Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Quantitative Evaluation and Reevaluation of Security in Services

Leanid Krautsevich

Supervisor

Dr. Fabio Martinelli

October 20, 2013

Abstract

Services are software components or systems designed to support interoperable machine or application-oriented interaction over a network. The popularity of services grows because they are easily accessible, very flexible, provide reach functionality, and can constitute more complex services. During the service selection, the user considers not only functional requirements to a service but also security requirements. The user would like to be aware that security of the service satisfies security requirements before starting the exploitation of the service, i.e., before the service is granted to access assets of the user. Moreover, the user wants to be sure that security of the service satisfies security requirements during the exploitation which may last for a long period. Pursuing these two goals require security of the service to be evaluated before the exploitation and continuously reevaluated during the exploitation.

This thesis aims at a framework consisting of several quantitative methods for evaluation and continuous reevaluation of security in services. The methods should help a user to select a service and to control the service security level during the exploitation. The thesis starts with the formal model for general quantitative security metrics and for risk that may be used for the evaluation of security in services. Next, we adjust the computation of security metrics with a refined model of an attacker. Then, the thesis proposes a general method for the evaluation of security of a complex service composed from several simple services using different security metrics. The method helps to select the most secure design of the complex service. In addition, the thesis describes an approach based on the Usage Control (UCON) model for continuous reevaluation of security in services. Finally, the thesis discusses several strategies for a cost-effective decision making in the UCON under uncertainties.

Acknowledgments

First of all, I would like to deeply thank my supervisor Dr. Fabio Martinelli for his support, suggestions and advices during my work, without him this thesis would not have been possible.

My further thanks are to the security group of Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche for the great scientific environment that helped me to elaborate my research skills. Special gratitude is to Dr. Artsiom Yautsiukhin and Dr. Aliaksandr Lazouski for the close collaboration that we had.

I am very thankful to Prof. Pierpaolo Degano and Prof. Francesco Pegoraro for the organization of "Galileo Galilei" Ph.D. School and for the opportunity to carry out the research at the University of Pisa. Special thanks are to Prof. Pierpaolo Degano for his patience and his help in making decisions during the most intricate moments of my Ph.D. studentship.

I am especially grateful to external reviewers of my thesis Prof. Stefanos Gritzalis and Prof. Ketil Stølen for their useful feedback.

I thank the research projects EU-FP7-ICT ANIKETOS, EU-FP7-ICT CONTRAIL and EU-FP7-ICT NESSoS that partially supported my work.

I would like to thank all my friends that stayed with me and that encouraged me during my Ph.D. studies.

Last but not least, there are my warm thanks to my parents and my sister that believed in me all the time.

to Yana

Contents

List of Figures

List of Tables

Nomenclature

A	Set of elements
$a \in A$	Element of a set
$A := \{a : \text{properties of } a\}$	Definition of a set
$\lvert A \rvert$	Number of elements in a set
$A \subset B$	Subset of a set
$A \cup B$	Union of sets
$A \cap B$	Intersection of sets
$A \setminus B$	Subtraction of sets
$A \times B$	Cartesian product
$\min A$	Minimal element of a set
$\max A$	Maximal element of a set
$a + b$	Addition
$a - b$	Subtraction
$a \cdot b$	Multiplication
$\frac{a}{b}$	Division
$\mathbf{Pr}[a]$	Probability of an event
$\mathbf{Pr}[b\|a]$	Conditional probability
$\mathbf{Pr}[a \cup b]$	Probability of exclusive events
$\mathbf{Pr}[a \cap b]$	Probability of simultaneous events
$a \wedge b$	Logical “and”
$a \vee b$	Logical “or”

Chapter 1

Introduction

The modern Internet is oriented to *services* [9, 19, 36, 127]. The purpose of the services is to provide some utility to a service consumer. For instance, the services can be an infrastructure (e.g., disk space, bandwidth), a development platform or a software tool [111]. We understand services as "a software component or system designed to support interoperable machine or application-oriented interaction over a network" [150]. Several advantages make the services popular [29, 128, 169, 173]. The services allow outsourcing complex parts of a business process. The services can be deployed on an existing infrastructure, thus, there is no need for investments into a new infrastructure. Moreover, the services have great interoperability, i.e., can interact with each other and become a part of complex services.

A service consumer selects a service according to her service requirements among several offers proposed by service providers. The service consumer starts to look for an appropriate service analyzing functional aspects of the services. Functional aspects of the services are usually described using Web Service Description Language (WSDL) and published in a service registry by a service provider [138]. Since several services may have required functionality, the service consumer analyzes non functional aspects of the services (e.g., an average response time of a service) in addition to functional ones. Non functional aspects of the services are commonly described in a service level agreement (SLA) template and represent quality of service (QoS) [112]. After the service consumer selects the service, she starts the exploitation of the service. The exploitation may last for a long period, e.g, several days or weeks.

An important part of non functional aspects of services is *security*. We understand security as "preservation of confidentiality, integrity and availability of information" [69, 70]. A special attention should be paid to the management of the security because of several reasons [108, 109]. First, security threats are the source of possible significant damage both to a service consumer and a service provider. Second, the number of security threats grows each year even more increasing the possibility of damage. In this thesis, we consider the security of the services mainly from the service consumer's side. The service consumer aims at selecting a service with the required security level, thus, the security of the service needs to be eval-

uated. Then, the service consumers wants to be sure that the security level still satisfies the requirements during the exploitation of the service. Therefore, the security of the services needs to be continuously reevaluated during the exploitation of the service. Evaluation and reevaluation of security in services require sorting out several theoretical and practical issues which we consider further.

A service consumer requires the security of services to be evaluated during the selection of a service in order to understand whether security is at a required level. We consider security evaluation as "process of obtaining evidence of the security level and performance in systems, products and services" [146]. A possible way to evaluate security is to exploit general security metrics (e.g., the number of attacks existing to a system) [145, 162]. There are qualitative and quantitative security metrics. Qualitative metrics are rather subjective and rough however they are easy to exploit and to understand. Quantitative metrics allow a more precise evaluation of the security however the evaluation procedure is more complicated. While several quantitative metrics were proposed, there is still no general formal model that allows describing quantitative security metrics. A formal model can help to understand the actual meaning of the metrics and to analyze the metrics, check their validity, find overlapping and relations between metrics.

Another approach for the evaluation of security is risk assessment. The main goal of risk assessment is to compute the amount of possible losses which are caused by occurrences of various events. The risk assessment has many advantages: the technique is general enough to be applied to any system, results of the assessment provide the complete vision of security, it helps to justify investments in security, and such justification is understandable for company managers. Sometimes the risk assessment is criticized for providing results with low precision and consuming huge amount of time [74, 153]. Usage of security metrics contributing into the overall risk may facilitate the assessment, e.g., may help to make a preliminary assessment. However, a formalization of risk is necessary to analyze the dependencies between security metrics and risk.

Easy interoperability allows simple basic services to be composed into more complex ones. Selection of complex services (i.e., their design) also requires the evaluation of security. Security of complex services depends on security of all basic services constituting the complex one. A method for the evaluation of security of complex services can be based on security metrics values of basic services [108, 109]. The method should allow evaluating security of a complex service on the basis of different metrics because different metrics can be useful for the evaluation of security. In this case, we assume that a value of a certain metric is available for each simple service constituting the complex one. In addition, the method should allow mappings between metrics. This is necessary when values of different metrics are available for different simple services. In this case, the metrics can be mapped to the needed one, and the value of the needed metric may be derived for the complex service.

Services stand under constant control of service providers. This allows the ser-

vice providers to modify their services promptly in order to meet the needs of service consumers. The modifications may occur during the exploitation of services because the exploitation may last for a long period. Thus, functional and non functional aspects of services may change during the exploitation. Particularly, the security level of the services may change. Therefore, it is not enough to evaluate the security of a service just once before the exploitation starts [12, 107, 136]. The continuous reevaluation of the security is required during the exploitation in order to understand the impact of changes. Recently, the Usage Control (UCON) model [129] was proposed to control a resource usage during long-lasting interactions. The UCON model should be adapted for the continuous reevaluation of security in services during the exploitation.

The evaluation of security is followed by security decisions that aim at improving security. The security evaluation and, thus, the security decisions highly depend on the input data [96, 122, 131]. The example of input data are security preferences of a service provider (e.g., encryption mechanisms currently used by the service provider). The data should be correct and trustworthy, in addition data should be timely delivered for the continuous reevaluation of security. In a distributed environment like a service-oriented architecture (SOA), the data may be delivered with delays, may be intentionally or unintentionally corrupted, i.e., the data used for the security decisions, may be uncertain. A method for making security decisions should take into account possible uncertainties of the input data. Moreover, the continuous query of data is sometime impossible due to limited resources of a system (e.g., network bandwidth, sensor battery). In this case, an effective strategy for querying data is required.

This thesis proposes a framework that focuses on evaluation and continuous reevaluation of security in services and should achieve the following *goals*:

1. *propose a formal model for security metrics and risk*;
2. *allow evaluation of security of complex services on the basis of different security metrics*;
3. *enable continuous reevaluation of security in services during their exploitation*;
4. *take into account possible uncertainties of the data used for security decisions in services.*

1.1 Contributions of the Thesis

We list the main *contributions* of the thesis. Each contribution is the important part of the framework for evaluation and continuous reevaluation of security in services. At the same time, each contribution is valuable as an independent method. Note, that the number of contribution in the list corresponds to the number of a goal which the contribution is supposed to achieve.

1. We developed a formal model for general quantitative security metrics and risk. The model considers the interactions between a system and an attacker as two communicating processes. We proposed possible formal definitions of several general quantitative security metrics and a definition of risk. We made an initial analysis of validity of several security metrics. The analysis showed that all these metrics are valid. The contribution was presented in [92, 93].

   (a) During the work on *Contribution 1*, we determined that metrics highly depend on the model of attacker's behavior that is used during the evaluation. We developed a refined model of the attacker's behavior in order to allow a finer-grained evaluation of security. The model is based on Markov Decision Processes theory. In contrast to existing models, our model considers the attacker as an entity with limited resources and partial knowledge about a target system. Moreover, in our model the attacker may change an initially selected attack path during the attack execution, thus, the behavior of the attacker is adaptive. We presented an algorithm that allows simulating the adaptive behavior of the attacker. We see the model as an extension of *Contribution 1*. The model was introduced in [95].

2. We proposed a general method for the evaluation of security of complex services on the basis of different metrics. The evaluation helps to select the most secure design of a complex service. The method is based on semiring algebraic structures. The method allows the exploitation of a single algorithm for the evaluation of the security of the complex service using different metrics. Moreover, the semirings allow mappings between metrics which are useful for the evaluation of security of the complex service when different metrics are used for the evaluation of simple services. We defined several security metrics as semirings to illustrate the method. The method was described in [94].

3. We developed a method for the continuous reevaluation of security of services using UCON model enhanced with qualitative risk assessment. The model allows a service consumer to select a service with required security. Moreover, the method allows continuous reevaluation of service security and decision making about interactions with the service. Finally, the service provider can exploit our method to determine directions for the improvement of the security of her service in order to better satisfy the needs of the consumers. The approach was presented in [87].

4. We considered the impact of unintentional uncertainties on decision making in the UCON model. We proposed threshold and flip coin strategies to make decision making cost-effective. Moreover, we derived cost-effective strategies for periodic and aperiodic queries of the mutable data for the case when continuous queries are not possible. We presented an architecture for the UCON

enforcement that allows cost-effective decision making and cost-effective data query. The cost-effective methods for the UCON enforcement under uncertainties were described in [85, 86, 88, 89].

1.1.1 List of Publications

This thesis is based on the following publications:

- L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Risk-based usage control for service oriented architecture. In *Proceedings of 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 641–648. IEEE, 2010

- L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Risk-aware usage decision making in highly dynamic systems. In *Proceedings of 5th International Conference on Internet Monitoring and Protection*, pages 29–34. IEEE, 2010

- L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Formal approach to security metrics: what does "more secure" mean for you? In *Proceedings of 4th European Conference on Software Architecture: Companion Volume*, pages 162–169. ACM, 2010

- L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Influence of attribute freshness on decision making in usage control. In *Proceedings of 6th Workshop on Security and Trust Management*, pages 35–50. Springer, 2010

- L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Formal analysis of security metrics and risk. In *Proceedings of 5th Workshop on Information Security Theory and Practice of Mobile Devices in Wireless Communication*, pages 304–319. Springer, 2011

- L. Krautsevich, F. Martinelli, and A. Yautsiukhin. A general method for assessment of security in complex services. In *Proceedings of 4th European Conference ServiceWave*, pages 153–164. Springer, 2011

- L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Cost-effective enforcement of ucona policies. In *Proceedings of 6th International Conference on Risk and Security of Internet and Systems*, pages 1–8. IEEE, 2011

- L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Towards modelling adaptive attacker's behaviour. In *Proceedings of 5th International Symposium on Foundations and Practice of Security*. Springer, 2012

- L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Cost-effective enforcement of access and usage control policies under uncertainties. *IEEE Systems Journal, Special Issue on Security and Privacy in Complex Systems*, 7(2):223–235, 2013

Several other papers were published during completing the Ph.D. curriculum:

- L. Krautsevich, A. Lazouski, F. Martinelli, P. Mori, and A. Yautsiukhin. Usage control, risk and trust. In *Proceedings of 7th International Conference Trust, Privacy and Security in Digital Business*, pages 1–12. Springer, 2010

- L. Krautsevich, F. Martinelli, C. Morisset, and A. Yautsiukhin. Risk-based auto-delegation for probabilistic availability. In *Proceedings of 4th International Workshop on Autonomous and Spontaneous Security*, pages 206–220. Springer, 2011

- L. Krautsevich. Parametric attack graph construction and analysis. In *Proceedings of Doctoral Symposium of International Symposium on Engineering Secure Software and Systems 2012*, pages 29–34. CEUR-WS.org, 2012

- L. Krautsevich, A. Lazouski, P. Mori, and A. Yautsiukhin. Quantitative methods for usage control, 2012. Presented at the International Workshop on Quantitative Aspects in Security Assurance

- L. Krautsevich, A. Lazouski, F. Martinelli, P. Mori, and A. Yautsiukhin. Integration of quantitative methods for risk evaluation within usage control policies. In *Proceedings of 22nd International Conference on Computer Communications and Networks*. IEEE, 2013

1.2 Structure of the Thesis

The thesis continues as follows.

Chapter 2 introduces definitions, provides background information for the thesis and divides goals of the thesis into smaller objectives.

Chapter 3 reviews the state of the art.

Chapter 4 presents a formal model for general quantitative security metrics and risk. The chapter provides a simple check of validity of security metrics.

Chapter 5 discusses modeling adaptive attacker's behavior.

Chapter 6 introduces semiring-based methods for the evaluation of security of complex services on the basis of different metrics.

Chapter 7 is devoted to a UCON-based approach for the continuous reevaluation of security of services.

Chapter 8 focuses on the problem of decision making in the UCON under uncertainties.

Chapter 9 validates whether the goals of the thesis are achieved.

Chapter 10 recalls contributions of the thesis and highlights our ideas about the future work.

Chapter 2

Problem Characterization

The thesis is devoted to a framework for evaluation and continuous reevaluation of security in services. This section contains terminology and background information of the areas considered in the thesis. The section also divides the goals of the thesis into smaller objectives which should be fulfilled in the thesis.

2.1 Background and Terminology

We provide some background information on services and security and recall the terminology related to these areas.

2.1.1 Services

During the epoch of Web 1.0 in 1990s, the Internet was a bunch of non-interactive rather static web resources that possessed limited communicating abilities. Since early 2000s, the Internet moved towards web applications that provided rich abilities for communicating with users and between the applications themselves. The epoch of Web 2.0 arrived and the Internet became oriented to services. Generally speaking, a *service* is a utility provided to a user by the service invocation. One type of services are *web services* that are "software components or systems designed to support interoperable machine or application-oriented interaction over a network" [150]. Further, we usually assume a web service saying "service".

A *service-oriented architecture* (SOA) was proposed to connect service consumers and service providers [128, 132]. The SOA is "an architectural style of building reliable distributed systems that deliver functionalities as services" [52]. We consider the SOA for web services that provides an opportunity for interoperability between web services. Basically, the SOA contains three main entities. A *service provider* is an entity that implements, supplies, maintains and describes a service. The service provider publishes a service in a *service registry* which is an authoritative, centrally-controlled store of information about services. A *service consumer* looks in the

service registry for a service which aspects satisfy the service consumer requirements, selects an appropriate service and exploits the selected service.

A service description contains an information about the different aspects of the service, for example, its capabilities, interface, behavior, and quality. A service consumer analyzes the service aspects during the selection of a service. The service consumer starts with the analysis of functional aspects of services. Functional aspects are usually described with Web Services Description Language (WSDL) [138]. If there are services with required functionality then the service consumer analyses non functional aspects of services. The non functional aspects of the service are usually described as a part of a service level agreement (SLA) template and represent quality of service (QoS) [8, 43]. We are interested in *security* of services which is usually described within the QoS (see Section 2.1.2). We assume that there is a QoS *certifier* in the SOA which is responsible for the certification of QoS aspects of services [138]. The certifier checks whether the QoS aspects of a service are at the level claimed by a service provider and then the certified QoS aspects are published in a service registry.

Essential peculiarity of services is that they are under the constant control of service providers. A benefit of such situation is that the service providers can easily and regularly update their services. The updates are required to meet the needs of the market, i.e., to provide a wider functionality and a better QoS to service consumers. Thus, the aspects of the services may change during the services exploitation which may last for a long period. The service consumers should be aware about these changes. Therefore, continuous reevaluation of the aspects of the services is required. In particular, the security of the services should be continuously reevaluated during the exploitation of the service.

Complex Services

An important feature of services is their interoperability. Simple basic services can be composed into more complex services that can perform complex activities (e.g., business processes) [120, 151, 173]. Two basic concepts for the aggregation of services are orchestration and choreography [130]. The orchestration corresponds to the case when a single party is responsible for the control of the resulted complex service. The choreography allows each service to define its part of interaction. We consider the evaluation of security of complex orchestrated services in the thesis. The evaluation is required for the selection of the most secure design of a service.

Many notations for the description of complex services could be used as a starting point for the security evaluation. For example, Business Process Execution Language (BPEL) [1] is one of the most well-known and wide-spread notations. The main disadvantage of using BPEL for the purpose of the analysis is that this language requires too many low-level details, which are not used for the security evaluation. On the other hand, the process can be described with Business Process Modeling Notation (BPMN) [4]. BPMN is a high-level notation and, thus, is more suitable

for the high-level evaluation of security. In the thesis, we consider complex services described in BPMN notation.

2.1.2 Security

We understand *security* as information security, i.e., as "preservation of confidentiality, integrity and availability of information" according to the definition in ISO/IEC 27002:2005 [69]. Confidentiality is "property that information is not made available or disclosed to unauthorized individuals, entities or processes", integrity is "property of protecting the accuracy and completeness of assets", availability is "property of being accessible and usable upon demand by an authorized entity".

We assume that a violation of security occurs as a result of a successful *attack* attempt. We see an attacks as a way "to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset" [69]. Attack attempt is a deliberate action or a sequence of deliberate actions (attack steps) to execute an attack. We distinguish between intentional and unintentional security violations. For example, a coordinated denial-of-service attack is deliberate and it is within the scope of interests of the thesis. An accidental data disclosure by an employee of a company is unintentional and it is not considered. However, in this thesis we consider how unintentional uncertainties (e.g., a delay of data delivery due to network latency) impact decision making in the UCON model.

We call *attacker* an entity that executes attacks. There are different types of attackers that posses different amount of resources, different skills and behavior. The attacker's behavior determines the way how the attacker selects her actions during the attack. The attacker's skill defines whether the attacker is trained enough to execute the actions of different complexity. The attacker's resources (e.g., time or/and money) bound the set of actions available to the attacker. Roughly speaking, the resources determine how much the attacker can pay for the execution of an action. Current models of attackers frequently do not take into account these parameters of attackers which may lead to a coarse evaluation of security.

Evaluation of Security

Popular approach to evaluate security is to exploit security metrics. A *metric* may be defined as "a proposed measure or unit of measure that is designed to facilitate decision making and improve performance and accountability through collection, analysis, and reporting of relevant data" [62]. Security metrics usually refer to a security level, a security performance, security indicators or a security strength [146]. Frequently, a metric is considered as a method for an evaluation together with a value obtained as a result of the evaluation [58]. Therefore, even if two methods produce the same value, these values are considered as different metrics.

Another popular approach for the evaluation of security is risk assessment [6, 69, 155]. We consider *risk assessment* as "activity to assign values to the probability

and consequences of a risk" which is only the part of risk assessment procedure defined in [69]. We see *risk* as "combination of the probability of event and its consequence" [69]. The following concepts contributes into risk. *Threat* represents "potential cause of an unwanted incident, which may result in harm to a system or organization", for example, an attacker threats to cause an unavailability of a service. *Vulnerability* is "weakness of an asset or control that can be exploited by a threat", for example, a vulnerability is a technical bug in the system that can be exploited by the attacker. *Impact* is "adverse change to the level of business objectives achieved", e.g., the impact can be monetary losses caused by a successful attack. *Asset* is "anything that has value to the organization", for example, some valuable data that should not be disclosed to competitors.

There is no single formal model for security metrics and risk. Thus, there are uncertainties in the real meaning of the metrics, it is difficult to analyze metrics, check their validity, find overlapping and relations between metrics and between metrics and risk. Unsurprisingly, NIST stated that one of the directions in security metrics should be the definition of formal models for the security metrics [73].

2.1.3 Security of Services

For the service consumer side according to ISO/IEC 27002:2005 [69], the security management of third party services should include three activities: (i)"it should be ensured that the security controls, service definitions and delivery levels included in the third party service delivery agreement are implemented, operated, and maintained by the third party"; (ii) "the services, reports and records provided by the third party should be regularly monitored and reviewed, and audits should be carried out regularly"; (iii) "changes to the provision of services, including maintaining and improving existing information security policies, procedures and controls, should be managed, taking account of the criticality of business systems and processes involved and re-assessment of risks.". In the thesis, we are focus on the issues connected with the implementation of the activities. The evaluation of security in services is required to guarantee that security is at a required level before the exploitation (activity (i)). The continuous reevaluation allows monitoring the changes of security and make corresponding security decisions during the exploitation (activities (ii) and (iii)).

From the service provider side, security of a service is reached by the implementation of controls which are "means of managing risks, including policies, procedures, guidelines, practices or organizational structures, which can be administrative, technical, management or legal in nature" [69]. We use interchangeably notions of security controls and security preferences of a service provider. The security preferences of a service provide impact the security level of the service and should be taken into account during the evaluation of security.

Evaluation of Security in Services

We consider the following approach for the evaluation of security by a service consumer. The service consumer derives the security level of the service on her own analyzing the security preferences of service providers which are available in a service registry. The service consumer continuously updates the information about the preferences during the exploitation of the service and reevaluates the security level if the preferences change. If security level becomes worse than required by the service consumer, she makes a decision about interactions with the service, e.g., stops the interactions and selects another service.

2.1.4 Access and Usage Control

Traditional access control models [143], such as Mandatory Access Control (MAC), Discretionary Access Control (DAC) or Role-based Access Control (RBAC), check that *subjects* hold the proper rights before granting them the access to the requested *objects*. In modern computer systems like services, access sessions last for a long time, thus it is not enough to check access rights only before granting the access. Further checks performed during the access sessions are required in order to verify that the access rights are still valid. The *Usage Control (UCON)* model [97, 129] was introduced to satisfy these needs.

We understand *policy* as "statements, rules or assertions that specify the correct or expected behavior of entity" [52]. The UCON model defines three types of policy statements: (i) *authorizations*, which are the predicates over subject and object attributes that are similar to access control policies; (ii) *conditions*, which are predicates over environmental attributes; (iii) *obligations*, which are actions which must be performed. Hence, the decisions process is based on the *attributes* that are characteristics of the requesting subject, the accessed object, and the execution environment. The UCON assumes the attributes as *mutable* characteristics that change over time. To address this issue, the UCON proposes continuous enforcement of the security policy, in order to interrupt ongoing accesses when the corresponding access rights are not valid any longer because of the new attribute values.

In this thesis, we adapt UCON model for evaluation and continuous reevaluation of security in services. We consider a service provider as a subject that is trying to access an object which is an asset of a service consumer, e.g., some valuable data of the service consumer that should be processed. We assume the security requirements of the service consumer and the security preferences of the service provider as the policies based on attributes of object and subject correspondingly. The example of policy is "the data is encrypted with a 128-bit key", the attribute is "the length of the key". The service consumer makes the decision whether the service provider is legitimate to access the data comparing her security requirements to a service and the security preferences of the service provider. The service consumer continuously reevaluates the security of the service after the access to the data is granted. Possible

security decisions for the service consumer in case she identifies that the security of the service is not at the required level are to revoke the access to the data for the current service provider and to select another service provider or to ask the service provider to adjust her security preferences.

Attributes values are data that is used for the decision making in the UCON. Decisions making in the UCON model highly depends on obtaining correct, trustworthy, and fresh attribute values. Attribute values are collected from attribute providers and sent to the UCON authorization system which exploits them for the decision process. Some attributes are remote, i.e., they are managed by attribute providers located outside the administrative domain of the authorization system. Therefore, the attribute values may be received with delays (e.g., due to delays in delivery or processing) or may be corrupted. In this case, some attributes values may be uncertain, i.e., the values possessed by the authorization system are different from the real attributes values. Thus, the access decision may be incorrect and may lead to a violation of the policy. Moreover, in order to save resources of the attribute provider (e.g., a sensor) and bandwidth of the network, up-to-date values of attributes are sent to the UCON authorization system periodically. Hence, some attribute values may be missed which may cause unnoticed violations of policies. The problem of making decisions under uncertainties is general for the UCON model, while also important for the reevaluation of security in services.

2.2 Objectives

The main *goals* of the thesis are:

1. *propose a formal model for security metrics and risk*;
2. *allow evaluation of security of complex services on the basis of different security metrics*;
3. *enable continuous reevaluation of security in services during their exploitation*;
4. *take into account possible uncertainties of the data used for security decisions in services.*

We describe several objectives that we are going to fulfill in order to satisfy the main goals of the thesis.

2.2.1 Formal Model for Security Metrics

- *The model should be capable of formalizing general quantitative security metrics.* There is no single formal model that can describe security metrics. The formal model for security metrics can be useful for the analysis of security metrics, for instance to analyze validity of the metrics, to understand overlapping and relations between metrics.

- *The model should be capable of formalizing risk.* Formal definition of risk may help to understand the relations between security metrics and risk. Risk may incorporate some security metrics, but it is not clear how different metrics contribute into the overall risk value. A preliminary calculation of security metrics contributing into the overall risk may facilitate the risk assessment.

- *The model should help to analyze several general quantitative security metrics and risk.* We would like to exploit the formal model for the analysis of security metrics and risk. At least we are going to perform a simple check of security metrics validity assuming that valid metrics "allow different entities be differentiated from each other" [80].

2.2.2 Security Evaluation of Complex Services

- *The method should allow selecting the most secure design of a complex service.* The evaluation of security of complex services is required for the selection of the most secure design.

- *The method should allow the evaluation of complex services on the basis of different metrics.* This is necessary because several security metrics may be useful for the evaluation of security. The orchestrator of the service can use the method to analyze designs of the service from different points of view.

- *The method should allow mapping between different metrics.* The method should allow evaluating security of a complex service when the same metric is not available for each simple service constituting the complex one.

2.2.3 Continuous Reevaluation of Security in Services

- *The UCON model should be adapted for the continuous reevaluation of services.* We need to define interactions and properties of service providers and service consumers in terms of the UCON model. This should allow exploiting the UCON model for the continuous reevaluation of security in services.

- *The method should help a service consumer to evaluate and continuously reevaluate security of services.* Evaluation and reevaluation of security should assist the service consumer in selecting the most secure service and in decision making about interactions with the service during the service exploitation.

- *The method should be useful for a service provider.* The service provider can improve the security of her service analyzing security requirements of service consumers. Our method should help to obtain the direction of improvements for the service provider.

2.2.4 Impact of Uncertainties on Security Decision Making

- *We should identify uncertainties that may impact decision making in the UCON.* We need to determine intentional and unintentional uncertainties that may impact decision making in the UCON.

- *The method should allow decision making under uncertainties.* We need to enhance the decision making procedure of the UCON in presence of uncertainties. We should analyze the impact of uncertainties on the decision making. In case of too high impact, an access decision should be adjusted to be cost-effective by minimizing possible losses caused by uncertainties.

- *The method should allow computing a strategy for attributes values checking.* The strategy serves for the enforcement of UCON policies when the continuous checks of attributes values are not possible. The strategy is supposed to determine the moment when it is necessary to check attribute values in a cost-effective way that minimizes the impact of uncertainties caused by non-continuous checks of the attributes.

- *The method should propose an architecture for the cost-effective enforcement of usage control policies under uncertainties.* The architecture for the UCON enforcement requires modifications for making the decisions under uncertainties and for computation and exploitation of cost-effective strategies for checking attributes values.

Chapter 3

State of the Art

In this chapter, we review existing quantitative methods and approaches for evaluation and reevaluation of security. We describe how attacker's behavior is modeled during the evaluation. We discuss and usage control models and quantitative methods for the improvement of decision making.

3.1 Evaluation of Security

We review methods and approaches for evaluation of security on the basis of general security metrics and risk.

3.1.1 Security Metrics

Several security metrics are computed on the basis of *attack graphs*. Attack graphs is a wide-spread model for description and analysis of systems security. Ortalo et al. [124] did one of the first works on attack graphs. The nodes in a graph represent possible privileges of an attacker and edges denote exploits which are required to get new privileges. The graph is exploited for the analysis of security of a system and for the comparison of security levels while the system is evolving.

Similar attack graphs were considered by Wing et al. [75, 76, 149] and by Phillips and Swiler [134]. The attack graph is considered as a set of nodes representing vulnerabilities in the system and the set of edges representing attempts (attack steps) to execute the vulnerabilities. Successful execution of a vulnerability gives new privileges to the attacker. The end nodes of the attack graph correspond to the vulnerabilities required to reach the attacker's goal.

Special tools for building attack graphs were developed [148, 157]. Construction of the attack graph usually consists of two steps. The first step is to find the vulnerabilities existing in the system. The second step is to compose possible attacks from the existing vulnerabilities. Both steps can be done either manually or automatically. After the attack graph is composed, it is analyzed to determine possible

security metrics.

The representation of attack graphs in [75, 76, 124, 134, 149] has poor scalability since amount of nodes and edges grows exponentially with the number of considered hosts [7, 125]. To overcome this problem another representation of attack graphs was proposed by Jajodia et al. [123, 165, 166]. In these graphs there are two types of nodes: conditions and exploits, which are linked with two types of edges: require and imply. Require edges connect conditions with exploits and show which conditions are required to exploit a vulnerability. Imply edges connect exploits with achieved conditions. The attacker starts with the initial number of conditions she may satisfy in the beginning (depending on the profile of the attacker) and moves to the final conditions which are her goal. The authors also proposed analysis using this type of attach graph: finding minimal cut-set which leads to impossibility to compromise a goal for an attacker [166] and evaluation of security using weakest-adversary metric [126].

A number of quantitative security metrics are defined on the basis of attack graphs: number of attacks [124], minimal length of attack [124], minimal cost of attack [126], probability of successful attack [163]:

- *Number of attacks* simply counts how many distinct attacks exist in the system, i.e., how many paths from initial nodes to goal nodes are in corresponding attack graph. This metric does not take into account the additional parameters of the attack.

