Lnu Licentiate No. 32, 2020

Design and Analysis of Self-Protection:

Adaptive Security for Software-Intensive Systems

Charilaos Skandylas

LINNAEUS UNIVERSITY PRESS

Design and Analysis of Self-Protection: Adaptive Security for Software-Intensive Systems

Licentiate Thesis

Charilaos Skandylas

DESIGN AND ANALYSIS OF SELF-PROTECTION: ADAPTIVE SECURITY FOR SOFTWARE-INTENSIVE SYSTEMS
Licentiate Thesis, Department of Computer Science and Media Technology, Linnaeus University, Växjö, 2020

Lnu Licentiate No. 32, 2020
ISBN: 978-91-89283-22-0 (tryckt) 978-91-89283-23-7 (pdf)
Published by: Linnaeus University Press, 351 95 Växjö
Printed by: Copycenter, Linnaeus University, Växjö

Abstract

Today's software landscape features a high degree of complexity, frequent changes in requirements and stakeholder goals, and uncertainty. Uncertainty and high complexity imply a threat landscape where cybersecurity attacks are a common occurrence, while their consequences are often severe. Self-adaptive systems have been proposed to mitigate the complexity and frequent degree of change by adapting at run-time to deal with situations not known at design time. They, however, are not immune to attacks, as they themselves suffer from high degrees of complexity and uncertainty. Therefore, systems that can dynamically defend themselves from adversaries are required. Such systems are called self-protecting systems and aim to identify, analyse and mitigate threats autonomously. This thesis contributes two approaches towards the goal of providing systems with self-protection capabilities.

The first approach aims to enhance the security of architecture-based self-adaptive systems and equip them with (proactive) self-protection capabilities that reduce the exposed attack surface. We target systems where information about the system components and its adaptation decisions is available, and control over its adaptation is also possible. We formally model the security of the system and provide two methods to analyze its security that help us rank adaptations in terms of their security level: a method based on quantitative risk assessment and a method based on probabilistic verification. The results indicate an improvement to the system security when either of our solutions is employed. However, only the second method can provide self-protecting capabilities. We have identified a direct relationship between security and performance overhead, i.e., higher security guarantees impose analogously higher performance overhead.

The second approach targets open decentralized systems where we have limited information about and control over the system entities. Therefore, we attempt to employ decentralized information flow control mechanisms to enforce security by controlling interactions among the system elements. We extend a classical decentralized information flow control model by incorporating trust and adding adaptation capabilities that allow the system to identify security threats and self-organize to maximize the average trust between the system entities. We arrange entities of the system in trust hierarchies that enforce security policies among their elements and can mitigate security issues raised by the openness and uncertainty in the context and environment, without the need for a trusted central controller. The experiment results show that a reasonable level of trust can be achieved and at the same time confidentiality and integrity can be enforced with a low impact on the throughput and latency of messages exchanged in the system.

Keywords: Self-Protection, Security Analysis, Self-Adaptation

Acknowledgments

I would like to first and foremost thank and express my sincere gratitude to my main supervisor, Narges Khakpour for her guidance, support, sharing of knowledge and experience, and patience over the first few years of my PhD studies. I would further like to thank my assistant supervisor Jesper Andersson, for his support and invaluable input to my research. I would finally like to thank my fellow PhD students and all other academic colleagues that I have learned so much from during this journey.

Charilaos Skandylas
Växjö, Sweden
November 2020

Table of Contents

Chapter

Introduction

The basic services of society have undergone a radical change, a digital transformation, in the last fifty years. The transformation is an ongoing process that, besides adding value for society through digitalization, makes society more sensitive to cyber-attacks, which pose a constant threat to society. Cybersecurity attacks have become commonplace, and their consequences are increasingly dire as they may impact all parts of human life, from everyday activities to critical systems that can have disastrous or fatal ramifications when compromised. The cyberattacks grow in sophistication and scale as perpetrators develop new strategies, tactics, and techniques. A continuously changing threat, where the next modus operandi may be unknown, poses a challenge for society to promptly adapt strategies and employ new tactics and techniques to defend or mitigate such attacks.

We may characterize the challenge by *response time* and *adaptivity*. Most organizations cannot fund organizational units responsible for continuously monitoring external threats and attacks and rapidly developing and deploying countermeasures. We find such units in the military, socially critical institutions, and in large companies. It is evident that a system's security challenges during its life-cycle are numerous and changing. The changes are due to *internal evolution*, e.g., the system itself evolves, and *external evolution* of attack strategies, tactics, and techniques. The internal and external evolution will take different directions, directions that are impossible to predict in advance. Hence, the evolution generates uncertainty concerning how to strategize a system's cyber-defense, including which tactics to employ and techniques to implement.

Uncertainty becomes a significant factor when studying a system's security, as multiple security characteristics are volatile. These characteristics include the system's attack surface, the known vulnerabilities of its components, and the attackers' capabilities. Moreover, as mentioned above, the attackers' toolset and capabilities develop, which warrants that a system that claims to protect itself from attacks must include *self-adaptation capabilities* to provide adequate defenses during its life-cycle. Such capabilities can mitigate threats and the inherent uncertainty efficiently, enabling systems to change security functionality and add defenses or reduce their attack surface to become less vulnerable to attacks.

Self-adaptive systems, i.e., systems that dynamically change their structure or behavior at run-time to cope with uncertainties in their environment and their

dynamic nature, have been often employed to counteract complexity, uncertainties in the environment and changing goals. The class of self-adaptive systems that are able to defend themselves from or mitigate the effects of attacks is called *self-protecting systems*. Self-protecting systems dynamically alter their capabilities at run-time to either provide defenses that can stop or lessen the consequences of attacks or alter their structure and/or behavior to proactively defend by making the attacks impossible or less likely. Self-adaptation in this case can aid in mitigating uncertainties in the security concerns of a system and provide timely countermeasures to complex and frequent attacks.

Self-adaptive systems however are not immune to attacks themselves. In particular, given the complexity and evolving nature of these systems, adversaries that target those systems can exploit their adapting nature to facilitate attacks that are often not possible on contemporary non-adaptive systems. These attacks are often based on the changing attack surface and further information exposed during a self-adaptive system's run-time. As the system adapts from one state to the next, an adversary can make use of information or capabilities gained by exploiting the previous state to compromise the security of the next state of a self-adaptive system. Therefore, it is necessary to analyze the security of self-adaptive systems as a whole, by including the adaptation mechanisms and possible future states of the system in the security analysis to ensure that the system can mitigate security threats as effectively as possible.

Adaptation can be employed in combination with security relevant techniques to enable dynamic protection in systems that feature uncertainty. In that context, static security enforcement approaches are enhanced with adaptation capabilities and can therefore dynamically alter their analysis capabilities and/or security enforcement to deal with uncertainties in the security policies, active entities, the maliciousness of entities and arbitration between different system components. These techniques are particularly relevant in open and distributed systems.

This thesis investigates self-adaptation and security based strategies, tactics, and techniques to enhance the security and/or enable self-protection capabilities in software-intensive systems.

This thesis contributes solutions that (i) enhance adaptation decisions with security concerns to reduce the attack surface in self-adaptive systems, and (ii) design and implement an adaptive security approach for open decentralized systems. This chapter is organized as follows. Section. 1.1 provides a brief overview of the domain and fundamental concepts on which this thesis is based. Section. 1.2 motivates the research challenges leading to this work and outlines the research questions that we attempt to answer. Section. 1.3 discusses our research methodology. Section. 1.4 gives a short summary of each paper included in the thesis and its scientific contributions. We close this chapter with a summary

of the research presented in this thesis and a short discussion about ongoing and planned future research to complete our five year research project.

1.1 Background

This section begins with a brief introduction to self-adaptive systems and security modeling and analysis. We then proceed with a short discussion of related work about security of self-adaptive systems and self-protection.

1.1.1 Self-Adaptive Systems

In this thesis, we refer to a self-adaptive system as a software-intensive system that can dynamically and autonomously change its architecture or behavior at run-time to more effectively achieve stakeholder goals and non-functional qualities. A self-adaptive system might additionally adapt itself to accommodate changes in its context or its environment.

Architecture A self-adaptive system architecture often comprises three basic elements [71]: (i) the environment, (ii) the managed system, and (iii) the managing layer. The environment contains the fragment of the external world that interacts with, and as a result, effects the operation of the system [33]. The environment includes all entities, physical or virtual, software, hardware etc, that the system interacts with but has no control over, i.e., the means to separate the system from its environment is via the notion of control. When the system is not able to control an element that it nevertheless interacts with, then that element is part of its environment. The managed system corresponds to the part of the system that deals with its domain-specific functionality. To realize its functionality, the managed system interacts with the environment. In order to support adaptation, the managed system has to provide mechanisms that enable monitoring and adaptation execution, i.e., the managed system needs to provide the means for an adaptation mechanism to receive relevant inputs that will aid in determining the current system state and need for adaptation (monitoring), and to be able to request or perform the required changes to the system in order to better align it with its goals.

