Département de formation doctorale en informatique
UFR STMIA

École doctorale IAEM
Université de Lorraine, France

Vulnerability Management for Safe Configurations in Autonomic Networks and Systems

DISSERTATION

(Extended abstract)

publicly presented on June 12, 2014

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Information and Computer Science

by

Martín BARRÈRE CAMBRÚN

Dissertation committee:

President:	Le président
Reviewers :	Dr. Michelle SIBILLA. Professor at Université Toulouse III Paul Sabatier, France. Dr. Raouf BOUTABA. Professor at University of Waterloo, Canada.
Examiners :	Dr. Emil LUPU. Reader at Imperial College London, U.K. Dr. Nacer BOUDJLIDA. Professor at Université de Lorraine, France. Dr. Rémi BADONNEL. Associate Professor at Université de Lorraine, France. Dr. Olivier FESTOR. Professor at Université de Lorraine, France.

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Table of Contents

Résumé / Abstract

Le déploiement d'équipements informatiques à large échelle, sur les multiples infrastructures interconnectées de l'Internet, a eu un impact considérable sur la complexité de la tâche de gestion. L'informatique autonome permet de faire face à cet enjeu en spécifiant des objectifs de haut niveau et en déléguant autant que possible les activités de gestion aux réseaux et systèmes eux-mêmes. Cependant, lorsque des changements sont opérés par les administrateurs ou directement par les équipements autonomes, des configurations vulnérables peuvent être involontairement introduites, même si celles-ci sont correctes d'un point de vue opérationnel. Ces vulnérabilités offrent un point d'entrée pour des attaques de sécurité. Les environnements autonomes doivent être capables de se protéger pour éviter leur compromission et la perte de leur autonomie. À cet égard, les mécanismes de gestion des vulnérabilités sont essentiels pour assurer une configuration sûre de ces environnements.

Cette thèse porte sur la conception et le développement de nouvelles méthodes et techniques pour la gestion des vulnérabilités dans les réseaux et systèmes autonomes, afin de leur permettre de détecter, d'évaluer et de corriger leurs propres expositions aux failles de sécurité. Nous présenterons tout d'abord un état de l'art sur l'informatique autonome et la gestion de vulnérabilités, en mettant en relief les défis importants qui doivent être relevés dans ce cadre. Nous décrirons ensuite notre approche d'intégration du processus de gestion des vulnérabilités dans ces environnements, et en détaillerons les différentes facettes, notamment : extension de l'approche dans le cas de vulnérabilités distribuées, prise en compte du facteur temps en considérant une historisation des paramètres de configuration, et application en environnements contraints en utilisant des techniques probabilistes. Nous présenterons également les prototypes et les résultats expérimentaux qui ont permis d'évaluer ces différentes contributions.

Mots clés: sécurité, gestion de réseaux, informatique autonome, gestion de vulnérabilités.

Over the last years, the massive deployment of computing devices over disparate interconnected infrastructures has dramatically increased the complexity of network management. Autonomic computing has emerged as a novel paradigm to cope with this challenging reality. By specifying high-level objectives, autonomic computing aims at delegating management activities to the networks themselves. However, when changes are performed by administrators and self-governed entities, vulnerable configurations may be unknowingly introduced. Vulnerabilities constitute the main entry point for security attacks. Hence, self-governed entities unable to protect themselves will eventually get compromised and consequently, they will lose their own autonomic nature. In that context, vulnerability management mechanisms are vital to ensure safe configurations, and with them, the survivability of any autonomic environment.

This thesis targets the design and development of novel autonomous mechanisms for dealing with vulnerabilities, in order to increase the security of autonomic networks and systems. We first present a comprehensive state of the art in autonomic computing and vulnerability management, and point out important challenges that should be faced in order to fully integrate the vulnerability management process into the autonomic management plane. Afterwards, we present our contributions which include autonomic assessment strategies for device-based vulnerabilities and extensions in several dimensions, namely, distributed vulnerabilities (spatial), past hidden vulnerable states (temporal), and mobile security assessment (technological). In addition, we present vulnerability remediation approaches able to autonomously bring networks and systems into secure states. The scientific approaches presented in this thesis have been largely validated by an extensive set of experiments which are also discussed in this manuscript.

Keywords: security, network management, autonomic computing, vulnerability management.

Chapter 1

General introduction

Contents

1.1 The context

Over the last years, the massive deployment of computer devices over interconnected heterogeneous infrastructures has dramatically changed the perspective of network management. In particular, the Internet has played a fundamental role providing a platform for thousands of mixed technologies which currently constitute our globalized digital world. Nowadays, almost any network including the Internet itself, is expanding as a response to many factors. As end-users become more connected with computing technologies, or maybe technology becomes more friendly with end-users, a growing demand for useful digital advances goes along with the process. These new requirements appear in different forms, which directly impact in the mechanisms and resources used to meet them. In the other way around, technology can be also observed as a proactive stream, which shapes end-users' behavior to some extent. This symbiosis between technology and end-users makes a vehicle for their evolution.

This evolution however, is not without dangers. Constructive approaches tend to test the boundaries of current technologies. Since networks became the key platform for exchanging information and providing all kind of services on top of them, their management rapidly shifted into higher levels of complexity. These scenarios in turn are dynamic, which challenges their limits even more. In that context, the autonomic computing approach has been conceived as a response to this problem. What if specific administration tasks to adapt our networks to each concrete service are not needed anymore ? What if we can express what we need from networks, and just tell the underlying infrastructure to manage it for us ? From a high-level viewpoint, these sentences may provide the intuition and spirit of autonomic computing. Indeed, autonomic computing aims at freeing administrators from the burden of heavy and error-prone management tasks. The main idea is to specify what networks have to do by means of high-level objectives, and delegate the responsibility of accomplishing these specific goals to the networks themselves.

Under this perspective, self-governed networks and systems can positively tackle the management of the overwhelming technological development we are witnessing today. However, in order to make this approach work, security is essential. The ability of being autonomous implies self-

protection. If this requirement is not met, autonomous entities might get compromised, not only affecting their own behavior but the surrounding environment as well. For this reason, **the ability of autonomic networks and systems to manage vulnerabilities and handle their own exposure is a critical factor for their survivability**. This matter constitutes the heart of our work. **This thesis aims at providing novel autonomous mechanisms for dealing with vulnerabilities, in order to increase the security of self-governed networks and systems**.

1.2 The problem

In computer security, vulnerabilities are flaws or weaknesses in the design, implementation, or configuration of a system that may allow an attacker to exploit them in order to bypass the security policies of such system. Vulnerabilities constitute the key entry point for breaking into computer systems and gaining unauthorized access to assets within these systems. Therefore, the ability to manage vulnerabilities is crucial for any computer system. Autonomic networks and systems are not an exception, although their autonomous nature challenges the vulnerability management process at higher levels. As a matter of fact, related tasks usually performed by human administrators over regular systems, must now be performed by self-governed entities on their own. The vulnerability management process basically involves the detection and remediation of vulnerabilities. Nevertheless, conceiving autonomic networks and systems featuring this process poses hard challenges. **How can we provide autonomic environments with mechanisms for increasing their vulnerability awareness ? What methods should be employed for identifying security weaknesses in an autonomous manner ? How should they proceed to mitigate and eradicate detected vulnerabilities while maintaining the system operative and safe ?** These and other questions constitute the issues that this thesis aims at dealing with.

Autonomic computing has opened new horizons for addressing problems where traditional methods seem to fail. Particularly, mechanisms able to properly scale with evolving dynamic networks and capable of reasonably tackling their increasing management complexity are simply, essential. Autonomic computing perfectly fits with these requirements. However, if autonomic infrastructures do not develop mechanisms and techniques to protect themselves from security threats, their real power and utility will eventually come apart. The focus of this thesis is to contribute in this direction, by providing a state of the art in autonomic computing and vulnerability management, and filling out missing scientific issues required to harden the foundations and security of autonomic computing. In the next section we present the structure of this document.

1.3 Organization of the document

This document is intended to provide a summarized overview of the content presented in this thesis. Fully detailed information about our research and contributions can be found in the main manuscript of this thesis. The present document is organized as follows. Chapter 2 presents key concepts of autonomic computing and vulnerability management, and the connection between both worlds. Chapter 3 presents the contributions of this thesis as well as implementation prototypes developed to validate our approaches. Chapter 4 ends this document with general conclusions and describes promising perspectives of this work.

Chapter 2

Autonomic environments and vulnerability management

Contents

2.1 Introduction

The growing development of networks and the multiplication of the services offered over them have dramatically increased the complexity of network management. The paradigm of autonomic computing has been introduced to address this complexity through the design of networks and services which are responsible for their own management. In this chapter, we first present essential concepts about autonomic computing and vulnerability management. Then, we explain why vulnerability management constitutes a critical activity to ensure real autonomy, and provide an overview of the approach taken in this thesis to achieve this goal.

2.2 Autonomic computing overview

The autonomic computing paradigm was proposed in 2001 by IBM [9], as a response to cope with the arising complexity of managing computing systems [14, 6]. The vision of autonomic computing is strongly inspired on the autonomous nervous system. Indeed, every time we breath, we do not do it conscientiously, we do not think about it. However, it is still a vital function that actually follows a high-level human law, being alive. Similarly, autonomic computing aims at providing an infrastructure where networks can be managed by establishing high level-objectives. The underlying networking components will then perform in accordance to these rules and the changing environment, without explicit human intervention. Indeed, autonomic solutions aim at emulating human behavior during the process, by reasoning and taking decisions for the successful operation of the system. This perspective aims at providing strong foundations for developing scalable and flexible infrastructures able to support a demanding and changing technological reality [21], [8]. In this thesis, we consider the following definition for an autonomic system.

