Management of Network Security Architectures

Nora Cuppens-Boulahia, Frédéric Cuppens,

Joaquin Garcia-Alfaro

IT TELECOM Bretagne CS 17607, 35576 Cesson-Sévigné, France

Abstract. Despite the advances in the field of network security technologies, such as access control, monitoring of traffic, and encrypted communications, there may always be errors that can degrade the security of a system. The use of automatic tools for the assistance of security configuration aims at providing assistance for the proper configuration of network security components — in order to optimize the degree of security deployed over the system. The configuration of the network security components is, however, very complex and error-prone. It is based on the distribution of several packages of security rules that define properties such as traffic that must be refused, traffic that must be surveyed, and traffic that must be protected. The assembly of all these properties over all the system components must be consistent, addressing always the same decisions under equivalent conditions, and avoiding conflicts or anomalies. Otherwise, the existence of misconfiguration will definitively lead to weak security architectures, potentially easy to be evaded by unauthorized parties. We provide in this paper a detailed set of functionalities that an ideal platform for the management of network security architectures shall provide to network administrators. We complement our work with an overview of available network management tools in today's market. We finally point out to some limitations of such support tools that might worth fulfilling by further research.

1 Introduction

Several components are required to build up a network security architecture, such as firewalls, intrusion detection systems, and VPN routers. These components must be properly configured in order to provide an appropriate degree of security to the system. The configuration process is a highly complex and error-prone task. There is, indeed, a clear necessity of guaranteeing the conformity of the process for managing the configuration of a network security architecture. Otherwise, the resulting system can lead to wrong configurations that do not provide the required level of security.

We set in this paper an overview of the main functionalities that any general purpose platform for the management of network security architectures must cover. Such functionalities can be summarized as follows: an appropriate expressive language capable of representing an abstract model of the security requirements and the properties of the system; an automatic process for translating and deploying such an abstract model into the concrete configuration of each security component located in the system; a complete audit process for analyzing the unavoidable modification of concrete configurations due to maintenance tasks; and an automatic process for the activation of reaction when attacks targeting the system are detected — in order to provide, for instance, eventual modifications of the abstract model that are required in order to react against the attacks.

Organization of the paper — Section 2 presents a detailed explanation of our proposed list of functionalities. Section 3 overviews available support tools in today's market that can be valuable for implementing the set of functionalities. Section 4 discusses some limitations of these tools and foresees some research perspectives. Section 5 closes the paper.

2 Functionalities of our Ideal Management Platform

We set in this section the core functionality of our ideal management platform. We recall that the goal of this platform is to assist the network administrator to obtain the appropriate configuration of network security components, according to the security requirements that must govern the system. A brief summary of the core four functions that we envision is presented below:

– The platform must provide the appropriate mechanisms to express in a unique and unambiguous manner the global policy that must rule the security of the system. Similarly, it must provide to the administrator with the appropriate language to describe the concrete information associated to the system, such as topological data and component capabilities. As a result, the platform must determine if the required degree of security expected to be enforced by the network security architecture is consistent, given the capabilities and functionalities deployed on it —- by means of the set of security components.
– If the result of previous step is positive, and the administrator can successfully define the appropriate policy that can govern the system given the components contained on it, it must then be capable of automatically deriving the appropriate set of configuration packages that makes the security system operative. Such a technical refinement can be efficiently be implemented by a downward translation of the abstract model that contains the security policy towards the concrete syntax of each component contained in the network security architecture.
– The cycle of life of the security system must consider the eventual maintenance of configurations. In other words, once the network security architecture is active, there will be daily or weekly updates, that dynamically adapts the configuration of the components to changes, updates, and any other related events. It is, then, necessary that the platform offers the administrator the appropriate mechanisms in order to verify that the current packages of configurations are still consistent to the global security policy. This third functionality must, therefore, provide an upward audit process of configurations, in order to report any eventual misconfiguration or anomaly at both single- or multi-component level.

– Finally, it is very likely that malicious activity detected by monitoring components of the network security architecture will eventually activate the reaction modules of the system. In such a case, the platform should provide the necessary mechanisms to assist the activation of responses and to update accordingly both the global security policy and the configuration of components. The former will provide a kind of short-term reaction, while the administrator — once warned — decides a more efficient reaction. The latter will simply consist on a re-deployment of the updated policy towards the affected components.

We present in the sequel a more deeply explanation of each functionality. We split, in turn, each functionality into specific technical processes; and refer the reader to appropriate literature for the implementation of each one of these processes.