The managing system controls the managed system and contains the adaptation logic required to achieve adaption goals. The managing system monitors both the environment and the managed system and adapts the latter when required to progress towards its adaptation goals. An adaptation goal is a concern of the managing system over the managed system, i.e., a *desired state* that the managed system should be in, or a *desired property* that it should satisfy to either meet or progress towards its stakeholder goals. Adaptation goals are often related to the software qualities of the managed system [72].

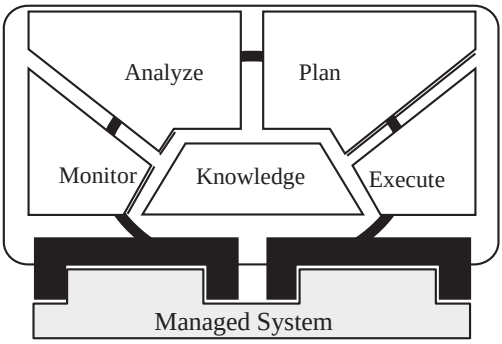

Figure 1.1: An Autonomic Manager

The primary building block of a managing system is an autonomic manager [36], the four phases of which are shown in Figure 1.1. These phases include *Monitoring*, *Analysis*, *Planning*, and *Execution*. All four phases share a common *Knowledge Base*. This particular model of an autonomic manager is therefore called a MAPE-K loop. The Monitoring phase gathers data from the managed system and the environment and updates the information stored in the Knowledge Base with the updated view of the world. The Analysis phase exploits the updated information stored in the Knowledge Base to determine whether an adaptation is required. When an adaptation is triggered, the Planning phase constructs a plan by combining a set of adaptation actions to meet the adaptation goals. The adaptation plan is executed by the Execution phase that performs the required changes to the managed system.

Architecture-based Adaptation Architecture based self-adaptation refers to the use of software architecture as the basis for self-adaptation [20]. In architectural-based adaptation, the system's software architecture is often used as the external model for dynamic adaptation [48]. The architecture of a software system is an abstract representation of the system as a composition of computational elements and their interconnections [62]. An architectural model is an abstract model that removes details and exposes the important system-level behaviors (operations) and architectural properties (values relevant to system qualities), required for the system to meet its goals. The architectural model is used to guide the adaptation decisions and therefore, is maintained and updated at run-time to correspond to an as accurate as possible representation of the actual managed system. In architecture-based adaptation, adaptation is guided by a set of analyses that make use of the architectural model and knowledge gathered through monitoring the managed system and the environment and embedded in the architectural model. Adaptation can be guided by one or multiple analyses, each of which might

be concerned with a different system goal or non-functional quality. Since, the adaptation is guided by analyses on an architectural model that represents the actual system, the closer the architectural model's architectural properties reflect the real qualities of the managed system, the more effective the adaptation will be. A strict separation of the managed and managing system is required to be able to properly estimate and tune the actual system values.

Centralized and Decentralized Control While the MAPE-K model is the prevalent model for designing adaptive behavior, its elements can be arranged in different ways and multiple MAPE-K loops can be employed at the same time to facilitate adaptation goals. The notion of control in a self-adaptive system refers to the utilization of one or multiple control loops to perform adaptation. Control in adaptation can therefore be centralized or decentralized [11]. Centralized control refers to the use of one or more control loops with a single central decision point that manages concerns over one singular system. Decentralized Control occurs when multiple control loops cooperate to achieve their goals. The control loops might be arranged in a control hierarchy or not, and might operate on the same or different parts of an adaptive system or system of systems. In [73], the authors describe a set of decentralized control architectures that can be combined to achieve the system's goals.

Uncertainty Uncertainty is the lack of complete knowledge about the actual consequences of alternatives [39]. Sources of uncertainty can be classified as external or internal. External uncertainty arises from the environment or domain in which the software is deployed. Internal uncertainty is rooted in the difficulty of determining the consequences of applying an adaptation to a self-adaptive system. In the context of self-adaptive systems, several sources of uncertainty have been studied [16]:

- Uncertainty due to simplifying assumptions. This source of uncertainty is rooted in inaccuracies in the analytical models that represent complex systems. These analytical models are used to reason about the impact of adaptation choices on a system.

- Uncertainty due to model drift. This source of uncertainty is rooted in the loose coupling between the managing system and managed system models (knowledge) used to make adaptation decisions. Due to model drift, the models maintained by the managing system may become inaccurate representations of the managed system.

- Uncertainty due to noise. This source of uncertainty is rooted in variations of a phenomenon, such as a monitored system parameter. A parameter rarely corresponds to a single value, but rather a set of values obtained over the observation period.

- Uncertainty of parameters in future operation. This source of uncertainty is rooted in the actual changes of the monitored phenomenon.
- Uncertainty due to humans in the loop. The actions of human agents are difficult to predict and classify.
- Uncertainty in the objectives. This type of uncertainty is rooted in the complexity of expressing user requirements and eliciting user preferences.
- Uncertainty in the context. Many self-adaptive software systems are intended to be used in different execution contexts. To that end, the managing system is expected to detect changes in the context and adapt the managed system to behave appropriately.

The approaches to handle uncertainty in self-adaptive systems, can be broadly split into three categories: (i) approaches that use probability theory [70] or Bayesian models [26], (ii) approaches based on fuzzy set theory [78], and (iii) approaches based on possibility theory [77]. Probabilistic approaches are used when the possible set of events is discrete and some information about the frequency of events is available. Fuzzy set theory approaches are used when there is vagueness or ambiguity in terms of which set an event belongs in. Possibly theory is used when the available information is incomplete.

1.1.2 Security Modeling and Analysis

Security Modeling Security modeling refers to the process of representing security relevant aspects of a system. The security models abstract away non-security relevant details in order to capture the security-related behavior. A security model usually has three components [6]. The first component is the *system model* that describes the behavior of the system to be analyzed. This model includes behavior under standard operation but can also specify how the system behaves under attack or due to unintended input or unexpected conditions. The second component is the *threat model*, which describes the attacker behavior and capabilities. The threat model includes all the resources and actions available to the attacker, and may or may not incorporate relevant defenses or defender actions. The third component is a set of security properties that express desired or undesired security states of the system. Desired properties correspond to states of the system that can guarantee security, while undesired properties indicate security flaws that may allow the attacker to compromise the system.

Several threat modeling techniques have been proposed in the literature, some examples include STRIDE [25], hTMM [44], QTT [54] and graphical threat models [27]. Graphical threat models which we make use of in this thesis, correspond to threat models that utilize a graph-based representation, such as attack graphs and attack trees [8,80]. Attack graphs are a class of graphical threat

models that show the exploitation of security weaknesses through vulnerabilities that lead to system compromise, i.e., they demonstrate attack scenarios in which an attacker exploits known vulnerabilities to compromise the security of a system [55]. Several varieties of attack graphs have been introduced in the literature, for instance, vulnerability cause graphs [3], Bayesian attack graphs [42], compromise graphs [43] and logical attack graphs [50]. Logical attack graphs consider logical cause-consequence relationships among vulnerabilities and are the type of attack graph that we use in this thesis.

Security Analysis Security analysis in a broad sense is defined as the process of checking whether the security policies of a system are preserved during its execution. A security policy considers all relevant aspects of confidentiality, integrity and availability of a system. With respect to confidentiality, it identifies the states in which information leaks to those not authorized to receive it. With respect to integrity, a security policy identifies the ways in which information may be altered and which entities are authorized to alter it. With respect to availability, a security policy describes what information, software or services can be accessed by authenticated users whenever they're needed. Security policies often express the desired security status during the system life time or a set of undesired states that must never be reached. Security analysis is performed by checking whether the system behavior is in conformance with the set of security policies or not. Different techniques can be used to perform security analysis including simulation, security risk analysis, verification and penetration testing among others.

Security risk analysis refers to the process of (i) identifying the valuable information regarding the system components and their vulnerabilities, (ii) revealing threats that may take advantage of those vulnerabilities to endanger the system, and (iii) estimating the possible damage and potential losses resulting from those threats [23]. Risk analysis can be categorized into two types of *qualitative assessment* and *quantitative assessment*. In qualitative assessment, classes and relative values are used to show the impact and probability of a particular threat scenario, i.e., qualitative information security risks are assessed using methods and principles with non-numerical (qualitative) values [4]. The input and output of qualitative risk assessment can be classified as range variables or linguistic variables for input, and range variables or rank variables for output. Linguistic variables are often useful for dealing with situations where the ranges of the variables are not well defined or cannot be easily quantified [60, 61]. Some qualitative assessment techniques, for example, [58] are designed to rank and prioritize the risks that an organization typically faces.

Quantitative assessment relies on numerical values, probability theory, and statistics to determine the level of risk exposure of a system [28]. Quantitative risk assessment is based on objective measurements, and the results are expressed as numerical quantifiable values e.g., as monetary costs, percentages, or probabilities

[17]. Quantitative assessment includes "expected value analyses" (EV) [67], where the impact of each attack scenario is assessed based on the expected quantifiable loss caused by that scenario.