Definition 1 (Autonomic system)**.** *An autonomic system is a self-governed entity, able to manage itself without any type of external control, and to perform required tasks without human intervention in order to accomplish the goals it was created for. While the purpose of the autonomic system is defined by high-level objectives, the achievement of this purpose is delegated to the system itself. The internal activities performed by the autonomic system may include environment perception, analysis, reasoning, decision making, planning and execution of actions that must ensure the successful operation of the system.*

Considering the previous definition, it is important to highlight that the fact of being governed by policies or high-level objectives is what makes the key difference between autonomic and automated solutions. Usually, autonomic solutions are composed of smaller components called autonomic entities. Autonomic entities are designed to serve a specific purpose such as monitoring a network device or providing routing services. These entities can then be combined to construct more complex autonomic solutions. In order to organize their self-governing nature, autonomic systems involve a set of functional areas called self-* properties, defined as follows [16]:

- **self-configuration**, providing mechanisms and techniques for automatically configuring components and services,
- **self-optimization**, covering methods for monitoring and adapting parameters in order to achieve an optimal operation according to the laws that govern the system,
- **self-healing**, for automatically detecting, diagnosing and repairing localized software and hardware problems, and
- **self-protection**, supporting activities for identifying and defending the system against potential threats.

Self-* properties are intended to autonomously solve high-level requirements, however, their implementation is complex and poses hard challenges. Along with administration tasks done by humans, changes performed by autonomic entities may inadvertently generate vulnerable states when following high-level objectives. Even though these changes can operationally improve the environment, insecure configurations may be produced increasing the exposure to security threats. As happens in the real world, autonomic elements coexist within dynamic environments, interacting with other autonomic and non-autonomic elements. If an autonomic element is compromised, its functions and abilities become untrustworthy and eventually disabled ; thus autonomic elements that use services of the former become compromised as well. This inevitably leads to distrust and failure of the autonomic environment. Autonomic systems must be able to manage their own state and perform required activities to achieve secure configurations. Autonomic elements unable to support this capability will age with time, becoming more vulnerable, insecure and useless. Real automation can be possible only if autonomic networks and systems fully integrate vulnerability management mechanisms for ensuring safe configurations. This is the target of this thesis, which is discussed in the next section.

2.3 Vulnerability management in autonomic environments

Autonomic systems must act on their own, taking any necessary decision to obey the rules that govern their behavior and to achieve their goals. As such, these systems must deal with security aspects, ensuring a proper functionality and guaranteeing their results. Therefore, vulnerability management, briefly assessing and remediating vulnerabilities, is essential. In this thesis, the concept of vulnerability in computing security is described by considering the following definition.

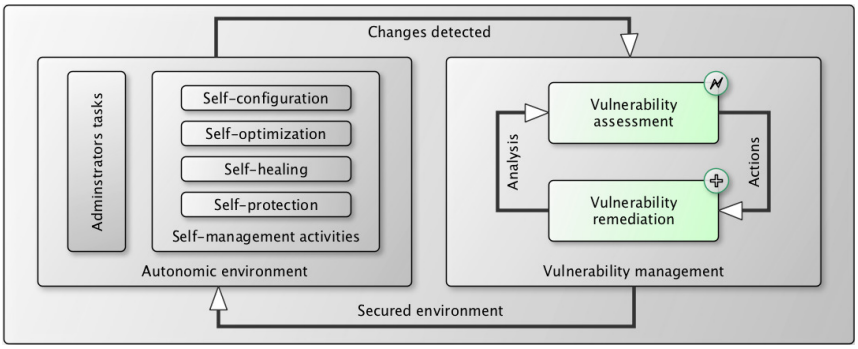

FIGURE 2.1 – Positioning of vulnerability management with respect to self-management activities

Definition 2 (Vulnerability)**.** *A vulnerability can be understood as a flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy [17], [11].*

Our objective is to provide autonomous and consistent mechanisms for assessing and remediating vulnerabilities, in order to ensure safe configurations within autonomic environments. Under this perspective, vulnerability management is a cross-cutting concern strongly related but not limited to self-configuration and self-protection activities of autonomic networks. This process is depicted in Figure 2.1 where a control loop enables the assessment and remediation of potential vulnerable states generated by both administrators tasks and self-management activities, thus securing the environment. The main idea is that actions and changes performed in the system are constantly monitored and analyzed looking for vulnerabilities. When vulnerable states are detected, corrective actions are performed until the environment is secured. The vulnerability management control loop remains active during the whole lifetime period of the autonomic environment under surveillance. In that context, the establishment of a secure process for dealing with vulnerabilities requires the specification of a policy defining the desired system state, and a well-known secure initial state to identify vulnerabilities and policy compliance [23].

The main activities performed during the lifecycle of the vulnerability management process can be mapped to the same activity line present in autonomic components. Figure 2.2 describes the general lifecycle of an autonomic component where the main activities done for dealing with vulnerabilities have been mapped to the task loop performed during the autonomic manager lifecycle [10]. As it can be observed, vulnerability identification activities take place in the moni-

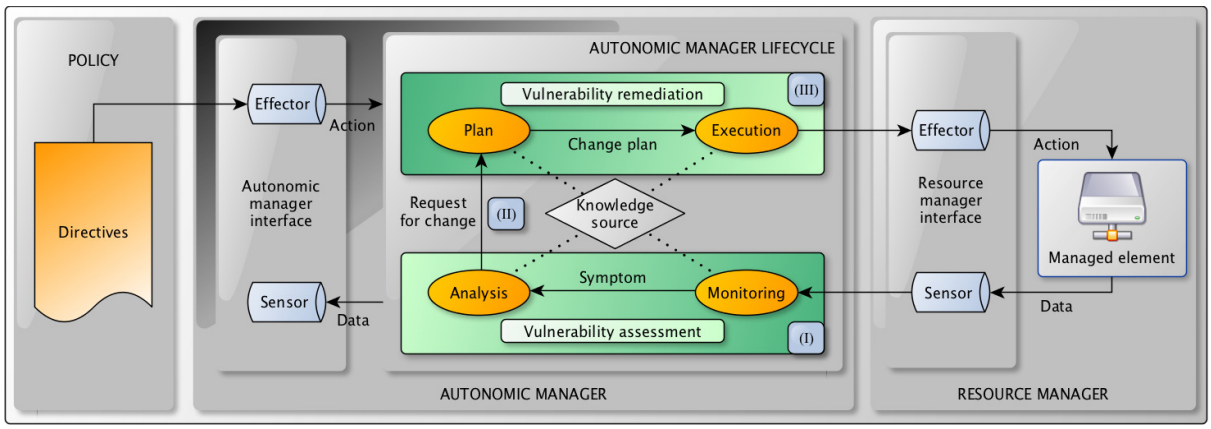

FIGURE 2.2 – Mapping of the vulnerability management activity into the autonomic lifecycle

toring phase where tasks for assessing and analyzing vulnerable states are performed (I) taking advantage of the available security knowledge. When a security problem is found, it is classified (II) and changes for correcting the situation must be performed. Therefore, vulnerability countermeasures are planned based on several factors such as importance, risks and impact. Finally, a change plan is generated and remediation tasks are executed (III) in order to maintain safe configurations and to be compliant with the current policy. Figure 2.2 illustrates the overall approach taken in this thesis for integrating vulnerability management activities into the autonomic plane. Actual existing methods and techniques for dealing with vulnerabilities within autonomic and non-autonomic systems are widely discussed in the main manuscript of this thesis.

Within our contributions, we have exploited the benefits of the Security Content Automation Protocol (SCAP) [2]. SCAP is a suite of specifications that standardize the format and nomenclature by which security software communicate information about publicly known software flaws and security configurations. These advisories are annotated with common identifiers and embedded in XML. In particular, we have heavily used the Open Vulnerability and Assessment Language (OVAL) [18] and the eXtensible Configuration Checklist Description Format (XCCDF) [25]. OVAL, supported by MITRE Corporation [15], standardizes how to assess and report upon the machine state of computer systems. XCCDF, supported by the National Institute of Standards and Technology (NIST) [17], provides support for authoring security benchmarks and reporting checklist evaluation results. Together, they not only allow to specify vulnerabilities but also to bring systems into compliance through the remediation of identified vulnerabilities or misconfigurations.

