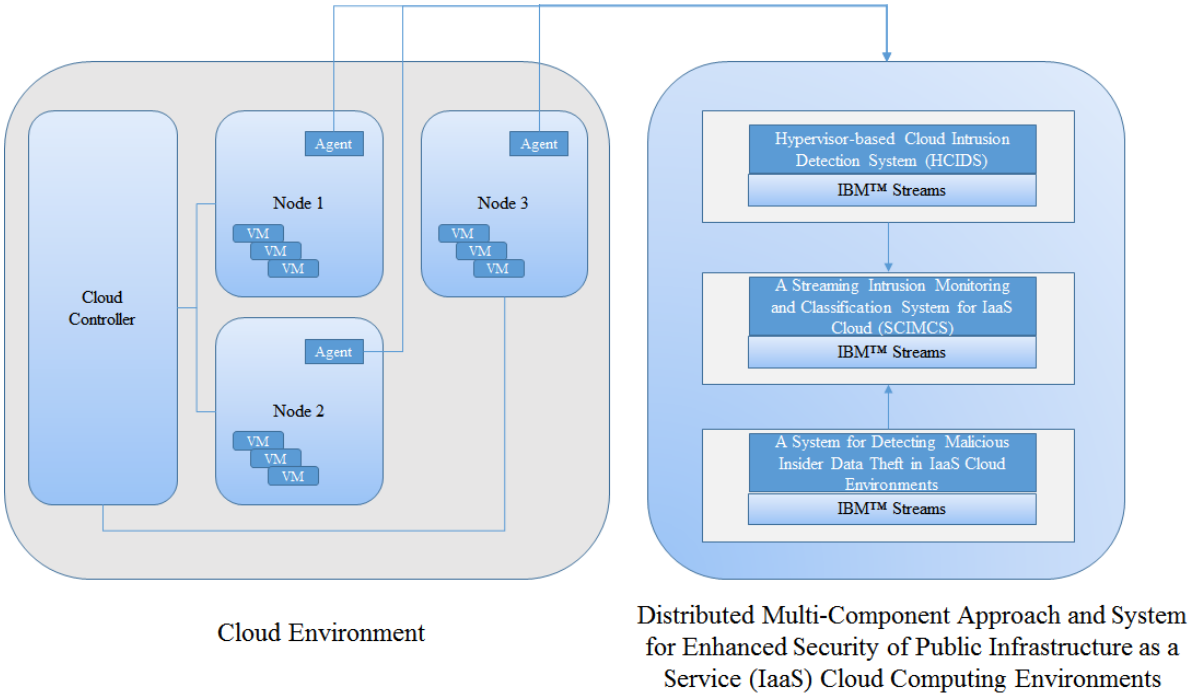

Figure 5. System Perspective of Research Components

Agents are installed in the cloud environment on the physical nodes that house tenant virtual machines. These agents collect network and system data from various sources and feed that data into the components of our system. More specifically, first, the Hypervisor-based Cloud Intrusion Detection System receives metric data from the hypervisors which control the tenant virtual machines. These metrics are used to detect anomalous behavior, then compare the anomalous patterns with known attack patterns and classify attacks. Second, the Streaming Intrusion Monitoring and Classification System for IaaS Cloud ingests

message alerts from security sensors in the cloud environment, including alerts from the other two research components. This system presents potential Black Swan events to the security administrator to help assist with triaging high priority events first. In addition, when trained, the system displays the type of attack based on previously observed patterns. Third, the System for Detecting Malicious Insider Data Theft in IaaS Cloud Environments examines login activity and data transfers on the physical node. Abnormal patterns in these two events coupled with system state data are shown to detect potential insider data theft in cloud environments with a high level of accuracy.

While each component contributes to the knowledge base, the combination of these three components makes up a system that provides a new set of security controls to improve the security posture of IaaS cloud computing environments. Future chapters in this dissertation reveal the detailed research and results for each component.

Research Methodology and Research Rigor

As discussed in chapter three, our research is accomplished using the design science research methodology. More specifically, we follow the Peffers, et al. Design Science Research Methodology to ensure research rigor. Table 1, summarizes the mapping of our research to the model.

Table 1. Design Science Methodology Component Mapping

Peffers, et al., Design Science Research Methodology Step	**Chapter 5: Hypervisor-based Cloud Intrusion Detection System (HCIDS)**	**Chapter 6: Streaming Intrusion Monitoring and Classification System (SCIMCS)**	**Chapter 7: System for Detecting Malicious Insider Data Theft**
Identify Problem and Motivate	A gap exists between protecting the cloud users from outsider attacks using perimeter security approaches and attacks from mischievous users who have penetrated the perimeter security controls.	A need to improve the monitoring of security system alerts and more importantly knowing which events to act upon immediately exists within cloud environments.	New security controls are needed to detect and prevent insider data theft. This is especially true in multi-tenant cloud environments.
Define Objectives of a Solution	The creation of a hypervisor-based intrusion detection system for cloud environments.	The creation of a system and approach to assist administrators with gaining a better understanding of important events and classification of such events.	The creation of a new approach and system for detecting insider data theft.
Design & Develop	The system is designed and developed using libvirt, Python, and IBM Streams.	The system consists of three steps: 1) Summarize and Score, 2) Detect Anomalies, and 3) Classify Attacks and is developed using Python and IBM Streams.	A system profiling approach is used for detecting abnormal login activity and data transfers from IaaS cloud computing nodes hosting tenant virtual machines using Python and IBM Streams.
Demonstration	We demonstrate and verify the effectiveness of the proposed system in a small cloud environment using the Eucalyptus infrastructure.	We demonstrate and verify the effectiveness of the proposed system in a small cloud environment using the Eucalyptus infrastructure.	We demonstrate and verify the effectiveness of the proposed system in a small cloud environment using the Eucalyptus infrastructure.
Evaluation	Using developed signatures, we are able to detect 100 percent of two types of denial of service attacks within a cloud environment: denial of service attacks against a cloud instance and a denial of service attacks from a cloud instance against another cloud instance.	We observe a total alert reduction of 95.9 percent with a zero miss rate for problematic alarms. In addition, we demonstrate a 100 percent classification rate for trained attacks.	We observe 100 percent detection of abnormal login activity and data copies to outside systems and a zero false positive detection rate when anomalies in active user counts and bytes transmitted is detected along with supporting system state data.
Communication	IEEE International Conference on Computing, Networking and Communication (ICNC), CNC Workshop, Honolulu, Hawaii, USA, Feb 3-6, 2014.	IEEE International Conference on Cloud Computing (CLOUD), San Francisco, USA, June 27 - July 2, 2016.	IEEE Global Communications Conference (GLOBECOM), December 4-8 December, 2016.

In accordance with the Hevner, et al. research framework, each component in our system contributes relevant work to the cloud computing environment along with rigorous design science research to the knowledge base. Table 2 summarizes the contributions of our work according to the Hevner, et al framework.

Table 2. Hevner, et al. Research Framework Mapping

Artifact	Application in the Appropriate Environment	Addition to the Knowledge Base
Hypervisor-based Cloud Intrusion Detection System	Technology: Infrastructure Technology: Applications	Foundations: Methods Foundations: Instantiations
SCIMCS	Technology: Infrastructure Technology: Applications	Foundations: Methods Foundations: Instantiations
Insider Data Theft Detector	Technology: Infrastructure Technology: Applications	Foundations: Methods Foundations: Instantiations

Each artifact of the overall system contributes to IaaS cloud computing infrastructure security through instantiation and demonstration of novel security components that reduce cloud deployment risk by improving security in these environments. Additionally, the research introduces new methods for detecting attacks and better understanding alerts in these environments.

Summary

In chapter four, we introduced each of our research components and discussed how the components fit into an overall system. Then, we elaborated on the methodology used to construct the artifacts and provided a mapping to information systems design science research concepts. Chapters five, six, and seven provide the details for each of the research components.

CHAPTER 5

HYPERVISOR-BASED CLOUD INTRUSION DETECTION SYSTEM

One of the significant challenges in Infrastructure as a Service (IaaS) cloud computing is the lack of ability for cloud users to control security protection in the cloud infrastructure. In a survey of more than 170 businesses, 50 percent of the respondents stated concerns with security issues relating to cloud computing resources (Biggs & Vidalis, 2009). To help address these concerns, controls have been proposed by the Cloud Security Alliance, many of which are process based and are subject to noncompliance and human error. For the controls that are automated and machine based, a gap exists between protecting the cloud users from outsider attacks using perimeter security approaches and attacks from mischievous users who have penetrated the perimeter security controls. These outside attackers become insiders within the cloud environment and can attack other virtual machines within the cloud infrastructure.

One automated security control recommended by the Cloud Security Alliance for cloud computing environments is an intrusion detection system (Cloud Security Alliance, 2011). There are two traditional types of intrusion detection systems: host based and network intrusion detection systems (Dhage & Meshram, 2012). Host based intrusion detection systems are composed of an agent on a host system that examines system calls, logs, file-system modifications, and other host activities to detect intrusions. Network intrusion detection systems monitor network traffic and the content of packets in order to discover malicious traffic.

Both host based and network based intrusion detection systems have advantages and limitations. Network intrusion detection systems attempt to address attacks from outsiders and generally have limited effectiveness against insider attacks (M. B. Salem, Hershkop, & Stolfo, 2008). These and other perimeter security controls, such as firewalls, may be less effective in cloud computing environments because of the shared multi-tenancy nature of cloud computing (Sheridan & Cooper, 2012). It is common for multiple cloud users to reside

virtually partitioned on a single physical machine which opens up the possibility for attacks over virtual or internal networks (Lori, 2010). Host based intrusion detection systems can be effective but typically must be monitored and managed by cloud users. This approach can be difficult for cloud users who have several instances in a cloud environment. Furthermore, host based intrusion detection systems can be disabled by a skilled attacker that has breached the instance.

In this chapter, we propose a new type of intrusion detection system, a Hypervisor-based Cloud Intrusion Detection System (HCIDS), to address some of the challenges with traditional intrusion detection systems in cloud environments. HCIDS examines system metrics for cloud instances directly from the hypervisor to seek out potential misuse patterns. Our contributions in this work include, but are not limited to:

- We propose a hypervisor-based intrusion detection system for cloud environments.
- We demonstrate and verify the effectiveness of the proposed system in a real cloud environment.
- Using developed signatures, we are able to detect 100 percent of two types of denial of service attacks within a cloud environment: denial of service attacks against a cloud instance and a denial of service attacks from a cloud instance against another cloud instance.

The remainder of the chapter is organized as follows. First, related work is discussed. Second, the system design of our hypervisor-based cloud intrusion detection system is presented. Third, the system is demonstrated and results are discussed. Fourth, our work is summarized and we discuss future works.

Related Work

Our work consists of three components: the use of performance signatures for detecting attacks on a system, detecting anomalies in virtualized environments from outside of the virtual machine, and an intrusion detection framework for cloud environments.

The use of performance signatures to detect malicious activity and intrusions is proposed by Avritzer, et. al and Oppenheimer and Martonosi (Avritzer, Tanikella, James, Cole, & Weyuker, 2010; Oppenheimer & Martonosi, 1997). Oppenheimer and Martonosi present a model for using performance signature data to detect system security violations.

More recently, Avritzer, Cole, and Weyuker demonstrate an approach for detecting attacks on software systems using system performance signatures. In their work, they model the performance characteristics of a system with a reasonable background load to simulate system usage and examine CPU, memory, I/O and network usage metrics to detect buffer overflow, stack overflow, SQL injection, denial of service (DoS), and man-in-the-middle attacks. The results from their work show promise for using performance signatures to profile attacks.

An approach for detecting attacks in a virtualized environment outside of the virtual machine instance is to use virtual machine monitor introspection (Christodorescu, Sailer, Schales, Sgandurra, & Zamboni, 2009). Garkinkel and Rosenblum present an architecture and prototype using virtual machine introspection to detect attacks in virtual machine instances (Garfinkel & Rosenblum, 2003). In their work, they demonstrate how introspection of the virtual machine can detect rootkits and backdoors, Trojan horses, packet sniffers, and worms by inferring software state based on a priori knowledge of its structure.

Cloud computing intrusion detection is an active research area. To address performance issues with intrusion detection in a cloud computing environment, Dhage, et al propose a distributed intrusion detection system which averts heavy loads on a central intrusion detection server (Dhage & Meshram, 2012). Their work places multiple mini intrusion detection instances throughout the cloud environment which communicate with a controller. The controller stores pertinent data in cloud logs and uses it for intrusion detection analysis. To enhance the effectiveness and efficiency of network intrusion detection systems, Lin, et al. present a technique for using hypervisors in the cloud to inventory operating systems and services on nodes (Lin, 2009). Larger security solutions are also suggested such as the Security Audit as a Service architecture for cloud computing environments posited by Doelitzscher, et. al (Doelitzscher, Reich, & Sulistio, 2010). Their six-layer security model utilizes modules, including a crypto module, a customer public key infrastructure, an SLA monitoring system, a policy module, a logging module, and an intrusion detection system.

Design and Development

Hypervisors have access to performance data for the virtual machines that they host. This data provides insight into the activities occurring within a virtual machine without having direct knowledge of the actual operating system, applications or private data residing

within the virtual machine. In our proposed Hypervisor-based Cloud Intrusion Detection System (HCIDS), we utilize performance metrics from hypervisors within a cloud environment to detect attack patterns. This approach differs from other performance based intrusion detection systems in that it removes the requirement of having software installed on the host computer, or virtual machine in a virtualized cloud environment. Forcing cloud computing users to install additional software in their instances can be problematic and may be resisted by cloud users. Furthermore, gathering performance metrics directly from the hypervisor and not from the operating system makes our solution operating system agnostic. Using patterns in streaming performance metrics from the hypervisor, we are able to detect and classify abnormal usage.

Performance Metrics

The performance metrics used in our work are retrieved from the hypervisors hosting virtual machines within the cloud computing environment. Performance metric data for network data transmitted, network data received, block device read data, block device write data, and CPU utilization is analyzed and is commonly available from all the major virtualization platforms using application programming interfaces (APIs). Our approach retrieves each of these metrics every second. We do not analyze memory utilization because memory allocation is performed once at startup and does not vary with usage making it irrelevant for detecting attacks.

Framework

Our proposed framework consists of three high level components: a controller node, end point nodes, and a notification service. First, the controller node is responsible for near real time analysis of performance data for every virtual machine in the cloud computing environment. Second, the end point nodes gather data on every virtual machine running in the cloud environment from the virtual machine hypervisor and present the data to the controller node. Third, the notification service provides notification when an attack signature has been identified. Figure 6 illustrates our proposed design architecture.