Quantitative security risk assessment is performed by measuring a few security metrics, often defined over threat models. In this thesis, we use the definition provided by [34]: A measurement is an observation of a relevant characteristic in a single point in time. A Metric on the other hand, is a value derived by analyzing measurements, often via comparison with an established scale. Measurements are generated by counting; metrics are generated from analysis. In other words, measurements are objective raw data and metrics are either objective or subjective human interpretations of that data. In computer security, these measurements are concerned with aspects of the system that contribute to its security, and as a consequence security metrics reason about security relevant properties that can be derived by analyzing those measurements.

Security Metrics can be classified into four categories [53]: metrics for measuring (i) vulnerabilities, i.e. security weaknesses that can allow exploitation, (ii) metrics for measuring defenses, i,e., actions or techniques to mitigate attacks, (iii) metrics for measuring threats, i.e., attacks that take advantage of multiple vulnerabilities or other factors such as human errors, and (iv) metrics for measuring situations, i.e., outcomes of attack-defense interactions. In this thesis, we are mainly interested in metrics for measuring vulnerabilities and metrics for measuring threats. In particular, we concentrate on metrics that deal with vulnerability severity e.g., the Common Vulnerability Scoring System (CVSS) [45] which is a free and open industry standard for assessing the severity of computer security vulnerabilities and metrics that measure the power of attacks that exploit multiple vulnerabilities [53]. Several security metrics have been proposed in the literature for security risk assessment based on attack graphs [30], e.g., shortest path that considers the shortest path that an attacker can take to achieve its goal, number of paths that counts the number of paths that an attacker can take, and mean of path lengths which is defined as the average length of all attack paths.

Formal (Security) Analysis There exist two major approaches to formal (security) analysis: *theorem proving* and *model checking*. Theorem proving, in similar fashion to human proofs, attempts to prove that a system is secure by mathematically demonstrating that no combination of attacker actions in the threat model can cause security policies to be violated. Theorem proving tools are categorized into two categories: fully automated theorem provers, and proof assistants. Automated theorem provers rely on decidability and satisfiability procedures to automate the proofs, e.g., SAT (Boolean Satisfiability) solvers [7], SMT (Satisfiability modulo theories) solvers [5], and fix-point engines [2]. Proof assistants, that are more expressive, are used when undecidable or higher order logics are involved. Model checking [10] is an exhaustive automatic method to verify whether a specification is satisfied by a given state transition model or

not. A specification is usually expressed in terms of a temporal logic such as Computation Tree Logic (CTL) [9] or Linear Tree Logic (LTL) [29], and the system behavior is expressed in terms of a state transition system. Probabilistic model checking is a technique employed to verify quantitative properties of a system that exhibits stochastic behavior [38]. Therefore, the specification is expressed in a temporal language that supports reasoning about probability (e.g., PCTL) while the model is usually described using a probabilistic automaton.

Information Flow Control Information flow control concerns itself with controlling how information flows in a system in order to guarantee that confidentiality and integrity will not be violated. Information flow is often formulated based on the semantic notion of *non-interference* [21]. Non-interference states that low-sensitive outputs should not be influenced by high-sensitive inputs. According to Denning [12], a model of information flow control is defined over a set of security objects, each of which is bound to a *security class*. Information is permitted to flow from an object a with security class C_a, to another object b with security class C_b, based on a *can-flow* relation $\rightarrow$ between C_a and C_b, i.e., if information is allowed to flow from the security class C_a to C_b, then it can also flow from a to b. If every operation in a sequence of operations satisfies the can flow relation $\rightarrow$, then an information flow control model is secure. The most common policy associated with this model is called the High-Low policy. It only contains two security classes H (high-sensitive) and L (low-sensitive), where information is allowed to flow from L to L and H (i.e., $L \rightarrow L$ and $L \rightarrow H$) and also from H to H (i.e., $H \rightarrow H$) but not from H to L.

Decentralized information flow control (DIFC) [46] concerns itself with protecting the system's confidentiality and integrity even in the presence of untrusted code or users [47]. DIFC allows application writers to control how data flows between pieces of an application and the outside world, by defining policies to be enforced by the system regarding the data flowing to and from a process [47]. Thus, each process in the system defines its own security policies to be enforced. DIFC has been employed in operating systems, e.g., Flume [37], histar [79], Asbestos [14], android applications [35], service-oriented architectures [15] and the cloud [51].

1.1.3 Security and Self-Adaptive Systems

Security in the context of self-adaptive systems can be approached from two perspectives. Either from the standpoint of analyzing the security status of the whole system before, under and after adaptation, which we refer to as security analysis of a self-adaptive system, or from the standpoint of providing mechanisms to identify, analyze and cope with threats autonomously that we define as self-protection. Self-protection approaches will always include at least some security analysis of the managed system to identify, analyze and mitigate

threats. Adaptation in self-protecting systems arises in terms of providing the capability to deal with attacks in a dynamic context. That usually means either adapting the system to proactively deal with incoming attacks, i.e., adapting the system to its most secure state available, or anticipating and dynamically crafting defenses for attacks not known in advance.

Security Analysis of Self-Adaptive Systems Security analysis of a self-adaptive system can be decomposed into the security analysis of its basic components and their interactions. Therefore we can identify three cases: (i) security analysis of the managed system, (ii) security analysis of the managing system and (iii) security analysis of the adaptations mechanism. Security analysis of the managed system is domain specific and corresponds to security analysis of the relevant application domain. Research has been carried out in cloud based systems [13,41], service-oriented systems [40,74], mobile ad-hoc networks [1,18], component based systems among other domains. To the best of our knowledge, there has been no research that analyzes formally or otherwise the security of the managing system or of the adaptation mechanisms associated with the adaptation process. Security analysis of the adaptation process is critical since self-adaptive systems are additionally vulnerable to classes of attacks that exploit the adapting or evolving nature of a system. These attacks exploit information exposed in one state of the system to compromise a later state that can be reached during or after adaptation.

In terms of providing adaptive security to autonomic systems, Tziakouris et.al [69], provide a survey of self-adaptive security in open systems environments. They survey self-adaptive solutions that explicitly target security and provide a taxonomy of 16 dimensions, among which the following dimensions of interest are included: architecture deployment, component dependency and anticipatory support. Architecture deployment refers to the nature of the feedback loops employed to perform adaptation. The results show that centralized control loops are far more common (57%), than hierarchical (7%), semi-decentralized (31%) and decentralized (5%) alternatives. Component dependency is described as the degree of coupling among the components of the self-adaptive system. The results show a high degree of coupling, with (81%) of the examined solutions following a controlled approach,d semi-controlled approaches taking up (10%) and autonomous solutions only being (8%) of the field. Anticipatory support corresponds to the nature of the defense of the system, with the majority of the examined solutions being reactive (92%) rather than proactive (8%).

Self-Protecting Systems Self-protection refers to the capability of the system to defend itself against adversarial attacks. That is accomplished by identifying, analyzing and mitigating external threats. Self-protection can be accomplished in a variety of ways using a wide berth of techniques. Yuan et al. [75] provide a taxonomy of self-protecting systems. Their taxonomy features the following interesting dimensions: self-protection levels, meta-level separation, theoretical

foundation, control topology and response timing. Self-protection levels refers to the extend to which the system is protected from attacks. The dominant self-protection level is monitoring and detecting threats (95%) rather than responding and protecting (86%) the level that comes second and planning and prevention (18%) that comes last. The results additionally show at least some degree of meta-level separation among the managing and managed layer, with most approaches (83%) showing some form of separation. Multiple theoretical foundations have been studied in the pursuit to provide self-protection, the vast majority (over 90%) of them however are based on heuristics [24,56,57,63,64], while other theoretical foundations in descending order of research works include optimization and learning [31,32,65,66] and formal methods [22,68]. Enforcement happens almost definitively at the system boundary level, while response timing is definitively reactive and the control topology is almost exclusively centralized.

When it comes to architectural self-protection which is the focus of this thesis, Yuan et.al., [76] provide a set of architectural security patterns to facilitate self-protection. These patterns include a protective wrapper pattern to defend against DoS attacks, a software rejuvenation pattern used to mitigate the effects of attacks that cannot be defended against and an agreement based redundancy pattern that is capable of defending against XSS and injection attacks. Each pattern details architectural adaptation, threat detection and mitigation. The authors finally demonstrate how to incorporate their approach into Rainbow [19], a well known and adopted architecture-based self-adaptation framework. Schmerl et.al., [59] build upon the previous work and provide a methodology to compose security-related tactics into higher level strategies to respond to DoS attacks. They base their solution in utility theory and combine security strategies with business qualities to select the strategy that better fits the business context. Probabilistic model checking is employed to select the strategies to be applied to the system by reasoning about the effect of a strategy both to security and other quality objectives of the system. The approach is incorporated to Rainbow and is demonstrated by adapting ZNN a well known self-adaptive exemplar news website.