Within the OVAL language, a specific vulnerability is described using an *OVAL definition*. An *OVAL definition* specifies a criteria that logically combines a set of *OVAL tests*. Each *OVAL test* in turn represents the process by which a specific condition or property is assessed on the

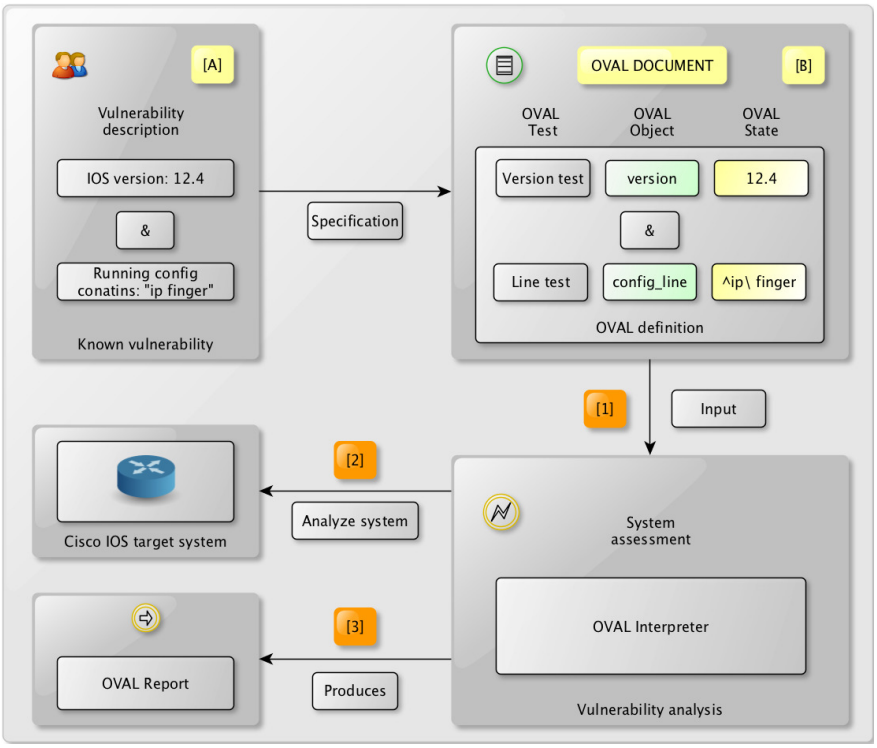

FIGURE 2.3 – OVAL example over Cisco IOS

target system. Each *OVAL test* examines an *OVAL object* looking for a specific *OVAL state*, thus an *OVAL test* will be *true* if the referred *OVAL object* matches the specified *OVAL state*. The overall result for the criteria specified in the *OVAL definition* will be built using the results of each referenced *OVAL test*.

As an example, we consider the situation illustrated in Figure 2.3 where a vulnerability for the Cisco IOS [12] has just been disclosed. For this vulnerability to be present, two conditions must hold simultaneously: (I) the version of the platform must be *12.4* and (II) the service *ip finger* must be enabled. Such vulnerability can be expressed within an *OVAL document* by defining an *OVAL definition* that arranges two *OVAL tests* as a logical conjunction. One test is in charge of assessing the system version and the other one must check the service status. The *OVAL objects* used in these tests will be an object that represents the version of the system and other object that represents the running configuration, respectively. Finally, the *OVAL states*, one for the version and one for the service, will express the states expected to be observed on each object for the tests to be true and hence, defining the truth or falsehood of the *OVAL definition*. Once the OVAL document has been specified, the actual assessment can be resumed in three main steps. At step 1, the document is consumed by an OVAL interpreter. At step 2, the target system is analyzed looking for present vulnerabilities. The OVAL analysis involves two parts, namely, the collection of required OVAL objects to be analyzed, and the comparison of collected OVAL items against the specified OVAL states. Finally, a report is produced at step 3 indicating the results of the assessment process. In this particular example, it is expected to observe the value 12.4 as the version of the system, and the running configuration file must have a line starting with the directive *ip finger*. If these two properties are observed, then the vulnerability is present on the target system.

The use of standard languages such as OVAL is essential for increasing the vulnerability awareness of autonomic environments. In addition, languages such as XCCDF highly contribute to address remediation activities in an autonomic manner. Autonomic networks and systems should be able to capitalize the knowledge provided by these security advisories in order to increase their ability to deal with vulnerabilities. In this thesis, we have pursued this goal as explained in the next chapter.

2.4 Synthesis

The overwhelming advent of new technologies featuring more power, ubiquity and usability in disparate contexts, requires novel techniques and methodologies for managing the underlying networks that support them. The technological landscape changes fast and users collaborate to mold its future as well. Therefore, it is important to leverage clean and adaptive approaches to face this evolving reality. Autonomic computing provides robust foundations that may encompass this evolution and can help to address several current network management challenges. However, autonomic systems must ensure safe configurations if we want to trust autonomic solutions. In that context, the aim of this thesis is to provide novel autonomous mechanisms for dealing with vulnerabilities, in order to increase the security of self-governed networks and systems. In the next chapter, we present a summary of our scientific contributions as well as implementation prototypes developed to validate our approaches.

Chapter 3

Contributions

Contents

3.1 Introduction

Nowadays, computing technologies spread fast, the number of end-users increases rapidly, and there is a constant demand for more and better services. Underneath, computer networks constitute the platform of this convoluted digital world. This accelerated evolution has tested the boundaries of traditional network management approaches, which do not scale properly with today's network requirements. Autonomic computing has emerged as a response to tackle this challenging issue by delegating management tasks to the network themselves. However, when changes are performed by administrators and self-governed entities, vulnerable configurations may be unknowingly introduced. Vulnerabilities constitute the main entry point for security attacks. Hence, self-governed entities unable to protect themselves will eventually get compromised and consequently, they will lose their own autonomic nature. In that context, vulnerability management mechanisms are vital to ensure safe configurations, and with them, the survivability of any autonomic environment. This thesis targets the design and development of novel approaches for managing vulnerabilities in autonomic networks and systems. In this chapter, we briefly describe in Section 3.2 our scientific contributions for autonomously managing configuration vulnerabilities, and our implementation prototypes in Section 3.3. Section 3.4 presents a synthesis which concludes this chapter.

3.2 An autonomic platform for managing configuration vulnerabilities

This section presents the contributions of this thesis. According to the vulnerability management process, we classify our contributions in two main categories, vulnerability assessment and vulnerability remediation. Figure 3.1 depicts our research work organized into different sections where dashed lines illustrate the main reading flow across them.

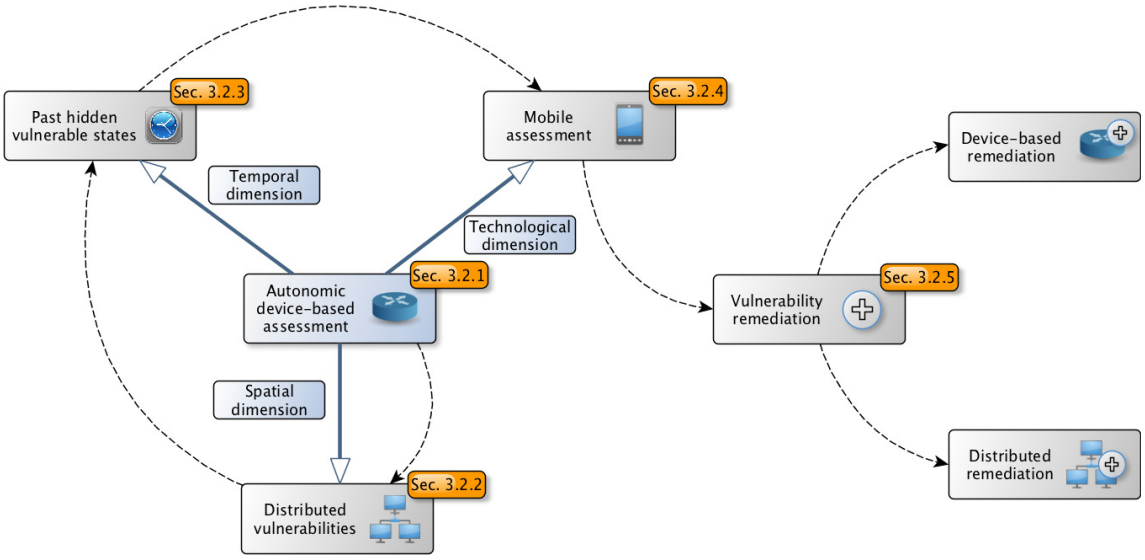

FIGURE 3.1 – Organization of contributions

The first contribution, presented in Section 3.2.1, consists of a device-based approach for autonomic vulnerability assessment. From here, three dimensions represented with solid lines extend the vulnerability assessment activity to novel scenarios considering spatial, temporal, and technological perspectives. Section 3.2.2 extends the concept of device-based vulnerabilities to composed vulnerabilities distributed across the network. We denominate this spatial extension, distributed vulnerabilities. Section 3.2.3 involves the second dimension which considers time. This approach allows to increase the present security of computer devices by analyzing hidden vulnerable states in the past. Section 3.2.4 captures the technological dimension, where we have investigated novel approaches for assessing vulnerabilities in constrained environments such as mobile networks. All these chapters, from 3.2.1 to 3.2.4, fall into the vulnerability assessment category. Section 3.2.5 closes the vulnerability management process by considering our contributions on vulnerability remediation activities. These contributions are divided in two parts, namely, remediation of device-based vulnerabilities, and distributed ones, both located in Section 3.2.5.

3.2.1 Autonomous vulnerability awareness

Changes that are operated by autonomic networks and systems may generate vulnerabilities and increase their exposure to security attacks. Our objective is to enable autonomic networks to take advantage of the knowledge provided by vulnerability descriptions in order to maintain safe configurations. In that context, our first contribution introduces an autonomous approach for assessing device-based vulnerabilities. To this end, we have integrated vulnerability descriptions into the management plane of autonomic systems. We have particularly chosen the Cisco IOS platform as a proof of concept [12]. By automatically translating these security advisories into

policy rules that are interpretable by an autonomic configuration system, autonomic agents distributed across the network become able to assess their own exposure. We have used the OVAL language [18] as a means for specifying vulnerability descriptions, and Cfengine [3] as the autonomic component of our solution. This approach, illustrated in Figure 3.2, provides an autonomous mechanism for increasing the vulnerability awareness of self-governed environments.