- *Minimal length of attack* describes the minimal number of attack steps required to compromise the system. The intuition behind this metric is simple. The less steps are required to complete the attack, the easier this attack is supposed to be. The assumption behind this metric is that the attack steps have equal probabilities of successful execution.

- *Minimal cost of attack* describes what minimal resources should attacker spend to reach the goal. Thus, the metric assumes that the security of the system should be considered versus the weakest attacker that could try to compromise the system. The metric takes into account the initial resources that the attacker must posses to start the attack.

- *Maximal probability of successful attack* requires an assignment of weights to the edges of the attack graph. Each weight denotes a probability of successful exploitation of an attack step. Finally, the probability of an attack consisting of atomic steps is derived from the probabilities of successful exploitation of these steps.

Attack surface metric [103, 104] proposes a different approach to the description of the security than attack graphs. The authors formally defined the notion of attack surface as sets of entry and exit points, channels, and untrusted data items. These

sets describes the resources that the attacker can exploit to perform the attacks. The bigger attack surface corresponds to the less secure system because the attacker in this case has more ways to execute an attack. The authors of the model formally proved that this metric is equivalent to risk, if the specific assumptions are taken into account.

Another metric is "mean time to failure" metric by Madan et al. [102]. The metric shows how much time an attacker spends to cause a failure of a system. This metric assumes that only one-step attacks are possible, and model the failure as a state transition diagram, that is a generic way to describe intrusion tolerant systems.

Wang et al. [164] defined a k-zero day safety metric. The metric aims at the analysis of an impact of zero day vulnerabilities on a system. A zero day vulnerability is a vulnerability that exists in the system but is not identified or publicly reported. The method of the evaluation of the metric considers an attack graph composed from existed vulnerabilities. Zero day vulnerabilities are modeled as new nodes added into the attack graph. The analysis is made to understand how the additional nodes impact security of the system. Eventually, the metric shows how many zero days vulnerabilities the system can withstand.

The Gap While several security metrics exist, there is no general formal model that is capable of expressing all metrics. Such a model is required to analyze the security metrics, check their validity and understand the relations among the metrics.

Models of Attacker

The evaluation of security depends on the attacker's model. Usually, the simple attackers models are defined as a part of methods for the evaluation of security. Frequently, the attacker is considered as an omniscient entity, that follows a single attack path. This attack path is selected on the basis of subjective functions defined by authors of an evaluation method (e.g., [98, 99]).

Sheyner et al. [148] defined deterministic attackers behavior. Authors proposed that an attacker selects a path in an attack graph on the basis of the highest probability to complete an attack. This path is supposed to be always followed by the attacker. Authors describe an initial idea to search the path using Markov Decision Processes (MDP) theory.

Ortalo et al. [124] introduced two models of the attacker behavior: Total Memory attacker and Memoryless attacker. Total Memory attacker considers further attack steps from the currently occupied node of the attack graph and also attack steps that was not tried from previously visited nodes. Memoryless attacker considers attack steps available only from currently occupied node. The models are used to fined shortest attack path in a graph.

LeMay et at. [98, 99] proposed an attacker-driven approach for the evaluation of security of computer systems. The authors consider the system that behaves probabilistically. Thus, the result of attacker's actions is not known in advance.

The authors introduced the set of functions based on the parameters of an attacker (e.g., possessed skills and time) that helps to compute the best possible attack path, i.e, the attacker selects her actions deterministically. Authors defined a model of the attacker that should enhance computation of security metrics.

Several attacker models which assume that attacker may select alternative ways for compromising a system were proposed for the analysis of cryptographic protocols [44, 105, 114]. The models assume that attackers know the system, i.e., the protocol, while have bounded resources. The models consider specific limitations relevant to the analysis of protocols like computational power and message manipulation capabilities. The models are not suitable for generic computer systems.

Sarraute et al. [144] proposed to use Partially Observable Markov Decision Processes (POMDPs) for attack planning during penetration tests. The authors analyze the system considering network configuration graph instead of an attack graph. In terms of knowledge collecting, authors introduces special actions that allow scanning network hosts.

The Gap Most current models of the attacker are quite simple. The attacker is considered as an entity that possesses full knowledge about the system. Moreover, the attacker usually behaves deterministically, i.e., follows a single path that she selects before the attack start. The evaluation of security on the basis of these models may be too coarse. A refined model of the attacker is required for a finer-grained evaluation of security.

3.1.2 Risk Analysis

An efficient practice for the evaluation of security is risk analysis [2, 6, 69, 155]. Risk analysis came to information systems from industries like chemistry and power engineering [54]. Although this practice was initially developed for managing unintentional threats (dependability), later it was adapted for dealing with intentional threats (security).

Usually, risk analysis is a formal well-defined procedure that contains several steps (like, risk management standards [69, 155]). Risk analysis starts with risk assessment which is activity to assign values to the probability and consequences of a risk [69]. Risk assessment is followed by risk mitigation process that is devoted to the addressing of identified risks. We review existing approaches to the risk analysis paying major attention to the risk assessment part.

Standards for Risk Management

ISO/IEC 27000:2009, 27001:2005, 27002:2005 [69, 70, 71] are well known standards for security management. The 27000 part describes overview and vocabulary of the standard, the 27001 part is devoted to organizational aspects of security, the 27002 part discusses security practices. The standard briefly describes the risk assessment

process but explicitly states that risk assessment is an essential part of security management.

NIST SP 800-30 [155] is a standard for risk management during information system development life cycle. The standard provides a well-defined basic description of risk management process. The standard consists of three phases of risk management process: risk assessment, risk mitigation, and evaluation and assessment, but focuses only on the first two. Risk is assessed using qualitative values (however, quantitative assessment is also possible). Risk mitigation strategy is determined using a trade off analysis.

General Risk Analysis Approaches

Operational Critical Treat, Assets, and Vulnerability Evaluation (OCTAVE) Criteria [6] is a well-defined and widely-known approach for risk analysis. Three phases of OCTAVE are to built asset-based threat profiles, to identify infrastructure vulnerabilities and to develop a security strategy and a plan for risk mitigation. The phases are described using principles, attributes and outputs. Attributes and principles are the features of evaluation process. Outputs define the outcomes that organization should achieve during evaluation process. While principles and attributes are the same for each phase of OCTAVE, outputs are different. OCTAVE Criteria can serve as a basis for the development of risk analysis methods for organizations [168].

CCTA Risk Analysis and Management Method (CRAMM) [3] is a universal method to perform risk management at any organization. CRAMM is supported by an automated tool based on qualitative risk assessment. The tool guides security specialist through the every stage of risk management. CRAMM is considered as a basis for other risk analysis methods, e.g, [78].

CORAS [25, 47, 55, 154] is a framework for model-based security risk analysis. The framework consists of four parts: a terminology, a library, a methodology, and a tool. The terminology combines terminology from security and risk analysis fields. The library contains two repositories. The experience repository contains reusable UML-models, checklists, procedures, etc. The assessment repository stores the results of the actual security analysis. CORAS is supported by an open-source tool. The method have been widely tested in the industry. Several other methods based on CORAS were developed [51, 63, 64].

Several methods for risk analysis are based on questionnaires and checklists. Karabacak and Sogukpinar [79] proposed Information Security Risk Analysis method (ISRAM). ISRAM is a quantitative approach that uses questionnaire results to analyze security risks. Special attention is devoted to creation of questionnaires. A questionnaire contains questions and possible answers about particular security problems. The questionnaire helps to receive the opinions of management and technical personnel about current security problems. Bennett and Kailay [18] introduced a qualitative questionnaire-based method which main goal is to provide a risk analysis for low and mid-size commercial companies. Farahmand et al. [45] described an

approach to evaluate the damage of security incidents using checklists. A checklist is exploited for the computation probabilities and tangible and intangible losses of incidents.

Shawn A. Butler [28] described cost-benefit analysis method called Security Attribute Evaluation Method (SAEM). The approach is used to analyze and to compare security designs of financial and accounting information systems. The method is based on the multi-attribute assessment, where analysis is performed using several criteria at once. For example, impact of different threats is considered using four criteria: the loss of productivity, the loss of revenue, regulatory penalties, and the loss of reputation. The overall impact for a threat is a weighted sum of these losses. Similar analysis is performed for the selection of the most appropriate protection strategy. Countermeasures are selected depending on how well they mitigate risk, how costly they are, and how much maintenance they require. In addition, the author proposed a coverage analysis, which is based on the idea of defense-in-depth and defense-in-breadth. The need of a countermeasures is determined by the aim to have at least some protection against all most dangerous threats and to have protection mechanisms on different levels.

There are many methods which are based on risk and have their small peculiarities. Wei et al. [167] proposed a cost-benefit analysis for network intrusion detection system. Tarr and Kinsman [158] used fault trees to identify the frequency of undesirable events and to determine the consequences of events. Ryan and Ryan [142] presented risk management method that uses system failure analysis based on the measures of relative risk. Relative risk is the ratio of two probabilities: the probability of a system failure before the countermeasures implementation and the probability of a system failure after the countermeasures implementation. McGraw [110] pointed out that risk should be compared with operational needs. The author did not provide any information about how this risk and operational needs could be calculated.

Risk Management Approaches for Software Development

Microsoft corporation introduces a risk management approach for software development based on Security Risk Management Discipline (SRMD) [113]. Two types of risks are considered: the first one is connected with the current project life cycle, the second one is related to the life cycle of computer system. Microsoft uses security risk statements to describe the risks. A security risk statement contains two parts. The first part describes the state of a system or a potential threat that can cause some damage. The second part describes the possible impact of an event from the first part. Microsoft follows a quantitative way for the evaluation of assets, losses, and costs of countermeasures as well as for calculating risk values.

Cigital Risk Management Framework (CRMF) [161] is a framework to manage risks during software development life cycle. CRMF has two main peculiarities: it stresses that risk management should be an iterative, continuous process and it

considers threats to business goals, not to assets.

Risk analysis of software systems based on security patterns is presented by Halkidis et al. [57]. The idea of security patterns is similar to the idea of design patterns, which allow to create well-structured and reusable software. The risk is used to represent the security extent of a pattern or a system. Security level of a system is evaluated during patterns merging.

The Gap Although risk is a popular mean for the evaluation of security, standards and approaches to risk analysis mostly consider a system as a static entity, take a lot of time to be applied, require experienced specialists to participate, and are expensive and difficult to be carried out. Due to these reasons usual practice is to perform risk analysis only few times per year (sometimes one time per several years) [140]. Modern dynamic systems like services change frequently and require continuous risk assessment to provide rapid and efficient response to system changes. Moreover, different methods propose different models for risk. A general formal model capable of describing risk is required for the analysis of relations between security metrics and risk. Preliminary evaluation of security using metrics may facilitate risk assessment.

3.2 Evaluation of Security in Complex Systems

While previously reviewed approaches mainly consider a computer system as a whole, it is frequently required to evaluate the security of a complex system that is composed from more simple parts. There are several approaches suitable for the evaluation of composed systems. Usually approaches aim at the evaluation of security of the complex system by aggregation of security levels of its parts [16, 17, 56, 101]. We consider several methods for the evaluation of the security of complex systems and also several methods devoted to complex services.

Freeman et al. [48] described risk assessment in large heterogeneous systems. Two approaches to risk assessment are studied. The first one is down-top approach when risk for subsystems is assessed in the beginning and then aggregated into the risk of the whole system. The second one is a top-down risk assessment approach when the system is considered as the whole during initial risk assessment and then risk assessment of subsystems is performed.

The important task of complex systems evaluation is to understand and reflect dependencies among presented parts. Balzarotti et al. [17] presented a system as a graph where nodes are system components and edges are the connections between the components. Components and links may contain vulnerabilities. Risk assessment is required to understand how difficult is for an attacker to compromise a component directly or through a some path. Another graph is constructed to describe the dependencies between vulnerabilities. Nodes represent vulnerabilities of components and vulnerabilities of links between the components, edges represent

dependencies between vulnerabilities. The method is intended for the analysis of different design choices, mitigation of risks if system has been modified, selection of security solutions.

Grunske and Joyce [56] modified attack trees to analyze component based systems. Basic attacks are mapped to each element of the power set of all system components. The probability of attack is assessed using attack profiles. The probability of attack depends on different attacker peculiarities such as available computational resources, skills, available amount of money, etc. This information about an attacker is saved in a profile. When probabilities of basic attacks are assessed, the attack tree is evaluated and the total risk of the system is obtained. The main problem of this method is the process should be repeated from the beginning after the system is reconfigured or countermeasures are applied.

Clark et al. [34] used a risk tree for the computation of risk for a system. The authors considered a tree constructed on the basis of the business objectives of an organization. The main business objectives are decomposed until small sub-objectives are found. A number of assets required for fulfillment of sub-objective are assigned to the corresponding leaves of the tree. Using this tree, an analyst determines the value of each asset. The vulnerabilities which can be used to damage assets are discovered and assigned to the corresponding assets. Then, the risk for the main business objectives is computed by aggregating risks for leaf nodes.

Zambon et al. [118, 171, 172] considered a graph constructed for a layered structure of an enterprise. In [171], the authors analyzed availability risks. In [118], authors used the graph to analyze risks in an enterprise where components are connected with each other and threats may propagate through the system.

Innerhofer-Oberperfler and Breu [68] considered a large enterprise and described how overall risk for an enterprise can be assessed considering various risks related to different parts of the enterprise. First, a dependency graph is constructed. The graph has four layers: (i) a business layer containing business objects, (ii) an application layer which contains applications required for business objects, (iii) a technical layer containing basic software (e.g., operation systems), and (iv) a physical layer which contains physical objects (e.g., laptops, servers). Different types of risk are assessed for different elements of the graph. There is a corresponding administrator who evaluate the risk according to the graph. Finally, risk of failure of security business goal is computed. The work was continued in [27] with a finer-grained analysis of the enterprise.

3.2.1 Evaluation of Security in Services

Massacci and Yautsiukhin [109, 108] proposed a specific method for the analysis of security of a complex service (a business process). The method transforms the business process into a tree and selects the most secure design according to the defined aggregation functions. The security level of business processes is evaluated on the basis of assurance indicators. The authors use probability of compromising an

activity as an assurance indicator. The total assurance level of a business process is calculated from the values of indicators of sub-processes using a directed graph. The authors also take into account that the values of assurance indicators which come from partners can be untrusted, therefore the authors modify the values according to the trust level of the partners. The algorithm can be adopted for the analysis of dynamic system.

Cheng et al. [32] proposed a framework for aggregation of downtime of a complex service. Derwi et al. [38] analyzed security in complex services using multi-objective optimization. The aim of the framework is to analyze the workflow in order to select a set of security solutions for a complex service. Moreover, authors consider 0-1 metrics which show whether a simple service is secure or it is not secure. Similar problems have been considered in non-security domains. For example, Jeager et al. [72] provided several aggregation functions for such criteria of services as minimal execution time, cost, etc. Yu et al. [170] proposed a method for the selection of alternative complex services designs using the graph theory.

There are several approaches for the evaluation of security in services that propose the comparison of security preferences of a service provider and security requirements of a service consumer. Ronda Henning [61] proposed to evaluate security of a service against 15 security domains using a level from 1 to 4. Casola et al. [31] proposed a method for selection of the best alternative based on the distance between two lists of security levels using the assessment results provided by the method of [61]. In addition, Casola et al. [30] proposed a more generic method for the aggregation of different evaluation results for security and quality of service.

The Gap There are several methods for the analysis of complex systems. However they are specific in different senses. The methods are system specific (e.g., consider a computer network) or metric specific, i.e., exploit just one way of the evaluation (e.g., compute risk). Simple services composing a complex service may be evaluated using different security metrics. Thus, a general method that can take into account several metrics is required.

3.3 Reevaluation of Security

As we described above, usual approaches to the evaluation of security (e.g., risk analysis) are manual, tedious, time consuming, and very expensive activities. They are performed regularly but only few times per year or rarer. Recently, approaches for continuous reevaluation of security were proposed in order to make the control of security efficient and effective for the application in dynamic systems.

Gehani and Keden [50] proposed an approach based on quantitative continuous risk assessment for an intrusion prevention for a network host. Authors supposed the host runs an operating system and an access control mechanism. The approach uses the idea of the host exposure which is the set of permissions granted to the

software by access control mechanism. The permissions restrict an access to the resources (e.g., files) of the host. Overall risk level is connected with the exposure and changes after each event that changes the exposure (e.g., granting or denying a permission of an access to a resource). There is a level of risk that the host can tolerate (threshold). If the risk is above the threshold then a cost benefit analysis is done to select countermeasures for an adjustment of risk level. The adjustment of the risk should be done with the minimum impact on the host performance. This issue requires solving a knapsack problem. In addition, the approach proposes relaxation of security restrictions to improve performance when general risk level becomes below the threshold. The approach considers a single host and does not suitable for implementation in distributed environment. The approach pays no attention to collecting of information for risk assessment and its trustworthiness and freshness.

Arnes et al. [10] presented a method for continuous risk assessment of network systems. The method is based on observations from network monitoring sensors or intrusion detection systems (IDS). The overall risk level of the system is computed by the aggregation of data from sensors. The information about an object provided by a sensor is weighted according to the trust level of the sensor. Sensors are monitored by agents that are special programs which are the part of IDS. The agents can communicate and cooperate with each other. Thus, an agent can receive the data from different sensors. The data form sensors may contain mistakes or be inconsistent, so the agent cannot be sure if this information about the state of the object is correct. As a result, real security state of the object is hidden from the agent. This is the reason to represent an object as discrete-time Hidden Markov model. Hidden Markov models are used for the computation of the additional probability of an object to be attacked. This probability is used together with the data obtained from sensors to compute the risk of the object to be attacked. The overall risk is the simple sum of risks of objects. The proposed approach was enhanced with continuous-time Hidden Markov model [60].

Miura-ko and Bambos [116] introduced an approach focused on dynamic risk mitigation. A computing system is presented as a set of nodes. Each node has a risk indicator which denotes risk as a function of time. A vector consisted of such indicators is called a risk profile. The set of risk profiles forms an integer lattice for basic Markov model. Subsets of risk profiles represent different risk zones (e.g., zone of low risk, medium, high). The risk changes and the risk profile transition occurs after each event occurred in the system. The system manager can control risk level applying different defend strategies. Markov model is extended to a non-Markov case using the notion of "risk shocks". A risk shock increases the value of risk indicator hitting the node, while a defense strategy decreases the risk value.

Refsdal and Stølen [141] introduced an approach based on risk indicators for providing dynamic risk picture. The threats in the system are modeled with CORAS threats diagrams. The diagrams help to identify key risk indicators and functions for threat execution likelihoods and impacts. The main element in the model is a Risk Monitor, which next to dynamic creation of risk picture also is responsible for

evaluating the consistency of and measuring the confidence in a risk picture.

Blyth and Thomas [23] proposed a real-time threat assessment of security incidents using data fusion of intrusion detection system (IDS) logs. The approach is based on the idea of a footprint. The footprint is a set of characteristics (e.g., actions and activities) observed during a period of time. The footprints allows identifying an attack and understanding how an attacker has accomplished this attack. The paper describes a framework for an automatic footprints analysis. Each host in the network contains a host IDS (HIDS) for its own attack detection, all HIDS observations are sent to the footprint manager which analyses general situation in the system.

The Gap Current approaches focus on the evaluation of security for a system owner. In services, the security should be evaluated for a service consumer that does not possess the system. Security requirements of the service consumer may be fulfilled partially by security preferences of a service provider. However, service consumer still needs to select and exploit a service.

3.4 Access and Usage Control

Access control is another area where the approaches for the continuous security control of the systems were developed. With new technologies where an access session lasts for long time (such as Web Services, Cloud computing, Grid) it is not enough to check access rights of a subject before granting an access to an object, but checks of rights during the access sessions are required. *Usage Control* model (UCON) was introduced to satisfy the needs [176] of security control over long lasting interactions. We discuss the approaches that tackle the impact of uncertainties on the decision making in the UCON. Moreover, we consider several other quantitative methods that aim at improvement of decision making in the UCON.

3.4.1 Impact of Uncertainties on the Decision Making

Data freshness is an important property for many computer systems (e.g., data caching, replication systems, data warehousing, etc). The property requires the data to be up-to-date to make the right decisions about a computer system. Usually data is not fresh because to the delays of data delivery or data processing. Thus, an unintentional uncertainty exists. Data freshness was studied by the computer science community during past years [24, 131]. Another unintentional uncertainty appears when the data is incorrect, for instance, corrupted by noise.

The importance of authorization information to be up to date during the access decisions was stated by Krishnan et al. [96] and Niu et al. [122]. Authors formally define two security properties: *weak stale safety* and *strong stale safety*. Weak stale safety holds in two cases: first, if a policy holds during the last update of the

attribute before the request and, second, if the policy holds during the check after the request and before the access is granted. Strong stale safety requires the policy to hold during the update after the request and before the access is performed. Authors design enforcement and decision points for group-based Secure Information Sharing (g-SIS) system as state machines and use model checking to show that the points satisfy defined properties.

3.4.2 Quantitative Methods for Access and Usage Control

There is a number of applications of quantitative methods in access and usage control. Usually, the access and usage control is enhanced with trust or risk. Cost-effectiveness of access and usage control is frequently analyzed on the bases of a risk notion. Risk is used to make an access decision taking into account that granting the access is connected with a threat. The threat is evaluated with a probability to occur and a cost in the case of occurrence.

Degano et al. [37] proposed a formal framework for the enforcement of quantitative security policies. The framework suggests two complementary models for the enforcement. The monitoring mechanism evaluates current state of a policy and aborts execution if the policy is violated. Moreover, continuous-time Markov chain is used to compute the probability of a possible policy violation in future. The execution is aborted if the probability of the violation is too high.

Belsis et al. studied the decision making for role-based access control in multi-domain environments. Decision making procedure is enhanced using fuzzy relations which allow to understand how a specific policy statement is satisfied in a multi-domain environment.

Aziz et al. [13] described reconfiguring role-based access control policies using risk semantics. The approach allows measuring a risk for a policy and reconfiguring the policy reducing the risk and saving policy strength. The policy language proposed by the authors contains three types of risks: operational, combinatorial, and conflict of interest. Han et al., [59] considered pre-evaluation of security policies enhanced with risk analysis. Dimmock et al., [42] demonstrated the method of architecture extension of access control with trust evaluation, creation of dynamic policy and risk management in role-based access control where risk is used as one of policy predicates. Zhang et al. [175] proposed the ordered semiring-based trust establishing model with risk assessment. The risk is used as a metric for credential analysis in a distributed environment.

Some authors use risk as a static parameter which simply helps to assign correct privileges taking into account possible losses [59, 100, 152]. For example, Skalka et al. [152] discussed a risk assessment approach for distributed authorization. The formal approach is used to assess and combine the risks of assertions that are used in the authorization decision. The method based on RT trust-management framework is called RT^R: framework for risk assessment in authorization. Li et al. [100] used quantitative risk assessment as the key factor for decision making about access

control in a grid environment. Zhang et al. [174] described a benefit and risk access control (BARAC) model. The idea is based on balancing between the risk of data compromising due to granting access and benefits from granting access.

Other authors use risk as a dynamically changing value which depends on the current value of possible losses and benefits as well as on the probability of abusing granted privileges by a concrete subject [33, 39, 110]. Diep et al. [39] proposed an access control enforcing mechanism with risk as a key component of decision making process during access control. Ni et al. [121] considered a risk-based access control system which assumes that the access to a resource can be granted to a risky subject if mitigation actions (post-obligations) will be applied in the future. The authors proposed an approach for the risk assessment under incomplete and imprecise data using fuzzy inferences.

The Gap Few methods focus on the decision making in presence of uncertainties. Moreover, the methods for decision making enhanced with quantitative methods mostly focus on access control, while usage control is left apart. The specific issue of usage control is that the unnoticed changes of attribute value may occur during the usage session, since continuous control of the attribute value is not always possible. Quantitative methods for access control do not take into account this issue.

Chapter 4

Formal Model for Security Metrics and Risk

We propose a formal model for general quantitative security metrics and risk. The model considers communications between an attacker and a system. The model is used to define security metrics and risk. Moreover, the model helps to analyze the validity of security metrics.

The chapter goes as follows. Section 4.1 recalls what metrics mean in theory of measurements and in the security community. Section 4.2 presents our basic formal model and proposes the definitions of the perfect security, the definition of "more secure" relation. Section 4.3 the definition of risk, and definitions and analysis of general security metrics. We conclude the chapter with Section 4.5 which contains a discussion of the initial results of the formalization of metrics.

4.1 Metrics in Mathematics and in Computer Security

In the measurement theory, we can find the following theorem which helps to establish a link between an empirical evidence and an objective measurement. This is a central theorem which is called a representation theorem [46, 156]:

Theorem 1 *Let Q be a set of elements and q_1 and q_2 be its members ($q_1, q_2 \in Q$). Let also $R = \{r_1, \ldots, r_n\}$ be a set of relations on Q. The tuple $\langle Q, R \rangle$ is called an empirical relational system. Measurement can be seen as an objective-empirical function $\mathscr{M} : Q \to \mathbb{R}$ which assigns a real value to an element and a set of relations $P = \{p_1, \ldots, p_n\}$ on $\mathbb{R}$ (real numbers) which is in a binary relation with R (i.e., each r_i corresponds to p_i). Then:*

$$\forall i, r_i(q_1, q_2, \ldots) \Leftrightarrow p_i(\mathscr{M}(q_1), \mathscr{M}(q_2), \ldots) \tag{4.1}$$

We consider binary measurement systems that are forms of representation theorem [46]:

Definition 1 *A nominal security measurement system is representative if the following holds:*

$$\forall q_1, q_2 \in Q, q_1 \sim_{sec} q_2 \Leftrightarrow \mathscr{M}(q_1) = \mathscr{M}(q_2) \tag{4.2}$$

where $q_1 \sim_{sec} q_2$ means that q_1 is equally secure as q_2.

Definition 2 *An ordinal security measurement system is representative if the following holds:*

$$q_1 \succeq_{sec} q_2 \Leftrightarrow \mathscr{M}(q_1) \geq \mathscr{M}(q_2) \tag{4.3}$$

where $q_1 \succeq_{sec} q_2$ means that q_1 is more secure than q_2.

Definition 2 shows that a measurement function must be monotone. Another definition of metrics in the measurement theory is based on interval system [81]:

Definition 3 *Metric is a function $\mathscr{D}$ on a set Q which determines the distance between two members of the set ($\mathscr{D} : Q \times Q \to \mathbb{R}$) and satisfies the following properties:*

1. $\forall q_1, q_2 \in Q, \mathscr{D}(q_1, q_2) \geq 0$ *(positivity);*
2. $\forall q_1, q_2 \in Q, q_1 = q_2, \mathscr{D}(q_1, q_2) = 0$ *(identity);*
3. $\forall q_1, q_2 \in Q, \mathscr{D}(q_1, q_2) = \mathscr{D}(q_2, q_1)$ *(symmetry);*
4. $\forall q_1, q_2, q_3 \in Q, \mathscr{D}(q_1, q_3) \leq \mathscr{D}(q_1, q_2) + \mathscr{D}(q_2, q_3)$ *(triangle inequality).*

The computer security community usually uses the term "metric" as it is shown in Definitions 1 and 2: one value is assigned to a system which defines how secure the system is. We stick to this common definition of "metric" accepted in the computer security community. Thus, we need to define what does it mean that two systems are equally secure or that one system is more secure than another one.

We aim at establishing a "more secure" relation. The relation should be similar to the ones from physics where without measurements we often can say that one object is hotter (or longer, or brighter) than another one. For example, the length of objects can be compared simply by putting the objects close one to another. Unfortunately, in the security community there is no such widely accepted answer. Usually, authors first define a way of measurement and then say that this means that the relation between two measurements defines relations between security levels of the two systems (e.g., [126]). We try to overcome this challenge and formalize "more secure" relation.

4.2 Formal Model

We propose a formal model that allows a more accurate discussion about security metrics and risk. We introduce definitions of a perfect security and "more secure" relation. We consider a system in and out of the context.

4.2.1 Security of a System out of a Context

The initial target of our analysis is a system which is applied out of a context, i.e., we do not consider parameters of attackers and possible impact of attacks. We will add context to our model in Sections 4.2.2. We use the notation of the process algebra [106] and define security as follows:

Definition 4 *Let S be a process modeling behavior of a system and X a process modeling behavior of an attacker. The system and the attacker perform actions $a_i \in A_S$ and $a_j \in A_X$ correspondingly and move from one state to another one: $S \xrightarrow{a_i} S'$ and $X \xrightarrow{a_j} X'$. We denote a trace of actions accomplished by the system as γ_S and by the attacker as γ_X. A trace $\gamma = \gamma_S \bullet \gamma_X$ is a result of merging one trace of actions with another one in a way that preserves the order of events. We say that the system S is (perfectly) secure if and only if:*

$$\forall X, \gamma = \gamma_S \bullet \gamma_X, \gamma_S \in S, \gamma_X \in X, S \xrightarrow{\gamma_S} S' \wedge X \xrightarrow{\gamma_X} X', \\ S \| X \xrightarrow{\gamma} S' \| X' \Rightarrow \mathscr{P}_{sec}(S' \| X') = \emptyset \tag{4.4}$$

Function $\mathscr{P}_{sec}(S' \| X')$ returns the set of possible goals successfully achieved by an attacker in the state $S' \| X'$ (e.g., the attacker has root access to a database) when the system and the attacker work in parallel. Equivalence to the empty set means that no goals are successfully achieved, i.e., the security [69] is preserved. We write $\gamma_X \in X$ to show that the attacker may execute a trace of actions and $\gamma_S \in S$ to show that the system may execute a trace of actions. A trace of actions is denoted in the following way preserving the order of actions: $\gamma = a_1 \circ a_2 \circ \cdots \circ a_n$. We use $a \in \gamma$ notation to denote that an action a is contained in the trace γ.

Definition 5 *An attack to a system S is a trace of actions γ_X:*

$$\forall X, \gamma = \gamma_S \bullet \gamma_X, \gamma_S \in S, \gamma_X \in X, S \xrightarrow{\gamma_S} S' \wedge X \xrightarrow{\gamma_X} X', \\ S \| X \xrightarrow{\gamma} S' \| X' \Rightarrow \mathscr{P}_{sec}(S' \| X') \neq \emptyset \tag{4.5}$$

Thus, the attack is the trace of actions of attacker that leads to a state where the attacker reaches her goal (set of goals).

We define the set of attacks relevant to a system.

Definition 6 *Let $\mathcal{X}_S$ be the set of attackers relevant to a system S then the set of attacks relevant to S is:*

$$\Gamma_S := \{\gamma_X : \gamma = \gamma_S \bullet \gamma_X, \gamma_S \in S, \gamma_X \in X, X \in \mathcal{X}_S, \quad (4.6)$$
$$S \xrightarrow{\gamma_S} S' \wedge X \xrightarrow{\gamma_X} X', S\|X \xrightarrow{\gamma} S'\|X' \Rightarrow \mathscr{P}_{sec}(S'\|X') \neq \emptyset\}$$

We derive a definition that determines the "more secure" relation.