1.2 Research Challenges

Self-adaptive systems alter their functionality and structure at runtime, and therefore can become vulnerable to different classes of attacks over their life-time. Since the information about these attacks is often not available during design time, a self-adaptive system must be able to reason about its security at runtime. To this end, the system must be able to model and analyze its security. As discussed in Section 1.1.3, research has been conducted on the security analysis of the managed system, however, security analysis of adaptations, to the best of our knowledge, has not yet been explored. Threat modeling is an approach to model (multi-stage) complex attacks that target the system in its entirety by

modeling the interactions between the vulnerabilities exploited on the underlying system components. Such modeling approaches are often supported with a few security metrics that are helpful to perform quantitative security risk analysis. If employed at runtime, threat modeling can aid in providing the system with dynamic security risk analysis capabilities to consider security aspects while performing adaptations. Threat modeling techniques have not been applied for security analysis of self-adaptive systems in the past. Hence, we formulate our first research question as the follows:

> Q-I: How can we incorporate threat modeling and quantitative security risk analysis to model and reason about security of self-adaptive systems at runtime?

While security metrics-based risk analysis methods are useful in practice, they suffer from a few limitations. Security metrics are not often expressive enough, e.g., they cannot express properties about the temporal ordering of events/attacks and their dependencies, or interactions with the system. It is not always possible to use such approaches to express refined, application-specific security policies and verify if they have been enforced or not. Additionally, it is important to be able to model the managed system, the adaptation behavior and their interactions with the attacker to have a suitable representation of the system security at each point in time. Furthermore, there might be uncertainties about various aspects of a self-adaptive system that can affect its security which cannot be captured by the threat models. A suitable analysis should consider different types of uncertainty, such as uncertainty about the evolution and adaptation of the system behavior, uncertainty about the environment including the behavior of external adversaries that may reside in it, uncertainty about humans in the loop etc. Therefore, we need approaches that provide the means to (i) model the system, adaptation and attacker behavior, and investigate their interactions, (ii) express and analyze complex security properties, and (iii) handle uncertainty. Formal security modeling and analysis methods are a class of approaches that are suitable candidates for this purpose. Hence, we formulate our next research question as:

> Q-II: How can we formally specify and analyze the security of a self-adaptive system under uncertainty at runtime?

The third research question that we attempt to address in this thesis deals with employing the security analysis results for self-protection. In this context, self-protection is implicit, i.e., it is a result of the reduced attack surface and control of the adaptation mechanisms to prohibit adaptations that would compromise

the system security. Reducing the attack surface can be achieved by making the adaptation mechanisms aware of the security concerns of the system or by taking control of the adaptation to enforce a warranted security level. We call adaptations that take security concerns into account, *security-aware adaptations*.

The following research question is therefore derived:

> Q-III: How can the results of runtime security analysis be utilized to reduce the attack surface of self-adaptive systems via security-aware adaptations? What are the implications of security-aware adaptations?

Decentralized systems feature a large number of entities that are required to cooperate to achieve their goals. Decentralized Information flow control (DIFC) approaches can be employed to allow each entity in the system to provide its own security policies to be enforced. Security in an open system is often based on the concept of trust [49], i.e., trust plays an important role when defining and enforcing security policies, as whether an entity should be allowed to access private information or be authorized to manipulate information, directly depends on how much that entity is trusted by the information source. Any approach aiming to provide security to such a system must therefore address trust establishment and update among the system entities. To employ a DIFC-based approach to enforce security in an open decentralized system, it is therefore required that trust is taken into account when considering security. There are, however, a few issues to be resolved first. Trust is dynamic and evolves over time, consequently a system that aims to enforce policies based on trust needs to adapt accordingly to handle the uncertainties that arise from the dynamic nature of trust. Changes in trust among the system entities entail changes to their security, and thus the system has to adapt its security enforcement capabilities accordingly, if it aspires to enforce the up-to-date security policies of its entities. The DIFC enforcement mechanism, therefore, needs to be equipped with adaptive capabilities to establish and dynamically update the trust between system elements, capabilities to change the nature of its enforcement at runtime and capabilities that modify which elements are under its control based on their mutual trust.

Since entities in the system can be considered to be mutually distrusting, a centralized controller is difficult to be agreed upon, thus the DIFC enforcement mechanisms also need to be decentralized. To facilitate communication among system entities that do not share the same controller, controllers need to be organized into a dynamic controller hierarchy with adaptation capabilities. These capabilities must feature mechanisms that deal with malicious or misbehaving controllers or entities. Self-organization is additionally required for trust handling and facilitating communication among the controllers. Since the system is then composed of communicating MAPE-K loops that operate independently, a

decentralized control approach is required. Thus, to achieve adaptive security, a combination of adaptive trust-aware DIFC and decentralized control is needed. We formulate the following research questions aimed at the design of adaptive security in open decentralized systems:

> Q-IV: How can trust, DIFC and adaptation be combined to enforce security in open decentralized systems?

> Q-V: How can adaptive security be achieved in an open decentralized system by combining adaptive trust-aware DIFC and decentralized control? What are the implications?

1.3 Research Project

Our research questions identify gaps in modeling, analysis, and implementation tactics for security aspects in self-adaptive software systems as well as in realizing adaptive security in open decentralized systems. Hence, we selected *design science research method* (DSRM) [52] for the research project design, as it constitutes a comprehensive method targeting research about techniques, processes, and methods. In this section, we discuss each design science activity, methods we used and their results. The design science research method comprises six phases [52], five of which are depicted in Figure 1.2. We describe and discuss each phase in detail in this section. Design Science allows for different types of research approaches identified by various starting points. This thesis is a comprehensive project that starts by identifying and motivating a problem. Hence, we use a problem-centered approach. In our presentation, both in Figure 1.2 and in the description of each phase, we omit the communication phase which was performed by publishing the scientific articles contained in this thesis.

1.3.1 DSRM Instantiation in this Thesis

Problem-centered approach This thesis is centered around the problem of designing secure adaptation and adaptive security for open systems. To achieve that goal, our solutions need to be able to analyze the security of the system to be protected, identify how to adapt the system to be able to guarantee a sufficient level of security, before, during and after adaptation and adapt the system in a secure fashion. The problem at hand was identified after a literature study on self-protecting systems, adaptive security approaches and security in self-adaptive systems.

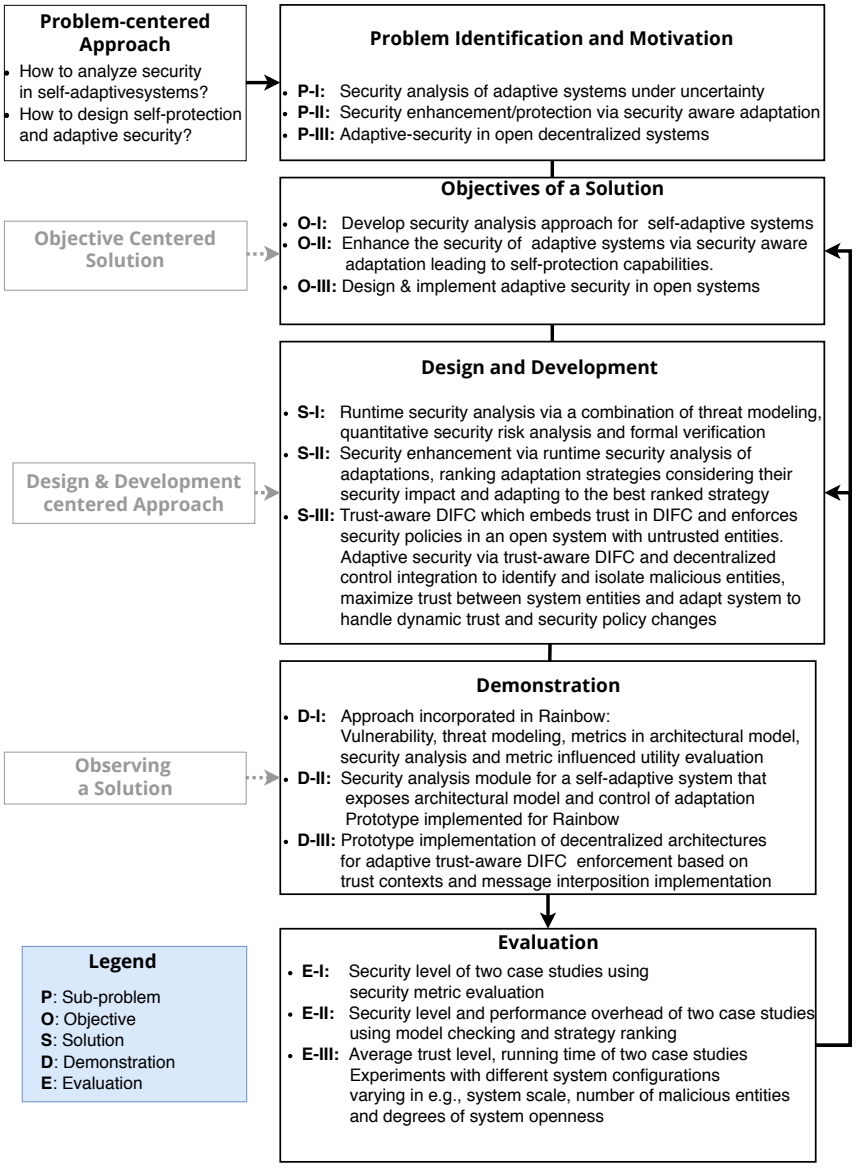

Figure 1.2: Our Design Research Method

Problem identification and motivation The problem of designing security-aware adaptive systems and self-protection can be decomposed in three sub-problems. Sub-problem **P-I** refers to the security analysis of self-adaptive systems which requires reasoning about the security of the system in all stages of

adaptation. The analysis results can then be utilized for protection via enhancing the system security by security-aware adaptation. Sub-problem **P-II** refers to protecting a self-adaptive system via security-aware adaptation when sufficient information about its components and vulnerabilities is available and control over its adaptation capabilities is assertable. Sub-problem **P-III** refers to the design and implementation of adaptive security in open (decentralized) systems where little about the maliciousness and intentions of entities residing in the system is known.