2.1 Policy Expression and Architecture Discovery

The objective of this first functionality involves building an abstract model of the security requirements expected to be enforced over the system. The formal meaning of this abstract model must be unique and unambiguous. In order to provide this model, it is first necessary to receive as an input some technical aspects of the system, such as its topology and the capabilities provided by the security components that are contained on it. A reconciliation between the degree of security expected by the system, and the real set of capabilities contained within the system, will conclude whether such a unique and unambiguous set of abstract security rules is indeed possible or not. Figure 1 depicts a simplified scheme of the two main processes that we consider crucial for the appropriate implementation of this first functionality: architecture discovery and reconciliation. We detail these two processes next.

Architecture Discovery — The purpose of this first process is to determine the topological data of the networking system, the set of security components, and some associated information, such as minimal paths, optimal routes, or precomputed lists of components crossed by any given packet (e.g., knowing its source and destination). We assume that the use of existing network support tools, e.g., set of tools surveyed in Section 3, that already provide features for network discovery, such as collection of technical configurations and collection of topology network cartography at the OSI levels 2 and 3, is sufficient to collect and encapsulate in a single and unique format the resulting system architecture of this first process.

Reconciliation — The objective of this second process is to formally validate whether the organizational security requirements can be successfully applied to the obtained system architecture. Indeed, this process should ensure that the enforcement of the security requirements, given the topology of the system, and the functionalities provided

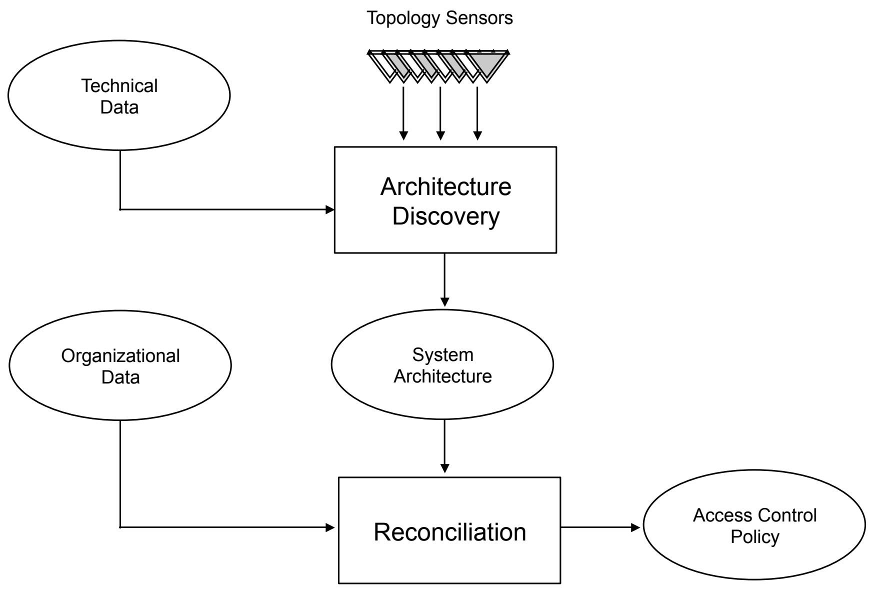

Fig. 1. Policy Expression and Architecture Discovery.

by the security components contained in such a system, is possible. Moreover, should ensure that it is possible to enforce the architecture with such a policy in a way that are not either conflict nor anomalies. Therefore, and in case that are no errors, the output data corresponds to the global security policy that is meant to govern the behavior of the system.

There exists in the literature several models in order to describe in a unique and unambiguous manner the resulting policy. However, not all the models might answer optimally the complete set of security requirements. Some models like the Discretionary Access Control (DAC) model of the Mandatory Access Control (MAC) model might require important additional administration efforts to sum up some properties that can, in turn, be satisfactorily described by more complete models, like the Role Based Access Control (RBAC) model [22] (e.g., introduction of notions such as roles and hierarchies). The same semantics proposed in the RBAC model, and many other like the use of delegation, and the activation of contexts, are inherent to the formalism of the Organization Based Access Control (OrBAC) model [1]. As we will see later, all these semantics might specially benefit the dynamic management of networking policies, specially when the system is composed by heterogeneous security mechanisms, such as filtering devices, monitoring modules, and VPN routers. The election of an appropriate model for representing the global policy is, therefore, crucial.