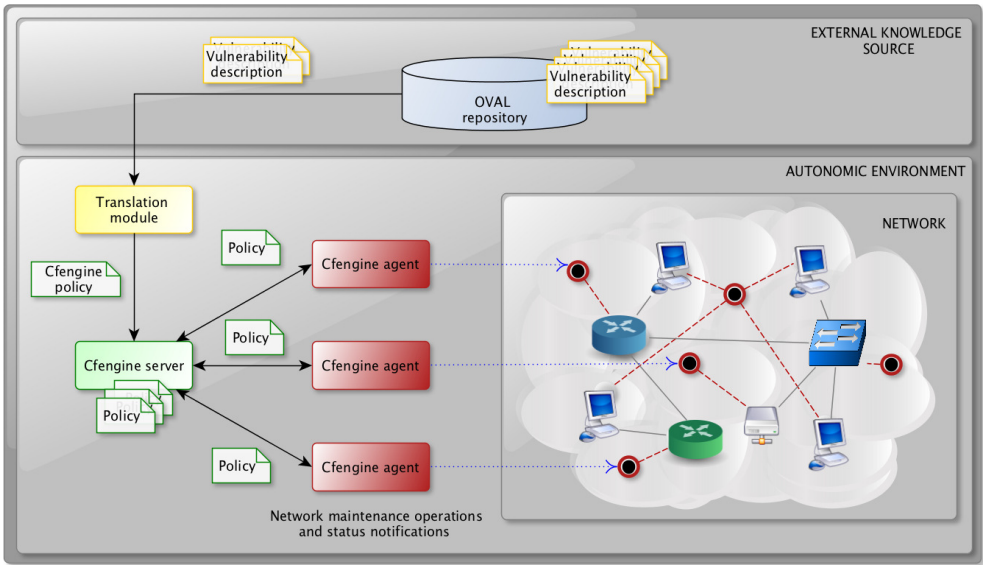

FIGURE 3.2 – High-level architecture

The proposed architecture involves an OVAL repository where the descriptions of known vulnerabilities are stored. Such descriptions are intended to be translated and introduced within a distributed Cfengine configuration. To do so, a translation module is placed between the OVAL repository and the Cfengine server. This module, further explained in Section 3.3, consumes available OVAL vulnerability descriptions from the repository and produces Cfengine policy rules that allow Cfengine agents to be aware of these security weaknesses. The Cfengine architecture is based on a client-server model. The server keeps these generated policies on its own and autonomous agents will pull these new policies from the server when convenient. In this manner, generated policies are deployed by the Cfengine server into its several Cfengine agents (points in the cloud). These autonomous agents are in charge of managing the devices present in the target network, in order to detect and prevent vulnerable configurations when self-management activities are performed. When a vulnerability is found on a specific monitored device, Cfengine agents are capable of generating specific alerts and shall be able to perform correction operations. Our approach has been validated through an extensive set of experiments whose results indicate its feasibility in terms of functionality and integration into the Cfengine autonomic maintenance tool. Even though we are focused on the Cfengine tool, currently used in millions of managed devices, our general approach could be applied to other policy-based configuration tools such as Puppet [20] or Chef [4].

Supporting vulnerability awareness constitutes the first step towards secure self-managed infrastructures capable of detecting and remediating potential security breaches. However, assessing vulnerabilities over individual network elements may not provide a global view of how vulnerable a network can be. This concept as well as our related contribution are explained in the next section.

3.2.2 Extension to distributed vulnerabilities

Vulnerability assessment activities are traditionally performed over individual network devices, independently of each other. Sometimes however, two or more devices combined together may produce a vulnerable network state that host-based approaches are not able to detect. We refer to these security weaknesses as distributed vulnerabilities, which constitute our extension within the spatial dimension. Distributed vulnerabilities must be assessed with a consolidated view of the network in order to detect vulnerable states that may simultaneously involve two or more network devices. In this thesis, we present a novel approach for describing and assessing distributed vulnerabilities in autonomic environments. No previous work has formally defined the concept of a distributed vulnerability. Therefore, we emphasize a mathematical construction to formally specify distributed vulnerabilities as well as a machine-readable language for describing them. We also present an autonomous framework for assessing distributed vulnerabilities that takes advantage of the knowledge provided by such descriptions. Hence, our strategy permits to increase the vulnerability awareness of both individual devices and the network as a whole.

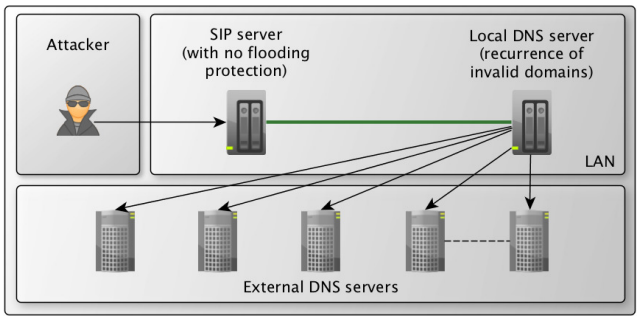

FIGURE 3.3 – Distributed vulnerability scenario

In order to explain the concept of a distributed vulnerability, we consider the example described in [24] and depicted in Figure 3.3. This scenario involves two related hosts, a SIP (Session Initiation Protocol) server and a DNS (Domain Name System) server. Each one has specific properties, however, they constitute together a potential exploitable network vulnerability. Indeed, a denial of service (DoS) attack over the SIP server can be performed by flooding it with unresolvable domain names that must be solved by a local DNS server. The local DNS server in turn, is configured for requesting the resolution of unknown domains to external servers, increasing the number of waiting requests and therefore the response time for each SIP request. Under these configuration states, flooding a SIP server with such type of messages will prevent it to respond to legitimate requests. It is important to highlight that both servers and the relationship between them are required conditions for the distributed vulnerability to be present. If the DNS server is not present or if it is not compliant with the required specific conditions, the SIP server would immediately respond to a SIP client that its SIP request has failed. Even in such a situation, thousands of SIP requests may collapse the SIP server anyway, though it is a slightly different scenario that could be specified using standard OVAL definitions. On the other hand, if there is no SIP server, it is quite clear that the distributed vulnerability has no place in this environment.

Formally, we define a distributed vulnerability as the compliant projection of a pattern involving specific host states (roles) and relationships between them over a target network as illustrated in Figure 3.4. We have designed the DOVAL language (Distributed OVAL), built on top of OVAL, as a means for describing distributed vulnerabilities in a machine-readable manner. Due to the size and dynamics of current networks, the assessment and detection of vulnerable

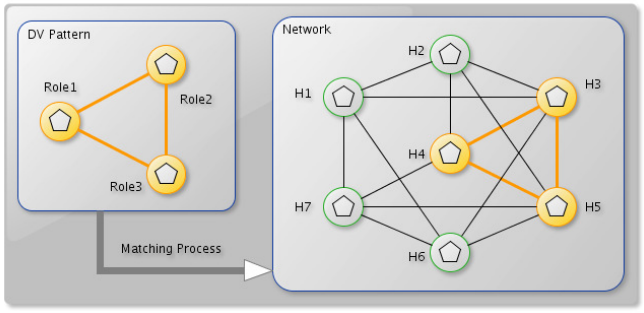

FIGURE 3.4 – Distributed vulnerability matching process

distributed states is not a trivial task. Therefore, we partition the problem into several steps, namely, (1) generation of a minimized loop-free topology of the underlying network, (2) collection of hosts and network information, and (3) assessment of DOVAL specifications over the gathered data. Figure 3.5 illustrates the steps and the architecture of the proposed approach.

Within this architecture, distributed vulnerabilities are specified using the DOVAL language and stored in a database. A Cfengine server is fed with such knowledge and translated as Cfengine policy rules, in the same way we have done before for host-based vulnerabilities. Our approach considers a deployment of Cfengine agents across the network, where each agent is in charge of controlling one network device. In order to evaluate the existence of a distributed vulnerability, a spanning tree is built on top of the target network to minimize paths and avoid network loops. The DOVAL specification is then transmitted across the tree and the required information is

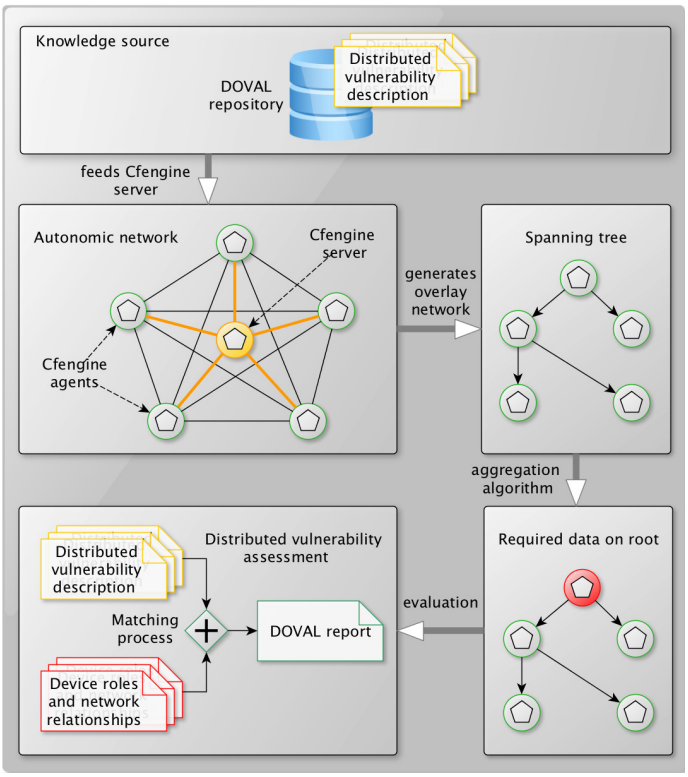

FIGURE 3.5 – Distributed vulnerability assessment process

gathered by performing an aggregation algorithm over the nodes. Each Cfengine agent assesses the device it controls in order to discover which roles such device can play within the distributed vulnerability specification. This information is returned back until all the information is stored at the root node of the spanning tree. Finally, the specification of the distributed vulnerability is projected over the information gathered from the network, and a DOVAL report is generated informing about distributed vulnerable states across the network. By using a parallel computing approach for assessing distributed vulnerabilities, our experiments have shown a linear behavior with respect to the size of the network, thus ensuring the scalability of our approach.