Objectives of a solution To address the above sub-problems, we identify the following objectives:

- Objective **O-I** entails the development of a security analysis approach for self-adaptive systems. The security analysis has to be be able to model and reason about the system security under uncertainty and aid in developing suitable protection techniques.
- Objective **O-II** aims at the design of a security-aware adaptation approach, supported by a prototype implementation, to enhance the security of a system by reducing its attack surface.
- Objective **O-III** entails the design and implementation of an adaptive security approach for open decentralized systems where uncertainty exists in terms of the entities residing in the system, their established trust and their maliciousness.

The research objectives have been initially identified via a literature study and were revised a few times as the project progressed, based on the results of our published scientific work and the scientific community feedback.

Design and development Sub-problem **P-I** is addressed by augmenting the architectural models with security-related information, such as vulnerabilities and then utilizing threat modeling at runtime to perform runtime security analysis. Given the augmented architectural model (containing information about the interactions and vulnerabilities of components), we build a logical attack graph for each of the possible system adaptations that shows the attacker behavior during and after adaptation. Based on the constructed graphs, we use quantitative security risk assessment and probabilistic model checking to analyze the security of adaptations.

In quantitative risk assessment, a set of security metrics is calculated for each enabled adaptation and the metric results are compared. The adaptation with the best metric scores is considered to be the most secure. To perform probabilistic model checking, for each adaptation, we construct a probabilistic automata network that contains the formalized attack graph in terms of a set of probabilistic automata, a system automaton and an adaptation automaton. A

set of temporal security properties expressed in PCTL are then verified on the automata network to evaluate the adaptation security. Both quantitative risk analysis and probabilistic model checking deal with uncertainty in probabilistic terms. In particular, we harness a probabilistic security metric that shows the probability of each node in the attack graph being reached as the basis for our proposed adaptation relevant security metrics and as the base properties used for model checking.

Sub-problem **P-II** is addressed by employing security analysis at runtime whenever an adaptation is triggered and harnessing the analysis results to facilitate security-aware adaptation. The system is able to keep an updated view of its security status before, during and after adaptation and select an enabled adaptation that leads to the most secure system state available. Based on the analysis type, we provide different ways to achieve security-aware adaptation.

When security analysis is performed via security metric calculation, the metric results are stored in the architectural model of the system. Security-aware adaptation is achieved by enabling the utility-based evaluation adaptation mechanisms to consider the security metric values when selecting the next adaptation to be executed. In this case, which adaptation will be executed is decided by utility evaluation and security is one of the dimensions that affect the decision. While the exposed attack surface is indeed reduced, this solution cannot guarantee protection. This can be demonstrated in a case where the only adaptations available might be harmful to the system security. Since this solution does not have control over the adaptation mechanisms but instead advises the utility evaluation on which adaptation is most secure, one of the harmful adaptations will be executed which might bring the system to a security critical condition.

In our approach for security analysis utilizing probabilistic model checking, adaptations are ranked based on the verification results and the best ranking one is selected for execution provided that it meets a minimum security threshold. This solution assumes control over the system's adaptation mechanisms and therefore can directly choose the adaptation to be executed. This solution can protect the system in two ways: (i) by reducing the attack surface of the system, thus limiting the attacks that can affect the system, and (ii) by guaranteeing that the system will never reach a security critical state, since those adaptations that lead to those states are expressively prohibited by the minimum security threshold.

The solution to Sub-problem **P-III** is based on a combination of decentralized information flow control, trust establishment and update, and decentralized control architectures. We extend a classical DIFC model and augment it with trust-related information and adaptation capabilities. Entities in the system are grouped based on their trust to form trust contexts, i.e., hierarchies of trusted elements or other trust contexts. Trust contexts can independently enforce DIFC,

establish and update the trust of the entities that they control. Each trust context can be adapted using three adaptation operations: *context merging*, *context splitting*, and *context or entity isolation*. Merging and splitting serve to maximize the trust among the entities in the trust context, while isolation serves to identify and isolate malicious or misbehaving entities. We organize trust contexts in trust context hierarchies which are formed based on strong trust between their entities. We call such a hierarchy of trust contexts a *trust architecture*. We instantiate the trust architectures as decentralized control architectures that are capable of taking higher level decisions about the trust context organization and adaptation of their underlying architecture. Adaptive security is achieved by the combination of runtime trust-aware DIFC enforcement and decentralized control architecture adaptations, to steer the trust architectures into more secure ones by isolating malicious entities and contexts, and maximizing trust in their underlying trust contexts.

Demonstration To demonstrate the applicability of our solutions, we implement and evaluate three prototypes. The first two prototypes are aimed towards security analysis and protection of self-adaptive systems, i.e., they implement the two approaches discussed in **S-I** and **S-II**. Essentially, **D-I** implements security analysis via metric evaluation and attempts to enhance the system security by adding a security dimension to the utility evaluation, and thus enabling it to consider security in adaptation selection. **D-II** employs UPPAAL to perform security analysis via probabilistic model checking. The verification results for each enabled strategy are aggregated and used to rank the adaptation strategies, the most secure of which is selected to adapt the system. The prototype comprises a security analysis module that performs security analysis, strategy ranking and directly controls adaptation. Based on the strategy ranking results, the highest ranking strategy is queued for adaptation provided that it passes a minimum security level threshold. If that threshold is not met, no strategy is executed.

We implemented **D-III**, that facilitates the approach presented in **S-III**. It implements a set of decentralized control architectures that employ our adaptive trust-aware DIFC enforcement solution to maximize average trust among system entities and enforce confidentiality and integrity in an open decentralized system. To control the messages exchanged between entities in the system, **D-III** employs message interposition. DIFC enforcement is achieved by trust-contexts and trust architectures that are capable of the adaptation operations described in **S-III**.

Evaluation Each of our solutions is evaluated by performing experiments using the implemented prototypes on two case studies. The case studies for the first two prototypes (**D-I**, **D-II**) include ZNN, a news site that has been used as an exemplar for self-adaptation research in the past, and InsecureDocumentStore, a document storage and retrieval application of our own design and implementation. The case studies for **D-III** include a microservice-based item shop and a randomly-generated decentralized system.

In the experiments **E-I**, that evaluate **D-I**, we evaluated the security level of the system by comparing the values of security metrics after each adaptation. We evaluated the approach effectiveness by comparing the metric results, when the security-enhancing capabilities are enabled, with the results of the same security metrics when our approach was disabled in Rainbow. The results show that when a high emphasis is placed on security, the selected adaptations will be more secure compared to when a lower emphasis is placed on security, or the case of not taking security into consideration at all. The results additionally show that the lack of adaptation control limits the protection capabilities of this approach, as the system security will keep deteriorating if only harmful adaptations are available.

In **E-II**, we performed two types of experiments: a set of experiments to evaluate the effectiveness of the approach, and a set of experiments to evaluate the performance of our solution. In the first type of experiments, we evaluate the security level of each enabled adaptation for the first few adaptations and demonstrate that our solution adapts to the most secure one that also meets a minimum security threshold. We also compared our approach to the no-security approach. We show that even when the security of the system would completely deteriorate if no security-enhancing capabilities are employed in the system, our solution will maintain the system security at an acceptable level by prohibiting adaptations that would endanger the security of the system past the minimum-set threshold. The performance analysis experiments demonstrated that the effectiveness of our security analysis comes at the cost of performance overhead, with security analysis consuming a large amount of the total adaptation time.

In **E-III**, we evaluate our adaptive trust-aware DIFC solution. For each of our experiments, we measure the average level of trust established and the accompanying running time. The experiments feature multiple security-relevant scenarios with different system configurations, varying in system scale, number of malicious entities and degrees of system openness etc. The results show that high levels of trust can be guaranteed with a small penalty to throughput and latency.

1.3.2 Mapping of Objectives to Contributions

Figure 1.3 depicts a mapping between our objectives and the corresponding research questions. A second mapping from the research questions to the published scientific articles is also shown. The first two articles feature some overlap in terms of their goals, however, the security analysis and security-aware adaptation realization are fundamentally different in each scientific publication. In Article-I, we analyze security by calculating security metrics and allowing the preexisting adaptation mechanisms in Rainbow to take security into account by incorporating the security metric values in Rainbow's utility evaluation. In Article-II, we extend the previous security modeling by employing an automata-

based technique and analyse the system security by model checking security properties on the derived automata. The second prototype has complete control over the adaptation to ensure that the minimum security requirements for the system will be met. The second prototype is therefore not tied to Rainbow or utility theory-based adaptation and can be deployed as an analysis component in any architecture-based adaptive system that exposes its architectural model and adaptation mechanisms to the analysis. The overlap is therefore, limited in the similar architectural modeling of the managed system and in the use of graphical threat modeling as a starting point for security analysis. The third objective shown is captured by two research questions and is answered in Article-III.