2.2 Downward Deployment of Technical Configurations

This second functionality addresses the downward deployment of rules by unfolding the obtained global security policy into the concrete configurations of the network security components. Notice that the previous stage (cf. Subsection 2.1) already guarantees that such a deployment can be done by translating the global policy into local configurations, while guaranteeing complete consistency. Figure 2 depicts a simplified scheme of the two main processes for the implementation of this second functionality: technical refinement and enforcement. We detail them below.

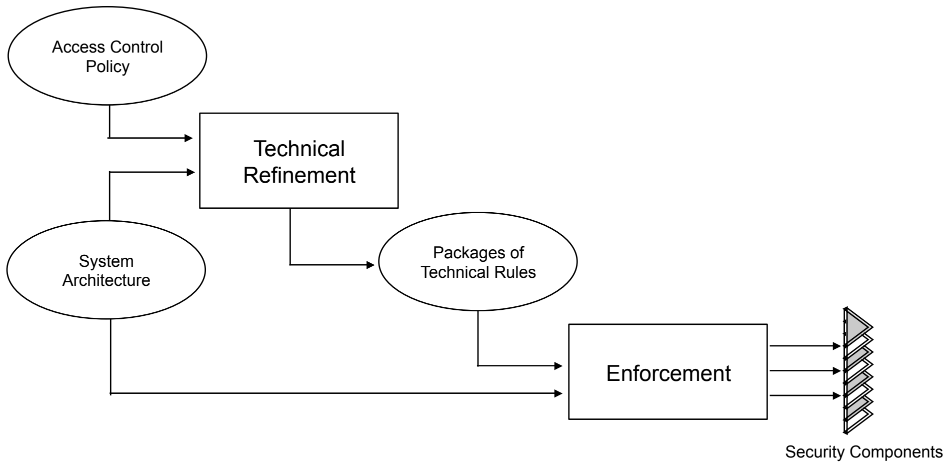

Fig. 2. Downward Deployment of Technical Configurations.

Technical Refinement — This process is in charge of translating the set of abstract rules contained within the global security policy into a set of concrete rules that will later be communicated to each security component of the architecture. The process can be seen as an iterative set of transformations, each complying with a given security technology instantiated in the system (e.g., firewalls, intrusion detection systems, or VPN routers). Each transformation can be conceived as the instantiation of those parts of the security policy that must be enforced in the system by means of the processed components. The global policy is, therefore, decomposed into several packages of local configurations.

Several proposals in the literature have been proposed for the implementation of this functionality. The main difference relays on the expressiveness of the model or language that expresses the set of abstract rules of the policy. In [6], for instance, we can find one of the earliest proposals for the refinement of a global security policy into local configurations of security components, such as filtering routers or firewalls. The approach, based on the RBAC [22] formalism, uses a traditional *Entity — Association* model that represents the abstract level. This abstract level is expected to be projected onto the network topology as a set of roles. The instantiation of the model is based on a specific language that allows a downward transformation of the global policy into a set of local firewall configurations.

The use of the role concept used in the previous approach, although valid for defining the network capabilities, becomes ambiguous in its semantics. In fact, the authors use the notion of *group* to identify topological data, such as network hosts. However, we can observe in [6] that the same notion is used later on to determine roles and sets of roles. We, therefore, consider that the approach does not ensure a clear separation between the network level and the security policy. This makes difficult the use of the proposed solution for modeling complex networks which, indeed, is very valuable in case the platform is targeting, for instance, at large enterprise configurations. The authors use, moreover, privilege inheritance through group hierarchies in order to derive permissions. If permission inheritance is related to the so-called *open group*, prohibitions are inherited through a *close group*. The notion of group clearly introduces ambiguities and seems to be useless at this abstraction level. Most of these limitations are solved in a more recent approach presented in [9]. The approach is based on the OrBAC model, and therefore benefits from inherent semantics of delegation and contexts in order to address the previous limitations. The solutions is, in turn, ameliorated in the work presented in [21], in order to include the deployment of a larger set of components, not only firewalls, but also intrusion detection systems and VPN routers.

Enforcement — The objective of this second process is to communicate the resulting packages of configuration rules towards the security components. Several communication protocols can address this issue. For example, the Simple Network Management Network (SNMP) protocol, the Common Open Policy Server (COPS) protocol, and the Netconf protocols are, actually, solutions that are easy to find implemented in most of todays' support tools (cf. Section 3 for more details). They appear, moreover, in the related literature in order to implement similar operations as the deployment method that we discussed above.