Definition 7 *Let Γ_{S_1} be a set of attacks relevant to a system S_1 and Γ_{S_2} be a set of attacks relevant for a system S_2. We say that the system S_1 is more secure than or equally secure to the system S_2 ($S_1 \succeq_{sec} S_2$) if a set of attacks Γ_{S_1} relevant to the system S_1 is included into a set of attacks Γ_{S_2} relevant to the system S_2 ($\Gamma_{S_1} \subseteq \Gamma_{S_2}$).*

Definition 7 does not allow distinguishing between equal or higher security in case $\Gamma_{S_1} \subset \Gamma_{S_2}$ thus we call this definition *non sensitive*. The *sensitive* definitions can be formalized in the following way.

Definition 8 *We say that the system S_1 is more secure than the system S_2 ($S_1 \succ_{sec} S_2$) if a set of attacks Γ_{S_1} relevant to the system S_1 is included into a set of attacks Γ_{S_2} relevant to the system S_2 ($\Gamma_{S_1} \subset \Gamma_{S_2}$).*

Definition 9 *We say that the system S_1 is equally secure to the system S_2 ($S_1 \sim_{sec} S_2$) if a set of attacks Γ_{S_1} relevant to the system S_1 is equal to a set of attacks Γ_{S_2} relevant to the system S_2 ($\Gamma_{S_1} = \Gamma_{S_2}$).*

Though the definitions (Definition 7 or Definition 8) indicate that one system is more secure than the other one, it is a rare case when the set of possible attacks for one system is completely included into the set of possible attacks for another system. However, the definition can still be good for the initial analysis of general quantitative security metrics.

4.2.2 Security of a System in a Specific Context

We extend our model considering security of a system in a specific context. The context includes protected assets and possible attackers. In particular, we need the amount of possible losses, caused by affecting valuable assets, and parameters of attackers.

We start with a more detailed model of attacker. We consider only entities that try intentionally violate security of a system.

Definition 10 *An attacker is a process which is characterized by the following tuple: $\langle goal, \Gamma_X, skill, intang, tang\rangle$, where goal is the goal of the attacker[1]; Γ_X is a set of attacks the attacker knows; skill is the level of skills the attacker possesses;*

[1]In our model every attacker has only one goal

intang is the amount of intangible resources (e.g., time) the attacker can spend to achieve her goal; tang is the amount of tangible resources (e.g., money) the attacker can spend in order to make an attempt to compromise the system.

Considering every attacker separately is an impractical approach. We consider similar attackers as one collective entity, or an attacker profile $\mathcal{X}$. We assume that all members of the same profile of attackers have the same goal *goal*. For example, cyber-terrorists aim at shutting down a system for a long time, cyber-thieves (hackers) want to receive economical benefits, insiders commit a fraud. Thus, we group the attackers according to their goals assuming that the attackers which have the same goal have also similar skills and resources (i.e., we assume small dispersion). Sometimes, there are attacker profiles which have a similar goal but should be grouped differently (e.g., terrorists which usually have high skills and large amount of resources, and simple hooligans which have very limited amount of resources). Such groups can be separated, and this separation will not affect our further discussions.

We would like to consider tangible and intangible resources required for an attack apart from each other. In our model, tangible resources are needed for buying the tools without which the attack is impossible (e.g., in order to crack a safe a special drill is required). When the attacker starts her attack she spends intangible resources in order to achieve her goal. The more intangible resources are spent the more chances for success the attacker has (e.g., the more time a bugler spends for studying and attempting to open a lock the more probably she will be able to open the safe). Sometimes tangible and intangible resources are connected, e.g., the more money is spent on a computer the more powerful it is and the less time is required to perform a brute-force attack on a cipher-text. We consider the resources as two distinct sets in order to show the different nature of these expenditures.

Each attack has a cost, thus, the attacker spends her resources during the attack. Cost may be considered as a one-time payment which an attacker has to make in order to exploit a vulnerability, i.e., to make a single step of an attack. An example could be the average amount of money required for bribing an employee in order to get access to the network or to buy information about an unknown vulnerability on a black market [147]. Such model is not entirely correct. First, one-time payment is usually an indispensable condition, but not a sufficient one. Possessing the information about an existing vulnerability and required tools do not always imply its successful exploitation. Second, in many cases different amount of investments may result in different probabilities of success. For example, the higher the bribe is, the higher the probability that it is accepted. Third, in contrast to the real world criminals (e.g., buglers or thieves), hackers often do not need special equipment, but a computer, tools (likely, simply downloaded) and access to the Internet (or to the internal network). In other words, exploitation of most of vulnerabilities often does not require one-time investments.

Therefore, we propose to consider two types of cost: a fixed cost C_{fix} and a

changing cost C_{chg}. The first cost is the common one-time investment. Such investment is required to allow the attacker to make an attempt to execute an attack. The changing cost is the investment which influences the probability of successful exploitation of a vulnerability. Such investment is often only the time the attacker devotes to exploitation of a vulnerability. We can express this time in currency by simple multiplication of the time spent by the cost of an hour of the attacker (a way of transformation does not affect the further discussion). The idea behind this cost is the following one: anyone can exploit a vulnerability spending some time trying to do this (see, for example, the work of Jonsson and Olovsson [77] where even unskilled attackers were able to compromise a system after considerable time).

In order to model such dependency we can use either lognormal [119] or Weibull distributions. Both these distributions are used for modeling faults. In our case we can see the problem as how long the system withstands an attack. We also can apply multiplicative degradation argument here. In every small amount of time an attacker gets a tiny amount of knowledge about how to exploit a vulnerability. In this case system is "degrading" until it is broken. Such degradation is modeled by lognormal distribution [14].

We consider how a probability of successful execution of a single action depends on a cost in order to analyze how the probability of successful execution of the whole attack depends on the resources possessed by the attacker. The probability of successful execution of action a_k is a conditional probability that the action is successfully executed if the attacker spent $C_{chg}(a_k)$:

$$\mathbf{Pr}[a_k|C_{chg}(a_k)] = \frac{\mathbf{Pr}[a_k \cap C_{chg}(a_k)]}{\mathbf{Pr}[C_{chg}(a_k)]} \tag{4.7}$$

where $\mathbf{Pr}[C_{chg}(a_k)]$ corresponds to the probability that the attacker belongs to a certain profile because $C_{chg}(a_k)$ is specific for the attacker profile (attacker skill level), $\mathbf{Pr}[a_k \cap C_{chg}(a_k)]$ is a joint probability that cost $C_{chg}(a_k)$ is spent and the action is successful. We assume that the probability $\mathbf{Pr}[a_k \cap C_{chg}(a_k)]$ also depends on parameters of attacker profile.

Definition 11 *The probability of successful attack γ_X executed by the attacker X is the maximal probability to accomplish successfully all required actions, if the overall sum of resources spent for the overall attack is less than or equal to the amount of resources the attacker has:*

$$\begin{aligned}\mathbf{Pr}_v[\gamma_X, X] = \max\{ \prod_{\forall a_k \in \gamma_X} \mathbf{Pr}[a_k|C_{chg}(a_k)] : \gamma = \gamma_S \bullet \gamma_X, \gamma_S \in S, \gamma_X \in X, \\ S \xrightarrow{\gamma_S} S' \wedge X \xrightarrow{\gamma_X} X', S\|X \xrightarrow{\gamma} S'\|X' \Rightarrow \mathscr{P}_{sec}(S'\|X') \ni goal, \\ \sum_{\forall a_k \in \gamma_X} C_{chg}(a_k) \leq intang\}\end{aligned} \tag{4.8}$$

The subscript v of probability $\mathbf{Pr}_v$ stands for a successful *violation* of security which we use in the same sense as a successful attack. We assume that the successful exploitation of each attack action is an independent event. Therefore we can compute the probability of the successful execution of the attack as a product of probabilities of the successful execution of steps.

Definition 12 *The fixed cost is used for defining the set of attacks available to an attacker X in a system S:*

$$\Gamma_{S_{|X}} := \{\gamma_X : \gamma = \gamma_S \bullet \gamma_X, \gamma_S \in S, \gamma_X \in X, S \xrightarrow{\gamma_S} S' \wedge X \xrightarrow{\gamma_X} X', \\ S\|X \xrightarrow{\gamma} S'\|X' \Rightarrow \mathscr{P}_{sec}(S'\|X') \ni goal, \sum_{\forall a_k \in \gamma_X} C_{fix}(a_k) \leq tang\} \tag{4.9}$$

4.3 Definitions and Analysis of Security Metrics

In this section, we present formal definitions for several general quantitative metrics which are used for evaluating security. We analyze formally defined metrics against Definitions 2 and 7. While the definitions are coarse, they are still useful for a simple check of metrics validity. We assume that a metric is valid if it "allows different entities be differentiated from each other" [80]. We formally represent the check of each metric as a criterion.

We consider the following metrics:

- number of attacks,
- minimal length of attack,
- minimal cost of attack,
- maximal probability of successful attack,
- attack surface metric.

For a metric $\mathscr{M}$ we write $\mathscr{M}(S)$ to denote that the metric is computed for a system S, e.g., for a workstation with all hardware and software installed.

Number of attacks Number of attacks metric defines how many attacks to a system exist. The idea behind this metric is that the more attacks for a system exist the less secure the system is. This metric is applied for the simplest analysis of attack graphs [124, 126]. Number of attacks also can be used for the analysis of results of the penetration testing.

Definition 13 *Number of attacks $N_{att}(S)$:*

$$N_{att}(S) = |\Gamma_S| \tag{4.10}$$

We introduce the relation $\succeq_{N_{att}}$.

Definition 14 *We say that the system S_1 is more secure than the system S_2 according to the number of attack metric if the number of attacks existed to the system S_1 is less than the number of attacks existed to the system S_2:*

$$S_1 \succeq_{N_{att}} S_2 \Leftrightarrow N_{att}(S_1) \leq N_{att}(S_2) \tag{4.11}$$

Criterion 1 *The number of attacks is a valid security metric:*

$$S_1 \succeq_{sec} S_2 \Rightarrow S_1 \succeq_{N_{att}} S_2 \tag{4.12}$$

Proof

$$S_1 \succeq_{sec} S_2 \Rightarrow \Gamma(S_1) \subseteq \Gamma(S_2) \Rightarrow N_{att}(S_1) \leq N_{att}(S_2) \Rightarrow S_1 \succeq_{N_{att}} S_2 \tag{4.13}$$

□

Minimal Length of Attack An intuition behind this metric is the following: the less steps an attacker has to make, the simpler is to execute the attack successfully, and the less secure the system is. The attack length is:

Definition 15 *The length L_{γ_X} of attacks γ_X is:*

$$L_{\gamma_X} = |\gamma_X|, \gamma = \gamma_S \bullet \gamma_X, \gamma_S \in S, \gamma_X \in X, S \xrightarrow{\gamma_S} S' \wedge X \xrightarrow{\gamma_X} X', \quad S \| X \xrightarrow{\gamma} S' \| X' \Rightarrow \mathscr{P}_{sec}(S' \| X') \neq \emptyset \tag{4.14}$$

Here we slightly abuse the notation using $|\gamma_X|$ to determine the number of steps in a sequence.

Definition 16 *The minimal length of attack $L^{min}(S)$ is:*

$$L^{min}(S) = \min_{\forall \gamma_X \in X} \{L_{\gamma_X} : \gamma_X \in \Gamma_S\} \tag{4.15}$$

We introduce the relation $\succeq_{L^{min}}$.

Definition 17 *We say that the system S_1 is more secure than the system S_2 according to the minimal length of attack metric if the minimal length of attack existed to the system S_1 is greater than the minimal length of attack existed to the system S_2:*

$$S_1 \succeq_{L^{min}} S_2 \Leftrightarrow L^{min}(S_1) \geq L^{min}(S_2) \tag{4.16}$$

Criterion 2 *The minimal length of attack is a valid security metric:*

$$S_1 \succeq_{sec} S_2 \Rightarrow S_1 \succeq_{L^{min}} S_2 \tag{4.17}$$

Proof

$$S_1 \succeq_{sec} S_2 \Rightarrow \Gamma(S_1) \subseteq \Gamma(S_2) \Rightarrow L^{min}(S_1) \geq L^{min}(S_2) \Rightarrow S_1 \succeq_{L^{min}} S_2 \tag{4.18}$$

□

Minimal cost of attack Minimal cost of attack (see Definition 18) has sense only for the fixed cost C_{fix}, but as we noted, possessing this amount of money does not always guarantee successful exploitation. The changing cost C_{chg} simply cannot be minimal because even with a little effort an attacker has a chance (but a very small chance) to achieve her goal. Example could be the password cracker who finds a strong password after a couple of attempts by sheer luck.

Definition 18 *Minimal fixed cost of attack $C_{fix}^{min}(S)$:*

$$C_{fix}^{min}(S) = \min_{\forall \gamma_X \in X} \{ \sum_{\forall a_k \in \gamma_X} C_{fix}(a_k) : \gamma_X \in \Gamma_S \} \tag{4.19}$$

We introduce the relation $\succeq_{C_{fix}^{min}}$.

Definition 19 *We say that the system S_1 is more secure than the system S_2 according to the minimal fixed cost of attack metric if the minimal fixed cost of attack existed to the system S_1 is greater than the minimal fixed cost of attack existed to the system S_2:*

$$S_1 \succeq_{C_{fix}^{min}} S_2 \Leftrightarrow C_{fix}^{min}(S_1) \geq C_{fix}^{min}(S_2) \tag{4.20}$$

Criterion 3 *The minimal fixed cost of attack is a valid security metric:*

$$S_1 \succeq_{sec} S_2 \Rightarrow S_1 \succeq_{C_{fix}^{min}} S_2 \tag{4.21}$$

Proof

$$S_1 \succeq_{sec} S_2 \Rightarrow \Gamma(S_1) \subseteq \Gamma(S_2) \Rightarrow C_{fix}^{min}(S_1) \geq C_{fix}^{min}(S_2) \Rightarrow S_1 \succeq_{C_{fix}^{min}} S_2 \tag{4.22}$$

□

Maximal probability of successful attack The probability to accomplish an attack successfully describes the most probable way to compromise the system [163].

Definition 20 *We define the maximal probability of successful attack as follows:*

$$\mathbf{Pr}^{max}(S) = \max_{\forall \gamma_X \in X} \{ \mathbf{Pr}_v[\gamma_X, X] : \gamma_X \in \Gamma_S \} \tag{4.23}$$

We introduce the relation $\succeq_{\mathbf{Pr}^{max}}$.

Definition 21 *We say that the system S_1 is more secure than the system S_2 according to the maximal probability of successful attack metric if the maximal probability of successful attack existed to the system S_1 is less than the maximal probability of successful attack existed to the system S_2:*

$$S_1 \succeq_{\mathbf{Pr}^{max}} S_2 \Leftrightarrow \mathbf{Pr}^{max}(S_1) \leq \mathbf{Pr}^{max}(S_2) \tag{4.24}$$

Criterion 4 *The maximal probability of successful attack is a valid security metric:*

$$S_1 \succeq_{sec} S_2 \Rightarrow S_1 \succeq_{\mathbf{Pr}^{max}} S_2 \tag{4.25}$$

Proof

$$S_1 \succeq_{sec} S_2 \Rightarrow \Gamma(S_1) \subseteq \Gamma(S_2) \Rightarrow \mathbf{Pr}^{max}(S_1) \leq \mathbf{Pr}^{max}(S_2) \Rightarrow S_1 \succeq_{\mathbf{Pr}^{max}} S_2 \tag{4.26}$$

□

Attack surface metric This metric has been proposed by Howard [65] and Manadhata and Wing [103].

Definition 22 *Let us have three assets which can be affected by an attack:* method (m), data items (d), channel (c). *Let us know the damage-potential level* $dmg_{pot}(\gamma_X)$ *of each asset and the level of privileges* $priv(\gamma_X)$ *required for execution of an attack* γ_X *(maximal difference in level of privileges among required actions of the same attack). Then, for every system we can assign the following tuple:*

$$ASM(S) = \langle Risk^m, Risk^c, Risk^d \rangle \tag{4.27}$$

where:

$$Risk^m = \sum_{\forall \gamma_X \in \Gamma^m} \frac{dmg_{pot}(\gamma_X)}{priv(\gamma_X)};\ Risk^c = \sum_{\forall \gamma_X \in \Gamma^c} \frac{dmg_{pot}(\gamma_X)}{priv(\gamma_X)};\ Risk^d = \sum_{\forall \gamma_X \in \Gamma^d} \frac{dmg_{pot}(\gamma_X)}{priv(\gamma_X)} \tag{4.28}$$

where $\Gamma^m, \Gamma^c, \Gamma^d$ *are the sets of attacks leading to compromise of the corresponding asset.*

We introduce the relation $\succeq_{ASM}$.

Definition 23 *We say that the system* S_1 *is more secure than the system* S_2 *according to the attack surface metric if the attack surface of the system* S_1 *is smaller than the attack surface of the system* S_2*:*

$$S_1 \succeq_{ASM} S_2 \Leftrightarrow ASM(S_1) \leq ASM(S_2) \tag{4.29}$$

Criterion 5 *The attack surface is a valid security metric:*

$$S_1 \succeq_{sec} S_2 \Rightarrow S_1 \succeq_{ASM} S_2 \tag{4.30}$$

Proof

$$S_1 \succeq_{sec} S_2 \Rightarrow \Gamma(S_1) \subseteq \Gamma(S_2) \Rightarrow ASM(S_1) \leq ASM(S_2) \Rightarrow S_1 \succeq_{ASM} S_2 \tag{4.31}$$

□

N_{att}	L^{min}	C_{att}^{min}	$\mathbf{Pr}^{max}$	ASM
+	+	+	+	+

Table 4.1: Analysis of validity of security metrics

Table 4.1 gathers the results of criteria, where "+" means that the metric is valid and "-" means that the metric is not valid.

4.4 Definition of Risk

We formally define risk which is one of the most general methods for the evaluation of the security of a system considered in a specific context.

Let a number of attacker profiles be $N_{\mathcal{X}}$ and let each profile $\mathcal{X}_j$ make $|\mathcal{X}_j| = N_{\mathcal{X}_j}$ attempts to execute attacks, the total number of attempts is $N_X^{ttl} = \sum_{j=1}^{N_{\mathcal{X}}} N_{\mathcal{X}_j}$. Suppose $\Gamma_{S|\mathcal{X}_j}$ is the set of attacks available to the profile $\mathcal{X}_j$. Then the number of attacks available to $\mathcal{X}_j$ are $N_{\Gamma_{S|\mathcal{X}_j}} = |\Gamma_{S|\mathcal{X}_j}|$.

Definition 24 *The risk $Risk(S)$ of a system S to be compromised is:*

$$\forall \mathcal{X}_j, \gamma = \gamma_S \bullet \gamma_i, \gamma_S \in S, \gamma_i \in \mathcal{X}_j, \tag{4.32}$$

$$S \xrightarrow{\gamma_S} S' \wedge \mathcal{X}_j \xrightarrow{\gamma_i} \mathcal{X}_j', S \| \mathcal{X}_j \xrightarrow{\gamma} S' \| \mathcal{X}_j' \Rightarrow \mathscr{P}_{sec}(S' \| \mathcal{X}_j') \ni goal_j,$$

$$Risk(S) = \sum_{j=1}^{N_{\mathcal{X}}} N_{\mathcal{X}_j} \cdot \sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j] \cdot dmg(\gamma_i, \mathcal{X}_j)$$

Where $\mathbf{Pr}_v[\gamma_i, \mathcal{X}_j]$ is the probability of successful execution of the attack γ_i by $\mathcal{X}_j$, $\mathbf{Pr}_t[\gamma_i, \mathcal{X}_j]$ is the probability of selection of attack γ_i by an attacker from the profile $\mathcal{X}_j$, $dmg(\gamma_i, \mathcal{X}_j)$ is the damage which $\mathcal{X}_j$ causes by successful execution of γ_i.

Note, that an attacker which is going to attack the system has to select one of the available attacks leading to achievement of her goal. Therefore, we have complete probability space here: $\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j] = 1$. On the contrary, we assume that the probability of successful execution of an attack does not depend on other attacks. Therefore, the complete probability space for the probability of successful execution of attack γ_i by $\mathcal{X}_j$ is $\mathbf{Pr}_v[\gamma_i, \mathcal{X}_j]$ and $1 - \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j]$.

If we know that a randomly taken attempt of attack is made by the profile $\mathcal{X}_j$ with probability $\mathbf{Pr}_{\mathcal{X}_j}$ we can find the number of attempts made by the profile if the overall amount of attempts is known.

$$N_{\mathcal{X}_j} = N_X^{ttl} \cdot \mathbf{Pr}_{\mathcal{X}_j} \tag{4.33}$$

Naturally, $\sum_{j=1}^{N_{\mathcal{X}}} \mathbf{Pr}_{\mathcal{X}_j} = 1$.

Proposition 1 *Definition 24 is a fine-grained form of the classical formula for computation of risk (annualized losses) [53, 74]:*

$$Risk = \sum_{j=1}^{N_{\mathcal{X}}} ARO_j \cdot SLE_j \tag{4.34}$$

Where ARO_j is the annual rate of successful achieving $goal_j$ by the attacker from profile $\mathcal{X}_j$ and SLE_j is single loss expectancy from achieving $goal_j$.

Proof ARO_j gives us the average number of successful attacks which realise $goal_j$. Let $\mathbf{Pr}^{scc}_{\mathcal{X}_j}$ be the probability that an attempt of attack on the system is successful and realises $goal_j$. Then, the number of successful attacks can be found if a number of attempts to compromise the system and probability $\mathbf{Pr}^{scc}_{\mathcal{X}_j}$ are known $ARO_j = N^{ttl}_X \cdot \mathbf{Pr}^{scc}_{\mathcal{X}_j}$. To complete an attack, an attacker has to select a goal, i.e., to be from a certain profile and then successfully achieve the goal. Expanding $\mathbf{Pr}^{scc}_{\mathcal{X}_j}$, ARO_j can be seen as $ARO_j = N^{ttl}_X \cdot Thr_j \cdot Vln_j$, where Thr_j is the probability that $goal_j$ is selected and Vln_j is the probability that $goal_j$ is *successfully achieved*.

The selection of $goal_j$ is equivalent to the probability that an attack attempt is made by the profile $\mathcal{X}_j$ therefore $Thr_j = \mathbf{Pr}_{\mathcal{X}_j}$. We can compute the average probability that a concrete goal will be successfully achieved if we know all attacks which lead to realisation of this goal.

$$Vln_j = \sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j] \tag{4.35}$$

SLE_j is the expected damage in case of $goal_j$ is achieved. In practice the expected damage is computed using the data collected from previous successful attacks. We compute SLE_j using the usual equation for the computation of expected values:

$$SLE_j = \frac{\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}^{scc}_{\gamma_i} \cdot dmg(\gamma_i, \mathcal{X}_j)}{\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}^{scc}_{\gamma_i}} \tag{4.36}$$

where $\mathbf{Pr}^{scc}_{\gamma_i}$ is the probability that the attack i is selected in the realization of the $goal_j$ and then successfully achieved. Thus, SLE_j is:

$$SLE_j = \frac{\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j] \cdot dmg(\gamma_i, \mathcal{X}_j)}{\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j]} \tag{4.37}$$

Now, if we multiply and divide at once the part of Equation 4.32 after the first sum by $\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j]$ and substitute $N_{\mathcal{X}_j}$ as shown in Equation 4.33:

$$Risk(S) = \sum_{j=1}^{N_{\mathcal{X}}} \left(N_X^{ttl} \cdot \mathbf{Pr}_{\mathcal{X}_j} \cdot \left(\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, X] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j] \right) \right. \tag{4.38}$$
$$\left. \cdot \frac{\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j] \cdot dmg(\gamma_i, \mathcal{X}_j)}{\sum_{i=1}^{N_{\Gamma_{S|\mathcal{X}_j}}} \mathbf{Pr}_v[\gamma_i, \mathcal{X}_j] \cdot \mathbf{Pr}_t[\gamma_i, \mathcal{X}_j]} \right)$$

Finally, using Equations 4.35 and 4.37 and recalling that $Thr_j = \mathbf{Pr}_{\mathcal{X}_j}$ we get:

$$Risk(S) = \sum_{j=1}^{N_{\mathcal{X}}} N_X^{ttl} \cdot Thr_j \cdot Vln_j \cdot SLE_j = \sum_{j=1}^{N_{\mathcal{X}}} ARO_j \cdot SLE_j \tag{4.39}$$

□

4.5 Discussion

We would like to discuss two important outcomes of the formal model.

4.5.1 Attacker Models and Metrics

The first outcome of the analysis, is that the security evaluation depends on the behavior of an attacker. Considering security strength, even not in a specific context, we should take into account possible behavior of an attacker. For example, we say that a castle with thicker walls is more secure than the one with thinner walls. In this case, we implicitly assume that an attacker is going to break the walls with cannons or catapults. Definition 4 already takes into account all possible ways which an attacker can follow to break the system. However, the definition does not consider *how* the attacker is going to select an attack to execute among several alternatives and does not say what kind of knowledge the attacker possesses. We consider two simple models of an attacker in order to show different behaviors of attackers.

Omniscient deterministic attacker is a "worst-case attacker". The attacker has a complete knowledge of the system: knows all possible attacks, costs she has to pay to execute each attack and also the probability that an attack will be successful. With all this knowledge the attacker will always select the "easiest" way (less costly or more probable). Thus, the existence of other attacks rather than the "easiest" one does not affect the overall security level because these attacks will never be used. In this case, such metrics as minimal cost of attack or the most probable attack are the most appropriate choice for the evaluation of security.

Although the omniscient deterministic attacker is popular in the literature she is not suitable for estimation of a real security level which security metrics do have to evaluate. If such attackers were possible all attempts of attacks on the same system would be the same. On the contrary, we see the diversity of attacks (see, for example, the experiment described in [77]).

Blind adaptive attacker is another extreme: the attacker does not know anything about the system. The attacker finds the first possible attack and tries to execute it because there is no knowledge of how easy the attack is. In other words, the attacker selects attacks randomly. With such an attacker in mind every attack will contribute to the overall security strength, but not only the "easiest" one. Therefore, metrics like minimal cost of attacks are not appropriate for estimation of security strength, because they do not register improvement of security strength caused by hardening of other attacks, except the "easiest" one.

Neither the first nor the second model are good for description of the behavior of attacker, since an attacker always has some knowledge about the system, but this knowledge is not complete. Therefore, new and more realistic models of an attacker are required. On the other hand, two extreme models illustrate that different conclusions can be derived with respect to the considered behavior of an attacker. Our initial model that captures different possible attackers behaviors is presented in Chapter 5.

4.5.2 Metrics and Stakeholders

Using only Definitions 7, 8 we cannot decide which metric is more suitable for the evaluation of security. Such decision cannot be done because the criterion we use is too coarse and, thus, we need other criteria to make a finer-grained analysis of metrics. Therefore, we have to accept that several metrics can be used. The selection of more appropriate metric can be done depending on who needs this metric:

- A security team or administrators are more interested in what has to be done to reduce amount of penetrations. Thus, a number of possible attacks is more useful for these stakeholders. Also attack surface can be useful too to see how assets can be better protected.

- Minimal cost of attack, minimal length of attacks and maximal probabilities of successful attack are more useful for the analysts studying attackers. After an analysis these metrics can be provided to security staff which can improve the system knowing the weakest places. In addition, these values are interesting for an attacker, who wants to attack a system in the most efficient way.

In Chapter 6, we propose a general method for evaluation of security of complex services using different security metrics.

Chapter 5

Modeling Adaptive Attacker's Behavior

As we have discussed in Chapter 4, the model of an attacker and her way to select the attacks significantly impact the values of security metrics during the evaluation. In this chapter, we strive for a refined attacker model which should allow a finer-grained evaluation of security. We introduce the attacker's view of a system, which is sometimes different from the real system. This view drives the actions of the attacker depending on the knowledge and resources the attacker possesses. Moreover, in our model an attacker may give up on her current attack and follow an alternative attack path. We use Markov Decision Process (MDP) to model the behavior of the attacker as the method of the selection of attack steps.

The chapter is organized as follows. Section 5.1 explains our concerns on uncertain knowledge of an attacker about a system. Section 5.2 focuses on models of attacker's behavior.

5.1 System and Attacker

We consider an attack graph $G = (S, A)$ that represents the ways to compromise the system [75, 148]. The set S of nodes $s_i \in S$ denotes a successfully exploited vulnerability and the set A of edges $a_{ij} \in A$ denotes an attempt of exploitation of vulnerability s_j after previously exploited vulnerability s_i. Thus, successful exploitation of vulnerabilities leads an attacker to new states with new privileges.

Similarly to Chapter 4 we group attackers into attacker profiles. Within a profile, attackers have the same goal and similar parameters such as money, skills, etc. Formally, the attacker profile is the tuple $\mathcal{X} = \{goal, \Gamma, skill, intang, tang\}$ where *goal* is the goal of the attacker, Γ is the set of attacks $\gamma \in \Gamma$ known by the attacker, *skill* defines how trained is the attacker, *intang* is an amount of intangible resources possessed by the attacker, e.g., time, *tang* is the amount of tangible resources possessed by the attacker.

We modify the attack graph to capture properties of the attacker. First, we add to the graph an initial node $s_{init} \in S$ corresponding to initial privileges of the attacker. Second, we define the subset of goal nodes $S_{end} \subset S$ that correspond to vulnerabilities that complete the attack (the ultimate step of each attack).

We assume that the attacker has certain amount of time units to perform the attacks. She spends a unit of time for executing a single attack step. The attacker stays in a goal state if she reaches it before spending all units of time. This situation is modeled by adding edges that start and end in the same goal state.

We distinguish between the *real system* and the *attacker belief* about the system. When the attacker is omniscient, her view of the system coincides with the real system. We consider a more realistic case, when the view does not coincide with the real systems. The attacker's knowledge about the system determines the set of vulnerabilities that the attacker believes present in the system. These believed vulnerabilities define a new graph G_B. This graph is similar to the attack graph for the real system while has believed vulnerabilities as nodes:

$$G_B = (S_B, A_B), S_B = S_{true} \cup S_{false}, A_B = A_{true} \cup A_{false} \tag{5.1}$$

Where $S_{true} \subseteq S$ and $A_{true} \subseteq A$ are the subset of vulnerabilities and the subset of attack steps really existing in the system and also believed by the attacker to exist, S_{false} and A_{false} are the set of vulnerabilities and the set of action that are believed to exist but are absent in reality.

The set of vulnerabilities that are believed by the attacker is further reduced according to attacker's skills and her tangible resources. Finally, the attacker has her own *view* (a graph $G_{\mathcal{X}}$) of the system:

$$G_{\mathcal{X}} = (S_{\mathcal{X}}, A_{\mathcal{X}}), S_{\mathcal{X}} \subseteq S_B, A_{\mathcal{X}} \subseteq A_B \tag{5.2}$$

We assume that the system behaves probabilistically. We introduce probability $\mathbf{Pr}_{ij}$ of system transition from the state s_i to the state s_j in response to an attacker's action. For the attacker this probability is:

$$\mathbf{Pr}_{ij} = \mathbf{Pr}^{p}_{ij} \cdot \mathbf{Pr}^{exp}_{ij} \tag{5.3}$$

where $\mathbf{Pr}^{p}_{ij}$ is the probability that the vulnerability s_j presents in the systems and $\mathbf{Pr}^{exp}_{ij}$ is the conditional probability that the vulnerability may be successfully exploited in case it exists in the system. The probability $\mathbf{Pr}_{ij}$ depends only on the successive state s_j while we use both indexes i and j for the uniformity with usual definition of transition probabilities.