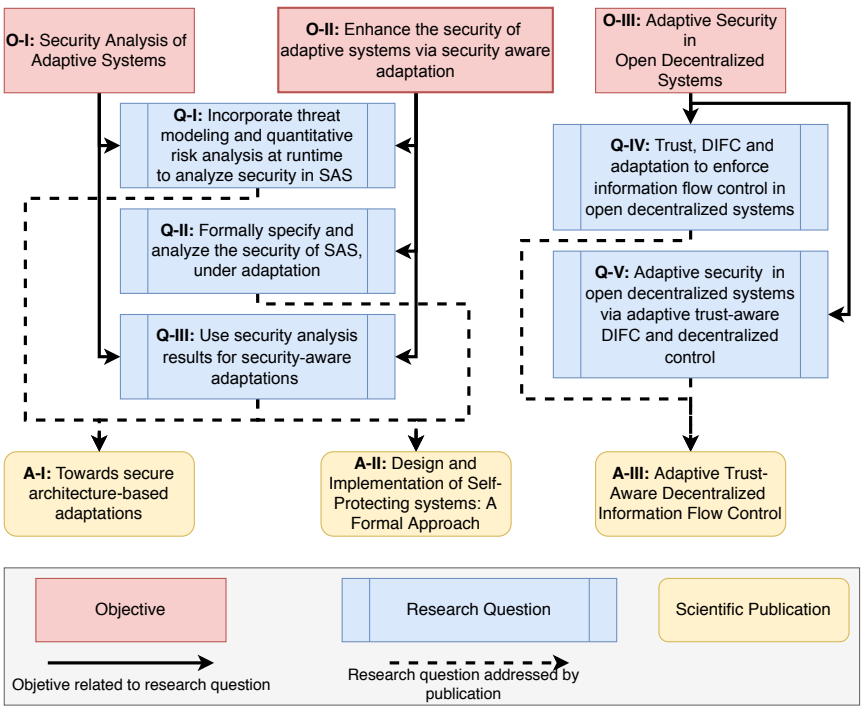

Figure 1.3: High Level Overview of Our research

1.4 Contributions

In this section, we give a summary of the included works and outline their contributions.

Article I (Chapter 2)

Narges Khakpour, **Charilaos Skandylas**, Goran Saman Nariman, and Danny Weyns,
Towards secure architecture-based adaptations,
In Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '19),
IEEE Press, 2019, pp. 114–125,
https://doi.org/10.1109/SEAMS.2019.00023.

Summary and Contributions This article discusses threat modeling and quantitative security risk assessment in self-adaptive systems detailed in **S-I**, and describes reducing the attack surface by considering security aspects before performing an adaptation, as detailed in **S-II**. The accompanying prototype is described by **D-I** and evaluation was performed as explained in **E-I**. This article contributes an approach to perform threat modeling and security analysis of adaptations in an architecture-based self-adaptive system to enable it to select the most secure adaptation available based on the analysis results. It also provides an example reference implementation for a security analysis module that can be incorporated to Rainbow with minor changes, so that it can reason about its security.

Candidate's authorship contribution Co-author, Writing, Investigation, Conceptualization, Methodology, Software, Validation

Article II (Chapter 3)

Charilaos Skandylas, Narges Khakpour,
Design and Implementation of Self-Protecting systems: A Formal Approach,
Future Generation Computer Systems,
Volume 115, 2021, pp. 421-437, ISSN 0167-739X,
https://doi.org/10.1016/j.future.2020.09.005.

Summary and Contributions

In this article, we formally specify the security of a system before, during and after adaptation and formally model the adaptive system during adaptation to facilitate formal security analysis under uncertainty. This article additionally describes probabilistic formal modeling and security analysis via probabilistic model checking, as described by **S-I**. Security enhancement is facilitated using the results of model checking and strategy ranking as described by **S-II**. The corresponding prototype is described in **D-II** and the approach evaluation is detailed by **E-II**. This paper contributes an approach to formally model and analyze the security of a self-adaptive system under uncertainty and takes the system behavior, the attacker and their interaction into account. Further,

contributions include a ranking method based on the verification results and an implementation-agnostic security analysis module that can be incorporated in an architecture-based self-adaptive system that holds a runtime instance of its architecture and exposes its adaptation mechanisms.

Candidate's authorship contribution Main Author, Writing, Investigation, Conceptualization, Methodology, Software, Validation

Article III (Chapter 4)

Charilaos Skandylas, Narges Khakpour and Jesper Andersson,
Adaptive Trust-Aware Decentralized Information Flow Control,
2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS),
Washington, DC, USA, 2020, pp. 92-101,
`https://doi.org/10.1109/ACSOS49614.2020.00030`.

Summary and Contributions In this article, we provide an adaptive security approach that can enforce decentralized information flow control in open decentralized systems. The approach is described by **S-III** and the corresponding prototype implementation is detailed in **D-III**. The results of the approach are evaluated in **E-III**. This paper contributes a novel DIFC approach, that incorporates trust, that (i) enforces security policies without the need for centralized control, and (ii) provides a good degree of trust among the entities of an open system. Moreover, we provide a decentralized control loop approach that can be instantiated into decentralized feedback loop architectures to balance between the cost incurred by DIFC enforcement and latency or throughput in an open system.

Candidate's authorship contribution Main Author, Writing, Investigation, Conceptualization, Methodology, Software, Validation

1.5 Validity and Ethical Considerations

We tackle the challenges to validity of each of our solutions in the corresponding published article. Since no synthetic data or human input was used to build our solutions, but only to evaluate them in the experiment phase, we do not discuss construct or internal validity. No ethical considerations are relevant to the research and development of this thesis. In this research, we have not in any form used or dealt with private, sensitive or objectionable information. No studies with human involvement were performed.

1.6 Summary and Future Work

In this thesis, we contribute two approaches to model and analyze the security of a self-adaptive system and enhance its security. These two approaches together aim to answer our first three identified research questions.

In answer to Q-I, threat modeling can be incorporated in an architecture-based system by augmenting the architectural model with vulnerability information that is maintained at runtime. Threat modeling can be then performed at runtime by deriving a logical attack graph for each of the possible adaptations of the system. Each logical attack graph models the attacker behavior in terms of the interactions between vulnerability exploitation in the system. Then, security analysis can be carried out using quantitative security risk assessment via measuring a few security metrics introduced to assess the security level of the system.

To answer Q-II, we derive a formal model in terms of probabilistic automata from the architectural model, the adaptation plan and the attacker behavior (in terms of an attack graph), and their interactions. These automata are combined with hand-crafted system model and adaptation behavior automata to generate a network of probabilistic automata modeling the security of the self-adaptive system and the attacker. Formal analysis is then performed by verifying security relevant properties on the automata network using probabilistic model checking. The verification results show to what extent the security properties of the system are satisfied or violated and therefore represent the security state of the system.

In answer to Q-III, security-ware adaptation allows the system to enhance its security and reduce the exposed attack surface by allowing the adaptation mechanisms to consider security in the adaptation decisions. When the most secure adaptation available is executed, the system exposes the smallest attack surface available by any of its available adaptations, thus it can prevent more threats. We provide two solutions to that end. In the first solution based on security metric evaluation, the metric results are used by utility evaluation to guide the adaptation towards the most secure state available. In the second solution based on model checking, for each enabled strategy, a number of security properties are evaluated and their results are used as the input to a strategy ranking function. The strategy selected is the one that is ranked the highest and additionally meets a minimum security threshold. If no strategies meet the minimum security threshold, then the adaptation is canceled. The implications of security-aware adaptation can be categorized in two aspects: protection capabilities and performance overhead. When it comes to self-protection capabilities, we have noticed that in order to effectively provide proactive self-protection, both security-aware adaptation and tight control of the adaptation mechanisms are required. Regarding the performance overhead, we have identified a direct relationship between security and performance overhead; higher security guarantees impose higher performance overhead.

We have also described an approach that is capable of guaranteeing a good level of trust along with enforcing confidentiality and integrity in open decentralized systems. This approach aims to address our last two research questions.

We answer Q-IV as follows. Trust-aware decentralized information flow control enhanced with trust allows each entity in the system to express (i) its own security policies in terms of confidentiality and integrity, and (ii) the level of trust required for the information to be shared to other entities in the system. Our adaptive trust-aware decentralized information flow control approach partitions the system into groups of mutually trusted entities called trust contexts. Each trust context is responsible to enforce the security policies of its entities and to facilitate communication and trust handling for them. Trust contexts form a trust hierarchy and are equipped with adaptation capabilities to maximize the trust between their entities, identify and isolate malicious entities and provide efficient DIFC enforcement mechanism.

Our final research question is answered as follows. Adaptive security is achieved by dynamically restructuring the trust hierarchies to steer the system into more secure states by isolating malicious entities and contexts, and maximizing trust in their underlying trust contexts. The implications of adaptive security include a small increase in the message latency in the system which entails a small decrease to the overall system throughput. These downsides however, come with high degree of average trust and guaranteed DIFC enforcement even in the presence of malicious nodes.