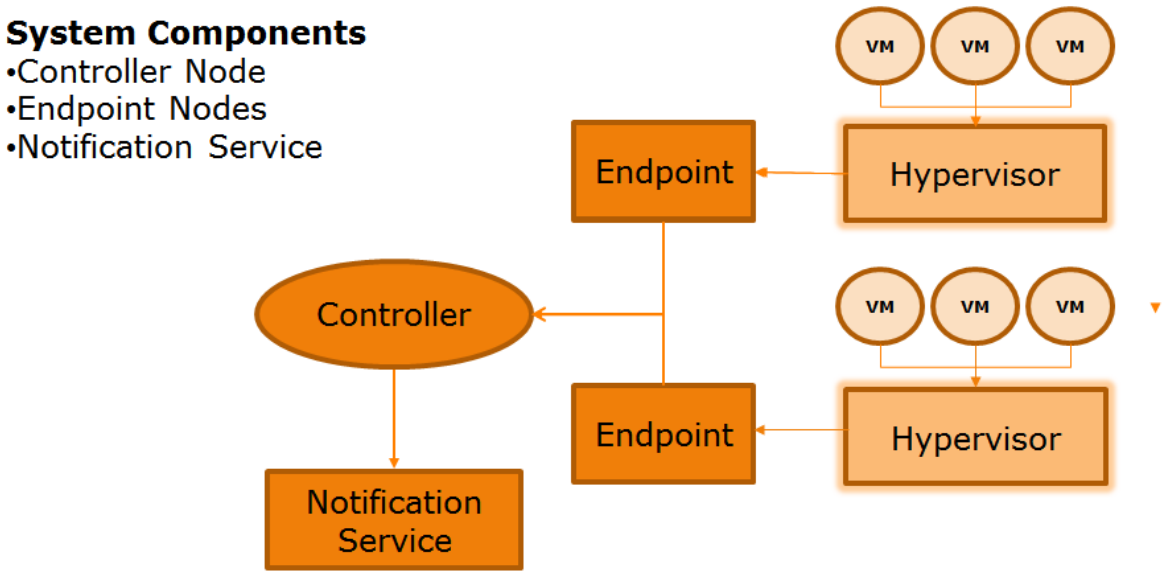

Figure 6. Conceptual Diagram of Proposed System

Controller Node

The controller node is a service that resides within the cloud environment. Its purpose is to collect and analyze data in near real time from the end point nodes. As data arrives, it is analyzed using a sliding window approach. Windows of performance metrics are analyzed for signatures that suggest suspicious activity.

Endpoint Nodes

Endpoint nodes are a conceptual component. They may be agents on each physical system that contains a hypervisor, built within a hypervisor, or API calls to the hypervisor. The purpose of these nodes is to gather and format data from hypervisors and send it to the controller node for analysis. These nodes reside outside of virtual machines and cannot be controlled or manipulated by cloud computing users.

Notification Service

The notification service is used to provide alerts that the system has detected a signature of a potential attack. The notification can be a message in a log file, an email, or input into another intrusion detection system.

Demonstration and Evaluation

We demonstrate the feasibility of using hypervisor performance metrics to detect attacks on virtual machines in a cloud computing environment using the Eucalyptus infrastructure. Eucalyptus is a private and hybrid cloud solution that is in use by a number of large organizations.

Eucalyptus Test Environment

The Eucalyptus cloud computing system is composed of five main components: cloud controller, Walrus, cluster controller, storage controller, and one or more node controllers ("Eucalyptus Components," 2013). In our test cloud environment, the cloud controller, Walrus server, cluster controller and storage controller all reside on a single physical server. Furthermore, our environment consists of two node controllers which reside on independent physical machines. Table 3 summarizes the hardware configuration of our test environment.

Table 3. Simulated Cloud Environment Specification

Component	Configuration
Controller	AMD Athon™ 64x2 Dual Core Processor 4 GB RAM Two gigabit network interface cards (NICs)
Endpoint 1	AMD Phenom™ II X4 965 Quad Core Processor 8 GB RAM Two gigabit network interface cards (NICs)
Endpoint 2	AMD Phenom™ 9150e Quad Core Processor 6 GB RAM Two gigabit network interface cards (NICs)

Simulation Implementation Detail

This section describes the components of our system. First, the controller node is explained. Second, the endpoint nodes are examined. And, third, the notification service is discussed. Figure 7 illustrates the implementation detail and flow between components.

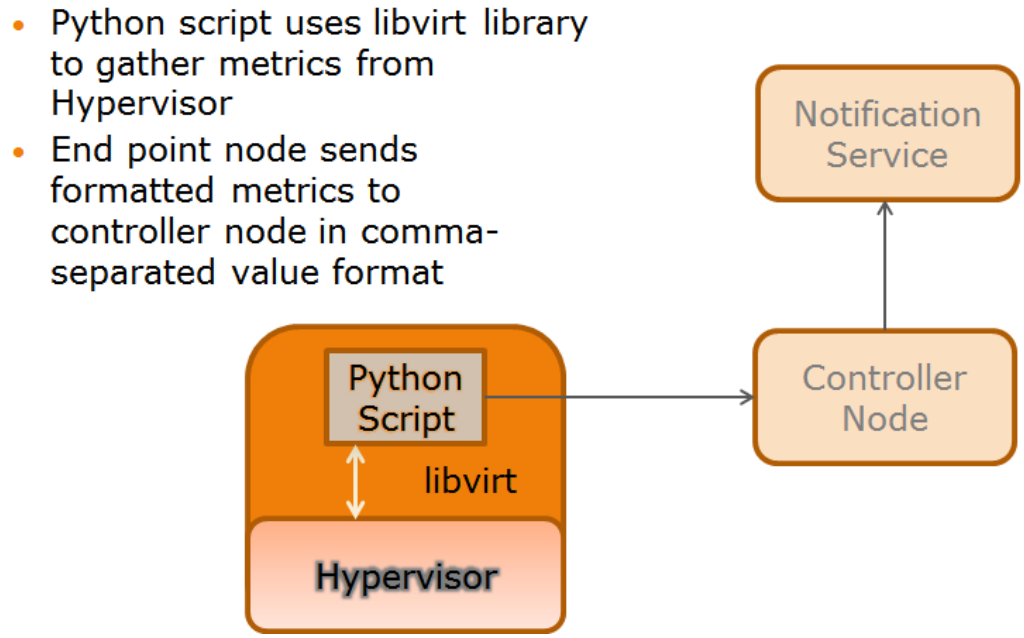

Figure 7. Implementation Detail

Controller Node

The controller node resides on a machine outside of the Eucalyptus infrastructure and is prototyped using the IBM Streams product. The controller node logic is implemented using IBM Streams Processing Language (SPL) (Zikopoulos & Eaton, 2011) and listens on a UDP socket. Hypervisor performance data is rapidly ingested and analyzed using two sets of sliding windows. First, ten second sliding windows aggregate data on CPU percent utilization, block device reads, block device writes, network packets received, and network packets transmitted. As metric values enter sliding windows, the mean and maximum values are calculated. Anomalies are defined as values that exceed two times the mean. Second, a three second sliding window is used to detect attacks. This window populates with anomalies detected from the first sliding window. If an anomaly occurs three times, consecutively, it is

labeled as a potential attack pattern and compared to a set of known attack patterns. Unrecognized patterns are ignored.

Endpoint Nodes

Two Eucalyptus nodes are used which contain multiple virtual machines. A Python script gathers CPU, block device, and network device metrics using the libvirt API and is deployed on each node. This script samples performance metrics every second and sends the data in comma separated value (csv) format to the Controller Node using the UDP protocol for each virtual machine running on the node.

Notification Service

The IBM Streams product performs the role of the notification service. The attacks are visualized in a table and written to a file. The file can be monitored using any common file monitoring tool.

Simulated Activities

The effectiveness of the system is demonstrated by running denial of service attacks from and against a virtual machine in the cloud environment with and without a simulated valid user workload.

Simulated Workload

An Apache web server resides on the cloud instance which serves up a web page that randomly performs different sized reads and writes at intervals of two and five seconds. For each simulated activity, three runs are conducted three times. The first run is performed without a user workload. The second run is conducted with five concurrent users accessing the cloud instance's HTTP server webpage. And, the third run is performed with 10 users concurrently accessing the cloud instance's HTTP server webpage.

Simulated Attacks

Two types of denial of service (DoS) attacks are performed to examine the effectiveness of our approach: a HTTP flood attack against a cloud instance and a syn flood

attack from the cloud instance against another virtual machine in the cloud environment. First, a denial of service attack against the cloud instance is performed using the tool DoSHTTP ("Socketsoft.net," 2013) from outside the cloud environment using a Windows 7 machine. This attack uses 500 sockets to issue 10,000 requests. Second, hping3 (Sanfilippo, 2013) is used to create a syn flood attack from within the virtual machine to attack another virtual machine in the cloud.

Attack analysis approach

The primary purpose of our analysis is to determine whether hypervisor performance metrics can be used to detect and classify attacks while minimally flagging normal usage as attacks. We do this by manually observing patterns in performance data when DoS attacks are occurring and creating signatures from these patterns. There are three goals for the attack signatures. The first goal is to reduce or eliminate false positives. A false positive occurs when normal usage is labeled as an attack. An intrusion detection system with a high false positive rate will be ignored or disabled. The second goal is to detect all valid attacks. The more attacks not detected, the less effective the system becomes. And, the third goal is to properly classify the type of an attack. Proper classification is useful for responding to attacks.

With these goals in mind, we perform both DoS attacks three times for approximately 15 minutes under three stress conditions: no users, five concurrent users, and a load of 10 concurrent users. The variability in workload improves the quality of the experimentation by better reflecting a real world cloud application. Furthermore, each run is performed three times to examine the repeatability of results.

We manually observe repeatable anomaly patterns in the hypervisor performance metrics while the attacks occur and create signatures for the attacks. Each signature is composed of five commonly available hypervisor metric variables: Packets Transmitted (Packets TX), Packets Received (Packets RX), Block Device Read Requests (Block Device Read Req), Block Device Write Request (Block Device Write Req), and CPU Utilization (CPU Util.)

A signature is defined by Boolean values for each performance metric. A metric is considered true in a signature if it is repeatedly detected as an anomaly for three consecutive

10 second sliding windows. As previously described, an anomaly is defined as a metric value exceeding twice the mean in a 10 second sliding window. We find that this technique reduces false positives caused by normal system variability. Using this approach, we code patterns for the DoS attacks from and against a cloud instance. The patterns represent the signature for an attack.

The system is applied to normal running conditions without attacks in order to measure false positives. The same three system stress conditions (e.g., no user activity, five concurrent users, and 15 concurrent users) are performed and the results are recorded.

Results and Observations

The coding of performance metrics reveals repeatable signature patterns for the two DoS attacks. Table 4 summarizes the attack signatures derived from manual observations during multiple system runs under the three stress conditions.

Table 4. Signatures

Attack	CPU Utilization	Block Device Read Request	Block Device Write Request	Packets Received	Packets Transmitted
HTTP DoS attack against cloud instance	True	False	False	True	True
Syn Flood attack from cloud instance	True	False	Any	False	True

To measure the accuracy of signatures, three test runs are performed: DoS attack against the instance, DoS attack from the instance, and no attack for a baseline measurement. Each test run is conducted over a 45 minute period consisting of three 15 minute stress conditions: no user activity, five concurrent users, and 10 concurrent users. Attacks are issued three times during each stress condition, or nine times total per test run. Table 5 summarizes the results of our findings.

Table 5. Accuracy

Attack	**False Positives** (number of false positives / number of metric sets analyzed that were not attacks)	**False Negatives** (number of attacks not detected / number of attacks issued)	**Misclassifications** (number of attacks incorrectly classified / number of attacks issued)
HTTP DoS attack against cloud instance	<1% (8/3043)	0% (0/9)	0% (0/9)
Syn Flood attack from cloud instance	1.4% (43/3179)	0% (0/9)	0% (0/9)
No attack	0% (0/3091)	N/A	N/A

A false positive is counted when a set of performance metric data is detected as an attack during normal usage. A false negative is defined as an attack that is not detected. And, misclassifications are attacks that are incorrectly classified as other attacks (e.g., a syn flood attack is classified as a HTTP DoS attack.)

The data from our findings indicate that streaming hypervisor performance metrics can be used to detect denial of service attacks within a cloud environment. Every denial of service attack performed by the instance and against the instance is detected and properly classified. The false positives are mostly detected during the 10 user workload run. We theorize that this workload may emulate a denial of service attack in the environment. Additional investigation is needed to prevent these false positives. Also, it is noteworthy that no false positives are detected when an attack is not applied.

Comparison with other approaches

The work presented in Avritzer et al (2010) and Oppenheimer & Martonosi (1997) uses a host-based intrusion detection approach to run a monitoring agent on a computer to retrieve performance metrics from the operating system or applications. Our proposed approach does not require any additional software installed in virtual machines. In Christodorescu et al. (2009) and Garfinkel & Rosenblum (2003), virtual machine introspection is used. Virtual machine introspection is effective to detect malicious behavior in virtual machines. However, it examines the detailed state of the virtual machine such as memory and register contents and I/O device flags. Cloud users storing confidential data on a

cloud instance may have concerns with the snooping of memory on their virtual machines. Further, it also requires knowledge and modification of the underlying operating system. Our approach does not require direct knowledge of the operating system running in virtual machines. We examine the performance metric usage patterns over time. The work in Dhage & Meshram (2012), Doelitzscher et al. (2010), and Lin (2009) uses distributed intrusion detection system which averts heavy loads on a central intrusion detection server. In our work, an agent is installed on each hypervisor node which communicates with a central decision node.

HCIDS offers at least two advantages over existing intrusion detection techniques. First, monitoring is done outside of the virtual machine and is independent of the operating system or applications running within the virtual machines. This removes the burden of users having to install additional software in their images and cannot be compromised from within the virtual machine instance. Second, insider attacks that potentially would not be detected using perimeter security controls can be detected. If an attacker takes over an instance and then uses that instance to attack other instances in the cloud computing environment, perimeter firewalls and intrusion detection systems generally would not detect this malicious activity.

Conclusion and Future Work

The initial findings from this work indicate that hypervisor performance signatures can indeed be successfully used to detect attacks in a cloud computing environment. This approach offers advantages over other intrusion detection systems alone. First, our framework does not require knowledge of the underlying operating system or applications run on cloud instances. We examine patterns of performance metrics from outside of the instance directly from the hypervisor without placing a burden on the cloud user. Second, the proposed hypervisor-based cloud intrusion detection system can be integrated with and complement existing intrusion detection systems and perimeter defenses to improve the security within cloud environments. When using a defense in depth security strategy, multiple security systems should be considered to detect and thwart attackers.

Our experiments and testing demonstrate the feasibility of using hypervisor metrics for detecting denial of service attacks both against and from a cloud instance. Additional

statistical approaches to extract attack patterns and system tuning will be explored next. Further attacks, including enumeration, insider password cracking, and network sniffing will be profiled to test the accuracy of detection and classification systems. Approaches to reduce false positives will be examined. And, a comparison analysis of traditional approaches and our system will be conducted.

CHAPTER 6

A STREAMING INTRUSION MONITORING AND CLASSIFICATION SYSTEM

A data breach reported on December 18, 2013 occurred against the Target Corporation. It is believed that the bad actors captured 40 million payment card reports and 70 million customer records. The cause of this significant loss is believed to be the result of inadequate monitoring and alert response (Cobb, 2014). Unfortunately, security attacks and breaches occur against organizations of all types and sizes. Security professionals and researchers accept the premise that no system has perfect security (Mandiant, 2014). Furthermore, security challenges have been noted as primary reasons for avoiding adoption of cloud computing by organizations (Hay, Nance, Bishop, Brian, & Hay, 2011; Ren et al., 2012).

Cloud computing offers unique challenges over single tenant data centers. More specifically, cloud computing infrastructure as a service (IaaS) environments generally consist of multiple tenants running a variety of applications with differing privacy and confidentiality requirements. Although IaaS cloud environments introduce challenges above and beyond private data centers, the techniques for securing both environments are similar. The Cloud Security Alliance suggests an approach that includes a defense in depth (SANS, 2001) strategy where people, process, and layers of technology all play a role in securing the cloud environment (Cloud Security Alliance, 2011).

As observed from incidents such as the Target Corporation breach, monitoring intrusion detection system alerts and more importantly knowing which events to act upon immediately may be the difference between a minor breach and significant damage to an organization. This challenge is amplified in a multi-tenant IaaS environment where multiple tenants may host sensitive data and run services that perform diverse computing activities.