We measure $\mathbf{Pr}^{p}_{ij}$ assuming that the attacker knows which software is installed in the system but may not know whether the software is patched or it is not. The probability of presence of the vulnerability in the system depends on the period passed after the vulnerability was discovered: the more time passed since the discovery the lower the probability of presence of the vulnerability [133]. We assume

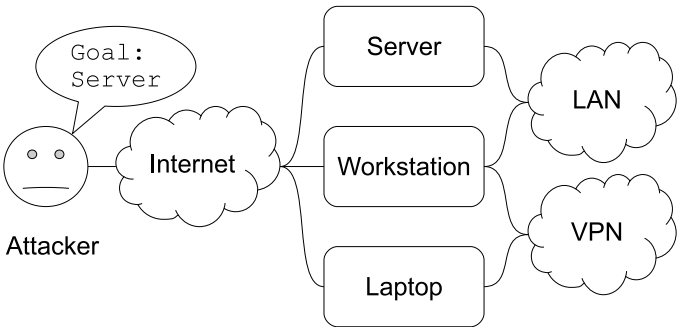

Figure 5.1: A network system

$\mathbf{Pr}_{ij}^{p}$ decreases linearly in time:

$$\begin{aligned} \mathbf{Pr}_{ij}^{p} &= -\tfrac{1}{T_{patch}} \cdot t + 1 \quad \text{if } T_{patch} \geq t \\ \mathbf{Pr}_{ij}^{p} &= 0 \qquad\qquad\quad \text{if } T_{patch} < t \end{aligned} \tag{5.4}$$

where T_{patch} is the time required for patching all systems, t is time passed since the release of a patch and for $t > T_{patch}$ we assume that all systems are patched. The probability $\mathbf{Pr}_{ij}^{exp}$ may be computed using the score from vulnerability databases similarly to [49] or by security experts. Our approach does not depend on the method of computation of $\mathbf{Pr}_{ij}^{p}$ and $\mathbf{Pr}_{ij}^{exp}$, thus, any other methods can be used.

Example 1 *We consider a company which saves information in an on-line database service. A competitor company would like to steal the information by attacking the server where the database is installed. The server operates FreeBSD 7 and MySQL 5. The database is managed by an administrator that uses a local workstation operated by Linux Mint 12 with Pidgin Messenger installed. Moreover, the administrator manages the database from her home laptop using a VPN connection to the workstation. The laptop runs Windows 7, Chrome browser, and TUKEVA Password Reminder. The whole system is depicted in Figure 5.1.*

The attacker composes the following attacks to the system[1]*:*

- *The shortest possible attack requires registration in the on-line database service and execution of vulnerability CVE-2012-0484 in MySQL.*
- *Another possible attack is based on vulnerability CVE-2011-3108 in Chrome browser and CVE-2009-4781 in TUKEVA where the administrator saves passwords to a database management tool.*
- *The attacker exploits CVE-2012-2369 in Pidgin gaining the access to the workstation. Then she causes a buffer overflow on the server using CVE-2011-4862 and exploits CVE-2012-0114 against MySQL.*

[1]Please, follow `http://nvd.nist.gov/home.cfm` for details of vulnerabilities.

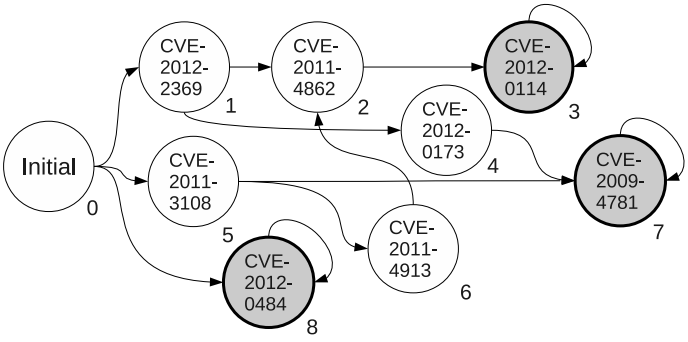

Figure 5.2: The attack graph of the network system

- *Since the laptop is connected by VPN to the workstation, the attacker gains the access to the laptop executing CVE-2012-0173 in Windows 7. Then she exploits CVE-2009-4781 in TUKEVA.*
- *The attacker may gain the access to the workstation after successful attack to the laptop by executing CVE-2011-4913 in the Linux kernel. Then she exploits CVE-2011-4862 in the FreeBSD server and CVE-2012-0114 in MySQL.*

The resulted attack graph is displayed in Figure 5.2. We enumerate the nodes for the sake of convenience. The node 0 is the initial node. The nodes 3, 7 and 8 (colored in grey) are goal nodes.

5.2 Models of Attacker's Behavior

We use Markov Decision Process (MDP) [137] to model decision making process of attackers. An attacker observes a system and can influence the behavior of the system by making actions at discrete moments of time (decision epochs). The system responds to an action probabilistically. The attacker does not make the decisions about actions blindly but takes into account past, current, and possible future states of the system and also possible rewards that are connected with the actions. The goal of the attacker is to maximize the expected total reward according to a some criterion.

Formally, MDP is a tuple $\mathcal{P} = \langle S, A, P, R, T \rangle$ where S is a set of system states s_i, A is a set of sets A_i of actions $a_{ij} \in A_i$ available for the attacker in the state s_i, P is a set probabilities $\mathbf{Pr}_{ij}$ that the system transits from state s_i to s_j in response to attacker's action a_{ij}, R is a set of rewards functions r_{ij} dependent on the state s_i and the action a_{ij}, T is a set of decision epochs t. Regarding transition probabilities, in general, the system may transit to any state available from s_i in response to the action a_{ij}. We assume that the system only transits to the state s_j with probability $\mathbf{Pr}_{ij}$ or stays in the state i with probability $1 - \mathbf{Pr}_{ij}$.

Algorithm 1 Computation of a deterministic policy

$t := N$ {number of decision epochs}
for all $s^N \in S$ **do**
  $u^N(s^N) := r^N(s^N)$
end for
while $t > 1$ **do**
  $t := t - 1$
  for all $s^t \in S$ **do**

$$u^t := \max_{a \in A_{s^t}} \left\{ r^t(s^t, a^t) + \sum_{a_{ij} \in A_i} \mathbf{Pr}_{ij}^t \cdot u^{t+1}(s_j) \right\}$$

$$A^*_{s^t,t} := \arg\max_{a \in A_{s^t}} \left\{ r^t(s^t, a^t) + \sum_{a_{ij} \in A_i} \mathbf{Pr}_{ij}^t \cdot u^{t+1}(s_j) \right\}$$

  end for
end while

We model attacker's behavior as an MDP policy π which determines how an attacker selects actions. We exploit finite-horizon policies to define attacker's behavior because the attacker has finite amount of time to perform an attack. The policy can be deterministic or probabilistic. The policy is deterministic if the next action of the attacker in the state is unambiguously defined by the history of actions and states. The policy is probabilistic if the attacker selects the next action at random. The probability of an action to be selected may also depends on the history. The policy is composed of decision rules. A decision rule is a procedure for the selection of an action for each moment of time. The decision rules either as the policy may be deterministic or probabilistic.

There is an optimal policy that maximizes an expected total reward obtained by the attacker during the attack. A total reward u_π obtained by the attacker as a result of the execution of policy π is computed on the basis of instant and terminal rewards. The attacker obtains an instant reward after execution of an action. The instant reward depends on s^t and a^t. The terminal reward $r^N(s^N)$ depends on the state of the process at the last decision epoch N. Thus, the total reward:

$$u_\pi = \sum_{t=1}^{N-1} r^t(s^t, a^t) + r^N(s^N) \tag{5.5}$$

Note, that we use upper index (e.g., s^t for a state) to denote the current value of a variable at a moment of time.

We evaluate every attacker's action as an amount of money. E.g., for an attacker that tries to cause maximal damage to a system, the reward are losses faced by the system owner in case of a successful attack.

Algorithm 2 Model of adaptive attacker

$\tau := N$ {number of decision epochs}
$t := 1$ {current decision epoch}
Run the backward induction algorithm using τ to obtain $A^*_{s^t,t}$
while $\tau \neq 0$ **do**
  if $a^t = a_{ij}$ is successful **then**
    for all $q, \mathbf{Pr}_{jq} \neq 0$ **do**
      $\mathbf{Pr}_{0q} := \mathbf{Pr}_{jq}$
    end for
    $\tau := \tau - 1$
    $t := t + 1$
  else
    if s_j exists **then**
      for all $k, \mathbf{Pr}_{kj} \neq 0$ **do**
        $\mathbf{Pr}_{kj} := \mathbf{Pr}^{exp}_{kj}$
      end for
    else
      for all $k, \mathbf{Pr}_{kj} \neq 0$ **do**
        $\mathbf{Pr}_{kj} := 0$
      end for
    end if
    $\tau := \tau - 1$
    $t := 1$
    Run the backward induction algorithm using τ to obtain $A^*_{s^t,t}$
  end if
end while

Omniscient Deterministic Attacker

The simplest model of attackers behavior [98, 148, 165] may be defined by an optimal deterministic policy of MDP. In this case, an attacker always prefers the best possible action in a state which belongs to the optimal attack path in the attack graph. The algorithm for the computation of optimal deterministic policies is the backward induction [137] (see Algorithm 1). Variables u^t and u^n describe intermediate values of total reward. The algorithm finds sets $A^*_{s^t,t}$ of actions that the attacker should follow in each state at each decision epoch. Sets $A^*_{s^t,t}$ maximize the expected total reward of the attacker.

Adaptive Attacker

We modify the behavior of the deterministic attacker so that she may reconsider her course of action when she cannot complete her current attack path (e.g., because the vulnerability initially thought to be in the system is really absent). We assume

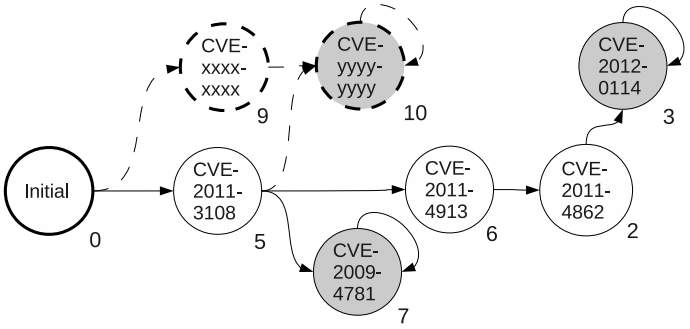

Figure 5.3: The view of the attacker at the first decision epoch

that the attacker sets $\mathbf{Pr}_{ij}^{p} = 0$ (and, thus, $\mathbf{Pr}_{ij} = 0$) when she cannot complete an attack step a_{ij} and understands that the vulnerability s_j is absent in the system. In addition, the attacker sets $\mathbf{Pr}_{kj} = 0$ for all other edges entering s_j from all states s_k. Then the attacker uses the backward induction algorithm to compute a new strategy using the updated attack graph and the amount of decision epochs τ left after the initial part of the attack.

The attacker sets $\mathbf{Pr}_{kj} = \mathbf{Pr}_{kj}^{exp}$ for all edges entering s_j from all states s_k if she cannot complete the action a_{ij} but understands that the vulnerability s_j exists in the system. Then the attack strategy is recomputed according to the backward induction algorithm with the rest τ of the decision epochs. If the attacker successfully exploits the vulnerability s_j she adds edges a_{0q} and sets $\mathbf{Pr}_{0q} = \mathbf{Pr}_{jq}$ for all states s_q reachable from s_j in one step. This modification is required to remember the privileges gained by the attacker for future adjustments in her strategy.

The modified behavior is described in Algorithm 2.

Example 2 *Suppose the attacker has the view of the system shown in Figure 5.3. The attacker supposes that nodes* 9 *and* 10 *are presented in the system, while they are actually not presented. Exploitation of nodes* 1 *and* 4 *(see Figure 5.2) is too expensive for the attacker. Moreover, the attacker does not have enough skills to exploit node* 8*.*

In our example, the attacker has $N = 5$ *decision epochs and gets terminal rewards (\$10K) only if she reaches states* 3, 7, 10 *i.e.* $r^N(s_3) = r^N(s_7) = r^N(s_10) = 10$ *other terminal rewards equal* 0*. Instant rewards also equal to* 0*. The first step is a computation of deterministic policies for the initial configuration of the system.*

For the attack graph presented in Figure 5.3, the policy is $\pi = (a^1 = a_{05})$ *at the initial state during the first decision epoch. The action is successful and the attacker sets the probabilities* $\mathbf{Pr}_{06} = \mathbf{Pr}_{56}$, $\mathbf{Pr}_{07} = \mathbf{Pr}_{57}$ *and* $\mathbf{Pr}_{0,10} = \mathbf{Pr}_{5,10}$*. Changes of probabilities are depicted as new edges of the attack graph (see Figure 5.4).*

The policy π *requires the attacker to select the action* $a^2 = a_{5,10}$ *at the second decision epoch. The attacker unsuccessfully tries the action understanding the vulnerability* 10 *is absent in the system. The attacker sets probabilities* $\mathbf{Pr}_{0,10} = 0$,

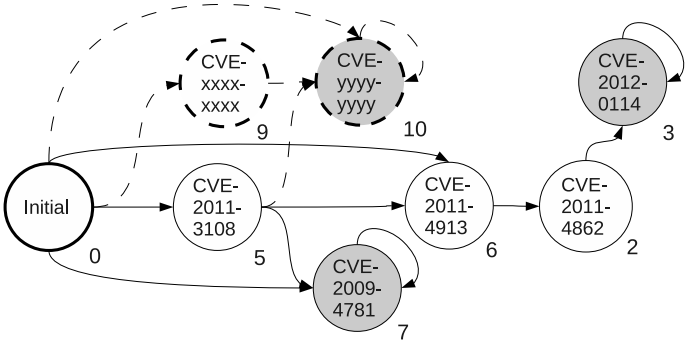

Figure 5.4: The view of the attacker at the second decision epoch

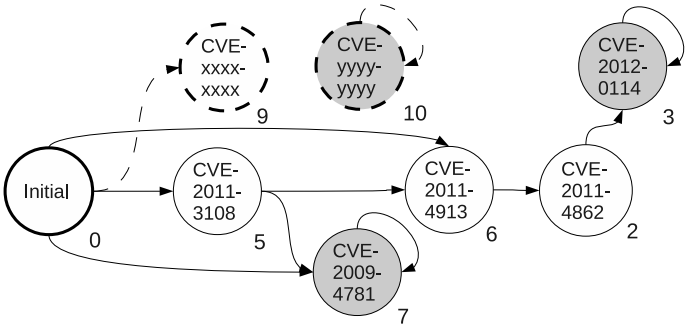

Figure 5.5: The view of the attacker at the third decision epoch

$\mathbf{Pr}_{5,10} = 0$ *and* $\mathbf{Pr}_{9,10} = 0$. *She reconsiders her initial policies using* $N - 2 = 3$ *decision epochs, and obtains a new initial deterministic policy* $\pi = (a^1 = a_{07})$.

Exploitation of Algorithm 2 allows running a simulation of interactions of an attacker and a system. We suppose that the security metrics should be estimated on the basis of several simulations (similarly to [98]). We suggest the following method for the evaluation of security metric. First, a security administrator should build an attack graph for a computer system she manages. Second, she needs to determine the parameters of the attacker that is supposed to attack the system. Finally, the administrator runs Algorithm 2 several times and obtains values of security metrics. We leave determination of exact formulas for the computation of other security metrics as a future work.

Chapter 6

Security Evaluation of Complex Services

W focus on the evaluation of security of complex services on the basis of different security metrics. The evaluation should help a service orchestrator to select the most secure design of a complex service. We assume that the complex service is composed of simple services evaluated with general quantitative security metrics. Essential peculiarity of our method is that we express security metrics as semirings which allow abstracting from the metric type during the evaluation. First, we consider a primitive decomposition of the complex service into a weighted graph which describes possible designs of the the complex service. Second, we evaluate the security using semiring-based methods for a graph analysis. Finally, we exploit semirings to describe mappings between security metrics which are useful when different metrics are used for the evaluation of security of different simple services.

The chapter is structured as follows. Section 6.1 presents a transformation of a complex service described in Business Process Modeling Notation (BPMN) into a graph. In Section 6.2, we evaluate overall security of a complex service analyzing the graph with help of semiring-based methods. Section 6.3 shows how mappings between security metrics can be described.

6.1 Decomposition of Complex Service into Design Graph

We consider a general complex service composed of simple abstract services. An abstract service describes a single job that should be done during the execution of the complex service. We follow BMPN notation for the description of a complex service. BPMN is a high-level notation and, thus, is suitable for the high-level evaluation of security. BPMN is a graphical notation, therefore an ad-hoc formalization is required for the further analysis.

We consider a complex service which is composed using the four basic structured

activities: sequence, choice, flow, and loop. *Sequence* describes a situation when services or structured activities are executed sequentially. *Choice* allows selecting a service on the basis of attributes of the complex service or events external to the complex service. *Flow* is used to denote two or more services or activities run in parallel. *Loop* supports the iterative execution of services and activities.

We extend this set with one more structured activity called *design choice* similarly to Massacci and Yautsiukhin [108, 109], which denotes a design alternative for a complex service. Design alternatives denote sub-processes which fulfill the same functional goal, but in different ways (i.e., these are different subprocesses). The alternatives provide different qualities in general, and security level in particular. Semantics of the design choice is similar to a regular choice, but the design choices are solved during the design of the complex service, while the regular choice is solved during the execution. We exploit a gateway with letter "X" inside to denote the regular choice and a gateway with letter "D" inside to denote the design choice in a complex service diagram.

Each abstract service has several real instantiations, *concrete services*. Concrete services are run by different service providers. For instance, an on-line trading platform may be provided by Amazon or eBay, an off-the-shelf e-mail solution may be provided by Google or Microsoft. We suppose that the security level of a concrete service is evaluated using general quantitative security metrics or risk. The security level is queried by an orchestrator from a service registry. An essential goal of the orchestrator is to solve all design choices and select instantiations for the abstract services in a way to obtain the most secure design of the complex service.

Example 3 *We consider an on-line shop as an example of a complex service (see Figure 6.1). First, a customer uses an on-line engine for searching and selecting items for buying. The owner of the shop would like to choose the way to implement the on-line engine. She considers two alternatives: to rent an on-line trading platform or to rent a server and install a content management system (CMS) there. Second, selected items are paid using a payment service. Third, items are shipped to the customer. Finally, the customer gets information about the payment and conditions of shipping by e-mail or VoIP service. The owner considers two opportunities to organise an e-mail service: to run an off-the-shelf e-mail solution or to organise her own e-mail server buying a hosting and installing an e-mail server software. We numerate abstract services for further convenience.*

6.1.1 Mathematical Model

We use a graph-based mathematical model of a complex service for description and analysis of complex services. A high level description of a complex service can be transformed into a graph in several ways (e.g., [108, 109]). In the current work, we simplify the transformation assuming that an orchestrator has information about usual execution of complex service in advance. Thus, all *choices* except design

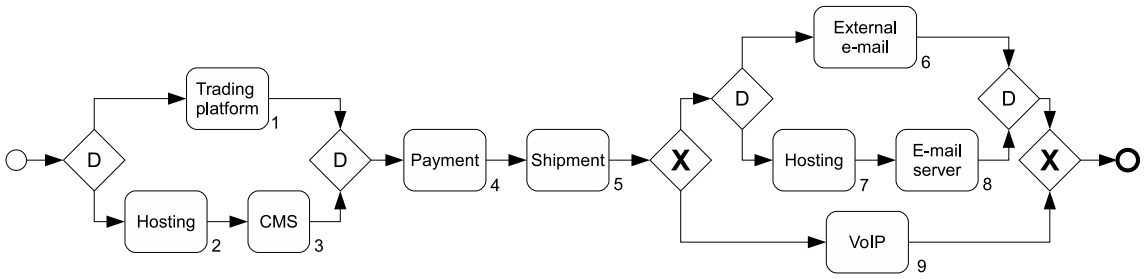

Figure 6.1: A complex service in the BPMN: an on-line shop

choices are known in advance and we can consider only a part of the initial complex service containing design choices only. *Loop* activity is considered as a number of the same executions in a raw. We assume that the orchestrator knows exact number of loops or uses the average number. The following technique is used to obtain the graph.

We call *Design Graph* a graph composed of concrete services connected with edges representing message flow in a complex service. The root node of the graph is an empty node representing the beginning of the complex service. For the sequential composition, the child of a node is the next executed service in a BPMN description. In case of parallel composition we select any activity first and then another one, hence, the parallel composition is a sequence of nodes in the graph. Intuition behind such transformation is that we consider the security of the complex service and all parallel branches should be successfully executed for the successful execution of the complex service. Regular choices are solved according to the assumption above. A node has several outgoing edges if corresponding service is followed by a design choice. We call such node an "or-node". Outgoing edges lead to nodes corresponding to the first services in design alternatives grouped by the design choice. In addition, "or-node" is used to represent a choice between concrete services. Finally, an empty node is used to conclude the graph. The direction of connections is the same as the direction of message flow in the BPMN diagram. Moreover, each node is assigned with a weight according to the value of a metric expressing service security. Source node and final nodes have zero weights. Now, we are able to formalize the Design Graph we receive after transformation of a complex service description.

Definition 25 *Let $S^A := \{a_i\}$ be a set of abstract services. Let also $S^C := \{c_{ij}\}$ be a set of concrete services and any $c_{ij} \in S^C$ is a j-th concrete service for an abstract service a_i. Then, we define the Design Graph as a tuple $\langle N, E, \mathscr{L} \rangle$. Where:*

- *$N := \{n_{ij}\} \cup \{n_0\} \cup \{n_\infty\}$ is a set of nodes, where nodes n_{ij} correspond to the concrete services c_{ij}, n_0 and n_∞ are initial and final nodes corresponding to the start and the end of the complex service;*

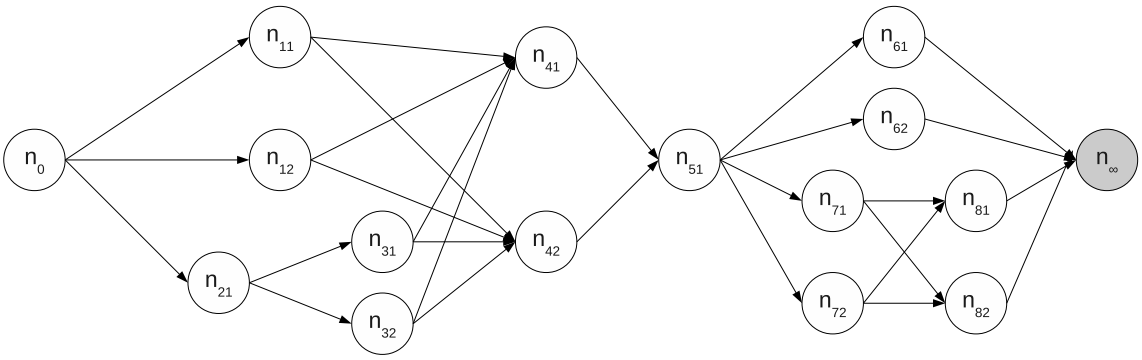

Figure 6.2: The design graph representing the on-line shop

- *E is a set of edges between nodes which correspond to the message flow in the complex service;*
- *$\mathscr{L} : N \to D$ is a labeling function which assigns to every node a number from the domain D of a security metric, the initial node and the final node are always assigned with zero value of the metric.*

Example 4 *We continue Example 3. Consider transformation from a complex service in Figure 6.1 into a Design Graph. Suppose the owner of the on-line shop knows that most of her customers prefer to be contacted via e-mail. This information helps an orchestrator of the complex service to remove the exclusive choice between a VoIP service and an e-mail service on the final step of the complex service.*

The design graph starts with the initial node n_0 which has three children n_{11}, n_{12}, and n_{21}. Nodes n_{11} and n_{12} describes the selection between concrete services instantiating a trading platform in Figure 6.1 (e.g., n_{11} is for Amazon and n_{12} is for eBay). The alternative design of the on-line engine is presented by node n_{21} which stands for a hosting service and two its children n_{31} and n_{32} denoting different CMSs. Nodes n_{41} and n_{42} represent payment services, n_{51} stands for shipping service, n_{61} and n_{62} denote external mailing services, n_{71} and n_{72} represent hosting for an e-mail server, n_{81} and n_{82} are the e-mail server software. The graph ends with the node n_∞ which stands for the end of the complex service. We display the resulted Design Graph produced from the complex service in Figure 6.2.

Every design of the complex service is represented as a path in the Design Graph. We define the set $P_{(n_0,n_\infty)} := \{\pi_{(n_0,n_\infty)}\}$ of all the possible paths π between n_0 and n_∞. Each path has its own weight obtained by aggregating weights of nodes belonging to the path. The weight of the path is representing the security metric for an design of a complex service. Aggregating weights corresponds to aggregating metrics values. The problem of the selection of the most suitable design of the complex service can be seen as *to find such path in a Design Graph that the weight*

of the path is the best one (e.g., maximal or minimal) among all possible. We call the path with optimal value of metric the *shortest path* and denote it as $\pi^{sh}_{(n_0,n_\infty)} \in P_{(n_0,n_\infty)}$. The most secure design of the complex service corresponds to the shortest path in the Design Graph and has the best value of the security metric.

6.2 Security-aware Selection of Complex Service Design

We analyze the Design Graph using semiring-based methods in order to select the design of a complex service which satisfies the desirable requirements of the service consumer. We select the most secure complex service design among available alternatives. If this selection does not satisfies the desirable customer's policies then no other design does.

We aim at the evaluation of the security of a complex service using different security metrics. However, in this section, we assume that the security of all concrete services is evaluated using the same security metric. This assumption will be relaxed in Section 6.3. Each node n_{ij} in the Design Graph is assigned with weight $w_{ij} = L(n_{ij})$. The initial n_0 and the final node n_∞ are assigned with a zero value. We look for a method that allows abstracting from the security metrics and using universal algorithms for the computation of the shortest path in graphs.

Mehryar Mohri [117] proposed a framework that contains algorithms for searching for the shortest path in a weighted graph, extending the work of Edsger Dijkstra [40]. The framework exploits the notion of *semiring* for the abstraction of weights and operators over weights. A semiring consists of the set of values D (e.g., natural or real numbers), and two types of operators: aggregation ($\otimes$) and comparison ($\oplus$) of values. Formally, the semiring is defined as follows [22]:

Definition 26 Semiring T *is a tuple* $\langle D, \oplus, \otimes, \mathbf{0}, \mathbf{1}\rangle$*:*

- D *is a set of elements and* $\mathbf{0}, \mathbf{1} \in D$*;*
- $\oplus$ *is an additive operator defined over (possibly infinite) set of elements* D*, for* $d_1, d_2, d_3 \in D$*, it is commutative* $(d_1 \oplus d_2 = d_2 \oplus d_1)$ *and associative* $(d_1 \oplus (d_2 \oplus d_3) = (d_1 \oplus d_2) \oplus d_3)$*, and* $\mathbf{0}$ *is a* unit *element of the additive operator* $(d_1 \oplus \mathbf{0} = d_1 = \mathbf{0} \oplus d_1)$*.*
- $\otimes$ *is a binary multiplicative operator, it is associative and commutative,* $\mathbf{1}$ *is its* unit *element* $(d_1 \otimes \mathbf{1} = d_1 = \mathbf{1} \otimes d_1)$*, and* $\mathbf{0}$ *is its* absorbing *element* $(d_1 \otimes \mathbf{0} = \mathbf{0} = \mathbf{0} \otimes d_1)$*;*
- $\otimes$ *is distributive over additive operator* $(d_1 \otimes (d_2 \oplus d_3) = (d_1 \otimes d_2) \oplus (d_1 \otimes d_3))$*;*

- $\leq_T$ *is a partial order over the set* D*, which enables comparing different elements of the semiring, the partial order is defined using the additive operator* $d_1 \leq_T d_2$ *(*d_2 *is better than* d_1*) iff* $d_1 \oplus d_2 = d_2$*.*

The weight $\delta^{sh}(\pi^{sh}_{(n_0,n_\infty)})$ of the shortest path $\pi^{sh}_{(n_0,n_\infty)}$ is computed using additive operator $\oplus$:

$$\delta^{sh}(\pi^{sh}_{(n_0,n_\infty)}) = \bigoplus_{\forall \pi_{(n_0,n_\infty)} \in P_{(n_0,n_\infty)}} \delta(\pi_{(n_0,n_\infty)}) \tag{6.1}$$

Where $P_{(n_0,n_\infty)}$ is the set of all paths from the initial node n_0 to the final one n_∞. The weight $\delta(\pi_{(n_0,n_\infty)})$ of the path $\pi_{(n_0,n_\infty)}$ is computed using multiplicative operator:

$$\delta(\pi_{(n_0,n_\infty)}) = \bigotimes_{\forall n_{ij} \in \pi_{(n_0,n_\infty)}} w_{ij} \tag{6.2}$$

Where we write $n_{ij} \in \pi_{(n_0,n_\infty)}$ to show that the node n_{ij} belongs to the path $\pi_{(n_0,n_\infty)}$.

We need to express security metrics as *semirings* for exploitation of universal algorithms for the search of the shortest path in a weighted graph.

6.2.1 Semirings for Expressing Security Metrics

We assume that security of the complex service and concrete services is evaluated using general security metrics or risk. Different semirings should be used to express different metrics. We describe several security metrics expressed as semirings.

- Weighted semiring $\langle R^+, min, +, \infty, \mathbf{0}\rangle$ represents the *risk* that a successful attack on a complex service occurs. A path in a tree computed under preferences that weighted semiring minimizes the overall sum of risks of successful attacks on simple services constituting the complex service. We assume that the complex service is compromised if a successful attack compromises at least one service included in the complex service.

- Semiring $\langle N^+, min, +, \infty, \mathbf{0}\rangle$ serves for identification of a path with the minimal *number of attacks.*

- Probability semiring $\langle [0,1], max, \cdot, \mathbf{0}, \mathbf{1}\rangle$ expresses the *probability of a successful operation* of the complex service (a resistance to all attack). In case we know the probability p_{ij} of compromising the service c_{ij}, then $(1 - p_{ij})$ is the probability that the service c_{ij} tolerates all attacks.

This is not a complete list of metrics and semirings that can be used for the search of the most secure design of a complex service. Other semirings can be defined for other metrics. Note, that semirings can serve also for the evaluation of non-security aspects of the complex service. For instance, semiring $\langle N^+, min, +, \infty, \mathbf{0}\rangle$ can be

used for identification of the minimal number of steps to reach the end goal of the complex service. Semiring $\langle R^+, max, min, \mathbf{0}, \infty \rangle$ allows evaluating the *latency* of the complex service if we assume that only one delay may occur during the execution of the complex service. Probability semiring $\langle [0,1], max, \cdot, \mathbf{0}, \mathbf{1} \rangle$ can be used to express users *trust* to the complex service.

We can apply a semiring-based Generic Single Source Shortest Distance algorithm [117] for searching of the shortest path after a semiring was chosen and the problem is defined by Equations 6.1 and 6.2. Note, that the algorithm uses the weights on the edges while we use the weights on the nodes. The algorithm can still be applied to Design Graph if we use the weights for the node as the weight of every incoming edge leading to this node.