Future Work Our current work focuses on completing the current iteration of the design research method by improving the scalability of our solutions to handle larger-scale systems. Our future work is planned to complement our current solutions by designing effective reactive and proactive defenses; specially identifying or synthesizing countermeasures to defend against zero-day attacks that are not known at design time.

While the results of our formal security analysis are promising in terms of selecting the most secure adaptation strategies available, we believe that scalability of the approach needs to be improved to reduce the performance overhead of our approach and support analysis of larger systems. As such, we are currently investigating modular generation of threat models that enables us to incrementally build the security models and incrementally verify security at runtime. This means that only the fragments of the threat models affected by a potential adaptation need to be updated, and subsequently, only the security properties that are affected by that adaptation need to be re-verified.

Our first research iteration was focused on reducing the attack surface by considering security aspects while performing adaptations or preemptively controlling the communication of entities in an open system. However, such defensive approaches are not always sufficient. For instance, zero-day vulnerabilities, undiscovered attack vectors or user error cannot be addressed by our current solutions.

Therefore, we plan to iterate over the design research method once more and focus on complementary objectives, i.e., to address classes of threats that cannot be mitigated by our current solutions. We also plan one final iteration and combine both proactive and reactive approaches to self-protection in one unified solution.

To be able to protect against unknown attacks that cannot be predicted at design time or cannot be detected and analyzed via vulnerability and threat analysis, e.g., some classes of denial of service or resource exhaustion attacks, we need to design effective reactive defenses that can be deployed smoothly at run-time. We, therefore, need to identify or develop effective countermeasures and the time and place to deploy them, so that they provide the maximum effect. Moreover, since we deal with a large amount of uncertainty, both in terms of which threats can occur and when an attack happens, we need to develop approaches to synthesize suitable countermeasures at runtime to deal with the threats that a system faces at the time.

Another problem to be addressed is how to satisfy the stakeholder goals when all available system adaptations are detrimental to the system security. Synthesis of secure strategies, e.g., via transforming available insecure strategies to secure ones by incorporating synthesized countermeasures and selecting the most secure component composition available to the system can aid in meeting some of the stakeholder goals while still satisfying the system security policies. The entry point for our second iteration over the design research method are the following objectives:

- Designing self-protection against unknown attacks by deploying effective reactive countermeasures.
- Countermeasure identification, synthesis and placement.
- Transformation of insecure strategies into secure ones such that the stakeholder's goals are fulfilled.

To summarize, the plan for the remainder of the research project follows.

- Complete our ongoing research on modular threat model generation and incremental security analysis to provide self-protection in self-adaptive systems.
- Perform a second iteration on the DSRM focusing on complementary techniques that will improve the effectiveness of our self-protection, i.e., developing reactive countermeasures against threats that are not known a priori and synthesizing secure strategies to ensure the system's stakeholder goals.

- Perform a final iteration on the DSRM to combine the proactive and reactive approaches, compare their effectiveness and evaluate the results of our work.

Bibliography

[1] H. Abie. Adaptive security and trust management for autonomic message-oriented middleware. In *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pages 810–817, 2009. doi:10.1109/MOBHOC.2009.5336915.

[2] José Almeida, Maria Frade, Jorge Pinto, and Simão Sousa. An overview of formal methods tools and techniques. 01 2011. doi:10.1007/978-0-85729-018-2_2.

[3] Shanai Ardi, David Byers, and Nahid Shahmehri. Towards a structured unified process for software security. In *Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems*, SESS '06, pages 3–10, New York, NY, USA, 2006. ACM. URL: http://doi.acm.org/10.1145/1137627.1137630, doi:10.1145/1137627.1137630.

[4] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Nist special publication 800-57. *NIST Special publication*, 800(57):1–142, 2007.

[5] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.

[6] Jason Bau and John Mitchell. Security modeling and analysis. *Security and Privacy, IEEE*, 9:18–25, 05 2011. doi:10.1109/MSP.2011.2.

[7] Armin Biere, Marijn J. H. Heule, H. Maaren, and T. Walsh. Handbook of satisfiability. 2009.

[8] S. Bistarelli, F. Fioravanti, and P. Peretti. Defense trees for economic evaluation of security investments. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8 pp.–423, 2006. doi:10.1109/ARES.2006.46.

[9] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Dexter Kozen, editor, *Logics of Programs*, pages 52–71, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[10] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[11] Rogério de Lemos et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems*, 2010.

[12] Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, May 1976. URL: http://doi.acm.org/10.1145/360051.360056, doi:10.1145/360051.360056.

[13] J. Du, X. Gu, and N. Shah. Adaptive data-driven service integrity attestation for multi-tenant cloud systems. In *2011 IEEE Nineteenth IEEE International Workshop on Quality of Service*, pages 1–9, 2011. doi:10.1109/IWQOS.2011.5931339.

[14] Petros Efstathopoulos, Maxwell Krohn, Steve VanDeBogart, Cliff Frey, David Ziegler, Eddie Kohler, David Mazières, Frans Kaashoek, and Robert Morris. Labels and event processes in the asbestos operating system. *SIGOPS Oper. Syst. Rev.*, 39(5):17–30, October 2005. URL: http://doi.acm.org/10.1145/1095809.1095813, doi:10.1145/1095809.1095813.

[15] Marwa Elsayed and Mohammad Zulkernine. Ifcaas: Information flow control as a service for cloud security. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 211–216, Aug 2016. doi:10.1109/ARES.2016.27.

[16] Naeem Esfahani and Sam Malek. *Uncertainty in Self-Adaptive Software Systems*, pages 214–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-35813-5_9.

[17] Fariborz Farahmand, Shamkant Navathe, Gunter Sharp, and Philip Enslow. A management perspective on risk of security threats to information systems. *Information Technology and Management*, 6:203–225, 04 2005. doi:10.1007/s10799-005-5880-5.

[18] Dewan Farid and Mohammad Zahidur Rahman. Anomaly network intrusion detection based on improved self adaptive bayesian algorithm. *Journal of Computers*, 5, 01 2010. doi:10.4304/jcp.5.1.23-31.

[19] D. Garlan, S. . Cheng, A. . Huang, B. Schmerl, and P. Steenkiste. Rainbow: architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004. doi:10.1109/MC.2004.175.

[20] David Garlan, Bradley Schmerl, and Shang-Wen Cheng. *Software Architecture-Based Self-Adaptation*, pages 31–55. 04 2009. doi:10.1007/978-0-387-89828-5_2.

[21] J. A. Goguen and J. Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*, pages 11–11, April 1982. doi:10.1109/SP.1982.10014.

[22] Joshua Guttman and Amy Herzog. Rigorous automated network security management. *Int. J. Inf. Sec.*, 4:29–48, 02 2005. doi:10.1007/s10207-004-0052-x.

[23] Shon Harris. *CISSP All-in-One Exam Guide, Fifth Edition*. McGraw-Hill, Inc., USA, 5 edition, 2010.

[24] Ruan He and Marc Lacoste. Applying component-based design to self-protection of ubiquitous systems. In *Proceedings of the 3rd ACM Workshop on Software Engineering for Pervasive Services*, SEPS '08, page 9–14, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1387229.1387233.

[25] Shawn Hernan, S. Lambert, T. Ostwald, and A. Shostack. Threat modeling-uncover security design flaws using the stride approach. *MSDN Magazine*, pages 68–75, 01 2006.

[26] Peter Hoff. *A First Course in Bayesian Statistical Methods*. 01 2009. doi:10.1007/978-0-387-92407-6.

[27] Jin B. Hong, Dong Seong Kim, Chun-Jen Chung, and Dijiang Huang. A survey on the usability and practical applications of graphical security models. *Computer Science Review*, 26:1 – 16, 2017. URL: http://www.sciencedirect.com/science/article/pii/S1574013716301083, doi:https://doi.org/10.1016/j.cosrev.2017.09.001.

[28] E. Hulitt and R. B. Vaughn. Information system security compliance to fisma standard: A quantitative measure. In *2008 International Multiconference on Computer Science and Information Technology*, pages 799–806, 2008. doi:10.1109/IMCSIT.2008.4747334.

[29] Michael Huth and Mark Ryan. *Logic in computer science - modelling and reasoning about systems*. 01 2000.

[30] Nwokedi C Idika. *Characterizing and Aggregating Attack Graph-Based*. PhD thesis, Purdue University West Lafayette, 2010.

[31] G. ". Jabbour and D. A. Menasce. The insider threat security architecture: A framework for an integrated, inseparable, and uninterrupted self-protection mechanism. In *2009 International Conference on Computational Science and Engineering*, volume 3, pages 244–251, 2009. doi:10.1109/CSE.2009.278.

[32] G. Jabbour and D. A. Menascé. Policy-based enforcement of database security configuration through autonomic capabilities. In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, pages 188–197, 2008. doi:10.1109/ICAS.2008.49.

[33] Michael Jackson. The meaning of requirements. *Annals of Software Engineering*, 3(1):5–21, 1997.

[34] Wayne Jansen. Directions in security metrics research. 01 2010.