Our work supports a defense in depth approach by leveraging multiple distributed intrusion detection and security system sensors in an IaaS cloud computing environment. We propose a streaming cloud intrusion monitoring and classification system (SCIMCS) to assist

cloud providers with multiple security systems by filtering noisy alert messages and classifying previously observed attacks.

Our approach consists of three steps: 1) Summarize and Score, 2) Detect Anomalies, and 3) Classify Attacks. In this chapter, we detail our approach and demonstrate its effectiveness through implementation and experimentation in an IaaS cloud environment using the Eucalyptus cloud framework. We observe a total alert reduction of 95.9 percent with a zero miss rate for problematic alarms. In addition, we demonstrate a 100 percent classification rate for trained attacks. Our contributions from this work include, but are not limited to:

- A weighted noisiness approach for alert prioritization and classification.
- A framework for IaaS cloud environments using the proposed approach consisting of five components: Sensor Agents, Ingestor, Analyzer, Detector, and Classifier.
- We demonstrate the effectiveness of our framework in an IaaS cloud environment using the Eucalyptus cloud infrastructure by executing and classifying real attacks.

The remainder of the chapter is organized as follows. First, related work is discussed. Second, the system design of our streaming cloud intrusion monitoring and classification system is presented. Third, the system is demonstrated and results are discussed. Fourth, our work is summarized and we discuss future works.

Related Work

In this section, first, we summarize existent related work in four subcategories: alert aggregation, alert correlation, alert ranking and classification, as well as cloud based approaches. Second, we summarize how our approach is novel.

Alert Aggregation

Debar and Wespi introduce a system using Tivoli Enterprise Console that aggregates and correlates data from probes. Correlation relationships are created using explicit rules and derived rules from configuration information. Aggregation relationships are created using an algorithm that groups events together using these rules. They demonstrate the effectiveness of their solution and display the results using alert and alarm views (Debar & Wespi, 2001). Fan, Ye, and Deng demonstrate a distributed alert aggregation system model that uses a

transform agent to convert intrusion detection system (IDS) messages. Original alerts are categorized into classes and actions which can be issued. Their work is tested using network data (Fan, Jihua, & Mingxing, 2009). Hofmann and Sick propose algorithms for alert aggregation based on probabilistic models of a current situation using alert attributes. They provide both an offline and data stream alert aggregation approach (Hofmann & Sick, 2011). Saad and Traore present semantic analysis and ontology engineering techniques to aggregate and fuse intrusion detection system alerts. Their work demonstrates an approach for lossless alert aggregation that does not require perfect matches of alert attributes (Saad & Traore, 2011).

Alert Correlation

Valeur, et al. present a correlation process and framework for addressing large volumes of messages output from intrusion detection systems. They use alerts from multiple systems and create intrusion reports or tag them as non-relevant. Their approach uses alert ids and alert names to perform mapping and normalization (Valeur, Vigna, Kruegel, & Kemmerer, 2004). Qin and Lee put forth an approach to analyze INFOSEC alerts and detect attack strategies. In their work, Bayesian inference and the Granger Causality test is used to correlate alerts (Singhal, Qin, & Lee, 2007). Ma, Li, and Zhang present an approach to fuse alerts based on timestamp and remove duplicates to reduce alerts. Evidence threat probability is calculated using Dempster-Shafer theory. Furthermore, they combine network and intrusion detection data and use Hidden Markov Models (HMM) to calculate a network risk distribution (Ma, Li, & Zhang, 2009). Wen, Xian and Zhou introduce a system for alert fusion and correlation. In their work, they use a target-oriented policy where alerts are clustered when requirements of duplication categorization and co-operating are met (Wen, Xiang, & Zhou, 2010).

Alert Ranking and Classification

Jiang, et al. present work on importance ranking of alerts using invariant relationships to map metric thresholds to other metric thresholds. They use Autoregressive models to learn linear relationships between metrics. And, they use alert peer review to assist with trustworthiness of alerts (Jiang, Chen, Yoshihira, & Saxena, 2011). Gupta, et al. present the

PIKE architecture which uses a binary classifier to classify an alert as relevant or irrelevant based on knowledge-based evaluation. They propose a classifier that uses a calculated score as the characteristic for classification. The effectiveness is demonstrated using contextual information as a basis of IDS alert classification (Gupta, Joshi, Bhattacharjee, & Mundada, 2012).

Cloud Based Approaches

The Cloud Security Alliance suggests an approach that includes a defense in depth strategy where people, process, and layers of technology all play a role in securing a cloud environment (Cloud Security Alliance, 2011). Furthermore, NIST provides Guidelines on Security and Privacy in Public Cloud Computing in SP800-144 (Jansen & Grance, 2011). Both of these pragmatic works provide recommendations for securing a cloud environment but leave the specific details to cloud providers.

Technical cloud solutions exist in literature. Gul and Hussain present a cloud intrusion detection model. They put forth a multi-threaded cloud intrusion detection system that monitors the network along with a third party monitoring system (Gul & Hussain, 2011). Log monitoring and management systems in cloud computing environments based on usage are also known (Lee, Park, Eom, & Chung, 2011). And, collaborative intrusion detection systems in cloud environments have been posited (Mohamed, Adil, Saida, & Hicham, 2013).

Our Approach

Information overload theory has been applied to the field of advertising (Anderson & de Palma, 2005). In marketing, the outcome of too much information is the reduced effectiveness of advertising investment. Information overload theory can be applied to security control effectiveness. Accuracy aside, the effectiveness of a security system dramatically decreases when an administrator is overwhelmed with too much information.

We set out to reduce the information overload problem in IaaS cloud by accomplishing three objectives: 1) to present cloud providers with alerts of importance from multiple systems while filtering lesser important alarms without sensor reconfiguration, 2) to properly classify attacks based on filtered alerts, and 3) to present a scalable and dynamic approach. We accomplish our objectives using a novel three step approach: 1) Summarize

and Score, 2) Detect Anomalies, and 3) Classify Attacks. To the best of our knowledge, we are the first to examine alert prioritization and classification in IaaS Cloud Environments using anomalies in calculated urgency scores. Using these reduced alerts, attacks are classified. We demonstrate the effectiveness of our framework by executing, detecting, and classifying real attacks. Unlike previous works, we do not use simulation on test data sets, but actual system state and attacks.

Design and Development

An IaaS cloud environment typically consists of virtual machines hosted on physical nodes. A single node may host virtual machine instances from multiple tenants. To secure this shared dynamic environment, the Cloud Security Alliance recommends a defense in depth approach (Cloud Security Alliance, 2011). Similarly, NIST provides security guidance for cloud providers (Jansen & Grance, 2011). However, managing a defense in depth approach is non-trivial. Several layers of security mechanisms must be deployed and monitored. When a threat is detected, it must be appropriately acted upon. A common problem with monitoring such approaches is the sheer volume of alarms generated, some of which are false positives, and others are informational. It is challenging for cloud providers to quickly interpret which events to act upon and the priority of such events.

Another challenge is the dynamic nature of cloud environments. Tenant instances may come and go. As the needs of providers grow, nodes are added dynamically to the environment. In other cases, nodes are removed due to system faults or maintenance. Security sensors may come and go in these environments. Therefore, our system must tolerate the inherent changing nature of these environments.

Approach

To address the problem of information overload from security sensors in dynamic IaaS cloud environments, we propose our distributed cloud intrusion monitoring and classification system. Our system consists of a three step approach: 1) Summarize and Score, 2) Detect Anomalies, and 3) Classify Attacks shown in Figure 8.

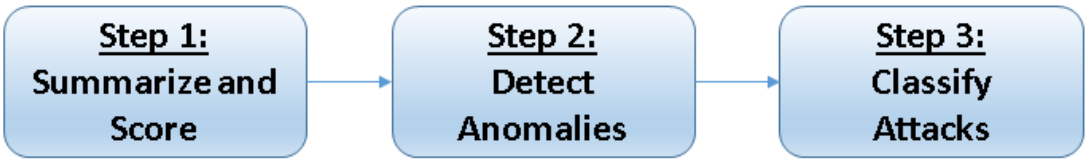

Figure 8. Three Step Approach

Summarize and Score

Summarize and score ingests alerts from security sensors in a cloud environment and calculates an urgency score for the alert message. This approach is based on the concept of black swan events, or rare events that have a significant future impact (Damiani, 2009; Taleb, 2010; Taylor & Williams, 2008). In an IaaS cloud environment with several sensors reporting data, we derive a formula for detecting these rare events. The urgency score is calculated using the number of times a message is reported historically in the entire cloud environment, by the individual node, and the count of alerts generated by the sensor.

First, we calculate the message noise in the overall cloud environment:

$$f(m,\ c) = (\frac{m}{c})$$

where m is the number of times the alert message occurred and c is the total count of all historical alert messages. Next, we calculate the message noise for the reporting node (e.g., we determine how many times this alert has occurred at the node level):

$$f(m',\ c') = (\frac{m'}{c'})$$

where m' is the number of times the alert message occurred on the node and c' is the total number of messages reported by the node. Third, we calculate the message urgency score:

$$U_M = 1 - \left(f(m', c') \times \left(1 - \left(1 - \frac{1}{n}\right)\right) + f(m, c) \times \left(1 - \frac{1}{n}\right)\right)$$

where n is the number of nodes in the cloud environment. In addition to message urgency, we calculate sensor urgency:

$$f(m_s, s) = (\frac{m_s}{s})$$

where m_s is the total number of messages reported by all sensors and s is the sensor count in the environment. Then, we calculate the urgency by the specific sensor reporting the message:

$$f(m_s', s') = (\frac{m_s'}{s'})$$

where $m_s{}'$ is the number of times the alert occurred from the sensor and s' is the total number of alerts reported from the sensor. Finally, we calculate the sensor urgency score U_S:

$$U_S = 1 - \left(f(m_s{}', s') \times \left(1 - \left(1 - \frac{1}{t}\right)\right) + f(m_s, s) \times \left(1 - \frac{1}{t}\right)\right)$$

where t is the total number of sensors in the cloud environment. Using these two scores, we derive an overall urgency value, U, for the alert:

$$U = U_M \times U_S$$

This value is used as the urgency score for the alert message in the system. A value of 1.0 is most urgent, or a possible black swan event, while a value of 0 is least urgent. In addition, sensors that infrequently report are given higher weight than those that commonly report. The combination of weighted event messages and sensor output increases the priority of rare anomalous events and ranks alerts properly amongst peers.

Detect Anomalies

Detect anomalies utilizes alert messages and the urgency score from Summarize and Score to detect abnormal patterns in scores using a time series k nearest neighbor approach (Sutton, 2012). We use window sizes of 15 and 30 tuples based on empirical experimentation. The choice to use this approach is based on simplicity, accuracy, and performance.

In addition, Detect Anomalies uses the mean and standard deviation of previously observed urgency scores to rank anomalous messages as important. Values that fall outside of one standard deviation and within two standard deviations from the mean are labeled with medium importance. Values that fall outside of two standard deviations from the mean are labeled as high importance. Both medium and high alerts are considered potential threats.

Classify Attacks

Classify attacks ingests anomalous alert messages from Detect Anomalies and uses 10 second tumbling windows to create signatures of alert messages and sensor ids. Bayesian classification (Mihaescu, n.d.) is applied to signatures for classification of the potential attack.

Naive Bayesian classification is chosen because the classifier model is easily implemented, performs quickly, and has a history of success for classification (Rish, 2001).

In addition to attack pattern classification, prioritized alert messages are displayed based on the standard deviation of the urgency score compared to previously observed scores. If an attack pattern is unknown, the window of alert messages may be used to train the classifier by applying the alerts to a classification rule. If an unclassified attack occurs, the classifier can be placed into train mode and the attack is replayed and labeled. Future similar patterns are recognized and classified as an attack defined by the specified label.

Framework

Our proposed framework consists of five components: Sensor Agents, Ingestor, Analyzer, Detector, and Classifier. Figure 9 provides a visual representation of the components.

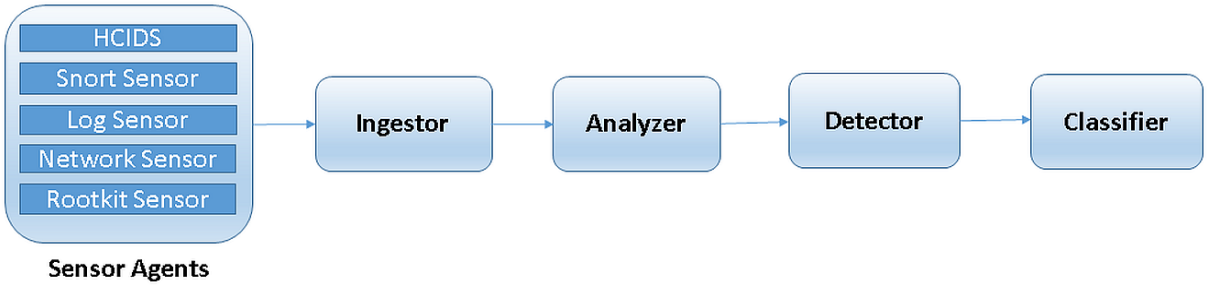

Figure 9. SCIMCS Framework Components

Sensor Agents

Sensor agents run on nodes hosting virtual machines. The sensor agents receive the input message from a security module, format the alert message, and pass it to the Ingestor. For example, the sensor agent for the common intrusion detection system, Snort, receives the output from the Snort system, formats the message and appends metadata consisting of time stamp, date stamp, node, and sensor id. The formatted message is then sent on to the Ingestor.

Ingestor

The Ingestor receives messages from sensor agents using the UDP protocol and has the capability to buffer events. A throttling control is implemented to avoid back pressure

from the Analyzer. The primary purpose of this component is to rapidly process event messages and to perform flow control to prevent message loss.

Analyzer

The Analyzer performs the Summarize and Score step of our system. As sensor events arrive from the Ingestor, the Analyzer inserts each message along with a count into a persistent in memory data store. Messages and counts are summarized at the cloud and node level. In addition, sensor ids for types and specific sensors are persisted with counts of messages reported by the sensor.

The Analyzer uses message counts to calculate an urgency value from the number of times the message has occurred historically in the cloud environment along with the number of messages generated by the sensor as described earlier in this chapter. Furthermore, a decay time is introduced to reduce the count of messages every n seconds. The reduction of message counts ensures that future critical messages are not overlooked.

Detector

The Detector performs two roles. First, it detects anomalies in urgency scores using k-nearest neighbors over a sliding window. The first time a high score arrives, it is flagged as an anomaly. As the same score continues to arrive, the anomaly score decreases, resulting in in a reduction of messages passed to the classifier. Second, the Detector labels alerts as high, medium, or low based upon the output from the Analyzer and standard deviations from the mean of alerts.

Classifier

The Classifier provides step three of our approach, Classify Attacks. This component uses Naive Bayes Classification to label the attack using concatenated sensor and alerts in 10 second data widows. As previously mentioned, The Classifier also provides a training mode for adding new attack patterns.

System Instantiation

We demonstrate our system in an IaaS cloud environment running Eucalyptus shown in Figure 10. The environment consists of three nodes containing up to four virtual machines each, a cloud controller machine, and two machines dedicated to the SCIMCS. This configuration is similar to a small business private IaaS cloud environment.