Example 5 *Suppose each concrete service is evaluated with the quantitative risk value. Weighted semiring* $\langle R^+, min, +, \infty, \mathbf{0} \rangle$ *is used to represent the risk. There are 48 possible paths in the graph presented in Figure 6.2. For simplicity, we consider just two paths. Let weights of nodes be* $w_{11} = 100$, $w_{41} = 120$, $w_{51} = 150$,$w_{61} = 90$, $w_{62} = 110$, $w_0 = w_\infty = 0$. *First, we find the weights for paths* $\pi^1_{(n_0,n_\infty)} = n_0 n_{11} n_{41} n_{51} n_{61} n_\infty$ *and* $\pi^2_{(n_0,n_\infty)} = n_0 n_{11} n_{41} n_{51} n_{62} n_\infty$. *The weights* $\delta^1(\pi^1_{(n_0,n_\infty)}) = 480$ *and* $\delta^2(\pi^2_{(n_0,n_\infty)}) = 500$ *are computed using multiplicative operator* $+$ *of weighted semiring. Second, the best weight is selected using additive operator min:* $\delta^{sh} = min\{\delta^1, \delta^2\} = 480$. *The shortest path is* $\pi^{sh}_{(n_0,n_\infty)} = \pi^1_{(n_0,n_\infty)}$. *Note, that here we used a simplified computation for this example, when the mentioned algorithms (e.g., [117]) are much more efficient.*

The main advantage of exploitation of semirings is that the orchestrator can evaluate the complex service using different security criteria and select several designs corresponding to different security metrics. The orchestrator can exploit an design satisfying the major part of criteria.

6.3 Interoperability of Services

Our previous idea requires concrete services being evaluated using the same metric. However in the real world a situation when security of all concrete services is evaluated using the same metric is not always possible. For instance, consider a situation when the security of the first part of services is evaluated using minimal number of attacks and the security of the second part is evaluated using risk. There is a need for a method that can evaluate the security of a complex service in case of several different metrics are used for the evaluation of security of different simple services. We propose to tackle the issue by mapping between security metrics. The relations may be expressed using mappings between semirings presented by Bistarelli et al. [20]. The analysis described in Sections 6.1 and 6.2 should be applied after the mapping is done.

Suppose there are two semirings $T = \langle D, +, \cdot, 0, 1\rangle$ and $\widehat{T} = \langle \widehat{D}, \widehat{+}, \widehat{\cdot}, \widehat{0}, \widehat{1}\rangle$. Our goal is mappings between these semirings. A Galois insertion $\langle \alpha, \gamma\rangle : \langle D, \leq_T\rangle \rightleftharpoons \langle \widehat{D}, \leq_{\widehat{T}}\rangle$ is used for the mappings. Here α and γ are two mappings such that:

- α and γ are monotonic,
- $\forall d \in D, d \leq_T \gamma(\alpha(d))$,
- $\forall \widehat{d} \in \widehat{D}, \alpha(\gamma(\widehat{d})) \leq_{\widehat{T}} \widehat{d}$.

For a constraint satisfaction problem (CSPs, [22]) H over semiring T we get a problem $\widehat{H} = \alpha(H)$ over semiring $\widehat{T}$ applying α. Similarly, we obtain the problem $H' = \gamma(\widehat{H})$ over semiring T applying the mapping γ to the problem $\widehat{H}$ over semiring $\widehat{T}$. The mapping has several useful applications. First, the mapping allows evaluating bounds for the solution of H if the solution of the problem $\alpha(H)$ is known. If there is the problem H over T, and $\widehat{h}$ is an optimal solution of problem $\alpha(H)$ with semiring value $\widehat{d}$ in $\alpha(H)$ and d in H, then there is an optimal solution h of H with semiring value $\overline{d}$ such that $d \leq \overline{d} \leq \gamma(\widehat{d})$. Second, a mapping is called *order preserving* if $\widehat{\bigotimes\limits_{d\in I_1}} \alpha(d) \leq_{\widehat{T}} \widehat{\bigotimes\limits_{d\in I_2}} \alpha(d) \Rightarrow \bigotimes\limits_{d\in I_1} d \leq_T \bigotimes\limits_{d\in I_2} d$, where I_1 and I_2 are two sets of elements from D. If the mapping is order preserving then the set of all optimal solutions of the problem H over T is the subset of optimal solutions of the problem $\alpha(H)$ over $\widehat{T}$.

A problem of searching the shortest path in a graph is a CSPs problem [117]. Thus, we are able to find bounds for a weight of the shortest path in a Design Graph if we do mapping between metrics using semirings. The bounds may be used as an approximated level of security of complex service. The bounds also may be used as a starting point for searching a precise value. If the mapping is order preserving, the set of shortest paths in the graph after the mapping contains all shortest paths for the graph before the mapping. Thus, we can also use this set as a starting point for searching a precise level of security for the complex service.

Chapter 7

Continuous Reevaluation of Security in Services

We consider service-oriented architecture (SOA) and propose a method for continuous reevaluation of security in services based on the Usage Control model (UCON) enhanced with qualitative risk assessment. For the service consumer side, we show how to use risk for selecting a service and how to continuously reevaluate risk during the service exploitation in order to make decisions about current service. For the service provider side, we show how to improve the service in order to mitigate risk and to keep or to attract the consumers.

The chapter is organized as follows. Section 7.1 summarizes interactions of a service consumer and a service provider and shows how to adapt the UCON for describing the interactions. In Section 7.2, we present our idea about computing the risk for a service consumer. We use the proposed method for selecting the most secure service and for continuous reevaluation of security during the usage of the service in Section 7.3. We propose a method for a service provider that allows selecting the best improvements for her security preferences in Section 7.4.

7.1 Interactions of Service Consumer and Service Provider

We consider the SOA (see Figure 7.1) where asset (e.g., data) belonging to a service consumer is processed by a service belonging to a service provider. During the processing the data can be compromised. The service consumer wants to be sure that her data is processed in the most secure (i.e., the least risky) way while it is under provider's control. The security should be high when access to the data is granted and should remain high during the data processing and, maybe, some time after. Therefore, the service consumer considers various service providers and decides which one provides the service with the best security. More important, the security should be constantly reevaluated by the service consumer after the access

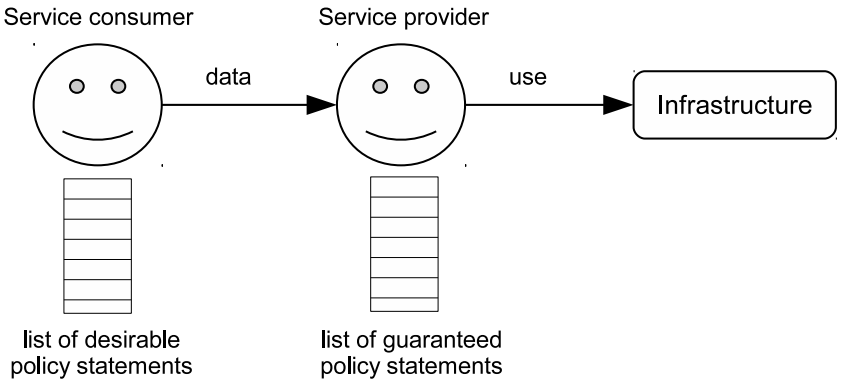

Figure 7.1: Interactions of a service consumer and a service provider in the SOA

to the data is granted. Hence, the usage of data must be controlled.

We assume that security requirements of a service consumer and security preferences of a a service provider can be expressed as attribute-based policies of the UCON. The attributes of the service consumer are collected into desirable policy statements that represent security requirements of the service consumer. The attributes of the service provider are collected into guaranteed policy statements that represent security properties of the service provider. Also environment itself has attributes (e.g., time, location, etc.) which may affect the decision to grant access to data or not to grant [5, 129].

Example 6 *For the policy statement "data must be deleted after 30 days", an attribute is "time after request about deletion", the value of the attribute is "30 days".*

Technically, we follow the SOA model presented in [138]. We assume that a service consumer can only observe policy statements of a service provider asking a SOA registry about security preferences of the service provider. The policy statements are certified by a certifier that confirms that security preferences announced by the service provider correspond to the reality. Moreover, we assume that the registry is notified about any changes of security preferences. For the sake of simplicity, we skip the intermediate interactions of the service consumer and the service provider with the registry and the certifier in further discussion.

We suppose that the service consumer should derive the security level of the service on her own analyzing certified security preferences. In Section 7.2 we discuss how the qualitative risk level may be assessed on the basis of security requirements and security preferences of the service consumer and the service provider.

We adapt the UCON model for security evaluation and reevaluation in the SOA (Figure 7.2). We suppose that service providers (subjects) are trying to access an object which is data of a service consumer. First, the service consumer starts

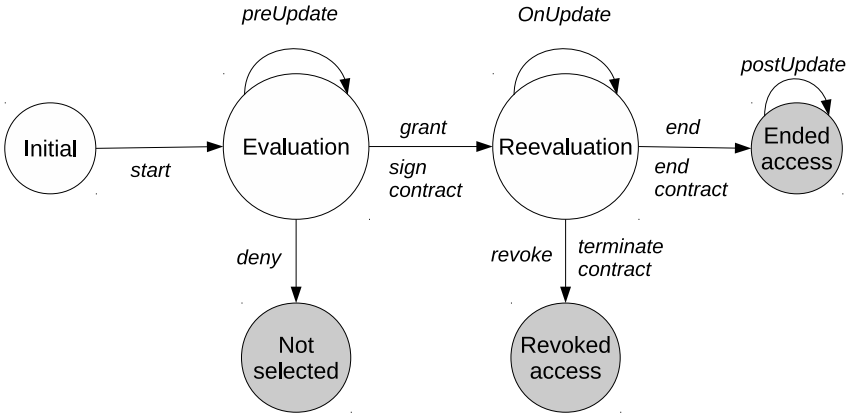

Figure 7.2: UCON model adapted for the SOA

selecting a service provider evaluating security of services. This step corresponds to *preUpdate* step of the UCON when initial attributes values are checked. If the service with appropriate security is found, the contract between the partners is signed and the exploitation of the service starts. This corresponds to *granting* the access to an object in the UCON. Otherwise, the service consumer *denies* the access to service providers because of inappropriate security level. The service provider gets the access to data and starts to process the data. During the usage of the data security is continuously reevaluated. The reevaluation corresponds to continuous checks of the attributes, hence, to *onUpdate* action of the UCON. If in result of reevaluation the service consumer understands that the security does not correspond to the required level, she terminates the contract, i.e., *revokes* the access to data. The access simply *ends* if the data is successfully processed by the service provider. Some *postUpdate* actions may be required after the access is ended, for instance, the service provider should delete the data from her servers.

7.2 Qualitative Risk Assessment for SOA

Every service consumer wants to minimize the risk related to possible abuse of her data during the processing. A service consumer has her requirements how the data should be used to minimize the risk of data to be revealed to the third parties, maliciously modified, become unavailable. For this purpose the service consumer specifies a list of desirable policy statements. Each policy statement specifies how the object must be used.

Policy statements have different importance for a service consumer. Violation of some policy statements may not harm the service consumer significantly. Other policies may be crucial. Moreover, the service consumer should consider which policies may be more likely violated during the processing while she determines the

importance of the policies. Only the service consumer has the necessary knowledge to determine the importance which is based on the nature of data and policies. We assume that the service consumer uses qualitative scale to rate the importance of policy statements assigning *high*, *medium* or *low* ranks of importance to each policy statement.

Definition 27 *Let P be a set of all desirable policies then we can define a function rank as follows:*

$$rank : P \rightarrow \{low, medium, high\} \tag{7.1}$$

Example 7 *Suppose that a service consumer wants two policy statements to be enforced: "the data must be deleted after 30 days" and "the audit must be performed twice a month". The first policy is not very important for the service consumer and has the level of importance "low". On the other hand, it is important for the service consumer that the service provider correctly processes the data and does not abuse it, e.g., for committing a fraud. Thus, the second policy statement has "high" rank of importance.*

A service consumer starts looking for a service provider which provides the required functionality and also satisfies the specified desirable policy statements. Obviously, not all statements can be addressed by every service provider in general case. The service consumer wants to grant the access to data to the service provider which guarantees the best security, i.e., the service provider which satisfies the desirable policy statements in the best way.

Policies may be expressed differently while have the same meaning. Thus, first of all, the service consumer should find the correspondence between her desirable policies and guaranteed policies of a service provider. We do not consider the problem of semantical mapping of policy statements here and refer the reader to the solutions already proposed in the literature, e.g., [26].

The next step is to consider if the desirable policies are addressed well enough by guaranteed policies. Every policy statement is based on some attributes [129]. This means that strength of statements depends on the values of the attributes. These attributes values may not be the same as the service consumer requires, but be slightly different. This means that the statement is not fulfilled entirely or sometimes fulfilled better than it is required.

Example 8 *The service consumer requires that "data must be deleted after 30 days" but the service provider may guarantee only that "data will be deleted after 90 days" because the local law requires to keep the data for this period.*

One solution is to search for a provider which can guarantee the policy statements as the service consumer requires, but this solution is too idealistic in the most real-world situations. We propose an alternative solution: accept a service provider

which has the "closest" attributes values to the desired ones. The service consumer should specifies the strength of policy statements depending on the attributes values. We define a function *str* which is specific for each policy statement and determines the strength of a statement depending on the values of the corresponding attributes.

Definition 28 *Let R be the set of all attributes used in policies of set P. Each attribute $r \in R$ has domain of values V_r. We define the set of tuples containing policy statements together with attribute values used in the statements:*

$$S = \{\langle p_i, v_{r_{i,1}}, \ldots, v_{r_{i,n}}\rangle : r_{i,1}, \ldots, r_{i,n} \in p_i, p_i \in P\} \tag{7.2}$$

Where p_i is a policy, $r_{i,1}, \ldots, r_{i,n} \in p_i$ means that attributes $r_{i,1}, \ldots, r_{i,n}$ are used in the policy p_i and $v_{r_{i,1}} \in V_{r_{i,1}}, \ldots, v_{r_{i,n}} \in V_{r_{i,n}}$ are values of attributes $r_{i,1}, \ldots, r_{i,n}$.

Then by the function str we mean the following mapping:

$$str : S \to \{perfect, high, medium, low, unacceptable\} \tag{7.3}$$

Together with usual levels (*high*, *medium* and *low*) we use two extremes: *perfect*, which means that a requirement is considered very well protected (almost perfect), and *unacceptable*, the strength level which the service consumer cannot accept. We admit that the first level can be rarely used because even the most robust requirement cannot give 100% protection. Nevertheless, we can assume that some attributes vales can make a policy statement almost perfect (e.g., encryption of data with a strong encryption algorithm has a very small probability to be broken). Unacceptable level is more useful and determines the border line that the service consumer does not wish to cross in any circumstances.

Example 9 *Continuing Example 8 for the policy statement "data must be deleted after N days" the service consumer specifies that in case $N \leq 30$ the strength of the policy is high and if $N \geq 60$ the strength is low. Actual deletion of data later than 180 days after the request about deletion is unacceptable. Thus, in our example (with $N = 90$) the strength is "low".*

We assume that every service provider has a huge list of guaranteed policy statements and every service consumer can find all her desirable policies in this list. Alternatively, a service consumer can assign the lowest strength level (depending on the defined *str* function) to the policy statements which were not found in the guaranteed list. Note, that it is not required to define the function *str* for all possible results. Especially, perfect and unacceptable levels can be often omitted.

We summarize the preparation procedures described above. First, ranks are assigned to desirable policy statements. Then, the desirable policy statements are mapped with the policy statements guaranteed by a service provider. Finally, the strength of policies is computed.

Now we are ready to determine how well a concrete service provider satisfies the requirements of the service consumer. For this purpose we would like to exploit risk

***rank*\\$\overline{str}$**	**no risk**	**low**	**medium**	**high**	**unacceptable**
low	no risk	low	low	low	unacceptable
medium	no risk	low	medium	medium	unacceptable
high	no risk	low	medium	high	unacceptable

Table 7.1: Qualitative calculation of risks

of each requirement to fail to fulfill its purpose. We use a simple approach for the assessment of risk using qualitative values but the method may be performed in a quantitative way, though the quantitative way is much harder [35, 153].

Risk can be seen as a combination of three components: *Impact*, *Threat*, and *Vulnerability* [6, 69, 155]. Frequently risk assessment methods find probabilities (ARO) of security violation as a combination of *Threat* and *Vulnerability* components, and the damage (SLE) caused by the violation as *Impact* component. Then, mathematically risk ($Risk$) is [53, 74]:

$$Risk = ARO \cdot SLE \tag{7.4}$$

Both probability and damage usually evaluated on the basis of statistics. We have to make a decision before the interaction between a service consumer and a service provider starts and thus we cannot use probability like it is done in classical risk assessment methods. On the other hand, we have $rank$ which represents combination of *Impact* and *Threat* components. The exposure to the violation of a policy (*Vulnerability*) can be seen as a reverse value of the strength str. The logic here is that the more robust a policy is the less chance is that it will fail to fulfill its purpose. The calculation of the reverse value is: $perfect \rightarrow no\ risk$, $high \rightarrow low$, $medium \rightarrow medium$, $low \rightarrow high$, $unacceptable \rightarrow unacceptable$ and vice versa[1].

Example 10 *The service consumer has rated the strength of the requirement "data must be deleted after 90 days" guaranteed by a possible service provider as having 'low' strength. This means that there is a 'high' exposure for the data to be abused in this period.*

For each requirement, it is possible to compute the risk using a table similar to the risk-level matrix [155] extended with the two extremes (see Table 7.1). In Table 7.1 rows are the rank levels and columns are exposure levels. Risk is the value of the cell situated on the intersection of given rank and exposure levels.

Definition 29 *We define Risk function as follows:*

$$\begin{aligned} Risk : &\{unacceptable, high, medium, low, perfect\} \\ &\times \{high, medium, low\} \\ &\rightarrow \{unacceptable, high, medium, low, no\ risk\} \end{aligned} \tag{7.5}$$

[1]We renamed level "perfect" to "no risk".

In other words, if we would like to compute the risk for a policy $p_i \in P$ which requires attributes $r_{i,1}, ..., r_{i,n} \in R$ the function will be:

$$Risk(p_i) = \overline{str}(p_i, v_{r_{i,1}}, \ldots, v_{r_{i,n}}) \cdot rank(p_i) \tag{7.6}$$

Where $\overline{str}$ is a reverse operation applied to the result of the function str, i.e., an exposure level, and $\cdot$ denotes multiplication of qualitative values according to Table 7.1.

The overall result of such calculation is a list of policy statements with assigned qualitative risks.

7.3 Reevaluation of Security for Service Consumer

We consider how a service consumer can exploit risk to select a service provider and then continuously reevaluate risk and make decisions about the interactions with the service.

7.3.1 Risk-based Access Control

We return to the comparison of different services in order to select the one which satisfies policy statements in the best way. We employ the idea presented in [160] for comparison of multi-objective qualitative values. First, all providers which have at least one "unacceptable" risk are eliminated from the further consideration. Second, the numbers of high, medium, and low risks are summed up separately for each service provider. In other words, a service consumer can assign the following tuple to each suitable service provider.

Definition 30 *Risk for a service provider sp is a tuple: $R_{sp} = \langle h, m, l \rangle$, where h is a number of risks with high value, m – with medium values, l – with low values. We define the functions $HIGH, MEDIUM$ and LOW: $RD_P \to \mathbb{N}$ where RD_P is the set of risks computed for the set of policies P and $\mathbb{N}$ is the domain of natural numbers. These functions simply return a number of high, medium and low risks taking a set of risks as an argument.*

The service consumer selects only service providers with the lowest number of high risk requirements. Among these service providers only those who have less medium risks are taken. Finally, those who have less low risks are determined. These service providers satisfy desirable policy statements in the best way and the service consumer should select one of these providers to process the data. Another approach for the service consumer can be to select a threshold for the risk and to consider all service providers with the risk below the threshold.

Example 11 *Assume that we found three service providers which got the following risks according to our computations:* $\langle 3, 5, 7\rangle$, $\langle 4, 4, 8\rangle$, $\langle 3, 6, 6\rangle$. *After the first check only the first and the third service providers are left. The second check indicates that the first service provider has less number of medium risks. Thus, the first service provider should be selected.*

7.3.2 Risk-based Usage Control

In the Section 7.3.1, we considered granting the access to a service provider which guarantees the best security. Since relations between the partners may last for a long period (days, months) the attributes values may change. The UCON is based on the idea that the usage of data must be controlled also after the access is granted. In this section we consider how a risk-based decision about further access can be made during usage of data.

Example 12 *After some time of interaction the service provider notifies the service consumer that from now on her data will be processed also by a specialized subsidiary situated in EU. Though this change violates one of the previously negotiated policy, i.e., "data must be processed only by the service provider", this change is not considered by the service consumer as a serious issue.*

Example 13 *Monitoring has shown that some data was removed 90 days after deletion request as this was agreed, but some data was removed only after 120 days after deletion request. The delay was detected by the monitoring mechanisms and mean that the service provider violates the negotiated policy: "completely remove data after 90 days from a delete request". The service consumer decides to take the maximal time to complete removal as a real value of the attribute and this adjustment changes the risk of the service provider.*

Our idea is not to jump to a quick decision and revoke access to the data because one attribute became less strong than it has been agreed. We propose to look at the overall risk and only then make a decision. The changed attributes values should be used for reevaluation of risk. If the risk is still acceptable, the access is continued. If the risk is higher than some previously predefined threshold, we consider two possible decisions for the service consumer. First, the service consumer can ask the service provider to adjust the security preferences. Second, if it is not possible to adjust the preference, the access to the data is revoked and the service consumer can select a service with better security.

7.4 Reevaluation of Security for Service Provider

A service provider may would like to improve her security in order to keep or to attract consumers. In other words, the service provider may install new security

controls, enforce more secure data management practices, change old encryption algorithms, etc. All these improvements mitigate risks and thus make the service more attractive to service consumers.

In a simple scenario, a service provider may notice that risk according to requirements of a service consumer is too high. If the service provider does not want to lose the consumer she can improve her system in order to satisfy the service consumer's requirements. We assume that the service provider has to know the strength functions and threshold risk level used by the service consumer.

Example 14 *Assume that after some time of operation the attribute of the requirement "delete data after 90 days" has changed because of the latest local law amendment. Now the data can be deleted only after 120 days. This fact raises the risk the service consumer thinks of changing the service provider. In order to decrease the high risk level the service provider decides to improve other attributes. The service provider decides to perform an external audit twice a month, but not just once as it was before.*

Usually, the service provider has limited budget for security improvements. Thus, the service provider aims at selecting security controls which improve her security in the best possible way. First, we should find the security controls which reduce the number of high risks. Then, the rest of the budget can be spent on the controls reducing medium risks. And finally we should consider low risks.

Definition 31 *We define the function chng which transforms attribute values if a security control (from a security control set CS) is installed as follows:*

$$chng : CS \times V_P \rightarrow V'_P \tag{7.7}$$

Where V_P is the set of sets of attributes values used in policies P before security controls are installed, and V'_P is the set of sets of attribute values used in policies P after security controls are installed

Now we can find how good is a security control cs_j, i.e., how many high risks become medium and low after its installation. For instance, let $\hat{h}_{cs_j}$ be this difference, then it can be computed as follows:

$$\begin{aligned}\hat{h}_{cs_j} = HIGH(\{\overline{str}(p_i, v_{r_{i,1}}, \ldots, v_{r_{i,n}}) \cdot rank(p_i) : \forall p_i \in P\}) \\ -HIGH(\{\overline{str}(p_i, v'_{r_{i,1}}, \ldots, v'_{r_{i,n}}) \cdot rank(p_i) : \forall p_i \in P\})\end{aligned} \tag{7.8}$$

Every security control has its cost c_{cs_j}. The overall security budget is W. Some security controls which should be implemented together in order to mitigate risks we consider as one complex countermeasure with the summarized cost. Note, that if the atomic security controls are able to mitigate some risks not only as a part of a complex countermeasure but by their own they must be also considered separately.

Now we can formalize the problem:

$$\text{maximise} \sum_{j=1}^{m} \hat{h}_{cs_j} \cdot x_j \tag{7.9}$$

$$\sum_{j=1}^{m} c_{cs_j} \cdot x_j < W \tag{7.10}$$

$$x_j = \{0, 1\}, j = 1, \ldots, m \tag{7.11}$$

Where m is the number of controls that can be installed, $x_j = 1$ in the formula means that the corresponding security control (cs_j) have been selected, $x_j = 0$ otherwise.

This is a classical knapsack problem [135]. The knapsack problem can be solved by methods of dynamic programming. After selection of the security controls mitigating high risks the mitigation should be continued for medium risks with the rest of the budget:

$$W_{medium} = W - \sum_{j=1}^{m} c_{cs_j} \cdot x_j \tag{7.12}$$

Here we assumed that there were no unacceptable risks in the beginning because a service provider is able to know the desires of service consumers only when she has contract with them. And since the contract exists we conclude that service consumers have not found unacceptable risks in the guaranteed policies. If a service provider can assess risk for service consumers which have not selected this provider (e.g., by collecting wishes through a questionnaire), then she has to eliminate as many unacceptable risks as possible first of all.

Usually, a service provider works with more than one service consumer. Thus, the provider has to improve her system in order to mitigate as many risks as possible. In order to solve this more complex problem we should simply consider policies for different service consumers together. Suppose the number of clients is n:

$$\begin{aligned} \hat{h}_{cs_j} = \sum_{l=1}^{n} (&HIGH(\{\overline{str}(p_i, v_{r_{i,1}}, \ldots, v_{r_{i,n}}) \cdot rank(p_i) : \forall p_i \in P_n\}) \\ &- HIGH(\{\overline{str}(p_i, v'_{r_{i,1}}, \ldots, v'_{r_{i,n}}) \cdot rank(p_i) : \forall p_i \in P_n\})) \end{aligned} \tag{7.13}$$

The rest of analysis goes as it was defined for the case of one service consumer.

Chapter 8

Enforcement of Usage Control Policies under Uncertainties

We propose a set of policy enforcement models which help to mitigate the uncertainties associated with mutable attributes. In our model, the reference monitor, as usual, evaluates logical predicates over attributes and, additionally, makes some estimates on how much observed attribute values differ from the real state of the world. The final access decision takes into account both factors. We assign costs for granting and revoking access to legitimate and malicious users and compare the proposed policy enforcement models in terms of cost-efficiency.

The chapter is structured as follows. Section 8.1 recalls important notes on UCON. Section 8.2 introduces the model of a mutable attribute. Section 8.3 enlists all types of uncertainties associated with mutable attributes. Section 8.4 presents models of a correct policy enforcement. Sections 8.5 and 8.6 outline a cost model and estimate an average profit for a policy enforcement under uncertainties for access and usage control. Section 8.7 presents the architecture of the reference monitor for enforcement of policies under uncertainties.

8.1 Peculiarities of Usage Control Model

Usage Control model (UCON) [129] requires continuous control over long-lasting accesses to computational resources (e.g., an interaction with a service, an execution of a job in Grid, a run of a virtual machine in Cloud). Continuity of control is a specific feature of the UCON intended to operate in a mutable context. The context is formed by attributes of a requesting subject, an accessed object and an execution environment.

An attribute is denoted as $h.r$ where h identifies a subject requesting an object, the object itself, or an environment, and r refers to the attribute. An assignment of an attribute maps it to a value in its domain V_r, i.e., $h.r = v$, where $v \in V_r$. For simplicity, we assume that there is only one attribute in the system denoted as r

and that this attribute has a finite domain of values.

Attribute mutability is an important feature of the UCON, which means that an attribute can change its value as a result of an access request or another uncontrollable factor. We define *the behavior of the attribute* as a sequence of values assigned to an attribute with time passage $v_0 v_1 \dots v_i \dots$ where v_0 refers to the attribute value when a subject sends an access request, and index $i \in \mathbb{N}$ refers to a time point at which the attribute changes its value. We define a strictly increasing function cl which assigns a real time value to any index, $cl : \mathbb{N} \to \mathbb{R}$.

Access decisions in the UCON are based on authorizations (predicates over subject and object attributes), conditions (predicates over environmental attributes), and obligations (actions that must be performed by a requesting subject). We consider security policies consisting of authorization and condition predicates only, that is, the $UCON_{AC}$ model [129]. We define a predicate p to be a boolean-valued function mapping an attribute value to either true or false, $p : V_r \to \{true, false\}$.

Another important feature of the UCON is that it specifies when access decisions are evaluated and enforced. There are two phases: preauthorization or *access control* when *preUpdate* checks of the attribute values are done, and continuous policy enforcement or *usage control* when continuous *onUpdate* checks of the attribute values are done.

Access control starts at time t_{try} when a user sends a request to the reference monitor. The reference monitor acquires an attribute value v_0, evaluates authorization predicates only once and grants the access at time t_{perm} if $p(v_0) = true$, and $t_{try} = cl(0)$, $t_{try} \leq t_{perm}$.

Usage control begins at time t_{perm} when the attribute takes value v_i, and $cl(i) \leq t_{perm} < cl(i+1)$. The reference monitor reevaluates authorization predicates each time the attribute changes its value. The access should be continued by time $t_{now} = cl(j)$ only if $p(v_i) \wedge p(v_{i+1}) \wedge \dots \wedge p(v_j) = true$. Although usage control ends as a result of the access revocation or at the subject's discretion, for simplicity we consider only the first scenario. When a new value v_k violates the policy, i.e., $p(v_k) = false$, the reference monitor revokes the access. Usage control is over at time $t_{rev} = cl(k)$.

8.1.1 Example

We consider a reputation of a service provider in a service-oriented architecture (SOA). The attribute changes its value based on "bad", "good" and "neutral" feedback received from other parties. The attribute domain is $V_r = \{$"general", "normal", "suspicious", "malicious"$\}$. There is a reputation management system (RMS) which measures the reputation value for all service providers in the SOA. We assume that the RMS is managed by a service registry that collects the feedback about previous interactions of service consumers and service providers. Every service consumer has an access and usage control system (AUCS) which allows processing of data belonging to a service consumer only if the reputation of a service provider is other than "malicious". Before granting the access to data, the AUCS (reference

monitor) pulls the reputation value from the RMS (attribute provider). If the value is "malicious" the AUCS denies access, otherwise the AUCS grants access to the service provider. During the usage session the AUCS periodically pulls the reputation from the RMS.

The problem is that, at the time of the access request, a service provider may be involved in several data processing jobs for which the RMS has no feedback yet. In other words, the reputation that the service provider has at the time of the access request could differ from the real one. The AUCS, which uses only the current version of the reputation, is opened to the following attack. A new service provider with a good or neutral reputation gets involved in many jobs in a short period of time. The service provider abuses her rights but, since feedback about her behavior is provided only at the end of a job, her reputation remains good for some time. During this time the malicious service provider is still able to have an access to consumers data. The AUCS should take into account the uncertainty which is in the system in order to make a right access decision.

After granting access to some data the AUCS should monitor the current reputation of the service provider. Now, during the usage of the resource, the AUCS has another problem which is also rooted in uncertainty. The AUCS has to define how often the reputation of the service provider has to be requested from the RMS. Continuous checks of the reputation value imply the use of computational resources and are expensive to perform. There is a need for a balance between security and benefits of usage of the resource.

8.2 Attribute Model

Our main concern in this chapter is the enforcement of a UCON policy based on a *remote* attribute with *observable mutability*. *Remote* means that an attribute is managed by the attribute provider which is not under the control of the reference monitor. *Observable mutability* means that the reference monitor observes only how the attribute behaves in time. Thus, for the same attribute we distinguish *real attribute values* which truly describe the attribute behavior in the system and *observed attribute values* which are obtained by the reference monitor and used to evaluate authorization predicates.