Distributed vulnerabilities constitute our extension within the spatial dimension which is essential to increase the overall security of autonomic networks. However, the ability to understand the state of autonomic elements in the past and therefore, potential hidden vulnerabilities, might provide robust support for enhancing security mechanisms in the present. This perspective is presented in the next section.

3.2.3 Detecting past hidden vulnerable states

Vulnerability assessment activities usually analyze new security advisories only over current running systems. However, a system compromised in the past by a vulnerability unknown at that moment may still constitute a potential security threat in the present. Indeed, a backdoor installed by an attacker for instance, may remain in the system even though the original vulnerability that gave room to such attack has been eradicated. Accordingly, past unknown system exposures must be taken into account. Our extension in the temporal dimension aims at increasing the overall security of computing systems by taking advantage of new security advisories in order to identify past hidden vulnerable states. In that context, we put forward a modeling for detecting unknown past system exposures, and an OVAL-based distributed framework for autonomously gathering network devices information and automatically analyzing their past security exposure.

The proposed framework considers two independent cyclical processes, namely, one process for imaging systems in an autonomous manner and the second one for actually detecting past security exposures. Figure 3.6 illustrates the proposed architecture identifying the main components as well as the communication processes between them.

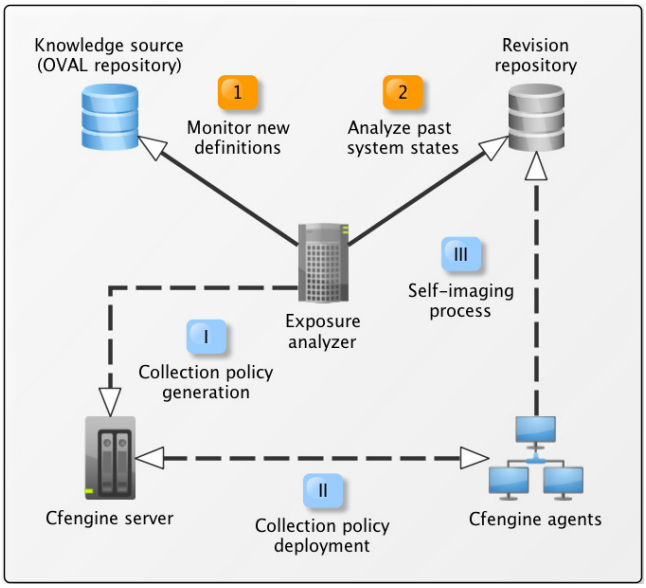

FIGURE 3.6 – High-level imaging and exposure detection process

The sequence denoted by steps I, II and III constitutes the image generation process. At step I, the exposure analyzer provides directives for data collection that will be used for building XML-based system images. These directives are specified by means of OVAL documents that are automatically translated to Cfengine policy rules. The ability to express OVAL objects without actually expecting any particular state allows us to use OVAL documents as the inventory of required objects to be collected. At step II the generated Cfengine policy rules are transmitted to the autonomic agents distributed in the network. These agents are in charge of controlling network devices and they will perform data collection activities in order to build their system images. Finally, these images are automatically stored in a SVN-based revision repository [22] at step III. The image generation process constitutes an autonomic activity and it is performed independently from the past exposure detection process. The latter, in charge of outsourcing vulnerability assessment activities, is composed of two steps. First at step 1, the exposure analyzer monitors the knowledge source on a regular basis checking for new vulnerability definitions. When a new definition becomes available, it analyzes at step 2 those system images stored in the revision repository which have been created after the public disclosure of the corresponding exploit. In this manner, our assessment strategy is able to autonomously identify periods of time in which network devices have been exposed to security threats, and therefore permitting to increase security efforts on specific devices that may still constitute an entry point for attackers in the present.

The ability to outsource vulnerability assessment activities provides a strong support to achieve autonomic features, specially on equipment with scarce resources such as mobile devices. Being focused on vulnerability management and autonomic environments, we decided to investigate to what extent autonomic computing may increase the security of mobile devices. Our investigation is presented in the next section.

3.2.4 Increasing mobile security

The development of mobile technologies and services has contributed to the large-scale deployment of smartphones and tablets. These environments are exposed to a wide range of security attacks and may contain critical information about users such as contact directories and phone calls. Assessing configuration vulnerabilities is a key challenge for maintaining their security, but this activity should be performed in a lightweight manner in order to minimize the impact on their scarce resources. In this thesis, we have conceived two complimentary approaches for assessing configuration vulnerabilities in mobile devices, which constitute our extension within the technological dimension. The first approach considers a self-assessment strategy which allows mobile devices to assess their own exposure. In order to reduce the workload on the mobile side even more, we also propose a probabilistic cost-efficient strategy integrated into a client-server architecture. Both approaches target the Android platform [1] as a proof of concept, though these approaches could be adapted to other mobile platforms as well.

Vulnerability self-assessment

Our first approach considers a self-assessment perspective where mobile devices are in charge of assessing their own security exposure. To that end, we have developed a mathematical model that supports efficient vulnerability assessment activities on resource-constrained devices. The overall architecture of our approach is shown in Figure 3.7, which considers a distributed infrastructure composed of three main building blocks: (1) a knowledge source that provides existing security advisories, (2) Android-based devices running a self-assessment service and (3) a repor-

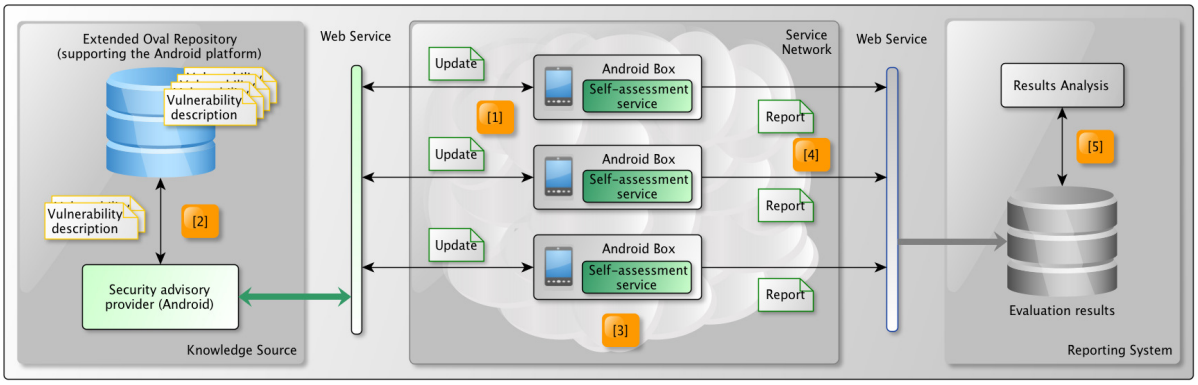

FIGURE 3.7 – OVAL-based vulnerability self-assessment architecture for the Android platform

ting system for storing analysis results and performing further analysis. The overall process is defined as follows. Firstly at step 1, the Android device periodically monitors and queries for new vulnerability descriptions updates. This is achieved by using a web service provided by the security advisory provider. At step 2, the provider examines its database and sends back new found entries. The updater tool running inside the Android device synchronizes then its security advisories. When new information is available or configuration changes occur within the system, a self-assessment service is launched in order to analyze the device at step 3. At step 4, the report containing the collected data and the results of the analyzed vulnerabilities is sent to a reporting system by means of a web service request. At step 5, the obtained results are stored in the external database. This information could be used later for different purposes such as forensic activities or statistical analysis.

Probabilistic vulnerability assessment

Delegating vulnerability assessment activities to mobile devices provides higher levels of autonomy. However, when these activities are performed, there is still a resource consuming process in the mobile side that must control the overall behavior of the analysis. We have realized that the externalization of this control may still provide autonomicity and decrease the workload on mobile devices. In that context, our second approach consists in centralizing main logistic vulnerability assessment aspects as a service. Mobile clients only need to provide the server with required data to analyze known vulnerabilities described with the OVAL language. By configuring the analysis frequency and the percentage of vulnerabilities to evaluate at each security assessment, our framework permits to bound client resource allocation and also to outsource the assessment process. Our strategy consists in distributing evaluation activities across time thus alleviating the workload on mobile devices, and simultaneously ensuring a complete and accurate coverage of the vulnerability dataset. This technique results in a faster assessment process, typically done in the cloud, and considerably reduces the resource allocation on the client side.