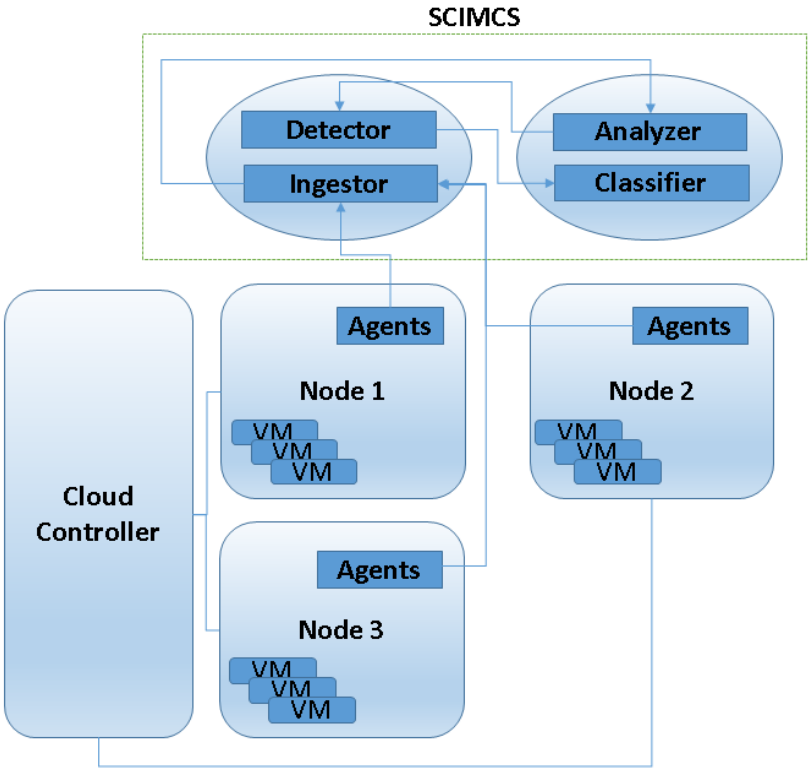

Figure 10. Cloud Environment

Cloud Environment

Figure 10 provides a conceptual view of the experimentation environment. This environment consists of five physical multicore systems connected over Gigabit Ethernet on a private network. The cloud computing environment is made up of the cloud controller and three node machines. The Streaming Cloud Intrusion Monitoring and Classification system is distributed across two physical machines. The Cloud Controller contains the management components for the Eucalyptus cloud infrastructure (Nurmi et al., 2009). The nodes contain security agents as well as the Eucalyptus node controller component for cloud management. The SCIMCS consists of two physical systems which communicate with Agents on the nodes.

SCIMCS Implementation

The SCIMCS implementation consists of the five components: Sensors, Ingestor, Analyzer, Detector, and Classifier.

Sensors

Five sensors are deployed on each of the nodes in the IaaS environment: HCIDS, Snort Sensor, Log Sensor, Network Sensor, and Rootkit Sensor. HCIDS is a hypervisor based cloud intrusion detection system proposed in (Nikolai & Wang, 2014). It monitors virtual machine performance metrics such as packets transmitted/received, block device read/write requests, and CPU utilization then seeks out anomalies. A signature based categorization approach is used to emit alerts, as described in chapter five.

The Snort sensor retrieves messages emitted from the Snort intrusion detection system (Kumar, 2012). The Snort deployment is configured with all rules turned on. This configuration ensures that valid attacks are not missed. It also demonstrates the dilemma security administrators face with rule based systems. If too few rules are active, attacks are missed. If too many rules are active, large volumes of messages are generated.

The log sensor retrieves node log messages from operating system logs. Log messages can reveal abnormal system state and attacks. We monitor two logs: /var/log/secure and /var/log/audit/audit.log. These logs provide operating system audit and security alerts.

The network sensor monitors network flow using a Python script to sniff traffic and IBM Streams Anomaly Detection operator (Cancilla, 2015) to find anomalous patterns. Two flow patterns are analyzed: the number of packets and packet rate. A 60 second training window trains the anomaly detector and 30 second windows are used to detect anomalies. When an anomaly is detected, an alert is generated.

Finally, the rootkit sensor runs a rootkit scan of the operating system which checks for signatures of abnormal behavior on a node. If checks fail, an alert is generated with text from the failed check. A modified version of the chkrootkit (Murilo & Steding-Jessen, n.d.) utility is used.

Ingestor

The Ingestor is written in the IBM programming language SPL and runs within the IBM Streams (Ballard et al., 2010) distributed architecture. This architecture provides the underlying high performance stream computing model and operators to throttle and buffer alert event messages. In our implementation, we buffer up to 1,000 tuples and throttle the flow rate to 20 tuples per second.

Analyzer

The Analyzer is coded in Python. Four dictionaries, which persist to disk upon program shutdown, are used to store summary data about alerts. More specifically, messages are stored along with counts by cloud, node, sensor, and sensor type. As described earlier in this chapter, the summary data is used to dynamically calculate the weighted urgency score based on message frequency. Labels of high, medium and low are given to alert messages based on the number of standard deviations from the mean of urgency scores in the environment.

Detector

The Detector is implemented using IBM InfoSphere Streams. The Streams Anomaly Detector operator (Cancilla, 2015) is used to perform anomaly detection using 15 values with a training data set of 30 tuples on the urgency scores generated by the Analyzer. When an anomaly is detected, all of the messages that occurred during the anomalous time window are passed to the Classifier.

Classifier

The Classifier is written in Python using the Reverend Python module for Bayesian classification (Bakhtiar, 2009) and has two modes of operation: train and monitor. In train mode, the administrator reproduces an attack. The anomalous event sensor ids and messages are concatenated together for each alert message event during an attack in 10 second windows. After attacks have been issued, the administrator applies a label defining the pattern (e.g., [nmap], [dos], etc).

In monitor mode, the Classifier applies Bayesian classification to message patterns using the classification data from train mode. When anomalous messages are detected by the

Detector, they are passed to the Classifier where concatenation with sensor ids is performed for each message in the 10 second window. Then, Bayesian classification is used to find the closest match and display the type of attack. A threshold of .50 is applied for displaying attack types based on empirical observation. Pattern matches with less than a .50 probability match are not displayed. In addition to the labeled attack, the Classifier outputs anomalous message alerts ranked as high, medium, or low based upon the score given by the Analyzer. An example of output from the Classifier is shown in Figure 11.

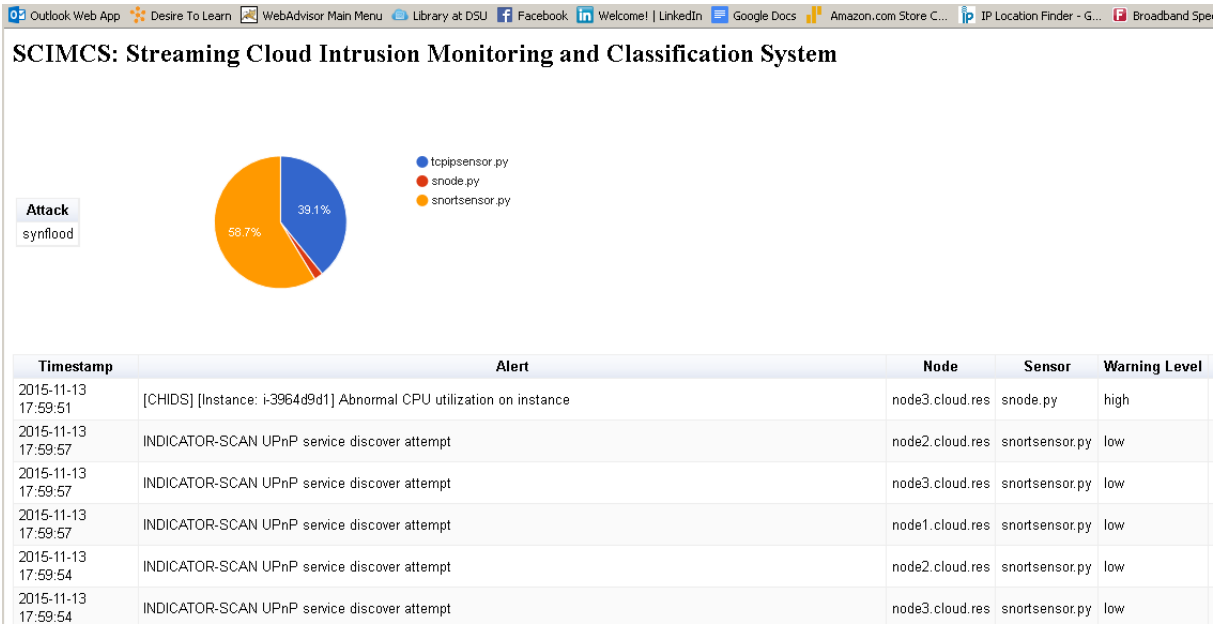

Figure 11. Classifer Output

Demonstration and Evaluation

This section details system analysis, training, experimentation attacks, and results. First, we demonstrate the effectiveness of our urgency score algorithm. Next, we detail our experimentation approach. Finally, we discuss the results.

Urgency Score

The proposed approach utilizes urgency score for alert prioritization. To illustrate the effectiveness of our posited urgency score, we run simulated event messages against the

system. First, 11 distinct messages (i.e., m_1, m_2... m_{11}) are simulated for a total of 2,047 messages. The messages are broadcast in powers of two after the first message is sent. In other words, m_1 is broadcast one time, m_2 is sent two times, m_3 is pushed four times and m_{11} is passed to the analyzer 1,024 times. Each unique message is generated from a different sensor on the same node.

The results are shown in Figure 12. The message urgency scores, U_M, sensor urgency scores, U_S, and overall urgency score, U, are plotted for message m_{11}. Note that curve U_M and curve U_S result in the same value because the message event is broadcast from a single node and sensor.

The first occurrence of message arrival is scored high and falls above three standard deviations (3σ) from the mean of all message occurrences. As previously mentioned, scores occurring beyond 1σ are considered potential threats, or black swan events. As illustrated, the urgency score, U, decreases in importance over time as expected.

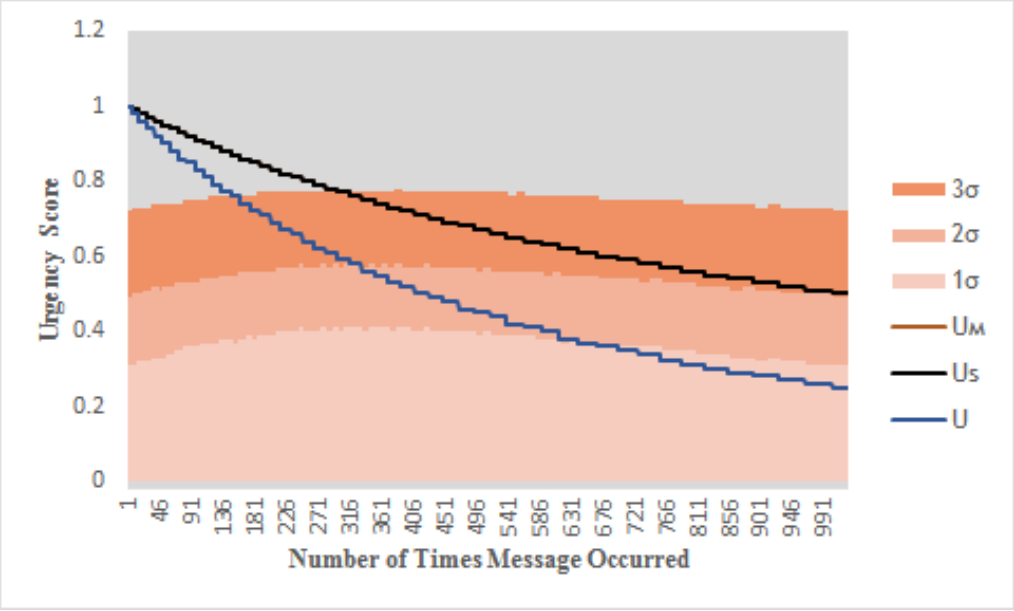

Figure 12. Urgency Score Simulation - One Node

To demonstrate the effectiveness of the overall urgency score using both the message urgency score and sensor urgency score, the simulation continues on two additional nodes. The scores for message m_{11} on node$_3$ are plotted in Figure 13. The overall urgency score is reduced in this instance when both message count and sensor count are taken into account. A

black swan event message from a sensor never previously reporting the message ranks higher than an event reported in the past.

The dynamic component of the scoring system assures rare events are properly scored. In addition, our system prevents rare events from getting stale over time by incorporating a decay factor which decrements each message count every n seconds of system operation. In our experimentation, n is set to 600.

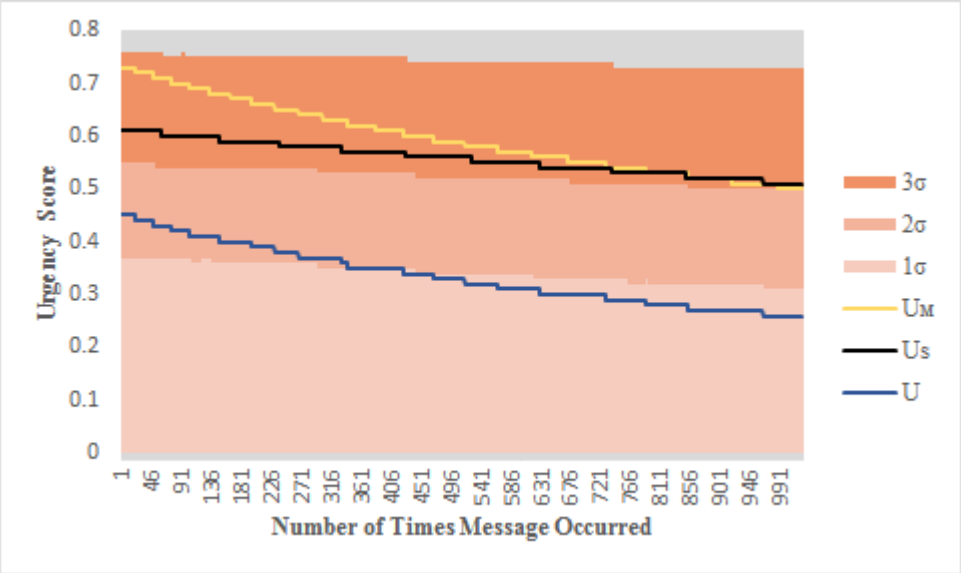

Figure 13. Urgency Score Simulation - Three Nodes

Streaming Intrusion Monitoring

Our system scores and analyzes streaming events from sensors by utilizing IBM Streams which provides a scalable and high performing infrastructure along with analytical support for anomaly detection. To demonstrate the effectiveness of our streaming system, we examine the times from sensor generation to classification. A sample of 500 messages reveals a minimum generation to classification time of less than one second, a maximum time of 20 seconds and a mean time of five seconds during our experimentation.

This sample suggests that an attacker would be detected within 20 seconds and if the attack has been previously classified, the attack type would also be known. Counter actions could be automatically initiated in a production system.

Training the System

Two components of the system are trained: Analyzer and Classifier. This section describes the training procedure.

Analyzer

The Analyzer self-trains over time as messages are generated by the sensors. High volume alert messages and noisy sensors are given a lower urgency score. We complete this training by running the system for one hour in normal operational state without attack.

Classifier

To train the Classifier, we run the component in training mode. Then, we conduct five attacks: an nmap scan, a syn flood attack, an ssh password crack attack, malware insertion, and we perform forensic counter measures by clearing logs. The detail for each attack is provided in the next section. After an attack is run, an appropriate label is assigned to the attack (e.g., [nmap], [synflood], etc.)

Experimentation Attacks

To demonstrate the effectiveness of our approach, we perform attacks on the test environment using a four-step attack approach (Dell, 2012) to simulate a real attack. The four steps consist of a reconnaissance phase, an intrusion and advanced attack phase, a malware insertion phase, and a cleanup phase.