8.2.1 Real Attribute Values

We assume that a change of the attribute's value can be modeled as a *random event*. Let $\omega : r = v$ denote this event which happens when the attribute r takes the value v. We define Ω_r to represent a set of all possible events ω. Since the attribute can take *any* value from its domain, there is a one-to-one correspondence between elements of Ω_r and V_r. Each change of an attribute is paired with the value the attribute takes as the result of this change.

In probability theory, it is often easier to deal with a value associated with the random variable rather than with the event itself. Therefore, we introduce a random variable A which gives a numerical description of the event ω. A is a real valued function on Ω_r, that is $A : \Omega_r \to \mathbb{R}$. The event $A = a$ represents the fact that the attribute r takes the value v, s.t., $A(\omega) = a_\omega$. Let probability of the event to happen be $\mathbf{Pr}[A = a]$. The function $\mathbf{Pr}$ has all properties of a probability function, e.g., for any event e, $0 \leq \mathbf{Pr}[e] \leq 1$. We write $e_1 \cap e_2$ for occurrence of both e_1 and e_2 and write $e_1 \cup e_2$ for the occurrence of either e_1 or e_2 (or both). Let the event $\mathcal{P}(A)$ denote the fact that an attribute takes *any* value which satisfies a policy, i.e., $\mathcal{P}(A) = \bigcup_{\omega \in \Omega_G}(A = a_\omega)$, and $\Omega_G = \{r = v : \ p(v) = \mathit{true},\ v \in V_r\}$. The event $\overline{\mathcal{P}}(A)$ specifies the fact that the attribute takes *any* value which violates the policy. Further, we use A to refer to the attribute value.

Let the behavior of a *real attribute* be specified by a scheme $\langle \mathbf{A}, \mathbf{CL}_{AP} \rangle$, where:

- $\mathbf{A} = \{A_i : i \in \mathbb{N}\}$ is a discrete-time stochastic process modeling a behavior of a mutable attribute. We call A_i the state of the process at i, and $A_i = a_i$ denotes that after i changes the attribute value equals a_i;
- $\mathbf{CL}_{AP} = \{cl_{AP}(j) : j \in \mathbb{N}\}$ is an ordered set of timestamps assigned to each attribute change by the attribute provider when it happens. We assume that $cl_{AP}(0) = t_{try}$ and for all $j \geq 1$, $cl_{AP}(j) = cl_{AP}(j-1) + \mathrm{T_j}$, where $\mathrm{T_j} > 0$ and it specifies a time interval between adjacent attribute changes.

Example 15 *A reputation attribute may be modeled as a random variable A with values $A(r =$ "general"$) = 1$, $A(r =$ "normal"$) = 2$, $A(r =$ "suspicious"$) = 3$, and $A(r =$ "malicious"$) = 4$. The mutability of the reputation attribute could be modeled as a discrete-time Markov chain [85, 86] uniquely defined by the one-step transition matrix. Thus, the entry in the i-th row and j-th column is the transition probability $\mathbf{Pr}[A_i = a \mid A_{i-1} = b]$ giving the probability that the attribute changes value to a if its current value is b.*

Figure 8.1 shows the Markov model for our example with the transition probabilities collected in a transition matrix. These probabilities could be used in order to find whether reputation has a certain value (e.g., $\mathbf{Pr}[A = 2]$). The transition probabilities are taken from the history of changes stored by the RMS and shared with the AUCS:

$$\mathbf{Prob} = \begin{bmatrix} 0.6 & 0.4 & 0.0 & 0.0 \\ 0.5 & 0.3 & 0.2 & 0.0 \\ 0.0 & 0.2 & 0.3 & 0.5 \\ 0.0 & 0.0 & 0.1 & 0.9 \end{bmatrix} \tag{8.1}$$

8.2.2 Observed Attribute Values

Only the attribute provider knows how the attribute behaves in time, but the reference monitor can also observe this process. There are two basic models how

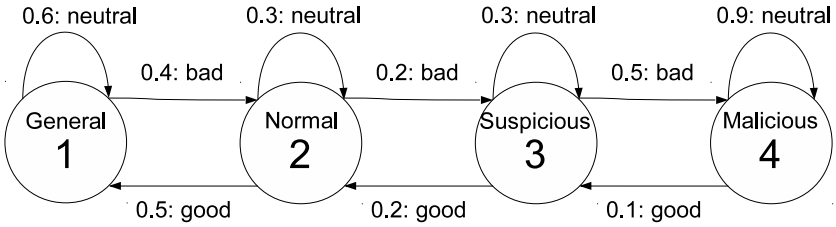

Figure 8.1: A reputation attribute model

attribute changes are delivered to the reference monitor: *push* and *pull*. The push model defines a scenario where every new attribute value is timestamped and pushed by the attribute provider to the reference monitor. The pull model defines a scenario where the reference monitor queries the attribute provider to give the current attribute value. The attribute provider replies with the value, its timestamp and some additional information.

By analogy with real attribute values, let *observed attributes* be specified by a scheme $\langle \tilde{\mathbf{A}}, \mathbf{CL}_{RM} \rangle$, where:

- $\tilde{\mathbf{A}} = \{\tilde{A}_i : i \in \mathbb{N}\}$ is a discrete-time stochastic process modeling an observation of attribute changes over time. $\tilde{A}_i = a_i$ denotes that an attribute value after i observations equals a_i;

- $\mathbf{CL}_{RM} = \{cl_{RM}(j) : j \in \mathbb{N}\}$ is an ordered set of timestamps assigned by the *reference monitor*. A timestamp j denotes when the j-th observation of an attribute value was processed and the appropriate access decision was enforced by the reference monitor. We assume that $cl_{RM}(0) = t_{perm}$.

Real and observed attribute values form a bipartite directed graph $\mathrm{W} = (\mathbf{A}, \tilde{\mathbf{A}}, \mathbf{E})$, where edges $\mathbf{E}$ connect real and observed attributes via push/pull queries. If there exists an edge e which connects A_c and $\tilde{A}_{c'}$, we say that A_c corresponds to $\tilde{A}_{c'}$ and denote this as $A_c \approx \tilde{A}_{c'}$. To evaluate authorization predicates, the reference monitor can exploit observed attribute values and timestamps of the corresponding real counterparts.

Example 16 *Figure 8.2 describes the exchange of attributes between the RMS and the AUCS from our example. The left part of the figure is devoted to access control. The attribute value A_0 sent by the RMS at $t_{try} = cl_{AP}(0)$ is observed by the AUCS at $t_{perm} = cl_{RM}(0)$ as $\tilde{A}_0$, i.e., $A_0 \approx \tilde{A}_0$.*

For the right part of the figure, i.e., usage control, the RMS sends the fourth change of the attribute A_4 at $cl_{AP}(4)$, which is observed by the AUCS as $\tilde{A}_2$ at time $cl_{RM}(2)$, i.e., $A_4 \approx \tilde{A}_2$.

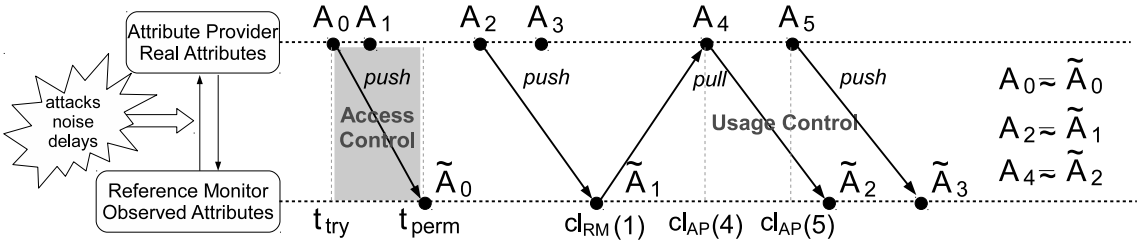

Figure 8.2: Real and observed attribute values

8.3 Intentional and Unintentional Uncertainties

Observed attributes values differ from their real counterparts due to attacks, noise, delays during delivery, missed values, etc. We call *uncertainty* a property on real and observed attributes values which specifies how these values vary. The closer observed values are to the real ones the more reliable the enforcement of the policy. We consider two types of uncertainties: *unintentional* (*freshness* and *correctness*), and *intentional* (*trustworthiness*).

Freshness of Attributes

Freshness is an unintentional uncertainty occurring due to the mutability of attributes. Generally, this property means that the latest observed value of an attribute is out-of-date, while the current real value of the attribute is unknown. We introduce three types of freshness uncertainties.

Freshness I (non-continuous checks) corresponds to scenarios where only part of attribute changes is detected because the checks are carried out through some time interval:

$$\exists c \geq 0,\ m > 0,\ \ c, m \in \mathbb{N}\ :\ A_{c+m} \approx \tilde{A}_c \tag{8.2}$$

In Figure 8.2 the attribute provider sends A_2 and A_4 values, while A_3 is not sent. Thus, the reference monitor making a decision after getting A_2 value ($\tilde{A}_1$) uses the wrong input for the decision.

Example 17 *After granting access to the service provider the AUCS has to monitor the reputation value during the usage. The RMS sends the current reputation value only once per hour in order to save resources. If the reputation of the service provider became "malicious" between the checks she will still process data because the AUCS is not aware of this change.*

Freshness II (delays in delivery and processing) implies that there are inevitable time delays in delivery of an attribute value (due to a network latency)

and decision making (evaluation of authorization predicates). That is:

$$\exists c' \geq 0,\ c'' \geq 0,\ c', c'' \in \mathbb{N} :\ A_{c''} \approx \tilde{A}_{c'} \quad cl_{RM}(c') > cl_{AP}(c'') \tag{8.3}$$

When the attribute provider gets a request for the attribute it sends value A_0 to the reference monitor (see Figure 8.2 again). Since the delivery takes some time $(t_{perm} - t_{try})$ the attribute changes to A_1 and the access control system uses the wrong value for the decision making.

Example 18 *A service provider asks the AUCS for an access. The AUCS asks the RMS for the current reputation value of the service provider and gets "suspicious". The problem is that because of the delay in the delivery when the AUCS makes the decision the value becomes obsolete, since a new feedback comes to the RMS and the reputation value changes to "malicious".*

Freshness III (pending updates) corresponds to scenarios where the current attribute value is uncertain since some update queries are pending at the time of the access reevaluation. In this case, the attribute provider sends two values: (i) the last certain attribute value, (ii) additional information on how the real value differs from the last certain value.

The presence of the uncertainty Freshness III implies:

$$\exists c' \geq 0,\ c'' \geq 0, m > 0,\ c', c'', m \in \mathbb{N} :\ A_{c''} \approx \tilde{A}_{c'} \quad cl_{AP}(c'' + m) \leq cl_{RM}(c') \tag{8.4}$$

In Figure 8.2, the reference monitor which is going to make a decision after getting value A_4 may already know, that this value is not certain. The attribute provider sending A_4 also sends additional information that there should be one more change $(m = 1)$ in the attribute between $cl_{AP}(5) - cl_{AP}(4)$.

Example 19 *The RMS updates the reputation only when a data processing job is ended and the RMS receives feedback from a service consumer. Data processing jobs run concurrently and each single execution may be long-lived and last for days. The access decision to allows data processing (made by the AUCS) is based on the reputation value dated by the last registered feedback and on the number of data processing jobs currently running by the service provider. Indeed, the service provider can perform malicious activities over the data but this fact will only be discovered afterwards. The only way to make the certain decision is to block the access until all running data processing jobs terminate. Instead, the AUCS should be set up to make an access decision with some uncertainty regarding the current reputation of the service provider. This uncertainty is contained in the amount of data processing jobs still active (value m).*

Correctness

Correctness is unintentional uncertainty occurring due to additive noises that usually exist in case of non-accurate measurements. For example, the location attribute can be sensed only with the given precision. Thus, observed attribute values differ from the real ones:

$$\exists c' \geq 0,\ c'' \geq 0\ , c', c'' \in \mathbb{N} :\ A_{c''} \approx \tilde{A}_{c'} \qquad (8.5)$$
$$\tilde{A}_{c'} = A_{c''} + N$$

where N is a random variable that models additive noises presented in observed attribute values. The reference monitor may know that the attribute value measured by the attribute provider is not precise. Thus, on getting a value (e.g., A_2) the reference monitor makes the decision taking the mistake N into account. This case cannot be shown in Figure 8.2 directly.

Example 20 *It is known that the RMS reputation values may differ from the real ones by a maximum of* 1 *for various reasons (e.g., some feedback could be lost). The AUCS should be aware of the possibility of such mistakes.*

Trustworthiness

Trustworthiness is an intentional uncertainty. It appears as a result of the attribute provider altering attributes or as a result of attacks during attribute delivery, storage, etc. Current approaches guarantee only the integrity of an attribute by validating a signature of the entity signing the attribute, but this does not guarantee trustworthiness. This uncertainty assumes that either an attribute value, or a time of issuance, or both can be modified. It implies that the reference monitor does not trust the attribute provider and assigns a confidence value for each observed attribute. This value represents the reliability of the attribute provider in the assertions it makes.

Approaches that consider trust as a probability of an interaction to succeed or to fail can be used for the analysis of a probability of a policy to be violated when the policy attributes can be either true or false (e.g., [41, 152]). For the computation of trustworthiness value a feedback collection mechanism is required, which is powerful enough to detect whether the received value was modified. Naturally, if such check could be performed timely for all received values there is no intentional uncertainty in the system. However, such check may require significant amount of resources and time (e.g., checking the logs of a service provider) and the information about trustworthiness of a user may be collected only from time to time just to compute the reputation value. The problem of trustworthiness for the attributes that have a wider domain of possible values is an open issue which is to be investigated. But our method only uses the probability of policy violation and does not depend on its way of computation.

The presence of the trustworthiness uncertainty states:

$$\exists c' \geq 0,\ c'' \geq 0\ , c', c'' \in \mathbb{N} :\ A_{c''} \approx \tilde{A}_{c'} \qquad (8.6)$$
$$\mathbf{Pr}[\tilde{A}_{c'} = A_{c''}] = \eta,\ 0 \leq \eta < 1$$

i.e., the probability that the observed attribute is equal to the real counterpart is below 1 and we assume that the reference monitor has the power to compute η. Similarly to Correctness this uncertainty cannot be shown in Figure 8.2.

Example 21 *The RMS sends to the AUCS the reputation attribute is equal to "normal". The AUCS does not trust the RMS entirely and based on its internal estimates the AUCS considers that the observed attribute has the "normal" value but with a probability of 0.8.*

In the following sections we continue to use our example taking into consideration only the uncertainties of Freshness III type for access control (Section 8.5) and Freshness I for usage control (Section 8.6).

8.4 Correct Policy Enforcement

The correct policy enforcement implies that having observed attributes the reference monitor enforces the policy exactly in the same fashion as with real attributes, and both observed and real attributes satisfy authorization predicates.

8.4.1 Correct Enforcement of Access Control

Access control starts at time $t_{try} = cl_{AP}(0)$ when the user sends the access request and the initial attribute value. The reference monitor evaluates a policy only once and grants an access to a resource at time $t_{perm} = cl_{RM}(0)$ if the policy holds. We say that the *policy holds for access control* if:

1. $\mathcal{P}(\tilde{A}_0)$ happens, i.e., the initial observed attribute value $\tilde{A}_0$ satisfies the policy;
2. $\mathcal{P}(A_m)$ happens, i.e., the real attribute value A_m at the time the decision is made also satisfies the policy and $cl_{AP}(m) \leq t_{perm} < cl_{AP}(m+1)$ where $m \geq 0$.

Note, that some attribute changes may happen between t_{try} and t_{perm}, but attribute values must satisfy security policy exactly when the request is issued and later when the access decision is evaluated.

Let H be an event specifying that the policy holds and $\overline{H}$ specifies the opposite. Clearly, the policy satisfaction and violation can be defined as:

$$H = \mathcal{P}(\tilde{A}_0) \cap \mathcal{P}(A_m) \qquad (8.7)$$
$$\overline{H} = \overline{\mathcal{P}}(\tilde{A}_0) \cup (\mathcal{P}(\tilde{A}_0) \cap \overline{\mathcal{P}}(A_m))$$

Definition 32 (**Correct Enforcement of Access Control**) *The reference monitor grants the access at t_{perm} if the policy holds and denies it otherwise.*

Let G be an event specifying that the reference monitor grants the access and $\overline{G}$ specifies the opposite (i.e., denies the access). Thus, the correct enforcement of access control is:

$$G = H, \quad \overline{G} = \overline{H} \tag{8.8}$$

8.4.2 Correct Enforcement of Usage Control

We say that a *policy holds for usage control* on a time interval $[t_b : t_e]$ if:

1. $\mathcal{P}(\tilde{A}_k) \cap \mathcal{P}(\tilde{A}_{k+1}) \cap ... \cap \mathcal{P}(\tilde{A}_l)$ happens and $cl_{RM}(k) \leq t_b < cl_{RM}(k+1), cl_{RM}(l) \leq t_e < cl_{RM}(l+1)$;

2. $\mathcal{P}(A_i) \cap \mathcal{P}(A_{i+1}) \cap ... \cap \mathcal{P}(A_j)$ happens and $cl_{AP}(i) \leq t_b < cl_{AP}(i+1), cl_{AP}(j) \leq t_e < cl_{AP}(j+1)$,

i.e., all real and observed attribute changes occurring within this interval do satisfy authorization predicates.

If there is at least one attribute value (either real or observed) which does not satisfy authorization predicates, we call this a *policy violation of usage control.*

Definition 33 (**Correct Enforcement of Usage Control**) *The reference monitor correctly continues the usage session at t_{now} if a policy holds on interval $[t_{perm} : t_{now}]$. The reference monitor revokes the access immediately when the policy violation occurs.*

8.5 Enforcement of Access Control under Uncertainties

Correct enforcement is not feasible in the presence of uncertainties since the reference monitor is unable to show that real attribute values satisfy a policy. The basic idea of the policy enforcement of *access control* under uncertainties is:

1. The reference monitor evaluates the policy with respect to observed attribute values.

2. If the observed values satisfy the policy, the reference monitor runs an experiment which estimates to what extent the observed attributes differ from the real ones. If this difference is negligible, the experiment succeeds and the reference monitor allows the access.

8.5.1 Models for Access Control Enforcement

We suppose that the reference monitor is powerful to get some probabilistic knowledge about a real attribute value based on the observed attribute $\tilde{A}_0 = a$:

$$\mathbf{Pr}_{RM} = \mathbf{Pr}[\mathcal{P}(A_m)|\tilde{A}_0 = a]$$

$\mathbf{Pr}_{RM}$ specifies a conditional probability that a value of real attribute A_m satisfies authorization predicates at time t_{perm} if the observed attribute value at time t_{perm} is equal to a. The reference monitor computes $\mathbf{Pr}_{RM}$ using the following data:

1. observed values of the attribute;
2. parameters of a stochastic process that models a real behavior of an attribute;
3. a list of uncertainties presented in the system.

Possible combinations of the last two factors produce a variety of techniques on *how* to compute $\mathbf{Pr}_{RM}$. As an example, we refer the reader to [85, 86] and Appendix A.1 and A.2 where the behavior of an attribute is modeled as a Markov chain and freshness uncertainties exist in the system. Another example given in [21] studies a static attribute (i.e. the attribute does not change its value over time) in the presence of the trustworthiness uncertainty. In our example for access control we compute $\mathbf{Pr}_{RM}$ considering only Freshness III uncertainty and model the attribute behavior as a discrete-time Markov chain.

Let Y be a random variable such that

$$Y = \begin{cases} 1 & \text{if uncertainties are acceptable} \\ 0 & \text{otherwise} \end{cases}$$

Let $\delta(x)$ be a function, that is

$$\delta(x) = \begin{cases} 1 & \text{if } x \geq th \\ 0 & \text{otherwise} \end{cases}$$

where th is a real-value threshold.

We propose two models of enforcement for access control under uncertainties: a *threshold* enforcement and a *flip coin* enforcement. The reference monitor chooses one of these models.

Definition 34 (**Threshold Enforcement of Access Control**) *The reference monitor computes* $\mathbf{Pr}_{RM}$ *and grants access at* t_{perm} *if:*

1. $\mathcal{P}(\tilde{A}_0)$ *happens;*
2. $Y = 1$, *where* $\mathbf{Pr}[Y = 1] = \delta(\mathbf{Pr}_{RM})$.

otherwise, the access is denied.

That is, if the initial observed attribute value satisfies authorization predicates, the reference monitor grants the access if the probability that the real attribute value A_m also satisfies authorization predicates is above a specified threshold th.

Definition 35 (**Flip Coin Enforcement of Access Control**) *The reference monitor behaves exactly as in the threshold enforcement but uses* $\mathbf{Pr}[Y = 1] = \mathbf{Pr}_{RM}$ *instead.*

Hence, if the initial observed attribute value satisfies authorization predicates, the reference monitor runs the random experiment that succeeds (returns grant) with probability $\mathbf{Pr}_{RM}$ and fails (returns deny) with probability $1 - \mathbf{Pr}_{RM}$.

In the notation of events, we get for the enforcement of access control under uncertainties (either threshold or flip coin):

$$\begin{aligned} G &= \mathcal{P}(\tilde{A}_0) \cap [Y = 1] \\ \overline{G} &= \overline{\mathcal{P}}(\tilde{A}_0) \cup (\mathcal{P}(\tilde{A}_0) \cap [Y = 0]) \end{aligned} \tag{8.9}$$

Example 22 *Consider the access control part of our example (see Figure 8.2). The AUCS gets value* $\tilde{A}_0 = 3$ *("suspicious") at time* t_{perm} *and it knows that there was one attribute change between* t_{try} *and* t_{perm}*. Now the AUCS should evaluate whether the current reputation value is still a good one, e.g.,* $\mathbf{Pr}_{RM} = \mathbf{Pr}[\mathcal{P}(A_m)|\tilde{A}_0 = 3]$.

The transition matrix (see Example 15) shows that if the initial attribute value is $A_0 = 3$*, then there are three possibilities for the value to evolve in one step: (i)* $(A_1 = 4)$ *with* $\mathbf{Pr}_{34} = 0.5$*; (ii)* $(A_1 = 3)$ *with* $\mathbf{Pr}_{33} = 0.3$*; (iii)* $(A_1 = 2)$ *with* $\mathbf{Pr}_{32} = 0.2$*. Since, the good states are 1, 2, and 3 then* $\mathbf{Pr}_{RM} = 0.3 + 0.2 = 0.5$.

8.5.2 Cost Matrix

We would now like to estimate the cost-effectiveness of the proposed enforcement methods. Our goal is to find the *expected profit* $\langle C \rangle$ for the enforcement of access control.

We assign monetary outcomes for granting and revoking access. Correct enforcement is impossible in the presence of uncertainties and mistakes in the decisions made by the reference monitor are unavoidable. We have four scenarios (events) of how the reference monitor acts under uncertainties:

- $G \cap H$ *true positive*: grant access when a policy holds;
- $G \cap \overline{H}$ *false negative*: grant access when a policy is violated;
- $\overline{G} \cap H$ *false positive*: deny access when a policy holds;
- $\overline{G} \cap \overline{H}$ *true negative*: deny access when a policy is violated.

Where *true positive* and *true negative* are well-chosen scenarios, while *false negative* and *false positive* are erroneous.

Each scenario has a monetary outcome, i.e. *cost*, the reference monitor loses or gains if a scenario happens. Let C_{tp} denote the cost of the true positive scenario, when the reference monitor grants the access and the policy really holds. C_{fn}, C_{fp}, C_{tn} are the costs of the remaining scenarios, respectively. The semantics of costs for access control corresponds to "pay-per-access", and specifies exact benefits and losses for a given access request. Naturally, well-chosen scenarios have positive values, i.e. $C_{tp} \geq 0, C_{tn} \geq 0$, while the erroneous ones have negative costs, i.e., $C_{fp} < 0, C_{fn} < 0$. Finally, let C_a be the cost to push/pull (observe) an attribute value.

Finding correct costs is not an easy task and usually requires a considerable amount of statistical data. Thus, we make the usual assumption for risk-based methods that the reference monitor has enough historical data to compute costs.

8.5.3 Cost of Access Control Enforcement

The expected profit received by the reference monitor processing a single access request is the sum of the costs of all 4 scenarios weighted on corresponding probabilities.

$$\begin{aligned}\langle C\rangle = {} & C_{tp}\cdot\mathbf{Pr}[G\cap H] + C_{fn}\cdot\mathbf{Pr}[G\cap\overline{H}] \\ & + C_{fp}\cdot\mathbf{Pr}[\overline{G}\cap H] + C_{tn}\cdot\mathbf{Pr}[\overline{G}\cap\overline{H}] + C_a\end{aligned} \tag{8.10}$$

Correct Enforcement

Since H and $\overline{H}$ are disjoint events, i.e. $\mathbf{Pr}[H\cap\overline{H}] = 0$ and $\mathbf{Pr}[H] + \mathbf{Pr}[\overline{H}] = 1$, from Equations 8.7, 8.8 and 8.10 we receive:

$$\langle C\rangle_{cor} = C_{tp}\cdot\mathbf{Pr}[H] + C_{tn}\cdot\mathbf{Pr}[\overline{H}] + C_a \tag{8.11}$$

$$\begin{aligned}\mathbf{Pr}[H] &= \mathbf{Pr}[\mathcal{P}(\tilde{A}_0)\cap\mathcal{P}(A_m)] \\ &= \mathbf{Pr}[\mathcal{P}(\tilde{A}_0)]\cdot\mathbf{Pr}[\mathcal{P}(A_m)|\mathcal{P}(\tilde{A}_0)]\end{aligned} \tag{8.12}$$

In what follows, we use $\mathbf{Pr}[\mathcal{P}(\tilde{A}_0)]$ interchangeably with α, and $\mathbf{Pr}[\mathcal{P}(A_m)|\mathcal{P}(\tilde{A}_0)]$ with β. Note, that for the correct access control $\beta = 1$. Finally,

$$\langle C\rangle_{cor} = C_{tp}\cdot\alpha\cdot\beta + C_{tn}\cdot(1-\alpha\cdot\beta) + C_a \tag{8.13}$$

Threshold Enforcement

We point out that the probability of a policy satisfaction for real attributes is conditionally independent of the estimates made by the reference monitor given that

observed attribute values satisfy the policy. Using this observation and Equations 8.7 and 8.9 we receive

$$\begin{aligned}
\mathbf{Pr}[G \cap H] &= \alpha \cdot \beta \cdot \mathbf{Pr}[Y = 1 | \mathcal{P}(\tilde{A}_0)] \\
\mathbf{Pr}[G \cap \overline{H}] &= \alpha \cdot (1 - \beta) \cdot \mathbf{Pr}[Y = 1 | \mathcal{P}(\tilde{A}_0)] \\
\mathbf{Pr}[\overline{G} \cap H] &= \alpha \cdot \beta \cdot (1 - \mathbf{Pr}[Y = 1 | \mathcal{P}(\tilde{A}_0)]) \\
\mathbf{Pr}[\overline{G} \cap \overline{H}] &= \alpha \cdot (1 - \beta) \cdot (1 - \mathbf{Pr}[Y = 1 | \mathcal{P}(\tilde{A}_0)])
\end{aligned} \tag{8.14}$$

We assume that all access requests come with the same initial attribute value a which satisfies authorization predicates. Such a situation is modeled with an assumption $\alpha = 1$. With this assumption, we get that $\mathbf{Pr}[Y = 1 | \mathcal{P}(\tilde{A}_0)] = \mathbf{Pr}[Y = 1 | \tilde{A}_0 = a]$ and $\beta = \mathbf{Pr}[\mathcal{P}(A_m) | \tilde{A}_0 = a] = \mathbf{Pr}_{RM}$.

We denote $C_g = \beta \cdot (C_{tp} - C_{fn}) + C_{fn}$ and $C_d = \beta \cdot C_{fp} + C_{tn} \cdot (1 - \beta)$. From Definition 34 and Equations 8.10 and 8.14 we get the average profit for a threshold enforcement:

$$\langle C \rangle_{th} = C_a + \begin{cases} C_g & \text{if } \beta \geq th \\ C_d & \text{otherwise} \end{cases} \tag{8.15}$$

Cost-effective enforcement implies that we should pick a threshold which gives the maximal profit for all possible average costs. Since the cost is a function of β which takes any value from 0 to 1, we should maximize the sum of costs for all β. The argument, for which this sum attains its maximum, constitutes the *optimal threshold value*:

$$\arg\max_{th} \int_0^1 \langle C \rangle_{th} \, d\beta$$

To obtain it, we solve the equation in which the derivative of the integral takes zero:

$$\left(\int_0^{th} C_d \, d\beta + \int_{th}^1 C_g \, d\beta \right)'_{th} = 0$$

Hence, the optimal threshold value is given by

$$th = \frac{C_{fn} - C_{tn}}{C_{fp} + C_{fn} - C_{tn} - C_{tp}} \tag{8.16}$$

Flip Coin Enforcement

All equations of a threshold enforcement are also valid for a flip coin enforcement. Taking the assumptions made in the threshold enforcement and Definition 35, we obtain the average profit for a flip-coin enforcement per access request:

$$\begin{aligned}
\langle C \rangle_{flip} = {} & C_{tp} \cdot \beta^2 + (C_{fp} + C_{fn}) \cdot \beta \cdot (1 - \beta) \\
& + C_{tn} \cdot (1 - \beta)^2 + C_a
\end{aligned} \tag{8.17}$$

Proposition 2 *Threshold strategy is more cost-effective than flip coin, except the points $\beta = 0$, $\beta = 1$, and $\beta = th$, where the strategies are equal: $\langle C \rangle_{th} \geq \langle C \rangle_{flip}$.*

Proof Consider the first case when $1 > \mathbf{Pr}_{RM} = \beta > th$. We do not consider the case when $\mathbf{Pr}_{RM} = \beta = 1$ since it is easy to see that the two strategies are equal at this point. Now let us derive the conditions where the flip coin strategy is better than the threshold one:

$$\begin{aligned} C_{tp} \cdot \beta^2 + (C_{fp} + C_{fn}) \cdot \beta \cdot (1 - \beta) + C_{tn} \cdot (1 - \beta)^2 + C_a \\ > \beta \cdot (C_{tp} - C_{fn}) + C_{fn} + C_a \end{aligned}$$

Algebraic transformations give:

$$\beta < \frac{C_{fn} - C_{tn}}{C_{fp} + C_{fn} - C_{tn} - C_{tp}} = th$$

We see that the condition for the flip coin strategy violates the initial preposition $\beta > th$. Thus, if $1 > \beta > th$ the threshold strategy is more profitable.

In the same way we can compare the strategies with the conditions $th > \beta > 0$. We exclude $\beta = 0$ point where the strategies are equal. In this case we get that $\beta > th$ that proves once again that the threshold strategy is better, except the points where the strategies are equal. □

Example 23 *We show how different strategies cope with Freshness III uncertainty in our example.*

The AUCS gets the attribute value $\tilde{A}_0 = 2$ at t_{perm} and can compute that there were exactly m attribute changes between t_{try} and t_{perm}. The AUCS must then compute the probability β that the policy holds at t_{perm} and choose the model of the policy enforcement. The probability matrix of the Markov chain was given in Example 15 and the probability β can be found as (see also [66, 85, 86] and Appendix A.1 and A.2):

$$\beta = \mathbf{Pr}[\mathcal{P}(A_m) | \tilde{A}_0 = 2] = \sum_{j \in \{1,2,3\}} (\mathrm{S} \times \mathbf{Prob}^m)_j$$

Where vector $\mathrm{S} = [0\ 1\ 0\ 0]$ specifies the initial attribute value, $\mathbf{Prob}^m$ means matrix $\mathbf{Prob}$ in power m, and $\times$ denotes a product of two matrices.