[35] Limin Jia, Jassim Aljuraidan, Elli Fragkaki, Lujo Bauer, Michael Stroucken, Kazuhide Fukushima, Shinsaku Kiyomoto, and Yutaka Miyake. Run-time enforcement of information-flow properties on android. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security – ESORICS 2013*, pages 775–792, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[36] J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36:41–50, 2003.

[37] Maxwell Krohn, Alexander Yip, Micah Brodsky, Natan Cliffer, M. Frans Kaashoek, Eddie Kohler, and Robert Morris. Information flow control for standard os abstractions. *SIGOPS Oper. Syst. Rev.*, 41(6):321–334, October 2007. URL: http://doi.acm.org/10.1145/1323293.1294293, doi:10.1145/1323293.1294293.

[38] Marta Kwiatkowska, Gethin Norman, and David Parker. *Probabilistic Model Checking: Advances and Applications*, pages 73–121. 2018.

[39] Emmanuel Letier, David Stefan, and Earl T. Barr. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, page 883–894, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2568225.2568239.

[40] Jianxin Li, Bo Li, Tianyu Wo, Chunming Hu, Jinpeng Huai, Lu Liu, and K.P. Lam. Cyberguarder: A virtualization security assurance architecture for green cloud computing. *Future Generation Computer Systems*, 28(2):379 – 390, 2012. URL: http://www.sciencedirect.com/science/article/pii/S0167739X1100063X, doi:https://doi.org/10.1016/j.future.2011.04.012.

[41] S. Ma and Y. Wang. Self-adaptive access control model based on feedback loop. In *2013 International Conference on Cloud Computing and Big Data*, pages 597–602, 2013. doi:10.1109/CLOUDCOM-ASIA.2013.94.

[42] Lingyu Wang Marcel Frigault. Measuring network security using bayesian network-based attack graphs. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 698–703, July 2008. doi:10.1109/COMPSAC.2008.88.

[43] M. A. McQueen, W. F. Boyer, M. A. Flynn, and G. A. Beitel. Quantitative cyber risk reduction estimation methodology for a small scada control system. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 9, pages 226–226, 2006. doi:10.1109/HICSS.2006.405.

[44] Nancy Mead, Forrest Shull, Krishnamurthy Vemuru, and Ole Villadsen. A hybrid threat modeling method. Technical Report CMU/SEI-2018-TN-002, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2018. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=516617.

[45] P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. *IEEE Security Privacy*, 4(6):85–89, 2006. doi:10.1109/MSP.2006.145.

[46] Andrew C Myers and Barbara Liskov. A decentralized model for information flow control. *ACM SIGOPS Operating Systems Review*, 31(5):129–142, 1997.

[47] Andrew C. Myers and Barbara Liskov. Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.*, 9(4):410–442, October 2000. URL: http://doi.acm.org/10.1145/363516.363526, doi:10.1145/363516.363526.

[48] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62, May 1999. doi:10.1109/5254.769885.

[49] Nardine Osman and David Robertson. Dynamic verification of trust in distributed open systems. pages 1440–1445, 01 2007.

[50] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 336–345, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1180405.1180446.

[51] Thomas Pasquier, Jatinder Singh, and Jean Bacon. Clouds of things need information flow control with hardware roots of trust. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 467–470, Nov 2015. doi:10.1109/CloudCom.2015.41.

[52] Ken Peffers, Tuure Tuunanen, Charles E Gengler, Matti Rossi, Wendy Hui, Ville Virtanen, and Johanna Bragge. Design science research process: A model for producing and presenting information systems research, 2020. arXiv:2006.02763.

[53] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. A survey on systems security metrics. *ACM Comput. Surv.*, 49(4), December 2016. doi:10.1145/3005714.

[54] Bradley Potteiger, Goncalo Martins, and Xenofon Koutsoukos. Software and attack centric integrated threat modeling for quantitative risk assessment. In *Proceedings of the Symposium and Bootcamp on the Science of Security*, HotSos '16, page 99–108, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2898375.2898390.

[55] Alex Ramos, Marcella Lazar, Raimir H. Filho, and Joel J. P. C. Rodrigues. Model-based quantitative network security metrics: A survey. *IEEE Communications Surveys Tutorials*, 19(4):2704–2734, Fourthquarter 2017. doi:10.1109/COMST.2017.2745505.

[56] Hans Reiser. Fault and intrusion tolerance on the basis of virtual machines. 01 2008.

[57] Hans Reiser and R. Kapitza. Hypervisor-based efficient proactive recovery. pages 83–92, 11 2007. doi:10.1109/SRDS.2007.25.

[58] Reginald E. Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In Sushil Jajodia and Javier Lopez, editors, *Computer Security - ESORICS 2008*, pages 18–34, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[59] Bradley Schmerl, Javier Cámara, Jeffrey Gennari, David Garlan, Paulo Casanova, Gabriel A. Moreno, Thomas J. Glazier, and Jeffrey M. Barnes. Architecture-based self-protection: Composing and reasoning about denial-of-service mitigations. In *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, HotSoS '14, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2600176.2600181.

[60] A. Sendi and M. Dagenais. Arito: Cyber-attack response system using accurate risk impact tolerance. *International Journal of Information Security*, 13:367–390, 2013.

[61] Alireza Shameli-Sendi, Mohamed Cheriet, and Abdelwahab Hamou-Lhadj. Taxonomy of intrusion risk assessment and response system. *Computers and Security*, 45:1 – 16, 2014. URL: http://www.sciencedirect.com/science/article/pii/S0167404814000613, doi:https://doi.org/10.1016/j.cose.2014.04.009.

[62] Mary Shaw and David Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, Inc., USA, 1996.

[63] F. M. Sibai and D. A. Menascé. Defeating the insider threat via autonomic network capabilities. In *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pages 1–10, 2011. doi:10.1109/COMSNETS.2011.5716431.

[64] F. M. Sibai and D. A. Menascé. Countering network-centric insider threats through self-protective autonomic rule generation. In *2012 IEEE Sixth International Conference on Software Security and Reliability*, pages 273–282, 2012. doi:10.1109/SERE.2012.40.

[65] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo. Resilient intrusion tolerance through proactive and reactive recovery. In *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, pages 373–380, 2007. doi:10.1109/PRDC.2007.52.

[66] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo. Highly available intrusion-tolerant services with proactive-reactive recovery. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):452–465, 2010. doi:10.1109/TPDS.2009.83.

[67] LILI SUN, RAJENDRA P. SRIVASTAVA, and THEODORE J. MOCK. An information systems security risk assessment model under the dempster-shafer theory of belief functions. *Journal of Management Information Systems*, 22(4):109–142, 2006. arXiv:https://doi.org/10.2753/MIS0742-1222220405, doi:10.2753/MIS0742-1222220405.

[68] Antonio Vicenzo Taddeo and Alberto Ferrante. Run-time selection of security algorithms for networked devices. In *Proceedings of the 5th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet '09, page 92–96, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1641944.1641963.

[69] Giannis Tziakouris, Rami Bahsoon, and Muhammad Ali Babar. A survey on self-adaptive security for large-scale open environments. *ACM Comput. Surv.*, 51(5):100:1–100:42, October 2018. URL: http://doi.acm.org/10.1145/3234148, doi:10.1145/3234148.

[70] Vladimir G Vovk and Glenn R Shafer. Kolmogorov's contributions to the foundations of probability. *Problems of Information Transmission*, 39(1):21–31, 2003.

[71] Danny Weyns. *Software Engineering of Self-adaptive Systems*, pages 399–443. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-00262-6_11.

[72] Danny Weyns and Tanvir Ahmad. Claims and evidence for architecture-based self-adaptation: A systematic literature review. In Khalil Drira, editor, *Software Architecture*, pages 249–265, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[73] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. *On Patterns for Decentralized Control in Self-Adaptive Systems*, pages 76–107. Springer, Berlin, Heidelberg, 2013.

[74] C. G. Yee, W. H. Shin, and G. S. V. R. K. Rao. An adaptive intrusion detection and prevention (id/ip) framework for web services. In *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pages 528–534, 2007. doi:10.1109/ICCIT.2007.422.

[75] Eric Yuan, Naeem Esfahani, and Sam Malek. A systematic survey of self-protecting software systems. *ACM Trans. Auton. Adapt. Syst.*, 8(4), January 2014. doi:10.1145/2555611.

[76] Eric Yuan, Sam Malek, Bradley Schmerl, David Garlan, and Jeff Gennari. Architecture-based self-protecting software systems. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, QoSA '13, pages 33–42, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2465478.2465479, doi:10.1145/2465478.2465479.

[77] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst.*, 100:9–34, April 1999.

[78] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. URL: http://www.sciencedirect.com/science/article/pii/S001999586590241X, doi:https://doi.org/10.1016/S0019-9958(65)90241-X.

[79] Nickolai Zeldovich, Silas Boyd-Wickizer, Eddie Kohler, and David Mazières. Making information flow explicit in histar. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 19–19, Berkeley, CA, USA, 2006. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1267308.1267327.

[80] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley. Rre: A game-theoretic intrusion response and recovery engine. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pages 439–448, 2009. doi:10.1109/DSN.2009.5270307.