First, in the reconnaissance phase, an attacker attempts to learn about potential target systems. In our experimentation, we use the nmap tool (Lyon, 2015) to scan systems in our cloud environment. We pass the nmap tool parameters '-T4 -A -v' to conduct an intense scan. The goal is to discover more information about vulnerable systems.

Second, the intrusion and advanced attack phase occurs after attackers find vulnerable systems. For this phase of attack we perform a synflood using the hping tool (Sanfilippo, 2015) with the parameter '–flood' against port 80 and a ssh password attack using the ncrack tool (Hantzis, 2015). The ncrack tool performs a dictionary attack against the root user on multiple nodes.

Third, in the malware insertion phase, malware is inserted into the environment to give the attacker future access to the system. In our experimentation, we use the netcat tool (Anonymous, 2007) to open a bindshell using the '-L' parameter.

And, fourth, in the cleanup phase, forensic evidence is removed from the system. We clear two logs to remove traces that we logged into the system by clearing the files /var/log/wtmp and /var/log/lastlog. These logs contain user login history to the system.

Results

We demonstrate the effectiveness of our approach by examining our initial goals. First, we show how our work reduces the volume of alarms without reducing the effectiveness of security systems. Second, we demonstrate through experimentation the effectiveness of using Bayesian classification for proper attack classification. Table 6 summarizes our findings for message reduction results.

Table 6. Message Reduction Results

Attack	Number of total alerts generated during attack	Number of alerts determined to be important	Number of attack messages not reported	Percentage of alert reduction
Reconnaissance: nmap	4795	163	0	96.6%
Intrusion and advanced attack 1: synflood	4598	561	0	87.8%
Intrusion and advanced attack 2: nCrack	3242	442	0	86.4%
Malware Insertion: bind attack	4112	50	0	98.5%
Clean-up: Remove logs	5050	86	0	98.3%
No attack 1 (20 minutes)	7468	88	n/a	98.8%
No attack 2 (20 minutes)	7441	122	n/a	98.4%

From Table 6, the results of our experimentation are promising. We observe an average 95.9 percent message reduction. To verify that critical alerts were not missed, we manually compare the generated alarm messages with those reported by the Classifier.

Table 7. Attack Classification Results

Attack	Number of attacks	Number of attacks properly classified	Number of attacks improperly classified	Attack classification accuracy
Reconnaissance: nmap	10	10	0	100%
Intrusion and advanced attack 1: synflood	10	10	0	100%
Intrusion and advanced attack 2: nCrack	10	10	0	100%
Malware Insertion: bind attack	10	10	0	100%
Clean-up: Remove logs	10	10	0	100%
No attack 1 (20 minutes)	54*	40*	14*	74.1%
No attack 2 (20 minutes)	58	58	0	100%

Table 7 summarizes our classification findings. When not under attack, sensors occasionally generate low volume messages which are displayed by the Classifier. To address this issue, we trained the Classifier to flag these patterns are none threats. After the attacks are trained in the system, we achieve a 100 percent accuracy in attack classification. However, as previously discussed, when attack patterns have not been trained (labeled with a * in the table), we observe a 25.9 percent misclassification rate in our environment. After training the message patterns, the classification accuracy returns to 100 percent. Table 8 provides a summary of the sensors generating messages by type of attacks.

Table 8. Signatures

Attack	Sensor				
	HCIDS	Snort Sensor	Log Sensor	Network Sensor	Rootkit Sensor
Reconnaissance: nmap	-	X	-	-	-
Intrusion and advanced attack 1: synflood	X	-	-	X	-
Intrusion and advanced attack 2: nCrack	-	-	X	-	-
Malware Insertion: bind attack	-	-	-	-	X
Clean-up: Remove logs	-	-	-	-	X

Other Observations

During our experimentation, a rogue log sensor was introduced into the environment. This noisy sensor emitted several messages per second. The system classified the noise as a potential synflood providing further supporting evidence for our approach.

Conclusion and Future Work

The multi-tenant, diverse nature of an IaaS cloud environment increases security complexity. Lack of control over the data and applications running on tenant instances makes securing these environments challenging for cloud providers. One approach recommended by the Cloud Security Alliance is a defense in depth strategy where multiple layers of protection provide a defense against bad actors. To implement this approach, multiple security technologies are deployed. Monitoring these technologies and knowing which alerts to act upon is non-trivial for cloud providers.

Our work demonstrates that sensors which ingest output from these security technologies can be used to feed a multistep approach that summarizes and scores alerts, detects anomalies, and classifies attack patterns. First, our approach utilizes security technology alert messages and sensor ids along with message and sensor volume data to score the importance of a particular alert message. Next, time series analysis is applied using k nearest neighbor anomaly detection over sliding windows of alert urgency scores. Finally, we classify the anomalous alerts using Bayesian classification. In addition to classification, alerts are output with a priority value of high, medium or low based upon how the alert score deviates from the mean of alert scores for all historical alerts in the IaaS cloud environment.

We demonstrate our approach through implementation and experimentation. During experimentation, we observe a total alert reduction of 95.9 percent with a zero percent miss rate for attack messages. In addition, a 100 percent classification rate is demonstrated for previously trained attacks. We suggest five areas for future research. First, more attack experimentation is needed to determine the effectiveness of Bayesian classification as well as other machine learning techniques. Second, we recommend introducing more sensors in the environment. As the number of sensors increase, the effectiveness of our classification approach should improve. We hypothesize that the use of more sensors which can detect specific attacks will improve our Bayesian classification technique as well as other similar techniques. Third, we encourage scaling the approach to a large IaaS cloud environment. Our test bed is a small cloud environment, similar to a small business. At this time, we can only extrapolate our results to larger cloud environments. Fourth, we suggest a technique of multilayer classification using our technique in this chapter. In other words, we hypothesize using multiple layers of classification could allow for detection of attack phases. Longitudinal analysis of detected attacks could reveal patterns relating to how far the bad actor has progressed in an attack. And, fifth, we encourage research in applying other machine learning, anomaly detection, and classification techniques.

CHAPTER 7

A SYSTEM FOR DETECTING MALICIOUS INSIDER DATA THEFT

On August 25, 2006, Amazon EC2, one of the leading Infrastructure as a Service (IaaS) cloud offerings, went into beta (Barr, 2006). Since then, cloud computing has become big business. The largest technology companies in the world now provide cloud computing offerings and solutions (Google, 2015; IBM, 2015; Microsoft, 2015). However, cloud computing is not without challenges. According to the Cloud Security Alliance (Cloud Security Alliance, 2013), data theft and insider attacks are two of the nine critical threats facing cloud security.

Insider attacks fall into three categories: malicious, accidental, and non-malicious (Willis-Ford, 2015). Malicious insiders conduct activities such as ip theft, information technology sabotage, fraud, and espionage, with intent of doing harm or for personal gain. Accidental insider attacks occur when unintentional misuse of systems is performed by a user without the intent of harm. And, non-malicious insider attacks are intentional attacks where the user attempts to perform self-benefiting activities but without malicious intent.

Technical controls exist for reducing the risk of insider attacks, including intrusion detection systems, security information and event management, data loss prevention, access control systems, and honey-tokens. In addition, non-technical controls are used and consist of psychology prediction models, education and awareness, as well as information security policies (Elmrabit, Yang, & Yang, 2015). While controls exist, not all insider attacks can be detected. Furthermore, several approaches for addressing insider attacks are reactive and not predictive in nature. Techniques for preventing such attacks are needed (Maxim, 2011).

Although no single approach can prevent all insider threats, a multi-faceted proactive technique can be used to reduce the risk of damage caused by inside attackers (Maxim, 2011). Several types of attacks exist, including unauthorized extraction of data, data tampering, asset destruction, illegal downloading, eavesdropping, spoofing, social engineering, resource misuse, and installing of malicious software (M. Ben Salem, Hershkop, & Stolfo, 2008).

Our work puts forth a new security control for detecting one type of insider attack, unauthorized extraction of data, or data theft. The importance of reducing the risk of data theft gained recent international attention when the National Security Agency contractor, Edward Snowden, downloaded and disseminated classified documents about intelligence programs (Elmrabit et al., 2015). A system to detect and possibly thwart such actions has significant potential to contribute to a successful defense in depth (SANS, 2001) security strategy and reduce the damage of data theft from inside attackers.

We posit a system profiling approach for detecting abnormal login activity and data transfers from IaaS cloud computing nodes hosting tenant virtual machines. This approach aims to address the problem of rogue administrators as described in Claycomb and Nicoll (Claycomb & Nicoll, 2012) who attempt to steal data from nodes as discussed in Duncan, Creese and Goldsmith (Duncan, Creese, & Goldsmith, 2012). Our approach uses k-nearest neighbors anomaly detection to detect abnormal variations in bytes sent over the network and number of active users on the cloud node. Furthermore, we examine system state data consisting of open files and network connections to improve detection and provide forensic data for investigation.

Unauthorized extraction attacks are especially important to address in IaaS cloud environments to prevent theft of tenant virtual machine data. Although encryption may reduce risk of insiders having the ability to use stolen data, encrypted virtual machine images and data store files may be copied from nodes and attacked off line.

In our system, agents are installed on cloud nodes hosting virtual machines and data. The system is trained under normal cloud operating conditions. Then, the system monitors for anomalies in transmitted network data and active user logins using k-nearest neighbors anomaly detection. This data is used to detect anomalies that exceed normal operating thresholds established during training.

Our results suggest that using k-nearest neighbors anomaly detection to monitor node network transmissions and number of active users along with system state information can be used to detect 100 percent of abnormal login activity and data copies to outside systems by users. Furthermore, we observe a zero false positive detection rate when anomalies in active user counts and bytes transmitted are detected along with supporting system state data.

The remainder of the chapter is organized as follows. First, related work is discussed. Second, the design of our system for detecting malicious insider data theft presented. Third, the system is demonstrated and results are discussed. Fourth, our work is summarized and we discuss future works.

Related Work

A number of works have been posited to reduce the threat of insider attacks. Stolfo, Salm, and Keromytis posit an approach to mitigate attacks using fog computing where they detect abnormal usage patterns and present potential attackers with misinformation (Stolfo, Salem, & Keromytis, 2012). Claycomb and Nicoll discuss the threat of rogue administrators and suggest process based approaches to deal with the threat. Furthermore, they discuss future research topics which include predictive models (Claycomb & Nicoll, 2012). Colombe and Stephens suggest an approach to visualize intrusion detection system data to detect insider attacks (Colombe & Stephens, 2004). Babu and Bhanu research an approach for using key stroke dynamics to detect insiders (Bondada & Bhanu, 2014). A more related and interesting technique for detecting insider attacks is the use of machine learning and rule based detection posited by Khorshed and Wasimi (Khorshed, Ali, & Wasimi, 2011). They suggest that rule based learning can be used to detect insider attacks in a cloud environment and that continuous cloud monitoring is an important part of cloud security. Sriram, Patel, and Lakshmanan posit a hybrid protocol using selective encryption and data cleaning along with user profiling and decoy technology to address the problem of inside attackers. One aspect of their work related to our approach is the use of a neural network that examines volume of data downloaded, nature of the operations, division of the task, ip address, and log files (Sriram, Patel, Harishma, & Lakshmanan, 2014). In previous work, we have put forth a system for detecting attacks from and against virtual machines in an IaaS cloud environment using anomalies in performance metrics obtained from the hypervisor (Nikolai & Wang, 2014).

To the best of our knowledge, applying the approach described in this chapter is novel. While insider attack detection and prevention is an active research area, we are unable to find existent works specifically targeting the problem of insider data theft using anomaly detection, system metrics, and system state information. Furthermore, we demonstrate our work through instantiation and experimentation with promising results.

Design and Development

An IaaS cloud environment typically consists of virtual machines hosted on physical nodes which run node controller agents. A single node may host virtual machine instances of different tenants. Each tenant may run various applications and workloads. In addition, virtual machines are dynamic and may be created and destroyed by tenant requests at any time resulting in an ever changing environment. Our work posits an approach to detect data theft within an IaaS cloud environment. More specifically, we seek to detect rogue administrators following the flow in Figure 14.

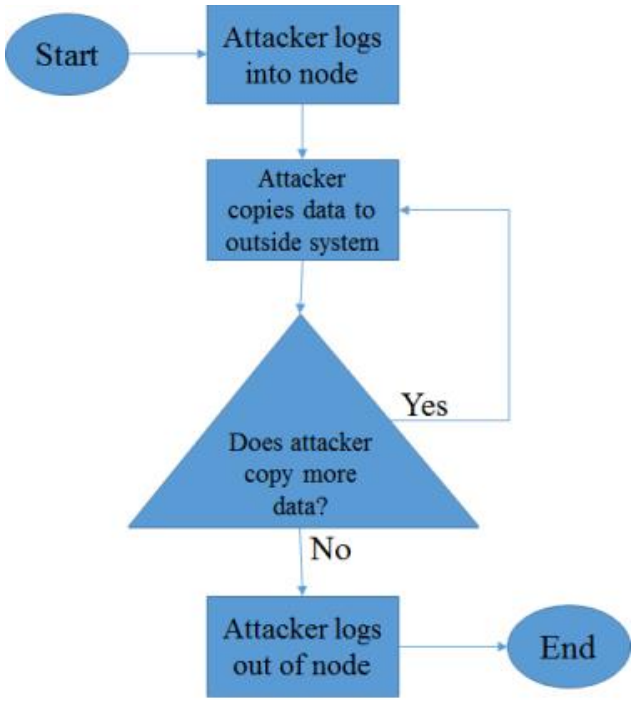

Figure 14. Insider Data Theft Flow Chart

Most production deployment policies restrict administrator login to systems. Our approach supports these controls by logging unusual login events. In addition, knowledgeable attackers are aware of forensic countermeasures. Hence, all of our detection and analysis must be performed on near real time data and persisted to a remote system.

The specific pattern that our system detects is shown in Figure 14 and consists of three steps: attacker logs in node, attacker copies data to outside system, and attacker logs out from node. Each event is considered an attack anomaly. In our approach, we examine the anomaly value for the number of active users and amount of data sent from the node. This approach allows the system to adapt to various environments and adjust to normal fluctuations that can occur in the environment. Furthermore, we examine system state forensic data for open connections and open files. In order for data theft to occur, an external connection must exist and a data file open for reading.

Approach

Our approach for detecting insider data theft uses a three step technique illustrated in Figure 15. First, the system is trained. Then, the system is put into monitoring mode. And, finally, a state-based rules approach is used to detect signatures of insider data theft.

Figure 15. Insider Data Theft Detection Approach

Train

A goal of training mode is to not burden security operations with excessive tuning in order to achieve accurate results. A system with a complicated training requirement lacks scalability. To achieve this goal, the system is placed into training mode while normal IaaS cloud activities occur. Our assumption is that attacks are not occurring during this period. Restricted access and additional manual monitoring may be applied during this period to reduce the likelihood of an attack.

Virtual machines are created and terminated. Tenants run various workloads. Data is sent from an agent on each cloud node hosting virtual machines to our system. During this time, we examine two system metrics for detecting insider attacks: number of active users on nodes and number of tcp bytes transmitted to the network from these nodes. Maximum k-

nearest neighbors anomaly scores derived using the IBM InfoSphere Streams anomaly detection operator (Cancilla, 2015) are calculated to be used later in monitoring mode.

Values arrive separately for each metric every second and are stored in memory. The first 20 values create the reference pattern. A current pattern of 10 values is compared to a subsequence of the reference pattern calculating an anomaly score. A total score is computed from the subsequence comparison scores. As each value arrives, a score is computed, and the window slides to the left. As anomalous events occur, the score increases. This total score is the anomaly value used by our system.