The SNMP protocol is one of the most widely used protocols by network management tools. This protocol relays on a traditional agent-manager paradigm, in which the agent is basically a program running over the equipments of a system; and the manager is a program centralized by a platform that controls and collects the execution results of each agent over the system. Most solutions of well known vendors like Cisco or ChecK Point base the synchronization and management of their tools on SNMP or other variants. In contrast, the COPS and Netconf protocols are essentially of a query-response nature. They have also been reported in recent literature aiming at exchanging policy information between servers and clients. COPS have been used in [7], for example, to dynamically distribute IPsec-based VPN policies. The main limitation reported in [15] seems to be the handling of XML encoding, as it is not natively handled by COPS, rather that other protocols like Netconf. A recent approach presented in [21] proposes the combination of the OrBAC formalism and the Netconf protocol, in order to deploy and communicate the configurations of security components over the network.

2.3 Upward Analysis of Technical Configurations

The dynamism of current networking threats forces the system to be periodically updated. This can affect both the global security policy and the local set of components' configurations which. Therefore, the system is often affected by either daily or weekly updates, with the objective of dealing with the real-world changes or threats. The management platform shall provide the necessary mechanisms in order to guarantee that the modifications do not lead to a misconfiguration of the system. In other words, it must verify that the current packages of local configurations that are enforcing the security components of the architecture are, indeed, still consistent with the global security policy. The platform must, therefore, assist the administrator to process an appropriate audit of the local configurations, and to report the existence of misconfiguration or inconsistency with the global policy. Moreover, it must propose the necessary modifications that will fix such problems — either manually, by requiring the approval of the administrator; or automatically, by reconfiguring the affected components without requiring further actions. Figure 3 depicts a simplified scheme of the main processes for the implementation of this third functionality: gathering configuration data, single-component audit, multi-component audit, and management of modifications. We detail them next.

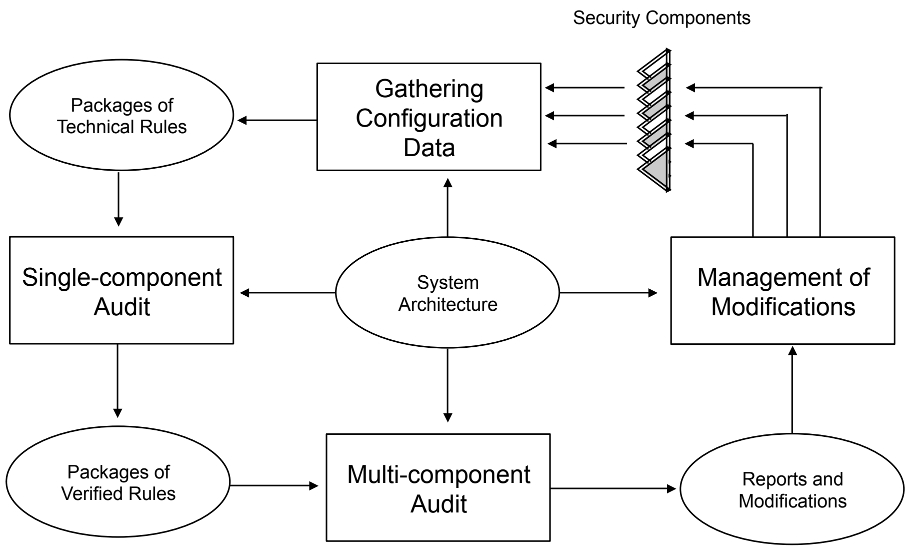
Security Components
Packages of Technical Rules
Gathering Configuration Data
Single-component Audit
System Architecture
Management of Modifications
Packages of Verified Rules
Multi-component Audit
Reports and Modifications

Fig. 3. Upward Analysis of Technical Configurations.

Gathering configuration data — This process aims at collecting the packages of local configuration rules deployed on each component of the network security architecture. Like in the architecture discovery process defined in Subsection 2.1, we also assume here that current support tools available in today's market, e.g., set of tools surveyed in Section 3, already provide the necessary features to configure the set of network components in order to divert their local configuration files towards a centralized repository of network component configurations. The use of a unique and unambiguous format shall be used in order to express and provide the resulting packages of local configuration rules derived from the gathering process.