The AUCS makes monetary estimations and determines the following costs: $C_{tp} = 10$, $C_{fn} = -15$, $C_{fp} = -1$, $C_{tn} = 0$ and to query an attribute we pay $C_a = -2$.

We performed a set of simulations in order to illustrate our theory. We computed the average profit per access request for the correct enforcement $\langle C \rangle_{cor}$, for the threshold enforcement $\langle C \rangle_{th}$, and for the flip coin enforcement $\langle C \rangle_{flip}$. We varied the uncertainties between real and observed attributes by increasing the number m of attribute changes that occur between t_{try} and t_{perm}. We start from $m = 0$ and go up to 30 unobserved attribute changes.

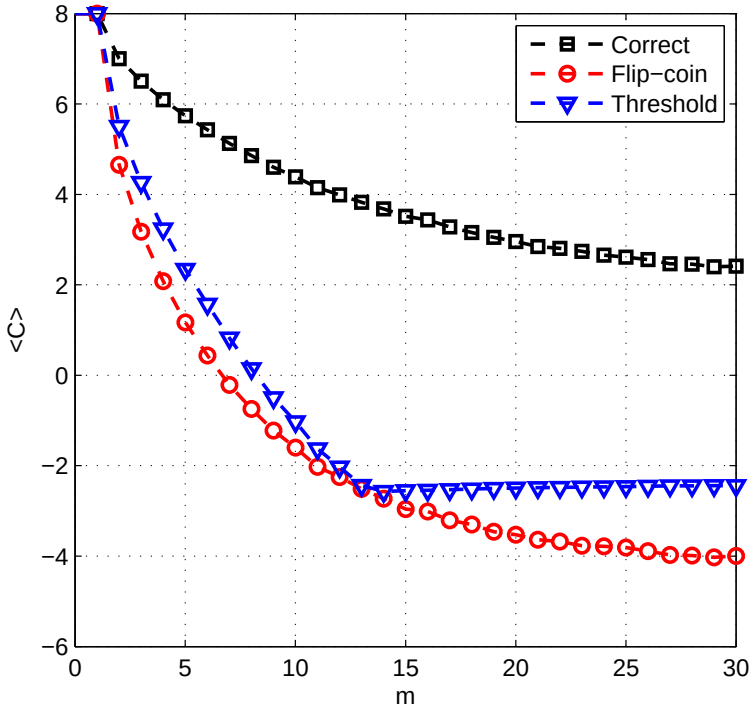

Figure 8.3: Cost-effective enforcement of access control

Figure 8.3 shows the obtained results. The average profit per access request for the correct enforcement is always higher. The decline of the correct curve occurs because while the delay increases the probability that the received value would fail the policy also increases (because of $\mathbf{Pr}[\mathcal{P}(A_m)|\mathcal{P}(\tilde{A}_0)] = \beta$*). Since the attribute cannot get a bad value in* $m = 0$ *or* $m = 1$ *steps (starting from state* 2*) all three curves have the same maximal value in these cases. The flip coin enforcement shows the worse results with respect to the threshold enforcement which tallies with our theoretical findings.*

8.6 Enforcement of Usage Control under Uncertainties

Our model of usage control enforcement under uncertainties imposes that the reference monitor iteratively performs three main activities.

1. Evaluates a policy and makes the decision based on the observed attribute values. If the access decision is "deny", the reference monitor terminates the usage session and halts.

2. Computes when the next attribute query should be performed.

3. Waits until the next check and when time elapses pulls a fresh attribute value.

The reference monitor executes these actions on each *check*. A check is a time interval $[t_b : t_e]$ between two adjacent observations of the attribute $\tilde{A}_{k-1}$ and $\tilde{A}_k$, where $cl_{RM}(k-1) = t_b$, $cl_{RM}(k) = t_e$. The time of the first check is t_{perm} when there is the observed attribute $\tilde{A}_0$. The *usage session* contains a sequence of n checks and $n \in \mathbb{N}$.

8.6.1 Models for Usage Control Enforcement

Decision Making

The basic idea of a decision making for usage control under uncertainties is the same as for access control (see Section 8.5). The only difference is that the reference monitor should take into account all possible changes occurred on a check. We assume that the reference monitor has the power to compute the probability that all real attributes satisfy a policy on the k-th check:

$$\mathbf{Pr}^k_{RM} = \mathbf{Pr}[\mathcal{P}(A_i) \cap ... \cap \mathcal{P}(A_j) | \tilde{A}_{k-1} = a_{k-1} \cap \tilde{A}_k = a_k]$$

where $cl_{AP}(i) \leq cl_{RM}(k-1) \leq cl_{AP}(i+1)$, $cl_{AP}(j-1) \leq cl_{RM}(k) \leq cl_{AP}(j)$.

We propose two models of a decision making for usage control under uncertainties: a threshold and a flip coin.

Definition 36 (**Usage Control Based on Threshold**) *The reference monitor continues the access after n policy checks at* $t_{now} = cl_{RM}(\tilde{A}_n)$ *if:*

1. $\mathcal{P}(\tilde{A}_0) \cap \mathcal{P}(\tilde{A}_1) \cap ... \cap \mathcal{P}(\tilde{A}_n)$ *occurs, i.e., all attribute changes observed within n checks do satisfy the policy,*

2. $\forall k = 1, .., n \ : \ Y_k = 1$, *where* $\mathbf{Pr}[Y_k = 1] = \delta(\mathbf{Pr}^k_{RM})$, *i.e., for each check the probability that a policy holds on this check should be above a specified threshold,*

otherwise access is revoked.

Definition 37 (**Usage Control Based on Coin Flip**) *The reference monitor behaves as in the threshold enforcement but uses* $\mathbf{Pr}[Y_k = 1] = \mathbf{Pr}^k_{RM}$.

Attribute Retrieval

Fresh attribute values could be pushed or pulled. Without loss of generality we assume that the reference monitor is responsible for pulling attribute values. Since frequent attribute queries are not always possible, expensive and lead to a performance slowdown, we assume that several attribute changes may occur on a single

check. Such scenario brings the inevitable Freshness I uncertainty since the reference monitor will observe only a part of attribute changes. The reference monitor should be aware that unnoticed attribute changes may violate a policy and result in a loss.

Our main concern is to find such intervals between queries that give the maximal profit for the enforcement of a usage session. We propose two models of attributes retrieval. The first one is *periodic pull of attributes* when the interval between attribute quires is constant. The second model is *aperiodic pull of attributes*. We assume that the reference monitor may increase the profit if it selects the interval between quires according to the history of observed attributes during the current session. Thus, there is a specific value of interval for each specific check.

8.6.2 Costs of Usage Control Enforcement

Possible combinations of decision making and attribute retrieval launch a variety of enforcement models. We discuss only the models relevant for usage control and do not consider models discussed previously for access control. We examine the cost-effectiveness of models when attributes are pulled periodically and aperiodically while the decision making is based on a threshold. In both models, we set the threshold value to 0 and assume that no uncertainties exist in the system except inevitable Freshness I. Such assumptions allows the reference monitor to skip the execution of the random experiment and just continue access if the observed attribute value satisfies a policy and revoke otherwise.

Cost of Usage Session in case of Periodic Checks

We start with a cost gained from the enforcement of a particular usage session. The semantics of costs for usage control corresponds to "pay-per-time-of-usage" attributes, and specifies the benefits and losses the system gains in a unit of time. The system receives profit if a policy holds on a time interval and this revenue is proportional to the duration of the interval. In opposite, the system suffers losses during the policy violation time. There are three costs for usage control: (i) c_{tp} is the gain per atomic interval of time when all changes of real attributes satisfy the policy; (ii) c_{fn} is the cost per atomic interval of time when the policy fails; and (iii) C_a is the cost paid for the attribute retrieval and the re-evaluation of access decision.

The usage session is associated with a sample sequence s of a stochastic process which models the behavior of a real attribute. That is:

$$s : (A_0 = a_0) \cap (A_1 = a_1) \cap \ldots \cap (A_l = a_l)$$

Let n state a total number of checks in the session before revocation. This means that after the last check the reference monitor revokes the session, i.e., $\overline{\mathcal{P}}(\tilde{A}_n)$ happens and $A_l \approx \tilde{A}_n$. Let q be a number of attribute changes on a check. Since checks are periodic, q is a constant for any check and $l = n \cdot q$. A cost C_s of a particular usage

session depends on the time τ_g when an attribute satisfies a policy, on the time τ_b when the attribute violates the policy, and a number of checks n:

$$C_s = c_{tp} \cdot \tau_g + c_{fn} \cdot \tau_b + C_a \cdot (n+1) \tag{8.18}$$

Let $\theta(x)$ be a function such that

$$\theta(x) = \begin{cases} 1 & \text{if } \mathcal{P}(A_x) \text{ happens} \\ 0 & \text{otherwise, i.e., a policy violation happens} \end{cases}$$

Then, τ_g and τ_b are given by

$$\tau_g = \sum_{j=0}^{l-1} (cl_{AP}(j+1) - cl_{AP}(j)) \cdot \theta(j)$$

$$\tau_b = cl_{RM}(n) - cl_{RM}(0) - \tau_g$$

In fact, s is a random event and let $\mathbf{Pr}[s]$ denote a probability that s occurs. Thus, the *average* cost of usage control enforcement will be a sum over every possible cost weighted by the probability of s:

$$\langle C \rangle_q = \sum_{s \in S} \mathbf{Pr}[s] \cdot C_s \tag{8.19}$$

where S contains all possible sample sequences associated with usage sessions enforced under uncertainties.

Cost-effective enforcement implies that the reference monitor should choose such q that maximizes profit: $\arg\max_q \langle C \rangle_q$.

Cost of Usage Session in case of Aperiodic Checks

In case of aperiodic checks, a number of attribute changes occurred on each check is different. There is a set $Q = \{q_1, q_2, \ldots, q_n\}$ and each q_i tells how many attribute changes happened on the i-th check. All formulas given for periodic checks are valid for aperiodic. Only a number of attribute changes is different, and for aperiodic checks we have that $l = \sum_{q \in Q} q$. We also use $\langle C \rangle_Q$ to denote the average cost of the usage control enforcement under aperiodic checks.

Cost-effective enforcement implies that the reference monitor should choose such Q that gives the maximal profit, i.e., $\arg\max_Q \langle C \rangle_Q$. The simplex method can be used to find Q for which $\langle C \rangle_Q$ attains the maximum. The application of such methods is left behind the scope of our work but initial ideas can be found in [11, 139].

Proposition 3 *Aperiodic checks are at least as good as periodic checks in terms of cost-effectiveness:* $\langle C \rangle_Q \geq \langle C \rangle_q$.

Proof The proof follows from the fact, that the method selects the set Q with the best average cost within all possible Q's. Periodic checks may be considered as a particular case of aperiodic checks when all intervals are equal. □

Example 24 *We continue our example comparing periodic and aperiodic checks.*

The AUCS selects the following costs $c_{tp} = 3$, $c_{fn} = -5$, and $C_a = -2$ on the basis of previous behavior of the reputation attribute. The AUCS exploits discrete-time Markov chain (Equation 8.1) to model the behavior of the reputation and find the best strategy for querying this attribute.

For the periodic checks, the probability $\mathbf{Pr}[s]$ is:

$$\mathbf{Pr}[s] = \mathbf{Pr}^{*}_{j_0} \cdot (\prod_{y=1}^{n-1} \mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q)) \cdot \mathbf{Pr}^{k_n}_{y_{n-1} y_n}(q)$$

Where $\mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q)$ is a probability of the reputation change from the value j_y to the value j_{y+1} taking the set of values k_y on the interval between changes, $\mathbf{Pr}^{*}_{j_0}$ is a probability that the attribute will have the certain good value at the first check.

$$\mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q) = \prod_{z=1}^{q-1} \mathbf{Pr}_{f_z f_{z+1}}$$

$\mathbf{Pr}_{f_z f_{z+1}}$ is an element of the matrix **Prob** *of one-time transition probabilities. Clearly $f_1 = j_y$, $f_q = j_{y+1}$, and k_y determines concrete values of $\{f_2, \ldots, f_{q-1}\}$. There are m^{q-2} possible $\mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q)$ if j_y and j_{y+1} are fixed.*

For aperiodic checks the computations are similar to ones above. However, since the reputation is modeled as a Markov chain, the probabilistic behavior of the reputation significantly depends on the current state of the random process. Thus, q now depends on the current value of the reputation and the AUCS selects a specific interval q_i on the basis of the last observed value $\tilde{A}_{i-1} = a_{i-1}$. Markov process quickly converges to a steady state. Therefore, the AUCS considers $q_i < q_{max}$, where q_{max} is the number of changes when the distribution of probabilities differs from the steady state distribution by some small value ϵ. For a more detailed description see Appendix A.3.

We performed several simulations to check the values provided by our theoretical equation. To evaluate the aperiodic checks we carried out an exhaustive search of the optimal lengths of intervals between checks and found the values $q_1 = 7$, $q_2 = 4$, and $q_3 = 1$ if the current observed value is "general", "normal", and "suspicious" respectively. The computations of q_i are only required ones the policy is deployed in the system.

The results of the simulations are shown in Figure 8.4. Since there is no single interval for aperiodic checks, we display aperiodic checks as a straight line.

First, both periodic and aperiodic checks are close enough to the theoretical curves. Second, the simulations illustrate our proposition regarding the fact that aperiodic checks are at least as cost-effective as periodic ones. In our example, aperiodic checks are about 15% more cost-effective then periodic checks. Third, the analysis of the periodic checks shows that the average cost of the session has the

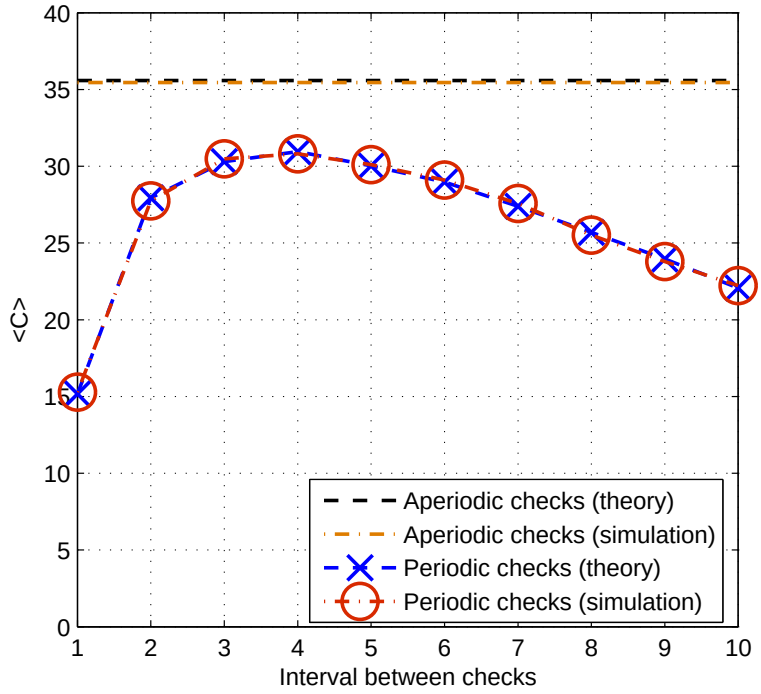

Figure 8.4: Cost-effective enforcement of usage control

maximum value when the interval between checks is 4. *The smaller interval is ineffective because we pay more for requesting an attribute. The bigger intervals are ineffective, because the system misses more policy violations.*

8.7 Architecture for Policy Enforcement under Uncertainties

The architecture of the reference monitor should be tuned to capture the presence of uncertainties. Figure 8.5 shows the overall architecture (in a distributed environment each component can run on a different host):

- Policy Enforcement Point (PEP) is a component which intercepts invocations of security-relevant access requests, suspends them before starting, queries the PDP for access decisions, enforces obtained decisions by resuming suspended requests, and interrupts ongoing accesses when the policy violation occurs.

- Policy Decision Point (PDP) is a component which evaluates security policies and produces the access decision.

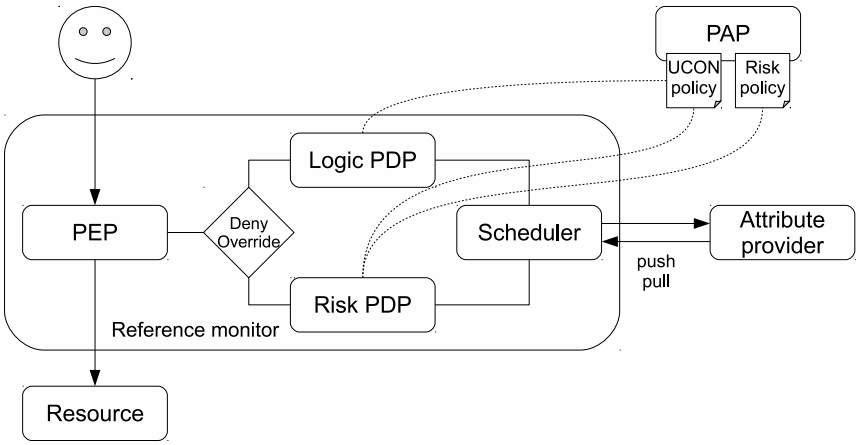

Figure 8.5: Architecture of reference monitor

- Attribute Provider (AP) is a component which manages attributes and knows their real values.
- Policy Administrative Point (PAP) is a component which provides and governs security policies.

The main novelty of the policy enforcement under uncertainties is that the PDP also consists of several components: the *logic PDP*, the *risk PDP*, and the *scheduler*.

The logic PDP behaves as a usual PDP [97] and evaluates logical predicates over observed attributes. The risk PDP computes all uncertainties associated with observed attributes and runs the random experiment to get a value of a random variable Y. If $Y = 1$, the risk PDP outputs "grant" and "deny" otherwise. Decisions of both PDPs are combined as "deny-override", i.e. the PDP sends "grant" to the PEP only if both the logical and risk PDPs grant the access.

Policies used by the logical PDP can be written in any appropriate language to formalize the UCON model, e.g., a POLPA language [15]. The risk PDP additionally uses risk policies, i.e., a cost matrix, specifications of stochastic processes which model the behavior of attributes, and values of thresholds. In fact, security and risk policies can be provided by different parties (security administrators).

The scheduler is managed by the risk PDP and is responsible to collect and process attribute observations. When a new attribute value is pushed to the reference monitor, the scheduler transforms it into the proper format and triggers both PDPs to reevaluate the access decision. During usage control, the scheduler usually pulls new attributes from the AP and then again processes them and forwards these observations to the PDPs. The risk PDP is responsible for informing the scheduler about how and when attribute queries should be initiated: either periodically or aperiodically.

Chapter 9

Validation of Contributions

We recall the main *goals* of the thesis:

1. *propose a formal model for security metrics and risk*;
2. *allow evaluation of security of complex services on the basis of different security metrics*;
3. *enable continuous reevaluation of security in services during their exploitation*;
4. *take into account possible uncertainties of the data used for security decisions in services.*

In Chapter 2, the goals were divided into several objectives. In this section, we analyze whether the objectives were successfully accomplished. Successful accomplishing the objectives should indicate that the main goals of the thesis were achieved.

9.1 Formal Model for Security Metrics

- *The model should be capable of formalizing general quantitative security metrics.* We proposed a model where a system and an attacker are considered as communicating processes. For formal definitions of metrics, we considered a system applied out of the context, i.e., we did not take into account the attacker parameters and impact of the attacks. We defined formally the following security metrics: number of attacks, minimal cost of attack, minimal length of attack, maximal probability of successful attack, attack surface.
- *The model should be capable of formalizing risk.* Formal definition of risk required us to extend the initial model adding the context to the system. We enhanced the initial model with parameters of the attacker which are a tuple of attacker's goal, set of attacks available to the attacker, her skills, tangible and intangible resources. We checked our definition of risk against the definition of risk accepted in the area and showed that the definitions are equal.

- *The model should help to analyze several general quantitative security metrics.* We performed simple analysis of validity of general security metrics. We showed that all considered metrics are valid and, thus, can be used for the evaluation of security in services. Although the model is supposed to allow the analysis of relations between metrics and between metrics and risk, we did not performed the analysis in the thesis.

9.1.1 Refined Model of the Attacker

During the work on the formal model for security metrics and risk we understand that they depend on the model of the attacker. We proposed the refined model of the attacker which is an additional contribution into the *Goal 1.* The model considered attackers with limited resources, partial knowledge about the systems and adaptive behavior. The model is based on the Markov Decision Processes theory. We proposed an algorithm for the simulation of adaptive attacker's behavior. While we suppose that the model should allow finer-grained evaluation of security, additional checks of the model usefulness are required. Moreover, methods for the computation of metrics on the basis of adaptive attacker behavior should be defined.

9.2 Security Evaluation of Complex Services

- *The method should allow selecting the most secure design of a complex service.* We proposed a method for decomposition of a complex service described in BPMN notation into a design graph which represents possible designs of the complex service. The decomposition is done under several assumptions on representing BPMN activities in the graph. We analyze the design graph using semiring-based methods for the evaluation of security. The analysis helps a service orchestrator to select the most secure design of the complex service.

- *The method should allow the evaluation of complex services on the basis of different metrics.* We defined several security metrics as semirings algebraic structures. The semirings allow utilizing a single algorithms for the analysis of the design graphs regardless what security metric is selected for the analysis. This allowed the analysis of complex services using different metrics.

- *The method should allow mapping between different metrics.* We exploited mappings between semirings to make possible mappings between security metrics. We described what conditions should be fulfilled by the metrics to make correct mappings. We reviewed limitations of the method based on mappings between semirings.

9.3 Continuous Reevaluation of Security in Services

- *The UCON model should be adapted for the continuous reevaluation of services.* We determined the interactions between a service consumer and a service provider in terms of the UCON model. We considered the service provider as subject that is trying to access objects which are assets of a service consumer. We considered security preferences of the service consumer and security preferences of service provider as attribute-based UCON policies. We showed how the UCON model can be used for reevaluation and security decisions during the exploitation of services.

- *The method should help a service consumer to evaluate and continuously reevaluate security of services.* We proposed a method for a qualitative risk evaluation on the basis of security requirements of a service consumer and security preferences of a service provider. The risk value is used to select a service with a proper security and to continuously reevaluate security of the service to make decisions about interactions with services. We considered only two security decisions for the case when the service does not satisfy security requirement of the consumer. First, the service consumer can revoke the access to the assets to the current service and select a different service with better security. Second, the service consumer may ask the current service provider to adjust the security level of the service. Currently, the method does nor work with complex services.

- *The method should be useful for a service provider.* The service provider uses risk computed by her clients to understand whether the security of the service should be improved. We formulated the risk mitigation strategy as a knapsack problem. This formulation allows exploiting existing algorithms for selecting security controls for the risk mitigation. Mitigating the risk helps the service provider to keep her service consumers and to attract new ones.

9.4 Impact of Uncertainties on Security Decision Making

- *We should identify uncertainties that may impact decision making in the UCON.* We listed unintentional and intentional uncertainties that can impact decision making. The uncertainties are: freshness, correctness (unintentional ones) and trustworthiness (intentional one).

- *The method should allow decision making under uncertainties.* We defined a correct policy enforcement for the UCON in absence of uncertainties. We

proposed cost-effective threshold and flip coin strategies for the decision making in presence of unintentional uncertainties. In these strategies, we evaluate possible monetary outcomes of the decision considering the case when fresh values of attributes are not available. To illustrate our theory, we made several simulations assuming the attribute changes follow Markov property.

- *The method should allow computing a strategy for attributes checking.* We developed periodic and aperiodic strategies for the attributes query. The strategies are cost-effective because they minimizing possible losses connected with impact of missed attribute values due to non-continuous checks. We considered periodic and aperiodic strategies for checking attributes values. We implemented the strategies as a software prototype assuming that the attributes behavior follows Markov property. We used the prototype to compare periodic and aperiodic checks. The comparison as well as our theoretical findings indicated that aperiodic checks are at least as effective as periodic ones.

- *The method should propose an architecture for the cost-effective enforcement of usage control policies under uncertainties.* We proposed a modification for the architecture for the enforcement of the UCON. We introduced three additional modules: a logic policy decision point, a risk policy decision point and a scheduler. New modules enabled the cost-effective decision making and attribute values checking in the UCON.

Chapter 10

Concluding Remarks

This thesis discussed the framework for quantitative evaluation and reevaluation of security in services. We remind the main contributions to conclude the thesis.

We proposed the formal model for the definition and analysis of general quantitative metrics and risk. We used the model to define several general quantitative metrics and risk. The definition allowed us to analyze validity of several security metrics: number of attacks, minimal cost of attack, minimal length of attack, maximal probability of successful attack, and attack surface metric. The analysis showed that all metrics are valid and, thus, can be exploited for evaluation of security in services. However, since the metrics are general they may be exploited also for the evaluation of other computer systems, for instance, computer networks.

The analysis of security metrics showed that metrics depend on the behavior of the attacker. While current models of the attacker are simplified, we introduced a refined model for the attacker's behavior. The model is based on the Markov Decision Processes theory. The model takes into account that the attacker has limited resources to attack the system. Moreover, the attacker has partial knowledge about a target system. The model allows the attacker to change her behavior during the attack, i.e., to behave adaptively. We presented an algorithm for the simulation of adaptive attacker's behavior. We suppose that the refined model of the attacker should allow a finer-grained evaluation of security.

The thesis presented a semiring-based method for the evaluation of security of complex services on the basis of different metrics and for the selection of the most secure design of complex services. We proposed a simplified decomposition of complex services described using the BPMN notation into a design graph. Then, the method exploits a semiring-based algorithm for the evaluation of design graphs. Moreover, the method allows semirings-based mappings between metrics. Such mappings are useful for the case when simple services are evaluated using different metrics. We defined several security metrics as semirings to exemplify our approach. The semirings can also be used for the definition of aspects not related to security, for instance, latency and trust. Hence, the method may be useful for the analysis of non-security QoS parameters of the complex services.

We proposed a method for the continuous reevaluation of the security in services. The method is based on the UCON enhanced with qualitative risk assessment. The method suits for the evaluation of security of a single service in the SOA. The method is useful both for a service provider and a service consumer. The method helps the service consumer to select a service according to security requirements and then to control whether the security requirements are satisfied during the interactions. The service provider can adjust security preferences using the method to provide a better service to her customers.

We addressed the issue of the UCON when a decision have to be made under uncertainties. We described the uncertainties that may impact decision making. We proposed threshold and flip coin strategies for cost-effective decision making. The threshold strategy is more cost-effective than the flip coin one. We introduced periodic and aperiodic strategies of attribute checks for usage control. Aperiodic checks are equally or more cost-effective than periodic checks. We checked our theoretical findings using simulations describing changes of an attribute as a random process that possesses Markov property. We proposed an architecture for the cost-effective enforcement of the UCON policies. While we considered the decisions making for the UCON in services, the problem of decision making under uncertainties is relevant for the UCON model applied in any system. Thus, the results of our work may be useful for any system where the UCON is applied.

10.1 Future Work

The results of the thesis give a rich background for the future work.

10.1.1 Formal Model For Security Metrics

A possible exploitation of the formal model is to analyze connections between security metrics and between security metrics and risk. The analysis can help to understand whether security metrics can be used interchangeably. Moreover, the analysis can help to understand how the metrics contributes into the risk. Such analysis can facilitate the security evaluation.

10.1.2 Modeling Adaptive Attacker's Behavior

The model of the attacker can be improved in several ways. The notion of decreasing tangible resources of the attacker can be introduced into the model. In this case, the attacker will spend the resources not only before the attack, but also during the attack. A better way of representing the time is necessary, i.e., in a real situation each attack step needs different amount of time. The model should be able to work with zero-day vulnerabilities which currently are not taken into account.

10.1.3 Security Evaluation of Complex Services

The model for the evaluation of complex services can be extended as follows. First, the assumptions on the transformation of a business process into the design graph can be relaxed, because an orchestrator often does not have an information required to avoid choices and loops. Second, explicit definitions of mappings between security metrics on the basis of their formal relations are needed.

10.1.4 Continuous Reevaluation of Security in Services

There is a number of ways how the main idea of continuous reevaluation of security in services can be elaborated. Detailed analysis and formalization of possible UCON policies in the SOA can facilitate the mapping of security requirements and security preferences. Also the method can be modified to work with complex services. Moreover, the quantitative risk can be exploited for evaluation and reevaluation of security.

10.1.5 Enforcement of Usage Control Policies under Uncertainties

For the decision making under uncertainties, the computation of probabilities of policy failure under intentional uncertainty is needed. Current state of the art can be applied only to static attributes and the model can be extended for more general cases, i.e. for the case of mutable attributes.

Bibliography

[1] Business process execution language for web services version 1.1, 2003. Was available via `http://public.dhe.ibm.com/software/dw/specs/ws-bpel/ws-bpel.pdf` on 13/04/2011.

[2] Systems security engineering capability maturity model version 3.0, June 2003.

[3] Cramm user guide, issue 5.1, July 2005.

[4] Business process model and notation (bpmn) version 2.0, January 2011. Was available via `http://www.omg.org/spec/BPMN/2.0` on 19/05/2011.

[5] M. Alam, X. Zhang, M. Nauman, T. Ali, and J.-P. Seifert. Model-based behavioral attestation. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, 2008.

[6] C. J. Alberts and A. J. Dorofee. OCTAVE Criteria. Technical Report CMU/SEI-2001-TR-016, CERT, December 2001.

[7] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002.

[8] A. Andrieux, K. Czajkowski, A. Dan, a. L. Kate Keahey, J. P. Toshiyuki Nakata, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification (ws-agreement), March 2007.

[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[10] A. Arnes, K. Sallhammar, K. Haslum, T. Brekne, M. E. G. Moe, and S. J. Knapskog. Real-time risk assessment with network sensors and intrusion detection systems. In *Proceeding of the International Conference on Computational Intelligence and Security*, 2005.

[11] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Publishing, 2003.

[12] B. Aziz, A. Arenas, F. Martinelli, I. Matteucci, and P. Mori. Controlling usage in business process workflows through fine-grained security policies. In *Proceedings of the 5th International Conference on Trust and Privacy in Digital Business*, 2008.

[13] B. Aziz, S. N. Foley, J. Herbert, and G. Swart. Reconfiguring role based access control policies using risk semantics. *High Speed Networks Journal*, 15(3):261–273, 2006.

[14] S. J. Bae, W. Kuo, and P. H. Kvam. Degradation models and implied lifetime distributions. *Reliability Engineering & System Safety*, 92:601–608, 2007.

[15] F. Baiardi, F. Martinelli, P. Mori, and A. Vaccarelli. Improving grid service security with fine grain policies. In *Proceedings of On the Move to Meaningful Internet System Workshop*, 2004.

[16] F. Baiardi, C. Telmon, and D. Sgandurra. Hierarchical, model-based risk management of critical infrastructures. *Reliability Engineering and System Safety*, 94(9):1403–1415, 2009.

[17] D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In *Quality of Protection: Security Measurements and Metrics*. 2006.

[18] S. P. Bennett and M. P. Kailay. An application of qualitative risk analysis to computer security for the commercial sector. In *Proceedings of the 8th Annual Computer Security Applications Conference*, 1992.