The proposed framework, named Ovaldroid, is described in Figure 3.8. Ovaldroid is composed of two main building blocks, namely, a server that manages the whole assessment process and clients located on the mobile network that use the vulnerability assessment service. Mobile clients periodically communicate with the Ovaldroid server (*Hello* message) in order to inform about their assessment availability. Based on the historical evaluation registry, the vulnerability manager subsystem located on the server side decides whether it is necessary to perform a new vulnerability assessment based on the pre-established assessment frequency. If it does, the probability-based test analyzer starts a process in which a sequence of OVAL tests is executed

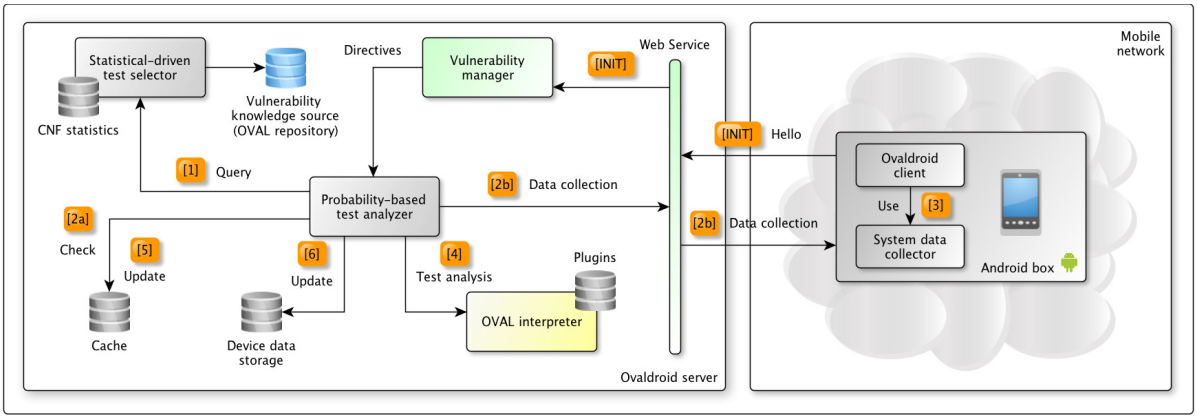

FIGURE 3.8 – Probabilistic vulnerability assessment for the Android platform

until the specified percentage of vulnerabilities to be evaluated is reached. OVAL tests are statistically selected according to their participation within the remaining vulnerabilities to evaluate as well as the impact of such vulnerabilities (step 1). The probabilistic component also considers the last time these tests have been evaluated in order to avoid test starvation. Our strategy integrates a cache for collected objects and tests results, which decreases the workload on the mobile side even more (step 2a). If the required data is not available in the cache, it is collected from the mobile device (steps 2b-3) and analyzed afterwards (step 4). Collected objects and obtained results are then stored in the server (steps 5-6). The process continues over steps 1 to 6 until the percentage of vulnerability coverage specified by the administrator is reached. Our experiments have shown promising results, indicating considerable workload reductions on the mobile side.

Vulnerability assessment activities play an essential role within the vulnerability management process. However, real autonomy can only be achieved if this process can be completed in a loop, permitting network devices to detect and also remediate security exposures by themselves. In the next section, we present our approach for remediating host-based and distributed vulnerabilities from an autonomic perspective.

3.2.5 Remediating configuration vulnerabilities

Once a vulnerability has been detected, remediation activities to eradicate such security weakness are essential. Indeed, the management of known vulnerabilities plays a crucial role for ensuring safe configurations and preventing security attacks. However, this activity should not generate new vulnerable states. In this section, we present two remediation approaches targeted on device-based and distributed vulnerabilities respectively. Our first approach formalizes the remediation decision process of device-based vulnerabilities as a SAT problem [5, 19]. In that context, we present an autonomous framework that is able to assess OVAL vulnerability descriptions and perform corrective actions by using XCCDF-based descriptions [25] of future machine states and the NETCONF protocol [7]. The second approach targets distributed vulnerabilities. There, we propose an autonomous strategy where network elements collaborate to remediate the vulnerabilities they are involved in.

Remediation of device-based vulnerabilities

The vulnerability remediation activity constitutes itself as a hard and challenging task. One single change may impact or activate other vulnerable states that were not present before the change. The same effect could occur over other system policies, in this work however, we only

deal with security configuration vulnerabilities. In that context, looking for correct changes that together can provide a safe system configuration becomes an explosive combinatorial activity. This is indeed a decision problem classified as an NP-complete problem [5]. In order to cope with this problem, we propose to formalize the change decision problem as a satisfiability or SAT problem [19]. Given a boolean expression, the SAT problem consists of finding an assignment for variables such that the formula evaluates to *true*. By specifying our vulnerability knowledge source as a propositional logical formula, we fix those system properties that we cannot change and free those variables for which changes are available. We use a SAT solving engine to determine which changes have to be made to secure the system. In order to provide proactive and reactive solutions, we propose the concept of a future state. This describes how a system will look after applying a specific change. These descriptions can be used for analyzing the security impact of changes without actually changing the system. When this information is not available, we use the NETCONF protocol [7] and its notion of candidate state where changes can be applied, analyzed and rolled back if necessary.

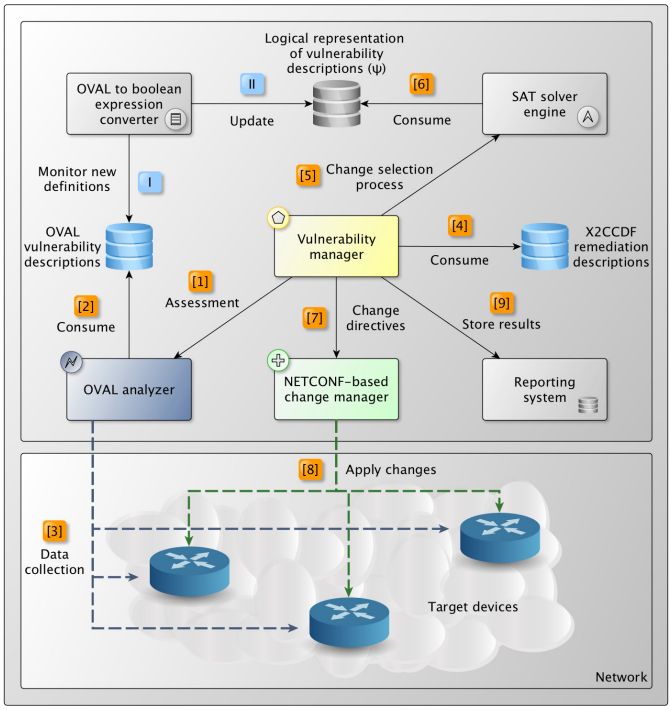

FIGURE 3.9 – SAT-based autonomous vulnerability management approach

Our SAT-based approach, illustrated in Figure 3.9, comprises two independent processes, namely, one process for maintaining logical representations of OVAL vulnerabilities descriptions up-to-date, and a second process for performing vulnerability management activities. The first process is in charge of monitoring the OVAL vulnerability descriptions database (step I) and converting new vulnerability descriptions into equivalent boolean expressions when they become available (step II). Independently, a second process is in charge of dealing with vulnerabilities, which is orchestrated by the vulnerability manager component. At step 1, it communicates with the OVAL analyzer in order to launch the assessment process. The analyzer consumes OVAL vulnerability descriptions from the repository at step 2 and collects the required data from those devices under control at step 3. Once the assessment is performed, the analyzer sends the results back to the vulnerability manager. If the system is found to be vulnerable, the vulnerability

manager analyzes the available remediation descriptions at step 4 and correlates them with the properties that can be changed in the target system. Considering the current system state and the available changeable properties, the SAT solver engine is used at step 5 to decide which changes must be applied in order to secure the system. At step 6, the SAT solver uses a logical representation ψ specifying that none of the vulnerabilities can occur. A solution provided by the SAT solver indicates which properties must be changed in the system to present a secure state. The vulnerability manager interprets this information and sends specific directives to the NETCONF-based change manager subsystem at step 7 in order to effectuate these changes. At step 8, the NETCONF protocol is used to communicate and perform the specified changes on the target system. Finally, the obtained results are sent to the reporting system at step 9. We have performed several experiments over the Cisco IOS platform to evaluate the feasibility of our approach, obtaining a stable linear behavior in terms of time and workload.

Towards the remediation of distributed vulnerabilities

The remediation of distributed vulnerabilities may involve several tasks to be performed on different devices. In order to specify remediation tasks in a machine-readable manner, we have designed DXCCDF, an XML-based language built on top of XCCDF. In this manner, we use DOVAL for specifying distributed vulnerabilities and DXCCDF for describing treatments. During the analysis of distributed vulnerabilities, some of the involved devices may present particular states that do not allow them or make it more expensive to perform specific corrective actions than other involved devices. Factors such as availability, capability, or even policy consistency must be considered during the remediation process. We refer to this spectrum of factors as the *cost* of the node for performing a corrective task. Potential mechanisms for actually computing task costs are beyond the scope of this work and they may involve several activities such as risk assessment and change management techniques.

The main process for detecting and remediating distributed vulnerabilities considers these costs as illustrated in Figure 3.10. Repositories of vulnerability descriptions as well as treatments specifications constitute the knowledge source of the network. At Step 1, a vulnerability description is consumed and assessed over the network as shown in Section 3.2.2. If there is one or more pattern matching instances over the network, a treatment analysis is launched at Step 2.1 and its corresponding distributed treatment is consumed from the treatments repository at Step 2.2.