The pattern sizes of 20 and 10 are derived through empirical analysis with a goal of accuracy and performance in mind. Tuning these values is beyond the scope of this work and is considered as future work. In addition, although thr number of active users and bytes sent over the network are the two metrics used for insider data theft detection, we collect metric anomaly data on user space, cpu usage, virtual memory, network connections, input/output read bytes, input/output write bytes, network bytes received, network bytes sent, number of users logged into node, and number of processes. In future work, we plan to investigate machine learning techniques with the goal of deriving more complex insider attack signatures.

Monitor

The system is placed into monitor mode with no attack assumptions. Similar to training mode, system metric data is sent from agents on cloud nodes to our system. Virtual machines are created, terminated, and tenant workloads run. Anomaly scores are calculated as previously done in training mode. However, instead of calculating maximum anomalous scores for each system metric, the calculated values are compared to the maximum values derived during the training period. Values that do not exceed the maximum scores are filtered from the system and ignored.

The plot in Figure 16 illustrates sample anomaly values for network transmission. Figure 17 shows the anomaly values compared to the trained reference pattern.

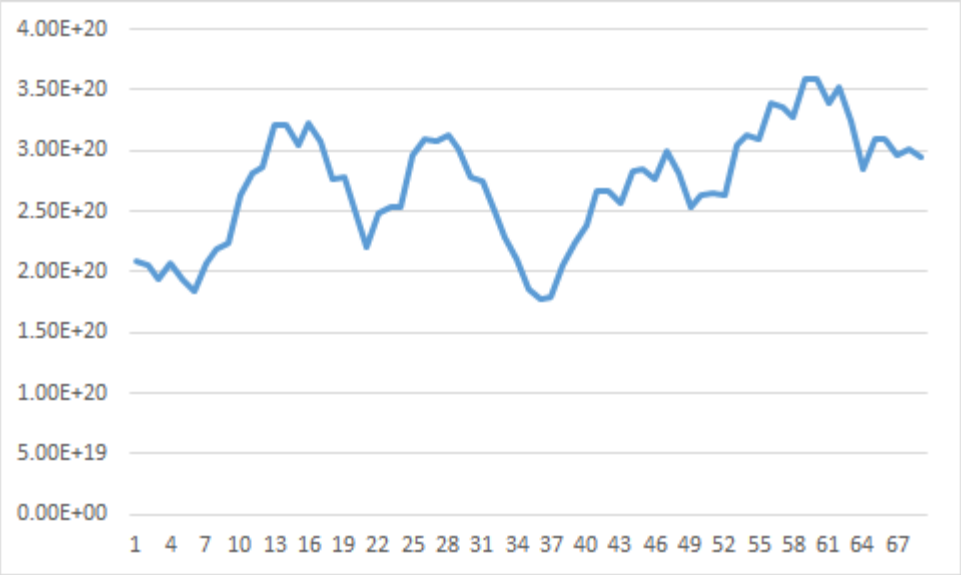

Figure 16. Network txbytes Anomaly Scores under Normal Conditions

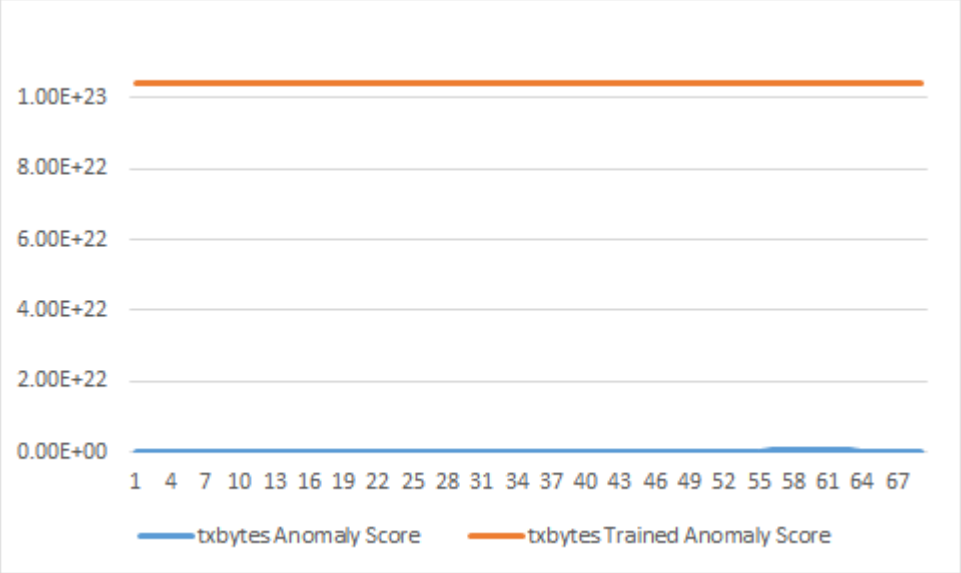

Figure 17. Network txbytes Anomaly Scores Normal Conditions with Trained Max

From Figure 17, one can observe that the trained maximum anomaly score is greater than the current tcp bytes transmitted anomaly score. Hence, anomalous activity is not detected.

Detect

Detection of insider data theft involves three events: a login anomaly (E1), a data transfer anomaly (E2), and forensic evidence (E3) as shown in Figure 18. When all three events are present (A3), we observe a 100 percent detection rate with a zero percent false positive rate. In this case, a login anomaly (E1) is detected followed by an abnormal data transfer (E2). And, forensic evidence (E3) is detected for both a network connection to the node and open files being copied. The forensic evidence is collected by the agent and is analyzed after E1 and E2 anomaly events occur.

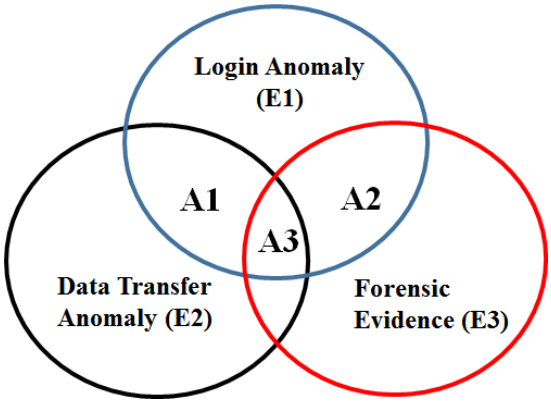

Figure 18. Detection of Insider Data Theft Venn Diagram

During experimentation, we examine condition A1 and A2 in isolation to determine whether a single event can be used for detection. We produce false positives for condition A2 by performing denial of service attacks between virtual machines hosted on nodes in the environment. In the case where E1 is present, a false positive is generated. This is considered a false positive because a denial of service attack is not a data theft scenario. To simulate A1, we turn off forensic evidence detection to examine the false positive and negative rates. We find that using both E1 and E2 as a vehicle to detect insider attacks is mostly successful. However, instances such as Denial of Service attacks or massive data transfers from virtual machines hosted on nodes can result in false positives and false negatives.

Figure 19 illustrates one sample from our experimentation for event E2.

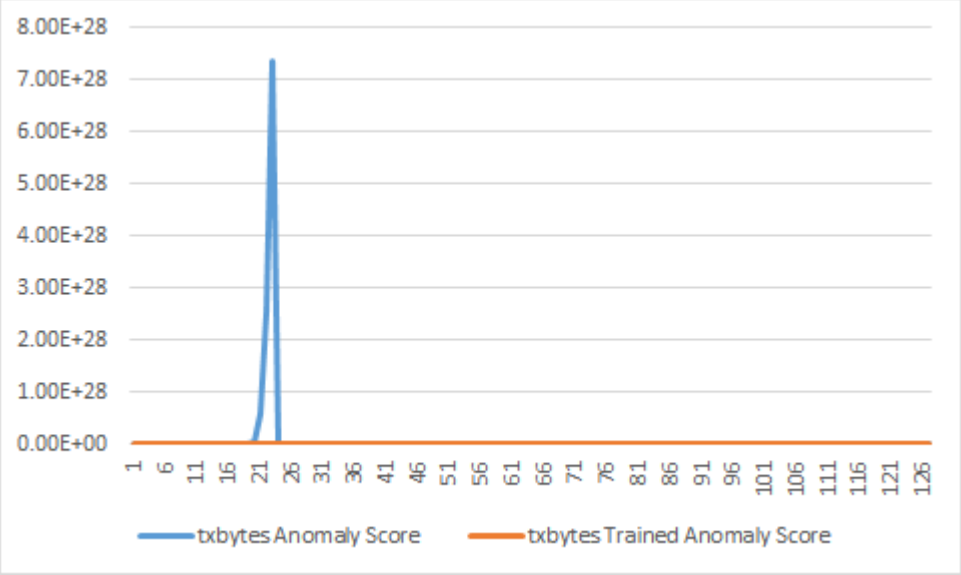

Figure 19. Network txbytes Anomaly

Figure 20 shows a sample from our experimentation for event E1.

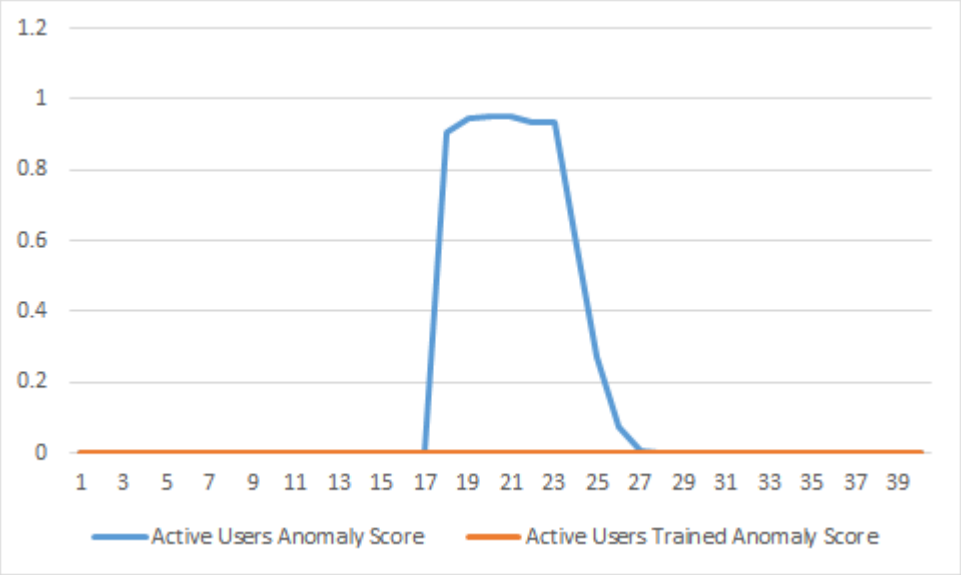

Figure 20. Active User Anomaly

Both Figure 19 and Figure 20 illustrate significant anomaly scores above the trained maximum for network transmitted bytes and active user logins. The detection of all three conditions is required to eliminate false positives.

System Instantiation

We demonstrate our system in an IaaS cloud environment running the Eucalyptus cloud infrastructure shown in Figure 21.

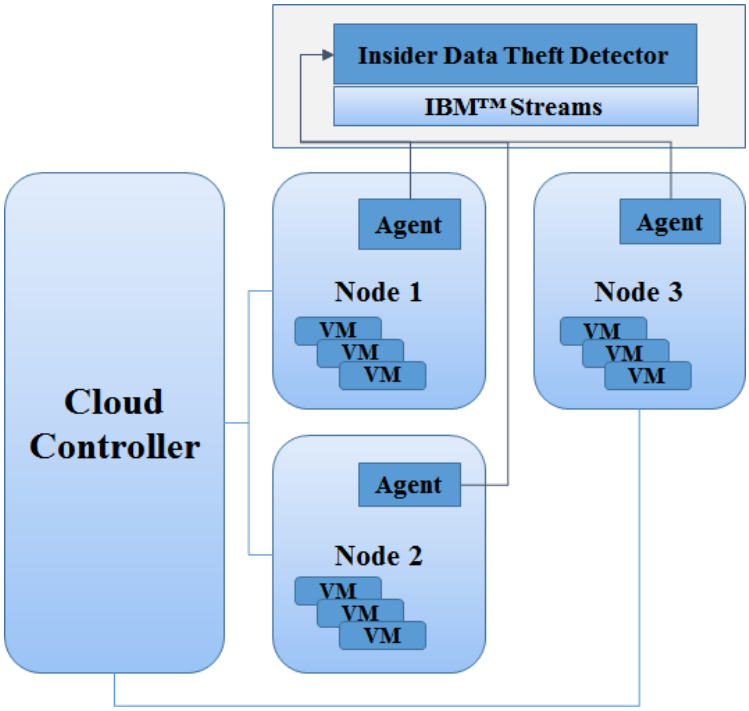

Figure 21. Cloud Environment

Cloud Environment

Figure 21 provides a conceptual view of the experimentation environment. The Eucalyptus cloud framework is used because of its similarities to the popular Amazon cloud infrastructure. IBM Streams provides a scalable infrastructure with built in analytical functions. Furthermore, both technologies are available free of charge for research purposes.

The physical environment consists of five multi-core systems connected over Gigabit Ethernet on a private network. The Cloud Controller contains the management components for the Eucalyptus cloud infrastructure (Nurmi et al., 2009). The Nodes contain Eucalyptus

node controllers and our agent written in Python to gather system metrics. The Insider Data Theft Detector runs IBM Streams and our system implementation code.

Insider Data Theft Detection System Implementation

Our system implementation consists of two components, agents that gather system metrics from cloud nodes and a detector which analyzes the data.

Agents

The Agent component is written in Python and runs on every node hosting virtual machines. It uses the psutil package along with calls to netstat to gather, format, and send data to the detector using a UDP socket connection. The output includes system metrics as described earlier and open connections as well as file state data.

Detector

The Detector is written in two programming languages: Python and Streams Processing Language (SPL). The Python script has two modes of operations, training and monitoring. Similarly, two SPL programs are used for training and monitoring.

In training mode, the Python script ingests metric data from agents and sends it to the SPL program. The anomaly detector operator is used to calculate an anomaly score and maximum training data is persisted in a JSON formatted file.

In monitoring mode, the Python script loads the JSON file into memory and enriches agent data with maximum anomaly values established during training. The SPL program ingests the data from the Python script and calculates the anomaly score similar to training mode. However, instead of saving scores to a file, anomaly values are compared in real time with the maximum values established during training. Values that exceed the maximum anomaly value for a given metric are passed to an alert script written in Python. System state data is persisted to a file as it arrives from each agent and acts as a forensic trail.

The alert Python script listens for abnormal login events and data transfer events. When an event is detected, the forensic data associated with the event is retrieved from the data file. If forensic data related to open connections and files is retrieved from the data file

for the event, an alert is logged indicating that a data theft attack occurred. Login events are always logged.

Demonstration and Evaluation

This section details training the system, data theft attacks executed, attack messages reported, and summarizes our results.

Training the System

Training of the system occurred over a one hour period of time. To test the effectiveness of our approach, we refrain from applying tenant virtual machine workloads during the training period. Instead, we train the system by launching and terminating several different virtual machines. We create up to 12 medium and small virtual machines with Centos 6, Centos 7 and Ubuntu precise images. After the virtual machines become accessible, they are terminated. This approach creates a baseline of activity for the Eucalyptus environment without attempting to predict the workload of users. Furthermore, this meets the goal for a practical and simplistic training approach.

Normal Operating Conditions

Under normal operating conditions, tenant virtual machines are randomly created and terminated placing the cloud environment into various states consisting of starting, stopping, and running virtual machines under load and in idle state. Furthermore, at times, tenant virtual machines place excessive network traffic load on 50 percent of the virtual machines transferring data to and from nodes. Load is placed on the virtual machine using system updates and web data transfers.