[19] N. Bieberstein, S. Bose, K. J. Marc Fiammante, and R. Shah. *Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap*. IBM Press, 2006.

[20] S. Bistarelli, P. Codognet, and F. Rossi. Abstracting soft constraints: Framework, properties, examples. *Artificial Intelligence*, 139:175–211, 2002.

[21] S. Bistarelli, F. Martinelli, and F. Santini. A semantic foundation for trust management languages with weights: An application to the rt family. In *Proceedings of the 5th International Conference on Autonomic and Trusted Computing*, 2008.

[22] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44:201–236, 1997.

[23] A. Blyth and P. Thomas. Performing real-time threat assessment of security incidents using data fusion of ids logs. *Journal of Computer Security*, 14(6):513–534, 2006.

[24] M. Bouzeghoub and V. Peralta. A framework for analysis of data freshness. In *Proceedings of the 2004 International Workshop on Information Quality in Information Systems*, 2004.

[25] F. Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen. Model-based security analysis in seven steps – a guided tour to the coras method. *BT Technology Journal*, 25(1):101–117, 2007.

[26] I. Brandic, D. Music, and S. Dustdar. Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services. In *Proceedings of the 6th International Conference Industry Session on Grids Meets Autonomic Computing*, 2009.

[27] R. Breu, F. Innerhofer-Oberperfler, and A. Yautsiukhin. Quantitative assessment of enterprise security system. In *Proceedongs of the 3rd International Conference on Availability, Reliability and Security*, 2008.

[28] S. A. Butler. Security attribute evaluation method: a cost-benefit approach. In *Proceedings of the 24th International Conference on Software Engineering*, 2002.

[29] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[30] V. Casola, A. R. Fasolino, N. Mazzocca, and P. Tramontana. An ahp-based framework for quality and security evaluation. In *International Conference on Computational Science and Engineering*, 2009.

[31] V. Casola, A. Mazzeo, N. Mazzocca, and M. Rak. A sla evaluation methodology in service oriented architectures. In *Proceedings of Quality of Protection Workshop*, 2005.

[32] F. Cheng, D. Gamarnik, N. Jengte, W. Min, and B. Ramachandran. Modelling operational risks in business process. Technical Report RC23872, IBM, July 2005.

[33] P.-C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. 2007.

[34] K. Clark, J. Dawkins, and J. Hale. Security risk metrics: Fusing enterprise objectives and vulnerabilities. In *Proceedings of the 6th Annual IEEE SMC Information Assurance Workshop*, 2005.

[35] F. Cohen. Managing network security – part 5: Risk management or risk analysis. *Network Security*, 1997:15–19, 1997.

[36] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, 2(6):86–93, 2002.

[37] P. Degano, G.-L. Ferrari, and G. Mezzetti. On quantitative security policies. In *Proceedings of the 11th International Conference on Parallel Computing Technologies*, 2011.

[38] R. Dewri, I. Ray, I. Ray, and D. Whitley. Security provisioning in pervasive environments using multi-objective optimization. In *Proceedongs of the 13th European Symposium on Research in Computer Security*, 2008.

[39] N. N. Diep, L. X. Hung, Y. Zhung, S. Lee, Y.-K. Lee, and H. Lee. Enforcing access control using risk assessment. In *Proceedings of the 4th European Conference on Universal Multiservice Networks*, 2007.

[40] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[41] N. Dimmock. How much is "enough"? risk in trust-based access control. In *Proceedings of the 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2003.

[42] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*, 2004.

[43] G. Dobson and A. Sanchez-Macian. Towards unified qos/sla ontologies. In *Proceedings of the 3rd International Workshop on Semantic and Dynamic Web Processes*, 2006.

[44] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[45] F. Farahmand, S. B. Navathe, P. H. Enslow, and G. P. Sharp. Managing vulnerabilities of information systems to security incidents. In *Proceedings of the 5th international conference on Electronic Commerce*, 2003.

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

[47] R. Fredriksen, M. Kristiansenand, B. A. Granand, K. Stølen, T. A. Opperud, and T. Dimitrakos. The coras framework for a model-based risk management process. In *Proceedings of the 21st International Conference on Computer Safety, Reliability and Security*, 2002.

[48] J. W. Freeman, T. Darr, and R. B. Neely. Risk assessment for large heterogeneous systems. In *Proceedings of the 13th Annual Computer Security Applications Conference*, 1997.

[49] L. Gallon and J.-J. Bascou. Cvss attack graphs. In *Proceedings of 7th International Conference on Signal-Image Technology and Internet-Based Systems*, 2011.

[50] A. Gehani and G. Kedem. Rheostat: Real-time risk management. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, 2004.

[51] G. Geri, H. S. Hilde, and R. Indrakshi. Aspect-oriented risk driven development of secure applications. In *Proceedings of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2006.

[52] Global Grid Forum. *Open Grid Services Architecture Glossary of Terms Version 1.5*, 2006.

[53] L. A. Gordon and M. P. Loeb. *Managing Cybersecurity Resources: a Cost-Benefit Analysis*. McGraw Hill, 2006.

[54] J. Gould, M. Glossop, and A. Ioannides. Review of hazard identification techniques. Technical Report HSL/2005/58, Health and Safety Executive, 2000.

[55] B. A. Gran, R. Fredriksen, and A. P.-J. Thunem. An approach for model-based risk assessment. In *Proceedings of the 23rd International Conference on Computer Safety, Reliability and Security*, 2004.

[56] L. Grunske and D. Joyce. Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles. *Journal of Systems and Software*, 81(8):1327–1345, 2008.

[57] S. T. Halkidis, N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides. Architectural risk analysis of software systems based on security patterns. *IEEE Transactions on Dependable and Secure Computing*, 5(3):129–142, 2008.

[58] J. Hallberg, A. Hustad, A. Bond, M. Peterson, and N. Pahlsson. System it security assessment. Technical Report FOI-R-1468-SE, Swedish Defence Research Agency, 2004.

[59] Y. Han, Y. Hori, and K. Sakurai. Security policy pre-evaluation towards risk analysis. In *Proceedings of the 2008 International Conference on Information Security and Assurance*, 2008.

[60] K. Haslum and A. Arnes. Multisensor real-time risk assessment using continuous-time hidden markov models. In *Proceeding of the International Conference on Computational Intelligence and Security*, 2007.

[61] R. Henning. Security service level agreements: quantifiable security for the enterprise? In *Proceedings of the 1999 Workshop on New Security Paradigms*, 2000.

[62] D. S. Herrmann. *Complete Guide to Security and Privacy Metrics: Measuring Regulatory Compliance, Operational Resilience, and ROI*. Auerbach Publications, 2007.

[63] I. Hogganvik and K. Stølen. A graphical approach to risk identification, motivated by empirical investigations. In *Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems*, 2006.

[64] S. H. Houmb and G. Georg. The aspect-oriented risk-driven development (aordd) framework. In *Proceedings of the International Conference on Software Development (SWDC/REX*, 2005.

[65] M. Howard. Fending off future attacks by reducing attack surface, February 2003. Was available via `http://msdn.microsoft.com/en-us/library/ms972812.aspx`.

[66] O. C. Ibe. *Fundamentals of Applied Probability and Random Processes*. Elsevier Academic Press, 2005.

[67] O. C. Ibe. *Markov processes for stochastic modeling*. Elsevier Academic Press, 2009.

[68] F. Innerhofer-Oberperfler and R. Breu. Using an enterprise architecture for it risk management. In *Proceedings of the Information Security South Africa from Insight to Foresight Conference*, 2006.

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

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

[71] ISO/IEC. *ISO/IEC 27001:2005 Information technology – Security techniques – Specification for an Information Security Management System*, 2005.

[72] M. C. Jaeger, G. Rojec-Goldmann, and G. Mühl. Qos aggregation in web service compositions. In *Proceedongs of the 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services*, 2005.

[73] W. Jansen. Directions in security metric research. Technical Report NISTIR 7564, National Institute of Standards and Technology, 2009. Was available via `http://csrc.nist.gov/publications/nistir/ir7564/nistir-7564_metrics-research.pdf` on 28/04/2010.

[74] A. Jaquith. *Security metrics: replacing fear, uncertainty, and doubt*. Addison-Wesley, 2007.

[75] S. Jha, O. Sheyner, and J. M. Wing. Minimization and reliability analyses of attack graphs. Technical Report CMU-CS-02-109, Carnegie Mellon University, 2002.

[76] S. Jha, O. Sheyner, and J. M. Wing. Two formal analys s of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, 2002.

[77] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Transactions on Software Engineering*, 23:235–245, 1997.

[78] A. Josang, D. Bradley, and S. J. Knapskog. Belief-based risk analysis. In *Proceedings of the 2nd Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, 2004.

[79] B. Karabacak and I. Sogukpinar. Isram: information security risk analysis method. *Computers & Security*, 24(2):147–159, 2005.

[80] B. Kitchenham, S. L. Pfleeger, and N. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 12(21):929–944, 1995.

[81] I. Kramosil and J. Michalek. Fuzzy metrics and statistical metric spaces. *Kybernetica*, 11(5):336–344, 1974.

[82] L. Krautsevich. Parametric attack graph construction and analysis. In *Proceedings of Doctoral Symposium of International Symposium on Engineering Secure Software and Systems 2012*, pages 29–34. CEUR-WS.org, 2012.

[83] L. Krautsevich, A. Lazouski, F. Martinelli, P. Mori, and A. Yautsiukhin. Usage control, risk and trust. In *Proceedings of 7th International Conference Trust, Privacy and Security in Digital Business*, pages 1–12. Springer, 2010.

[84] L. Krautsevich, A. Lazouski, F. Martinelli, P. Mori, and A. Yautsiukhin. Integration of quantitative methods for risk evaluation within usage control policies. In *Proceedings of 22nd International Conference on Computer Communications and Networks*. IEEE, 2013.

[85] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Influence of attribute freshness on decision making in usage control. In *Proceedings of 6th Workshop on Security and Trust Management*, pages 35–50. Springer, 2010.

[86] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Risk-aware usage decision making in highly dynamic systems. In *Proceedings of 5th International Conference on Internet Monitoring and Protection*, pages 29–34. IEEE, 2010.

[87] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Risk-based usage control for service oriented architecture. In *Proceedings of 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 641–648. IEEE, 2010.

[88] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Cost-effective enforcement of ucona policies. In *Proceedings of 6th International Conference on Risk and Security of Internet and Systems*, pages 1–8. IEEE, 2011.

[89] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Cost-effective enforcement of access and usage control policies under uncertainties. *IEEE Systems Journal, Special Issue on Security and Privacy in Complex Systems*, 7(2):223–235, 2013.

[90] L. Krautsevich, A. Lazouski, P. Mori, and A. Yautsiukhin. Quantitative methods for usage control, 2012. Presented at the International Workshop on Quantitative Aspects in Security Assurance.

[91] L. Krautsevich, F. Martinelli, C. Morisset, and A. Yautsiukhin. Risk-based auto-delegation for probabilistic availability. In *Proceedings of 4th International Workshop on Autonomous and Spontaneous Security*, pages 206–220. Springer, 2011.

[92] L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Formal approach to security metrics: what does "more secure" mean for you? In *Proceedings of 4th European Conference on Software Architecture: Companion Volume*, pages 162–169. ACM, 2010.

[93] L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Formal analysis of security metrics and risk. In *Proceedings of 5th Workshop on Information Security Theory and Practice of Mobile Devices in Wireless Communication*, pages 304–319. Springer, 2011.

[94] L. Krautsevich, F. Martinelli, and A. Yautsiukhin. A general method for assessment of security in complex services. In *Proceedings of 4th European Conference ServiceWave*, pages 153–164. Springer, 2011.

[95] L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Towards modelling adaptive attacker's behaviour. In *Proceedings of 5th International Symposium on Foundations and Practice of Security*. Springer, 2012.

[96] R. Krishnan, J. Niu, R. Sandhu, and W. H. Winsborough. Stale-safe security properties for group-based secure information sharing. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering*, 2008.

[97] A. Lazouski, F. Martinelli, and P. Mori. Usage control in computer security: A survey. *Computer Science Review*, 4:81–99, 2010.

[98] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. Model-based security metrics using adversary view security evaluation (advise). In *Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems*, 2011.

[99] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders. Adversary-driven state-based system security evaluation. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, 2010.

[100] Y. Li, H. Sun, Z. Chen, J. Ren, and H. Luo. Using trust and risk in access control for grid environment. In *Proceedings of the 2008 International Conference on Security Technology*, 2008.

[101] U. Lindqvist and E. Jonsson. A map of security risks associated with using cots. *Computer*, 31(6):60–66, 1998.

[102] B. B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K. S. Trivedi. A method for modeling and quantifying the security attributes of intrusion tolerant systems. *Performance Evaluation Journal*, 1-4(56):167–186, 2004.

[103] P. Manadhata and J. Wing. Measuring a system's attack surface. Technical Report CMU-TR-04-102, Carnegie Mellon University, 2004.

[104] P. K. Manadhata, K. M. C. Tan, R. A. Maxion, and J. M. Wing. An approach to measuring a systems attack surface. Technical Report CMU-CS-07-146, School of Computer Science, Carnegie Mellon University, 2007.

[105] D. Marchignoli and F. Martinelli. Automatic verification of cryptographic protocols through compositional analysis techniques. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, 1999.

[106] F. Martinelli. Analysis of security protocols as open systems. *Theoretical Computer Science*, 290(1):1057–1106, 2003.

[107] F. Martinelli, P. Mori, and A. Vaccarelli. Towards continuous usage control on grid computational services. In *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, 2005.

[108] F. Massacci and A. Yautsiukhin. An algorithm for the appraisal of assurance indicators for complex business processe. In *Proceedings of the 2007 ACM Workshop on Quality of Protection*, 2007.

[109] F. Massacci and A. Yautsiukhin. Modelling of quality of protection in outsourced business processes. In *Proceedings of the 3rd International Symposium on Information Assurance and Security*, 2007.

[110] R. W. McGraw. Risk-adaptable access control (radac). Was available via `http://csrc.nist.gov/news_events/privilege-management-workshop/radac-Paper0001.pdf` on 16/08/09.

[111] P. Mell and T. Grance. The nist definition of cloud computing, September 2011. Available on-line via http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf on 12/12/2012.

[112] D. A. Menasce. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[113] Microsoft. Securing windows 2000 server chapter 3: Understanding the security risk management discipline. Microsoft TechNet Archive, November 2004.

[114] J. C. Mitchell, A. Ramanathan, A. Scedrovb, and V. Teaguea. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Computer Science*, 353(1):118–164, 2006.

[115] M. Mitzenmacher and E. Upfal. *Probability and Computing Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[116] R. A. Miura-Ko and N. Bambos. Dynamic risk mitigation in computing infrastructures. In *Proceedings of the 3rd International Symposium on Information Assurance and Security*, 2007.

[117] M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7:321–350, 2002.

[118] A. Morali, E. Zambon, S. Etalle, and P. Overbeek. It confidentiality risk assessment for an architecture-based approach. In *Proceedings of the 3rd IEEE/IFIP International Workshop on Business-driven IT Management*, 2008.

[119] R. E. Mullen. The lognormal distribution of software failure rates: application to software reliability growth modeling. In *The 9th International Symposium on Software Reliability Engineering*, 1998.

[120] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *Proceedings of the 11th International Conference on World Wide Web*, 2002.

[121] Q. Ni, E. Bertino, and J. Lobo. Risk-based access control systems built on fuzzy inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, 2010.

[122] J. Niu, R. Krishnan, J. F. Bennat, R. Sandhu, and W. H. Winsborough. Enforceable and verifable stale-safe security properties in distributed systems. Technical Report CS-TR-2011-02, University of Texas at San Antonio, 2011.

[123] S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, 2004.

[124] R. Ortalo, Y. Deswarte, and M. Kaâniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25(5):633–650, 1999.

[125] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006.

[126] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the 2nd ACM Workshop on Quality of Protection*, 2006.

[127] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of the 1st International Semantic Web Conference*, 2002.

[128] M. P. Papazoglou. Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the 4th International Conference on Web Information Systems Engineering*, 2003.

[129] J. Park and R. Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies*, 2002.

[130] C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[131] V. Peralta. Data freshness and data accuracy: A state of the art. Technical Report TR0613, Universidad de la Republica Uruguay, 2006.

[132] R. Perrey and M. Lycett. Service-oriented architecture. In *Proceedings of Symposium on Applications and the Internet Workshops*, 2003.

[133] Y. N. Pettersen. Renego patched servers: A long-term interoperability time bomb brewing. Was available online on 20/07/2012.

[134] C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms*, 1998.

[135] D. Pisinger. Algorithms for knapsack problems. Technical Report DK-2100, University of Cophenhagen, 1995.

[136] A. Pretschner, F. Massacci, and M. Hilty. Usage control in service-oriented architectures. In *Proceedings of the 4th International Conference on Trust, Privacy and Security in Digital Business*, 2007.

[137] M. L. Puterman. *Markov Decision Processes Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[138] S. Ran. A model for web services discovery with qos. *Newsletter ACM SIGecom Exchanges*, 4(1):1–10, 2003.

[139] R. L. Rardin. *Optimization in operations research*. Prentice Hall, 1997.

[140] J. Rees and J. Allen. The state of risk assessment practices in information security: An exploratory investigation. *Journal of Organizational Computing and Electronic Commerce*, 18(4):255–277, 2008.

[141] A. Refsdal and K. Stølen. Employing key indicators to provide a dynamic risk picture with a notion of confidence. In *Proceedings of the 3rd IFIP WG 11.11 International Conference on Trust Management III*, 2009.

[142] J. J. C. H. Ryan and D. J. Ryan. Performance metrics for information security risk management. *IEEE Security and Privacy*, 6(5):38–44, 2008.

[143] P. Samarati and S. D. C. di Vimercati. Access control: Policies, models, and mechanisms. In *In Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, 2000.

[144] C. Sarraute, O. Buffet, and J. Hoffmann. Pomdps make better hackers: Accounting for uncertainty in penetration testing. In *Proceedings of the 26th Conference on Artificial Intelligence*, 2012.

[145] R. Savola. Towards a security metrics taxonomy for the information and communication technology industry. In *Proceedings of the International Conference on Software Engineering Advances*, 2007.

[146] R. M. Savola. A security metrics taxonomization model for software-intensive systems. *Journal of Information Processing Systems*, 5(4):197–206, 2009.

[147] S. Schechter. How to buy better testing. In *Proceedings of the International Conference on Infrastructure Security*, 2002.

[148] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 273–284, 2002.

[149] O. Sheyner and J. M. Wing. Tools for generating and analyzing attack graphs. In *Proceedings of the 2nd International Symposium on Formal Methods for Components and Objects*, 2003.

[150] A. Singhal, T. Winograd, and K. Scarfone. Guide to secure web services. Technical report, National Institute of Standards and Technology (NIST), 2007.

[151] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of web services using semantic descriptions. In *Workshop on Web Services: Modeling, Architecture and Infrastructure*, 2002.

[152] C. Skalka, X. S. Wang, and P. Chapin. Risk management for distributed authorization. *Journal of Computer Security*, 15:447–489, 2007.

[153] A. Stewart. On risk: perception and direction. *Computers & Security*, 23:362–370, 2004.

[154] K. Stølen, F. D. Braber, T. Dimitrakos, R. Fredriksen, B. A. Gran, S.-H. Houmb, S. Lund, Y. C. Stamatiou, and J. O. Aagedal. Model-based risk assessment the coras approach, presented at the 1st itrust workshop, 2002.

[155] G. Stoneburner, A. Goguen, and A. Feringa. Risk management guide for information technology systems. Technical report, National Institute of Standards and Technology (NIST), 2002. Was available via `http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf`.

[156] P. Suppes and J. L. Zinnes. Basic measurement theory. Technical report, Institute for mathematical studies in the social science, 1962.

[157] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *Proceedings of DARPA Information Survivability Conference*, 2001.

[158] C. Tarr and P. Kinsman. The validity of security risk assessment. In *Proceedings of the 30th Annual 1996 International Carnahan Conference on Security Technology*, 1996.

[159] H. C. Tijms. *A First Course in Stochastic Models*. Wiley, 2003.

[160] S. Venkataraman and W. Harrison. Prioritization of threats using the k/m algebra. In *Proceedings of Workshop on Software Security Assurance Tools, Techniques, and Metrics*, 2005.

[161] D. Verdon and G. McGraw. Risk analysis in software design. *IEEE Security and Privacy*, 2(4):79–84, 2004.

[162] A. J. A. Wang. Information security models and metrics. In *Proceedings of the 43rd Annual Southeast Regional Conference*, 2005.

[163] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In *Proceeedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2008.

[164] L. Wang, S. Jajodia, A. Singhal, and S. Noel. k-zero day safety: Measuring the security risk of networks against unknown attacks. In *Proceedings of the 15th European Conference on Research in Computer Security*, 2010.

[165] L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29(15):2917–2933, 2006.

[166] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 2006.

[167] H. Wei, D. Frinke, O. Carter, and C. Ritter. Cost-benefit analysis for network intrusion detection systems. In *Proceedings of the 28th Annual Computer Security Conference*, 2001.

[168] C. Woody and C. Alberts. Considering operational security risk during system development. *IEEE Security and Privacy*, 5(1):30–35, 2007.

[169] L. Youseff, M. Butrico, and D. D. Silva. Toward a unified ontology of cloud computing. In *Proceedings of Grid Computing Environments Workshop*, 2008.

[170] T. Yu and K.-J. Lin. A broker-based framework for qos-aware web service composition. In *Proceedongs of the 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services*, 2005.

[171] E. Zambon, D. Bolzoni, S. Etalle, and M. Salvato. Model-based mitigation of availability risks. In *Proceedongs of the 2nd IEEE/IFIP International Workshop on Business-Driven IT Management*, 2007.

[172] E. Zambon, D. Bolzoni, S. Etalle, and M. Salvato. A model supporting business continuity auditing & planning in information systems. In *Proceedings of the 2nd International Conference on Internet Monitoring and Protection*, 2007.

[173] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, 2004.

[174] L. Zhang, A. Brodsky, and S. Jajodia. Toward information sharing: Benefit and risk access control (barac). In *Proceedings of the 7th IEEE International Workshop on Policies for Distributed Systems and Networks*, 2006.

[175] M. Zhang, B. Yang, S. Zhu, and W. Zhang. Ordered semiring-based trust establish model with risk evaluating. *International Journal of Network Security*, 8(2):101–106, 2009.

[176] X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Transactions on Information and System Security*, 8(4):351–387, 2005.

Appendix A

Computational Problems of Markov Chains

In the appendix, we briefly present solutions of several computational problems related to the Markov chains theory. We discuss in details a method for computation of transition probabilities based on discrete-time (DTMC) and continuous-time (CTMC) Markov chains.

We consider an attribute r with a discrete domain of values and assume that the attribute satisfies the Markov property, which means that a future attribute value depends only on the present value and does not depend on its previous values. We model possible changes of the attribute value with the Markov chain. The Markov chain contains states and transitions between states. The states of the chain represent the values of the attribute, and the transitions describe the changes of the attribute. The values of the attribute can be grouped into two domains: the "bad" domain V_B and the "good" domain V_G. If the attribute takes a value from the "bad" domain then the policy is violated and the usage session should be revoked. If the attribute takes a value from the "good" domain then the policy holds and the usage is continued. The states of the Markov chain can be gathered into two groups I_B and I_G respectively. The set of all values of the attribute is $V_r = V_B \cup V_G$ and the corresponding set of states is $I = I_B \cup I_G$.

We define the following variables:

- $x \in V_r$ is a value of an attribute, x_i we denote the value of the attribute in the state i;
- t_0 is the time when we know the exact value of the attribute;
- t' is the time, when we make an access decision about the usage session.

A.1 Discrete-time Markov Chain

First, we consider a stochastic process represented as a DTMC. We compute a transition probability $\mathbf{Pr}_{ij}(q)$ of a stochastic process to be in a state j if the process started from the state i and exactly q changes occur. The case is useful for the computation of the probability of a policy failure after q changes of an attribute value.

There is a vector of transition probabilities:

$$S(q) = [\mathbf{Pr}_{1j}(q)\ \mathbf{Pr}_{2j}(q)\ \ldots\ \mathbf{Pr}_{jM}(q)]$$

where $M = |V_r|$ is the number of elements in the domain V_r. $\mathbf{Pr}_{ij}(n)$ can be found using the Markov chains theory and Kolmogorov-Chapman's equation in particular [159]. Assume that we know the initial value x_i where the process starts ($q = 0$). Thus, $\mathbf{Pr}_{ii}(0) = 1$ and others are 0, i.e., $S(0) = [0\ 0\ \ldots\ 1\ \ldots\ 0]$. The value of the vectors after q changes will be:

$$S(t') = S(t_0) \times \mathbf{Prob}^q \tag{A.1}$$

where $\times$ denotes multiplication of matrices, **Prob** is a transition matrix composed by probabilities of one-time transitions from a state a (row) to a state b (column), $\mathbf{Prob}^q$ shows the matrix in power q.

Sometime we need a probability $\mathbf{Pr}_{iI_B}(q)$ of transition from the state i to the set of states I_B. In this case, the vector of states $S(q)$ is computed according to Equation A.1, the probability $\mathbf{Pr}_{iI_B}(q)$ of the transition is:

$$\mathbf{Pr}_{iI_B}(q) = \sum_{j \in I_B} S(q)[j] \tag{A.2}$$

A.2 Continuous-time Markov Chain

Now, we consider a slightly different situation when we know only the time passed from the last check of an attribute. We assume that the average time between changes of the attribute value is exponentially distributed with the rate parameter ν. This assumption allows modelling the behaviour of attribute values using CTMC, where:

- ν_i is the rate parameter of an exponential distribution for the time of jumping from a state i to another state, the value $\frac{1}{\nu_i}$ is the average life-time of the attribute in the state x_i;
- $\mathbf{Pr}_{ij}$ is the one-step transition probability (the probability that the process makes a direct jump from a state i to a state j without visiting any intermediate state).

We assume that the values ν_i and $\mathbf{Pr}_{ij}$ can be determined and adjusted using statistical methods during the analysis of the past behaviour of the process. The history of attribute changes is required for this purpose. Using ν_i and $\mathbf{Pr}_{ij}$ we can evaluate the probability of the policy violation on the basis of the following approach.

The transitions between the states are described with the infinitesimal transition rates ($q_{ij} \in Q$). The infinitesimal transition rates are defined as:

$$q_{ij} = \nu_i \cdot \mathbf{Pr}_{ij}, \forall i, j \in I \text{ and } i \neq j \tag{A.3}$$

The infinitesimal transition rates uniquely determine the rates ν_i and one-step transition probabilities $\mathbf{Pr}_{ij}$:

$$\nu_i = \sum_{\forall j \neq i} q_{ij} \tag{A.4}$$

$$\mathbf{Pr}_{ij} = \frac{q_{ij}}{\nu_i} \tag{A.5}$$

Suppose, the value of an attribute is $x_i \in V_G$ (the state is $i \in I_G$) at time t_0 and we need to find the probability of $x_j \in V_B$ ($j \in I_B$) during the period from t_0 till t'.

We apply the uniformization method to compute transient state probabilities $\mathbf{Pr}_{ij}$ [67, 159]. The uniformization method replaces a CTMC by a DTMC analogue, which is more suitable for numerical computations. The uniformisation is done by replacing the transition rates of Markov chain ν_i with a sole transition rate ν:

$$\nu \geq \nu_i, \forall i \in I \tag{A.6}$$

Usually, the following strategy is applied: $\nu = \max_{\forall \nu_i \in V} \nu_i$.

The DTMC chain makes a transition from a state with probabilities:

$$\overline{\mathbf{Pr}}_{ij} = \begin{cases} \frac{\nu_i}{\nu} \cdot \mathbf{Pr}_{ij} = \frac{q_{ij}}{\nu}, & \forall i \neq j \\ 1 - \frac{\nu_i}{\nu}, & \forall i = j \end{cases} \tag{A.7}$$

Now we have all required parameters and we can skip the mathematical proofs, which can be found here [159, pages 167-168]. The transition state probabilities are:

$$\mathbf{Pr}_{ij}(t') = \sum_{n=0}^{\infty} e^{-\nu \cdot (t' - t_0)} \cdot \frac{(\nu \cdot (t' - t_0))^n}{n!} \cdot \overline{\mathbf{Pr}}_{ij}^{(n)}, \forall i, j \in I \text{ and } t' > t_0 \tag{A.8}$$

where $\overline{\mathbf{Pr}^*}_{ij}^{(n)}$ can be recursively computed from:

$$\overline{\mathbf{Pr}}_{ij}^{(n)} = \sum_{x_k \in I} \overline{\mathbf{Pr}}_{ik}^{(n-1)} \cdot \overline{\mathbf{Pr}}_{kj}, \; n = 1, 2, \ldots \tag{A.9}$$

starting with $\overline{\mathbf{Pr}^*}_{ii}^{(0)} = 1$ and $\overline{\mathbf{Pr}}_{ij}^{(0)} = 0$ for $i \neq j$.

For fixed $t' > t_0$ the infinite series can be truncated because of the negligible impact of the residue. The truncation number U (upper limit of summation) in Formula A.8 can be chosen as:

$$U = \nu \cdot t' + c\sqrt{\nu \cdot t'} \tag{A.10}$$

for some c with $0 < c \leq c_0(\varepsilon)$, where ε is a tolerance number [159, page 169].

Formula A.8 gives a matrix of transition probabilities if time $t' - t_0$ passed. The additional summation is required if we look for a transition into a set of states I_B:

$$\mathbf{Pr}_{vl}(t') = \sum_{j \in I_B} \mathbf{Pr}^*_{ij}(t') \tag{A.11}$$

where i is an index of the initial state, and $j \in I_B$ are the forbidden states.

A.3 Convergence of a Markov Chain to the Steady State

If the attribute behaviour follows the Markov property, the probabilities of the attribute to be in a certain state converges to a stationary (steady state) distribution. We can find the number n_{st} of transitions when the distribution of probabilities differs from the steady state distribution by any small value [115].

We first define a distance $\Delta_{a_s}(t)$ between steady state distribution π and distribution of probabilities $p^t_{a_s}$ obtained when a process starts from the initial state a_s and t transitions occurred:

$$\Delta_{a_s}(t) = \|p^t_{a_s} - \pi\| = \frac{1}{2} \sum_{a_s \in \Omega_{attr}} |p^t_{a_s}(a) - \pi(a)|$$

In addition, we define:

$$\tau_{a_s}(t) = \min\{t : \Delta_{a_s}(t) \leq \epsilon\}$$
$$\tau(\epsilon) = \max_{a_s \in \Omega_{attr}} \tau_{a_s}(\epsilon)$$

where $\tau_{a_s}(t)$ is the first step when the distance between $p^t_{a_s}$ ans steady state distribution becomes lower then ϵ, and $\tau(\epsilon)$ is maximum among all possible starting states.

A general result for the convergence of Markov chains is the following. Suppose there is $\tau(c) \leq T$ for some $c < 1/2$ where T is an amount of transitions. Then:

$$\tau(\epsilon) \leq \lceil \frac{\ln \epsilon}{\ln(2c)} \rceil T \tag{A.12}$$

Thus, the distribution $p^t_{a_s}$ is indistinguishable from the steady state distribution after $\tau(\epsilon)$ steps:

$$n_{st} = \tau(\epsilon)$$