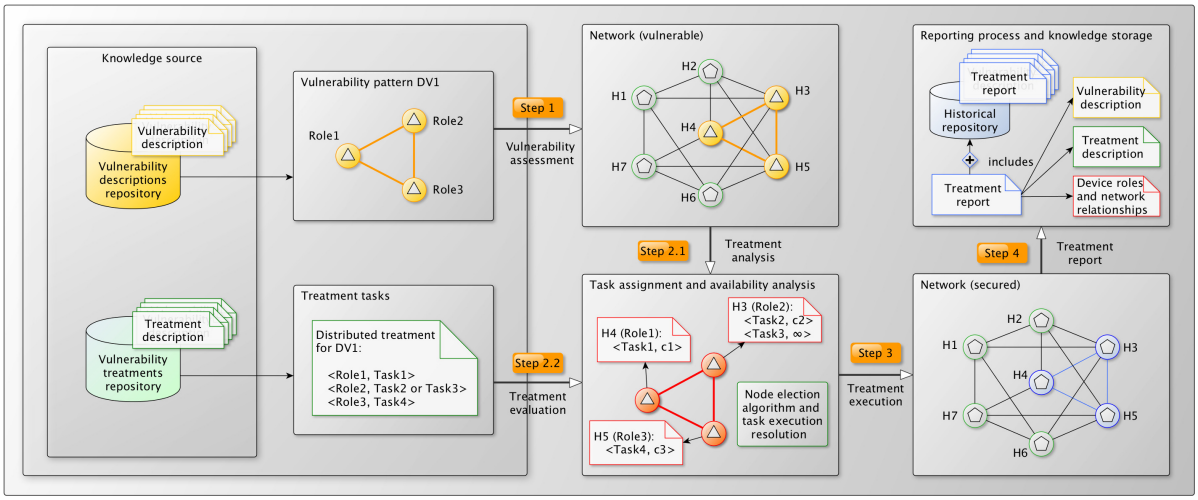

FIGURE 3.10 – Collaborative treatment for distributed vulnerabilities

Based on the available tasks for correcting the security vulnerability, devices are analyzed across the network in order to find a node for performing a remediation task. Once the treatment execution has been done and the network has been secured at Step 3, a treatment report is generated at Step 4. This report includes the vulnerability description, the treatment description used for remediating the vulnerability, and the information gathered from the network used to perform the corrective activity. Generated reports are stored in a historical database providing the ability to consider past experiences in future treatments.

3.3 Implementation prototypes

In order to evaluate the feasibility and scalability of the proposed approaches, we have developed different implementation prototypes that serve as the computable infrastructure for our experiments. We now provide a brief description of each one of them, though further documentation can be found in the main manuscript of this thesis. Our first prototype, called Ovalyzer, is an OVAL to Cfengine translation system written in Java [13], which permits the integration of OVAL vulnerability descriptions into the autonomic management plane. Ovalyzer takes up the role of the translation module depicted in Section 3.2.1, generating Cfengine policy rules that represent OVAL security advisories. Then, generated Cfengine policies are consumed by autonomic agents deployed in the network, thus becoming able to assess their own security exposure.

Our second implementation prototype aims at dealing with past unknown security exposures. Reusing the idea behind Ovalyzer, this prototype is able to autonomously generate XML-based snapshots of the state of the systems under surveillance, by following Cfengine policy rules. These images are then efficiently stored in an SVN-based repository. When new vulnerability descriptions become available, an exposure analyzer automatically assess stored images in order to identify past unknown security exposures.

Our third prototype, called Ovaldroid, targets vulnerability assessment activities on the Android platform. Indeed, we have implemented both approaches presented in Section 3.2.4. First, we have developed a lightweight self-assessment service able to monitor an external provider for new vulnerability definitions and assess its own exposure. Afterwards, we have built our probabilistic extension where the assessment activities are controlled and performed by an external server, thus reducing even more the workload on the client side.

A comprehensive set of experiments has been performed using these prototypes in their respective scenarios. Even though these prototypes are in an early development stage, and they can be clearly enhanced and further extended, they have provided a strong support to prove the scientific approaches presented in the previous sections, indicating promising results as well.

3.4 Synthesis

The aim of this thesis is to provide novel autonomous mechanisms for dealing with vulnerabilities, in order to increase the security of self-governed networks and systems. In this chapter, we have briefly presented our contributions. Particularly, we have presented an autonomous approach for increasing the vulnerability awareness of self-governed environments. We have extended this perspective by also considering distributed vulnerabilities (spatial dimension), past hidden vulnerable states (temporal dimension), and mobile devices (technological dimension). In addition, we have presented our approach for remediating both host-based and distributed vulnerabilities. The prototypes used for validating the scientific approaches presented in this thesis have been also discussed at the end of this chapter.

Chapter 4

General conclusion

Contents

4.1 Contributions summary

The large-scale deployment of disparate computing devices over evolving dynamic networks has profusely augmented the complexity of network management. Autonomic computing has become a very important research field within the scientific community, featuring strong foundations and promising perspectives to tackle this challenging problem. However, two main points require special attention. First, security issues have been poorly discussed in autonomic environments, particularly, vulnerability management mechanisms. Second, the expertise obtained from autonomic approaches has been barely experimented in non-autonomic environments. In this thesis, we have pursued both goals ; to investigate and develop novel vulnerability management approaches for autonomic environments, and to transfer autonomic principles to non-autonomic scenarios. In this chapter, we provide general conclusions about our research work as well as our technical implementations. Finally, we present research perspectives and further work.

4.1.1 Autonomic vulnerability management

Our contributions can be classified in two main categories according to the vulnerability management process, namely, vulnerability assessment and vulnerability remediation.

Vulnerability assessment is an essential activity that enables computer systems to increase their awareness about security threats. In this thesis, we have presented several approaches for autonomously assessing vulnerabilities in different scenarios. First, we have proposed an approach that integrates OVAL vulnerability descriptions (for Cisco IOS) into the autonomic management

plane. By translating these security advisories into Cfengine policy rules, autonomic agents deployed across the network become able to analyze their own security exposure. We have extended this approach to capture distributed vulnerabilities (**spatial dimension**). This concept considers situations where two or more network devices may present safe states, but when combined together, a vulnerable state arises. We also have proposed an approach for autonomously increasing the security of present computer systems by analyzing past hidden vulnerable states (**temporal dimension**). Our approach is able to identify periods of security exposure due to unknown vulnerabilities at that time, where malicious activities may have taken place. This feature may allow forensic activities to be performed in order to identify current security breaches. Additionally, we have presented an approach for autonomously assessing vulnerabilities in mobile environments (**technological dimension**). Indeed, we have proposed two complimentary approaches that aim at dealing with the assessment activity over resource-constrained devices. First, we have introduced a lightweight autonomous vulnerability assessment service that permits Android devices to assess their own exposure. Then, we have extended this approach by considering a probabilistic framework where assessment activities are outsourced to an external server which controls the overall assessment process, thus decreasing the workload of mobile clients even more.

In order to close the vulnerability management control loop, we have proposed two autonomic vulnerability remediation approaches focused on device-based and distributed vulnerabilities respectively. The first one considers the set of all known vulnerability descriptions as a conjunction of propositional logical formulas. Then, the problem is encoded as a SAT problem and a SAT solver is used to find safe configurations. Our experiments using the NETCONF protocol over the Cisco IOS platform confirm the feasibility of our approach. Our second approach proposes a collaborative mechanism for describing and remediating distributed vulnerabilities. This approach also considers correction advisories that are taken into account by Cfengine agents in the network. The remediation process involves a distributed algorithm which collects and analyze the information of network devices and selects a node to apply corrective actions based on the reported costs. Even though there is not a complete prototype implementation of our second approach, we have performed an analytical evaluation of its performance, obtaining successful linear costs when it is integrated into the vulnerability management process.

4.1.2 Implementation prototypes

With the objective of technically proving the feasibility of our previous contributions, we have developed three implementation prototypes which correspond to three different vulnerability assessment scenarios. First, we have developed Ovalyzer, an OVAL to Cfengine translation system. Ovalyzer generates Cfengine policy rules that represent OVAL vulnerability descriptions, enabling autonomic agents to perform self-assessment activities. Second, we have implemented a prototype for identifying past unknown security exposures. We have reused the idea behind Ovalyzer, but this time for autonomously generating XML-based images of the states of the systems being monitored. When new vulnerability descriptions become available, our prototype is able to analyze the history of system images looking for vulnerable periods according to this new information. Finally, we have developed Ovaldroid, an OVAL-based vulnerability assessment framework for Android, which runs as a lightweight self-assessment service inside the mobile device. In order to further reduce the load on the mobile side, we have implemented our probabilistic approach that free mobile devices from performing assessment activities themselves. These activities are outsourced in the server and then notified to mobile clients. All these prototypes have served as a computable infrastructure to prove the feasibility and scalability of our autonomic approaches.