Data Theft Attacks

The goal of our experimentation is to demonstrate the effectiveness for using our system to detect data theft of tenant data on IaaS cloud nodes and virtual machines. We use copies of actual virtual machine data which is approximately five gigabytes in size. In addition, we test the system with smaller data theft events, including the theft of data files 500

megabytes and 100 megabytes in size. Various data sizes provide supporting evidence for the effectiveness of the system.

Attack Messages Reported

We conduct 48 attacks during our experimentation. An attack is considered an unapproved login or data transfer event. Of these attacks, 48 are detected and reported. A sample of the system output is shown below in the following format: [node reporting],[date reported],[time reported], [alert message], [forensic data]. The forensic data is reduced because of space constraints.

...”node1.cloud.res”,”2016-03-12”,”14:09:42”,”[INSIDER] [Node: node1.cloud.res] [Attack Detected: Abnormal user login activity detected] ”, ”@sconn(...laddr=(192.168.1.98 22) raddr=(192.168.1.110 52925)...”

”node1.cloud.res”,”2016-03-12”,”14:13:49”,”[INSIDER] [Node: node1.cloud.res] [Attack Detected: Abnormally large data transfer detected] ”,”...popenfile(path= /root/theft1...”

From the sample data above, one can observe that the forensic data reveals the IP address of the attacker and file being copied along with supporting evidence to assist with an investigation.

Results

The goal of our work is to detect insider data theft in IaaS cloud environments. Our approach uses three events, login anomalies, data transfer anomalies, and forensic data to detect attacks. During experimentation, we perform attacks using all three events for attack detection and observe a 100 percent detection rate in under 60 seconds with zero false positives for 48 attacks contained in 233,829 data sets sent by node agents. The results are shown in Table 9.

Table 9. Experimentation Results

IaaS State	VM Workload	Type of Attack	Number of Attacks	Percent Detected
No VMs running	None	Login	3	100%
No VMs running	None	Five GB data theft	3	100%
10 VMs starting	None	Login	3	100%
10 VMs starting	None	Five GB data theft	3	100%
10 VMs running	None	Login	3	100%
10 VMs running	None	Five GB data theft	3	100%
10 VMs stopping	None	Login	3	100%
10 VMs stopping	None	Five GB data theft	3	100%
10 VMs running	Five VM workload	Login	3	100%
10 VMs running	Five VM workload	Five GB data theft	3	100%
10 VMs stopping	Five VM workload	Login	3	100%
10 VMs stopping	Five VM workload	Five GB data theft	3	100%
10 VMs running	Five VM workload	Login	3	100%
10 VMs running	Five VM workload	500 MB data theft	3	100%
10 VMs running	Five VM workload	Login	3	100%
10 VMs running	Five VM workload	100 MB data theft	3	100%

We also examine each event in isolation and find flaws in the use of single events:

Login Events (E1) Only

Examining login events in isolation using our approach detects 100 percent of anomalous user login and logout activity. However, this cannot be used to detect insider data theft.

Data Transfer Anomaly Events (E2) Only

Data transfer anomalies can be solely used to detect data theft events. However, we observe an unacceptably high 22.6 percent false positive rate under extreme operating conditions. These extremes occur during excessive starting and stopping of all virtual machines in the cloud environment and under heavy cloud tenant data transfer workload. Furthermore, when performing denial of service attacks between tenant nodes, we observe a 100 percent false positive rate using these events in isolation.

Forensic Data (E3) Only

System state data for open connections and open files also can be used to detect both abnormal login activity and data theft attacks. However, this approach is unreliable. During our experimentation, we observe normal connection activity between the node controller and the cloud controller. While rules could be created to filter out known connections, the complexity of creating rules and filters would complicate the system and not meet our requirement for usability and ease of use.

Conclusion and Future Work

We put forth a train, monitor, detect pattern for detecting insider data theft attacks. Our system profiling approach utilizes a combination of system metric anomalies and system state data. More specifically, we use a k-nearest neighbors anomaly detection algorithm to score the number of active users on nodes and bytes sent over the network. Excessive scores compared to scores calculated during training indicate an attack event. In addition, system state data on open connections and files is collected and analyzed. Our experimentation suggests that the combination of login events, data transfer events and system state events results in a 100 percent detection rate for insider data theft attacks with a zero percent false positive rate.

To expand on our work, three areas should be explored. First, scalability of the approach needs to be tested in a large IaaS cloud environment. Second, various anomaly detection approaches should be explored. And, third, leveraging machine learning techniques to find rules may reveal combinations of system metrics for better detection of insider attacks.

CHAPTER 8

CONCLUSIONS

In this dissertation, we set out to explore security and privacy concerns in IaaS cloud computing environments. More specially, our research focuses on the instantiation of new security control artifacts that make up a system of novel security controls to reduce the risk of deploying in these multi-tenant and dynamic environments. We set out to explore three research questions. First, how can we detect attacks on cloud tenant instances without specific knowledge of tenant applications in order to preserve privacy? Second, how can we assist cloud providers with interpretation of the output from security controls in an IaaS cloud environment to improve security? And, third, how can we help protect cloud tenants from insider data theft attacks?

Based on the research questions, we aim to achieve three research objectives through the instantiation of new security control artifacts making up an overall system. The first objective is achieved through the instantiation of a new system for detecting abnormal usage in virtual machines using system performance metrics obtained by the hypervisor, or our Hypervisor-based Cloud Intrusion Detection System. In this work, we demonstrate and publish our work which lays the ground work for using performance signatures from hypervisors and rule based attack classification to detect and halt attacks from and against virtual machine instances in cloud environments without knowing the precise workload running on the virtual machine. Instantiation and demonstration of the system reveals a 100 percent detection rate for denial of service attacks from and against the virtual machine.

The second objective is accomplished through the instantiation of a new approach and system for reducing the vast numbers of alarms that can occur from a defense in depth approach with many sensors in the cloud environment. We derive a three step approach consisting of summarize and score, detect anomalies, and classify attacks. Construction and demonstration of this system reveals that we are able to reduce the alarm volume by 95.9 percent. In addition, when properly trained, this approach has a 100 percent classification rate

using Bayesian classification. This work lays the ground work for future research on alert reduction and attack classification from multiple security sensors in cloud environments.

The third objective set forth in our research is the detection of insider data theft attacks. We develop and instantiate a system which detects the transfer of files from cloud nodes which can be used to detect the theft of virtual machines and data stores by examining system state along with anomalies in bytes transmitted and number of active users on the system. We demonstrate the effectiveness of this approach with a 100 percent detection rate of simulated data thefts in a cloud environment. Our research lays the ground work for further research in node system usage metrics and system state from the granular detection of insider attacks.

We accomplished the goals that we set out to achieve of examining the three research questions with an objective to create an artifact for each question. Furthermore, our work is or will soon be published in the proceedings from three flagship IEEE conferences (Nikolai & Wang, 2014, 2016a, 2016b) which further validates our work, demonstrates the relevance of our research, and contributes back to the knowledge base. Although we are successful in our research, we merely lay the ground work for additional exploration and research. Much work is still needed to explore the scalability of our approaches and system. Each artifact must be demonstrated in large scale cloud environments to ensure the effectiveness of the solution. Also, new anomaly detection approaches should be researched to determine optimal techniques for each component. Machine learning and advanced rules engines should also be considered in future research.

REFERENCES

Anderson, S. P., & de Palma, A. (2005). A theory of information overload. *Unpublished Manuscript, Department of Economics, University of Virginia*. JOUR.

Anonymous. (2007). Netcat: the TCP/IP swiss army. Retrieved June 23, 2015, from http://nc110.sourceforge.net/

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., … Stoica, I. (2009). Above the Clouds: A Berkeley View of Cloud Computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, *53*(UCB/EECS-2009-28), 07–013. JOUR. http://doi.org/10.1145/1721654.1721672

Avritzer, A., Tanikella, R., James, K., Cole, R. G., & Weyuker, E. (2010). Monitoring for security intrusion using performance signatures (pp. 93–104). CONF, ACM. Retrieved from internal-pdf://p93-avritzer-1674896640/p93-avritzer.pdf

Bakhtiar, A. (2009). Reverend : 0.4. Retrieved October 21, 2015, from https://www.versioneye.com/python/Reverend/0.4

Ballard, C., Farrell, D. M., Lee, M., Stone, P. D., Thibault, S., & Tucker, S. (2010). IBM Infosphere Streams: Harnessing Data in Motion. GEN, IBM.

Barr, J. (2006). Amazon EC2 Beta. Retrieved October 31, 2015, from https://aws.amazon.com/blogs/aws/amazon_ec2_beta/

Biggs, S., & Vidalis, S. (2009). Cloud Computing: The Impact on Digital Forensic Investigations (pp. 1–6). CONF, IEEE. Retrieved from internal-pdf://05402561-4005005324/05402561.pdf

Bondada, M. B., & Bhanu, S. M. S. (2014). Analyzing User Behavior Using Keystroke Dynamics to Protect Cloud from Malicious Insiders. *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*. CONF. http://doi.org/10.1109/CCEM.2014.7015481

Cancilla, J. (2015). Anomaly Detection in Streams. Retrieved October 21, 2015, from https://developer.ibm.com/streamsdev/docs/anomaly-detection-in-streams/

Christodorescu, M., Sailer, R., Schales, D. L., Sgandurra, D., & Zamboni, D. (2009). Cloud security is not (just) virtualization security: a short paper (pp. 97–102). CONF, ACM.

Claycomb, W. R., & Nicoll, A. (2012). Insider Threats to Cloud Computing: Directions for New Research Challenges. *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*. CONF. http://doi.org/10.1109/COMPSAC.2012.113

Cloud Security Alliance. (2011). *Security guidance for critical areas of focus in cloud computing v3.0* (article). Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Security+Guidance+Critical+Areas+of+Focus+for#0

Cloud Security Alliance. (2013). *The Notorious Nine Cloud Computing Top Threats in 2013* (Report). Cloud Security Alliance. Retrieved from https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

Cobb, S. (2014). Target breach 12 months on a year of lessons learned. Retrieved September 1, 2015, from http://www.welivesecurity.com/2014/12/18/target-breach-lessons-learned/#lessons

Colombe, J. B., & Stephens, G. (2004). Statistical profiling and visualization for detection of malicious insider attacks on computer networks. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* (pp. 138–142). CONF, ACM.

Damiani, E. (2009). Composite intrusion detection in process control networks. JOUR.

Debar, H., & Wespi, A. (2001). Aggregation and correlation of intrusion-detection alerts. *Recent Advances in Intrusion Detection*, 85–103. http://doi.org/10.1007/3-540-45474-8_6

Dell. (2012). Anatomy of a cyber-attack. Retrieved October 1, 2015, from http://software.dell.com/documents/anatomy-of-a-cyber-attack-ebook-24640.pdf

Dhage, S. N., & Meshram, B. B. (2012). Intrusion detection system in cloud computing environment. *International Journal of Cloud Computing*, *1*(2), 261–282. Journal Article.

Doelitzscher, F., Reich, C., & Sulistio, A. (2010). Designing Cloud Services Adhering to Government Privacy Laws (pp. 930–935). CONF, IEEE. Retrieved from internal-pdf://crl-2010-02-0785593356/CRL-2010-02.pdf

Duncan, A. J., Creese, S., & Goldsmith, M. (2012). Insider Attacks in Cloud Computing. *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. CONF. http://doi.org/10.1109/TrustCom.2012.188

Elmrabit, N., Yang, S.-H., & Yang, L. (2015). Insider threats in information security categories and approaches. *Automation and Computing (ICAC), 2015 21st International Conference on*. CONF. http://doi.org/10.1109/IConAC.2015.7313979

Eucalyptus Components. (2013). [Web Page]. Retrieved February 15, 2015, from http://www.eucalyptus.com/docs/3.1/ig/euca_components.html

Fan, G., Jihua, Y., & Mingxing, D. (2009). Design and implementation of a distributed IDS alert aggregation model. *Computer Science & Education, 2009. ICCSE '09. 4th International Conference on*. CONF. http://doi.org/10.1109/ICCSE.2009.5228172

Garfinkel, T., & Rosenblum, M. (2003). A virtual machine introspection based architecture for intrusion detection (Vol. 1, pp. 253–285). CONF, Citeseer. Retrieved from internal-pdf://garfinkel2003-0560777473/Garfinkel2003.pdf

Google. (2015). Google Cloud Platform. Retrieved October 31, 2015, from https://cloud.google.com/

Gul, I., & Hussain, M. (2011). Distributed Cloud Intrusion Detection Model. *International Journal of Advanced Science and Technology*, *34*, 71–82. JOUR.

Gupta, D., Joshi, P. S. S., Bhattacharjee, A. K. K., & Mundada, R. S. S. (2012). IDS alerts classification using knowledge-based evaluation. In *2012 Fourth International Conference on Communication Systems and Networks COMSNETS 2012* (pp. 1–8). http://doi.org/10.1109/COMSNETS.2012.6151339

Hantzis, F. (2015). Ncrack - High-speed network authentication cracker. Retrieved February 13, 2015, from https://nmap.org/ncrack/

Hay, B., Nance, K., Bishop, M., Brian, H., & Hay, B. (2011). Storm Clouds Rising: Security Challenges for IaaS Cloud Computing. In N. Kara & B. Matt (Eds.), *System Sciences*

(HICSS), 2011 44th Hawaii International Conference on (Vol. 0, pp. 1–7). CONF, Kauai, HI: IEEE Computer Society. Retrieved from internal-pdf://10-03-03-2160270622/10-03-03.pdf

Hofmann, A., & Sick, B. (2011). Online Intrusion Alert Aggregation with Generative Data Stream Modeling. *Dependable and Secure Computing, IEEE Transactions on*. JOUR. http://doi.org/10.1109/TDSC.2009.36

IBM. (2015). IBM Cloud. Retrieved October 31, 2015, from http://www.softlayer.com/

IDG. (2014). IDG Enterprise Cloud Computing Study 2014. Retrieved October 31, 2015, from http://www.idgenterprise.com/report/idg-enterprise-cloud-computing-study-2014

Jansen, W., & Grance, T. (2011). Guidelines on security and privacy in public cloud computing. *NIST Special Publication*, *800*, 144. JOUR.

Jiang, G., Chen, H., Yoshihira, K., & Saxena, A. (2011). Ranking the importance of alerts for problem determination in large computer systems. *Cluster Computing*, *14*(3), 213–227. http://doi.org/10.1007/s10586-010-0120-0

Kandukuri, B. R., Paturi, V. R., & Rakshit, A. (2009). Cloud Security Issues. In *Services Computing, 2009. SCC '09. IEEE International Conference on* (pp. 517–520). CONF. Retrieved from internal-pdf://05283911-1171854084/05283911.pdf

Khorshed, M. T., Ali, A. B. M. S., & Wasimi, S. A. (2011). Monitoring Insiders Activities in Cloud Computing Using Rule Based Learning. *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*. CONF. http://doi.org/10.1109/TrustCom.2011.99

Kumar, V. (2012). Signature Based Intrusion Detection System Using SNORT. *IJCAIT*, *1*(3), 35–41. Journal Article.

Lee, J.-H., Park, M.-W., Eom, J.-H., & Chung, T.-M. (2011). Multi-level Intrusion Detection System and log management in Cloud Computing. *Advanced Communication Technology (ICACT), 2011 13th International Conference on*. CONF.

Lin, C. (2009). Modeling and Analyzing Dynamic Forensics System Based on Intrusion Tolerance. In L. Zhitang, G. Cuixia, & L. Yingshu (Eds.), *Computer and Information Technology, International Conference on* (Vol. 2, pp. 230–235). CONF. Retrieved from

internal-pdf://3836b230-3469768729/3836b230.pdf

Lori, M. K. (2010). Can Public-Cloud Security Meet Its Unique Challenges?, *8*, 55–57. MGZN. Retrieved from internal-pdf://msp2010040055-0820948992/msp2010040055.pdf

Lyon, G. (2015). Nmap: The Network Mapper. Retrieved March 2, 2015, from https://nmap.org/

Ma, J., Li, Z., & Zhang, H. (2009). A fusion model for network threat identification and risk assessment. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on* (Vol. 1, pp. 314–318). CONF, IEEE.

Mandiant. (2014). M-Trends: Beyond the Breach - 2014 Threat Report. *2014*. Retrieved from https://dl.mandiant.com/EE/library/WP_M-Trends2014_140409.pdf

Maxim, M. (2011). Defending against insider threats to reduce your IT risk. *Security and Compliance, Jan*. JOUR.

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing.[Online] http://csrc.nist.gov/publications/nistpubs/800-145. *SP800-145.pdf*. JOUR.

Microsoft. (2015). Microsoft Cloud. Retrieved October 31, 2015, from http://www.microsoft.com/enterprise/microsoftcloud/default.aspx#fbid=Sz0L-G7AlOP

Mihaescu, C. (n.d.). Naive-Bayes Classification Algorithm. Retrieved October 15, 2015, from http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf

Mohamed, H., Adil, L., Saida, T., & Hicham, M. (2013). A collaborative intrusion detection and Prevention System in Cloud Computing. *AFRICON, 2013*. CONF. http://doi.org/10.1109/AFRCON.2013.6757727

Murilo, N., & Steding-Jessen, K. (n.d.). chkrootkit. Retrieved March 13, 2015, from http://www.chkrootkit.org/

Nikolai, J., & Wang, Y. (2014). Hypervisor-based cloud intrusion detection system. In *2014 International Conference on Computing, Networking and Communications (ICNC)* (pp. 989–993). http://doi.org/10.1109/ICCNC.2014.6785472

Nikolai, J., & Wang, Y. (2016a). A Streaming Intrusion Monitoring and Classification System

for IaaS Cloud. In *2016 IEEE 9th International Conference on Cloud Computing*.

Nikolai, J., & Wang, Y. (2016b). A System for Detecting Malicious Insider Data Theft in IaaS Cloud Environments. In *IEEE GLOBECOM 2016*.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., & Zagorodnov, D. (2009). The eucalyptus open-source cloud-computing system (pp. 124–131). CONF, IEEE.

Oppenheimer, D. L., & Martonosi, M. R. (1997). Performance signatures: A mechanism for intrusion detection. In *Proceedings of the 1997 IEEE Information Survivability Workshop*. Conference Proceedings.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. JOUR.

Ren, K., Wang, C., & Wang, Q. (2012). Security challenges for the public cloud. *IEEE Internet Computing*, *16*(1), 69–73. http://doi.org/10.1109/MIC.2012.14

Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, pp. 41–46). CONF, IBM New York.

Saad, S., & Traore, I. (2011). A semantic analysis approach to manage IDS alerts flooding. *Proceedings of the 2011 7th International Conference on Information Assurance and Security, IAS 2011*, 156–161. http://doi.org/10.1109/ISIAS.2011.6122812

Salem, M. B., Hershkop, S., & Stolfo, S. J. (2008). A survey of insider attack detection research. *Insider Attack and Cyber Security*, 69–90. JOUR.

Salem, M. Ben, Hershkop, S., & Stolfo, S. J. (2008). A survey of insider attack detection research. In *Insider Attack and Cyber Security* (pp. 69–90). CHAP, Springer.

Sanfilippo, S. (2013). Hping [Web Page]. Retrieved January 15, 2015, from http://www.hping.org/

Sanfilippo, S. (2015). hping. Retrieved July 12, 2015, from http://www.hping.org/

SANS. (2001). Defense in Depth. http://doi.org/10.1038/scientificamerican0502-101

Sheridan, J., & Cooper, C. (2012). *Whitepaper: Defending the Cloud* (Report). (R. I. Security,

Ed.).

Singhal, A., Qin, X., & Lee, W. (2007). Discovering Novel Attack Strategies from Infosec Alerts. *Data Warehousing and Data Mining Techniques for Cyber Security*, *31*, 109–157. Retrieved from http://dx.doi.org/10.1007/978-0-387-47653-7_7

Socketsoft.net. (2013). [Web Page]. Retrieved from http://www.socketsoft.net/

Sriram, M., Patel, V., Harishma, D., & Lakshmanan, N. (2014). A Hybrid Protocol to Secure the Cloud from Insider Threats. *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*. CONF. http://doi.org/10.1109/CCEM.2014.7015476

Stolfo, S. J., Salem, M. B., & Keromytis, A. D. (2012). Fog Computing: Mitigating Insider Data Theft Attacks in the Cloud. *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*. CONF. http://doi.org/10.1109/SPW.2012.19

Sutton, O. (2012). Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction. *University Lectures, University of Leicester*. JOUR.

Takabi, H., Joshi, J. B. D., & Ahn, G. J. (2010). Security and privacy challenges in cloud computing environments. *Security & Privacy, IEEE*, *8*(6), 24–31. JOUR.

Taleb, N. N. (2010). *The black swan:: The impact of the highly improbable fragility* (Vol. 2). BOOK, Random House.

Taylor, J. B., & Williams, J. C. (2008). *A black swan in the money market* (RPRT). National Bureau of Economic Research.

Valeur, F., Vigna, G., Kruegel, C., & Kemmerer, R. A. (2004). Comprehensive approach to intrusion detection alert correlation. In *Dependable and Secure Computing, IEEE Transactions on* (Vol. 1, pp. 146–169). JOUR. http://doi.org/10.1109/TDSC.2004.21

Wen, S., Xiang, Y., & Zhou, W. (2010). A Lightweight Intrusion Alert Fusion System. *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)*, (1), 695–700. CONF. http://doi.org/10.1109/HPCC.2010.120

Willis-Ford, C. (2015). Education & Awareness: Manage the Insider Threat. In *Fissea Working Group*. Retrieved from http://csrc.nist.gov/organizations/fissea/2015-conference/presentations/march-24/fissea-2015-willis-ford.pdf

Zikopoulos, P., & Eaton, C. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. Book, McGraw-Hill Osborne Media.

APPENDICES

APPENDIX A: HCIDS SYSTEM DESIGN

System Design

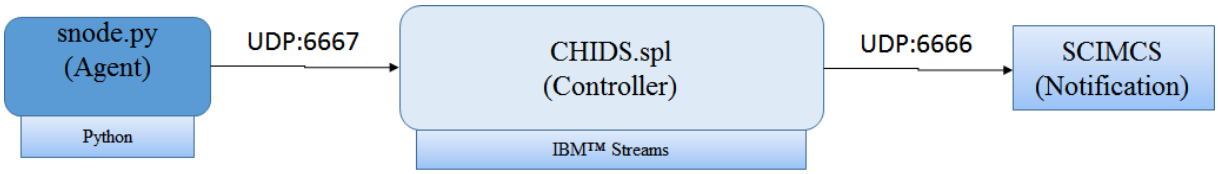

Figure 22. Hypervisor-Based Cloud Intrusion Detection System Design

Snode.py

This program retrieves system metric data every second from the hypervisor and passes it to CHIDS.spl for analysis.

CHIDS.spl

This program looks for abnormal spikes in metric data and then compares to known attack patterns. Then, it sends the attack information to the SCIMCS system.

APPENDIX B: SCIMCS SYSTEM DESIGN

System Design

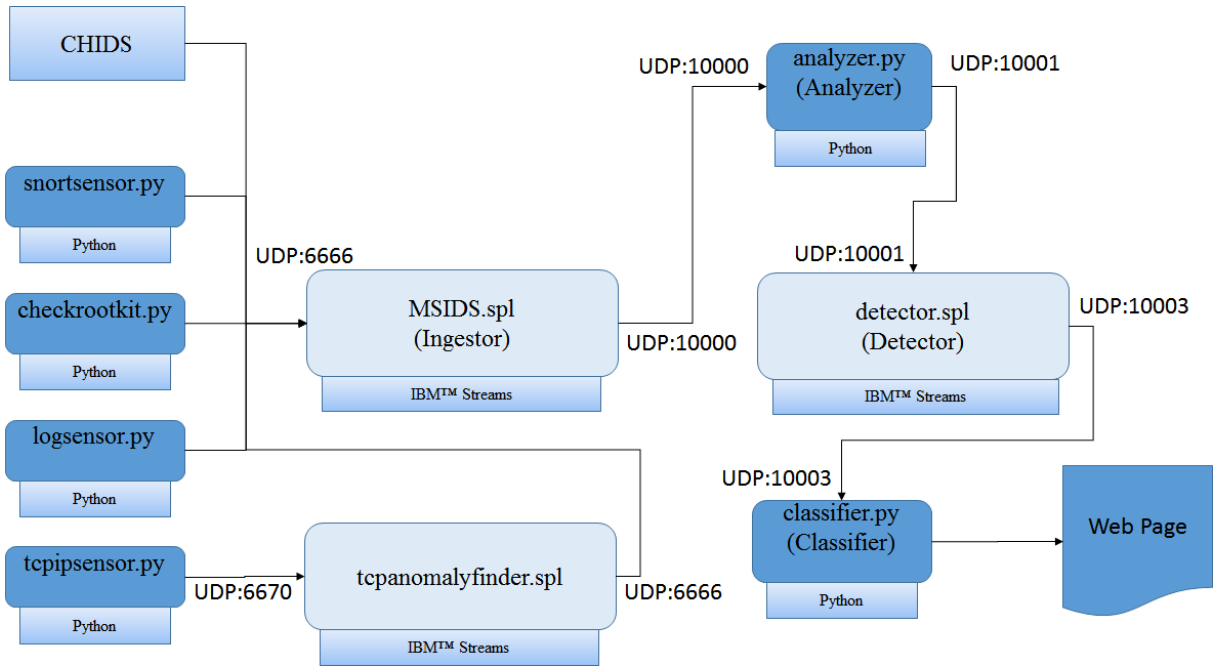

Figure 23. Streaming Intrusion Monitoring and Classification System Design

Launch.sh

This script was used for starting the sensors during experimentation.

Snortsensor.py

This program retrieves data from the popular Snort intrusion detection system, formats it, and passes it the MSIDS.spl.

checkrootkit.py

This program uses the chkrootkit package to search for known rootkits. It formats the output and passes any results to MSIDS.spl.

Logsensor.py

This program retrieves and formats data from log files. Then, it sends the data to MSIDS.spl.

Tcpipsensor.py

This code was originally written by someone going under the handle Silver Moon. It was modified to sniff tcp/ip traffic, format it and send the data to tcpanomalyfinder.spl.

MSIDS.spl

This program ingests data from sensors, buffers the data, and controls the flow rate into analyzer.py.

Tcpanomalyfinder.spl

This program ingests sniffed formatted data from tcpipsensor.py and looks for anomalies in the rate at which packets are sent and the size. Then, abnormal activity is labeled using rules based on observation. The output is sent to MSIDS.spl.

Analyzer.py

This program does the counting, weighting, and persistence of alerts. It can be considered the brain of the system. The major calculations are done here. As alerts come into the program, they are given a value and passed to detector.spl.

Detector.spl

This program looks for anomalies in alert message scores from analyzer. This performs a filtration effect to stop overwhelming administrators with redundant alerts. In addition, alerts are ranked high, medium, or low and passed to classifer.py

Classifer.py

This program has two modes: training and monitor. In training mode, it takes groups of alerts and stores. A label can be assigned. In monitor mode, the program looks for groups of messages that match previously observed patterns using Bayes classification. The alert data is sent off to be rendered on a web page.

visualize.html

This html script utilizes Google visualization APIs to visualize the output from classifer.py.

Get_data.cgi

This script feeds data to the visualize.html page for visualizing the output from classifer.py.

APPENDIX C: A SYSTEM FOR DETECTING INSIDER DATA THEFT DESIGN

System Design

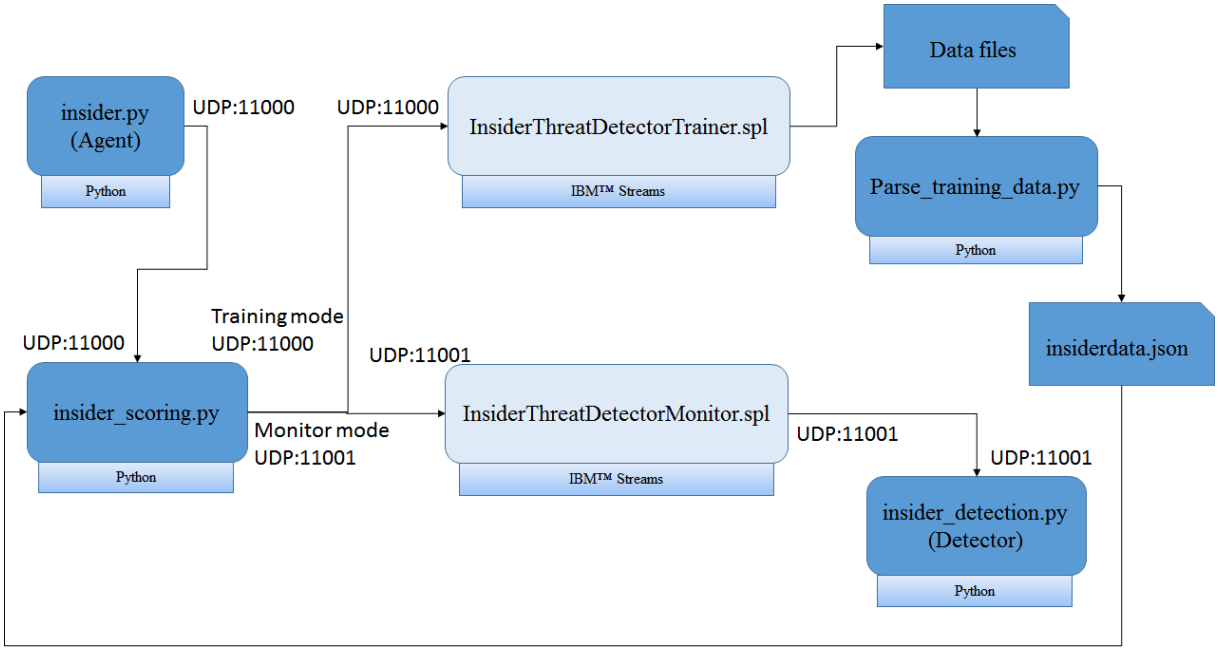

Figure 24. Insider Data Theft Detector System Design

Insider.py

This program gathers the data from the nodes and performs the agent role in the research.

insider_scoring.py

This program has two modes: training and monitoring. In training mode, data is sent to InsiderThreatDetectorTrainer.spl where normal patterns are learned. In monitoring mode, data is enriched from the training data and is sent to InsiderThreatDetectorMonitor.spl where abnormal usage is detected.

InsiderThreatDetectorTrainer.spl

This program calculates anomaly values for each metric passed in and then writes to data files.

Parse_training_data.py

This program reads data files and generates insiderdata.json with the max values observed during the training period.

InsiderThreatDetectorMonitor.spl

This program takes in metric data, calculates an anomaly score and compares the score to the maximum score observed during training.

insider_detection.py

This program ingests the anomaly data from InsiderThreatDetectorMonitor.spl and issues the alerts.