4.2 Perspectives

4.2.1 Proactive autonomic defense by anticipating future vulnerable states

In this thesis we have proposed approaches for analyzing vulnerabilities in the present and the past, which in turn are complemented with vulnerability remediation approaches. This schema could be completed by anticipating the trajectory of a system in order to avoid changes that lead the system to known vulnerable spaces. Indeed, we have shown how vulnerabilities and system states can be characterized by the properties they present. Considering that we have n properties we can model, a target system could be graphically located on a single point of an n-dimensional space. Known vulnerabilities would have their corresponding points in such space, and similar vulnerabilities would probably conform clusters or vulnerable subspaces. The idea is that observing the movement of a target system, its trend could be monitored and determined on this space. If such a trajectory indicates high closeness levels to vulnerable states, it could be deviated by averting changes that may get the system closer or even fall into these vulnerable subspaces. This approach could provide a continuous metric of vulnerability awareness, thus enabling autonomic systems to anticipate and avoid vulnerable configuration states.

4.2.2 Unified autonomic management platform

The approaches proposed in this work reinforce the security of a network from different perspectives, making it more reliable and stronger. However, these approaches need to be unified, over a common and consistent platform, able to provide all these features in a seamless manner. Changes that can lead a system to secure states may contradict existing operational requirements. Therefore, a main challenge is to provide mechanisms able to coexist with other policy-based systems, maintaining coherency at all levels, including operational and security perspectives. Additionally, our approach for managing distributed vulnerabilities requires more technical work, as well as further investigation on the metrics required to collaboratively perform forensic and remediation tasks. Therefore, the construction of a standard model and a system able to contemplate all these aspects under a single view, would be extremely useful for the community of autonomic computing, as a basis for autonomously managing vulnerabilities.

4.2.3 Autonomic security for current and emerging technologies

The security enhancement of current paradigms such as cloud computing, and emerging models like software-defined networks (SDN) and Internet of Things (IoT), is also extremely challenging. Briefly, cloud computing tackles availability and processing power by decoupling services from the underlying hardware. More recently, SDNs also separate the management (control plane) from the hardware that actually implement network functionalities (data plane). Both approaches aim at providing reliable and scalable services while decreasing the complexity of their management and accomplishment. This is where the autonomic perspective can be extremely helpful. By providing self-configuration and self-protection mechanisms, these operational management models can also become scalable and resilient in the security plane, which is essential to achieve reliability. IoT on the other hand, gives rise to a tremendous and vertiginous growth of disparate interconnected devices. This trend clearly states a need for scalable and adaptive management mechanisms, where their security must be also as much autonomous as these mechanisms will be. In that context, autonomic security solutions might be a key element in the evolution of this new challenging landscape.

4.3 List of publications

International peer-reviewed journals

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Vulnerability Assessment in Autonomic Networks and Services: a Survey.* **IEEE Communications Surveys & Tutorials**, 16(2):988-1004, May 2014. (Impact factor at acceptance date: 6.311).

Book chapters

- **Martín Barrère**, Gaëtan Hurel, Rémi Badonnel, and Olivier Festor. *Increasing Android Security using a Lightweight OVAL-based Vulnerability Assessment Framework.* In Automated Security Management, E. Al-Shaer et al, Eds. Springer International Publishing, 2013, ch. 3, pp. 41-58, ISBN: 978-3-319-01432-6. Book chapter based on our paper selected from the 5th International Symposium on Configuration Analytics and Automation (**SafeConfig'12**), October 3-4, 2012, Baltimore, USA.

International peer-reviewed conferences

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *A SAT-based Autonomous Strategy for Security Vulnerability Management.* In Proceedings of the IEEE/IFIP Network Operations and Management Symposium (**NOMS'14**), Mini-Conference, May 5-9, 2014, Krakow, Poland.

- **Martín Barrère**, Gaëtan Hurel, Rémi Badonnel, and Olivier Festor. *A Probabilistic Cost-efficient Approach for Mobile Security Assessment.* In Proceedings of the 9th IEEE International Conference on Network and Service Management (**CNSM'13**), October 14-18, 2013, Zürich, Switzerland. (Acceptance rate 18.1%, 21 out of 116 papers).

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Improving Present Security through the Detection of Past Hidden Vulnerable States.* In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (**IM'13**), Mini-Conference, May 27-31, 2013, Ghent, Belgium.

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Collaborative Remediation of Configuration Vulnerabilities in Autonomic Networks and Systems.* In Proceedings of the 8th IEEE International Conference on Network and Service Management (**CNSM'12**), Mini-Conference, October 22-26, 2012, Las Vegas, USA.

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Towards the Assessment of Distributed Vulnerabilities in Autonomic Networks and Systems.* In Proceedings of the IEEE/IFIP International Network Operations and Management Symposium (**NOMS'12**), April 16-20, 2012, Maui, Hawaii, USA. (Acceptance rate 26.2%, 55 out of 210 papers).

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Supporting Vulnerability Awareness in Autonomic Networks and Systems with OVAL.* In Proceedings of the 7th IEEE International Conference on Network and Service Management (**CNSM'11**), October 24-28, 2011, Paris, France. (Acceptance rate 14.6%, 24 out of 164 papers).

– **Martín Barrère**, Gustavo Betarte, and Marcelo Rodríguez. *Towards Machine-assisted Formal Procedures for the Collection of Digital Evidence*. In Proceedings of the 9th IEEE Annual International Conference on Privacy, Security and Trust (**PST'11**), July 19-21, 2011, Montreal, Canada.

– **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Towards Vulnerability Prevention in Autonomic Networks and Systems*. In Proceedings of the 5th International Conference on Autonomous Infrastructure, Management and Security (**AIMS'11**), Ph.D. Symposium, Springer, June 13-17, 2011, Nancy, France.

Demonstrations and seminars

– **Martín Barrère**, Gaëtan Hurel, Rémi Badonnel, and Olivier Festor. *Ovaldroid: an OVAL-based Vulnerability Assessment Framework for Android*. Demonstration Sessions of the IFIP/IEEE International Symposium on Integrated Network Management (**IM'13**), May 27-31, 2013, Ghent, Belgium.

– **Martín Barrère**. *Vulnerability Management for Safe Configurations in Autonomic Networks and Systems*. Ph.D. Seminar, **NSS Department of Loria**, March 28, 2013, Nancy, France.

– **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Ovalyzer: an OVAL to Cfengine Translator*. Ph.D. Student Demo Contest of the IEEE/IFIP International Network Operations and Management Symposium (**NOMS'12**), April 16-20, 2012, Maui, Hawaii, USA.

Bibliography

[1] Android. http://www.android.com/. Last visited on November, 2013.

[2] J. Banghart and C. Johnson. The Technical Specification for the Security Content Automation Protocol (SCAP). Nist Special Publication. http://scap.nist.gov/revision/, 2011. Last visited on January, 2013.

[3] Cfengine. http://www.cfengine.com/. Last visited on November, 2013.

[4] Chef. http://www.getchef.com/chef/. Last visited on November, 2013.

[5] S. A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[6] S. Dobson, F. Zambonelli, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, and N. Schmidt. A Survey of Autonomic Communications. *ACM Transactions on Autonomous and Adaptive Systems*, 1(2):223–259, December 2006.

[7] R. Enns, M. Bjorklund, J. Schönwälder, and A. Bierman. RFC 6241, Network Configuration Protocol (NETCONF). http://tools.ietf.org/html/rfc6241, June 2011.

[8] M. C. Huebscher and J. A. McCann. A Survey of Autonomic Computing–Degrees, Models, and Applications. *ACM Comput. Surv.*, 40:7:1–7:28, August 2008.

[9] IBM. http://www.ibm.com/. Last visited on November, 2013.

[10] IBM. An Architectural Blueprint for Autonomic Computing. *IBM White Paper*, 2006.

[11] V. Igure and R. Williams. Taxonomies of Attacks and Vulnerabilities in Computer Systems. *IEEE Communications Surveys & Tutorials*, 10(1):6–19, January 2008.

[12] Cisco IOS. http://www.cisco.com/. Last visited on November, 2013.

[13] Java technology. http://www.oracle.com/technetwork/java/. Last visited on November, 2013.

[14] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, January 2003.

[15] MITRE Corporation. http://www.mitre.org/. Last visited on November, 2013.

[16] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle. A Survey of Autonomic Network Architectures and Evaluation Criteria. *IEEE Communications Surveys & Tutorials*, PP:1–27, May 2011.

[17] NIST, National Institute of Standards and Technology. http://www.nist.gov/. Last visited on November, 2013.

[18] The OVAL Language. http://oval.mitre.org/. Last visited on November, 2013.

[19] M.R. Prasad, A. Biere, and A. Gupta. A Survey of Recent Advances in SAT-Based Formal Verification. *STTT*, 7(2):156–173, 2005.

[20] Puppet. `http://www.puppetlabs.com/`. Last visited on November, 2013.

[21] N. Samaan and A. Karmouch. Towards Autonomic Network Management: an Analysis of Current and Future Research Directions. *IEEE Communications Surveys & Tutorials*, 11(3):22–36, July 2009.

[22] Apache Subversion. `http://subversion.apache.org/`. Last visited on November, 2013.

[23] A. Williams and M. Nicolett. Improve IT Security with Vulnerability Management. `http://www.gartner.com/id=480703`, 2005. Last visited on November, 2013.

[24] G. Zhang, S. Ehlert, T. Magedanz, and D. Sisalem. Denial of Service Attack and Prevention on SIP VoIP Infrastructures using DNS Flooding. In *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'07)*, pages 57–66, New York, NY, USA, 2007. ACM.

[25] N. Ziring and S. D. Quinn. Specification for the Extensible Configuration Checklist Description Format (XCCDF). NIST (National Institute of Standards and Technology). `http://scap.nist.gov/specifications/xccdf/`. Last visited on January, 2013.