Single-component analysis — This process is intended to detect and remove conflicts between configuration rules, that could lead to the misconfiguration of components from a stand-alone perspective. The output of this function contains a group of verified rules, free of anomalies, and the corresponding information about the detected anomalies. Several solutions for implementing this process exist in the literature. The authors in [2] consider that, in a configuration set, two rules are in conflict when the first rule in order matches some packets that match the second rule, and the second rule also matches some of the packets that match the first rule. This overlap results either on anomalies that might affect the performance of the components (e.g., redundant rules that can be removed without affecting the policy that governs the components) or security conflicts (e.g., rules that shadow other rules with less priority in the configuration, and that make inconsistent the local policy of the component to the global system policy). In [3], the authors provide a more complete taxonomy of anomalies in firewall configuration, and a set of algorithms to detect them by analyzing, as it was proposed in [2], the relationships between rules two by two. The limitation of these two approaches is that anomalies due to the union of rules are not explicitly considered.

The proposal presented in [24], uses a model checking formalism for detecting inconsistencies and redundancies in single firewall configurations. This proposal handles the detection process by addressing directly the way how traffics is handled by the components. Their solution, as well as the above mentioned approaches, only addresses the audit analysis process. They do not propose a candidate set of rules for replacing the conflicts. Moreover, none of them have presented specific mechanisms for verifying configurations other than firewall configurations. To our knowledge, just the works proposed in [10, 19] cover the above mentioned limitations. This new work considers the audit of configurations of firewalls and intrusion detection systems, processing and detecting anomalies like redundancy and shadowing. The configurations are processed as a whole. Through a rule transformation process, the proposal derives from an initial set of configuration rules, an equivalent and valid one that is completely free of misconfiguration, such as shadowing and redundancy. This rewriting of rules allows, moreover, the derivation of a new configuration set whose rules are completely disjoint. Therefore, the ordering of rules is no longer relevant.

Multi-component analysis — This third process, complemented now by the knowledge of the system architecture, must verify that the configurations of each network component, from a distributed point of view, is consistent with the global policy. It must guarantee, moreover, the interoperability between the different components. The output of this function corresponds to the group of reports and modifications generated during the verification process. Proposals like [4, 20] extend the single-component audit analysis presented in [3, 19], in order to detect multi-component deployments that are not consistent. The approach presented in [4] provides a correlation process that compares the configuration rules of distributed architectures and derives inconsistencies hidden in their configurations. The detection process is based again on the comparison of rules two by two. Therefore, errors due to the union of rules are still not properly handled. The approach presented in [20] solves this limitation and allows, moreover, the audit of distributed architectures where firewalls and intrusion detection systems are in charge of enforcing the global policy. The detection checks, indeed, if there are errors in the distributed configurations by comparing the policy decision that should rule over each component that matches the exactly same traffic.

Regarding the analysis of VPN routers' configurations, a significant approach is the proposal presented in [16]. The authors provide a technique that simulates VPN tunneling processing and reports any violation of the security policy requirements. In their approach, if an access rule concerning a protected traffic between two points is implemented by configuring more than one VPN overlapping tunnel, the risk is that in some network zones the IP packets circulate without any protection. The authors present a discovery process to detect such situations and propose a high-level language to deal with VPN policies. An important limitation is that, although it can discover some violations in certain simulation scenarios, there is no guarantee that it discovers every possible misconfiguration present in the system. Moreover, the proposed technique only discovers VPN conflicts resulting from incorrect tunnel overlaps, but it does not address the complete taxonomy of all possible conflicts. Another attempt to audit and fix VPN misconfiguration is presented in [5], where the authors propose a central-entity approach with high level language conflict resolution techniques also. Unfortunately, their proposed algorithms remain unevaluated.

Management of modifications — The complete set of results and candidate modifications provided by the previous processes, and intended for fixing the detected misconfiguration or inconsistencies, should be centralized and reported to the administrator. The management platform should also leave open the option of either simply reporting the analyses results or automatically enforcing the proposed modifications. The former case means that the platform would simply inform the administrator about the results of the single and multi-component analyses. The administrator is, therefore, in charge of verifying the reports and manually activating the appropriate modifications. The alternative option is to directly accept the proposed modifications and re-enforce the security components that would require such a re-configuration. In both cases, a communication protocol is required. Valid alternatives are protocols like SNMP, COPS or Netconf. These protocols have already been overviewed in Subsection 2.2.

2.4 Handling of Reaction and Re-deployment of Configurations

The objective of this fourth functionality is the activation of reaction. We assume here the use of appropriate components for the management of alerts based, for instance, on the analysis of attack descriptions defined as actions that violate the global policy. We also assume that such components can anticipate the occurrence of possible attack scenarios, and that produce the appropriate diagnosis and candidate counter-measures to neutralize the attacks. For example, if the attacks have being specified in the policy as prohibitions, the achievement of an attack has to be interpreted as a violation of the security policy, and detected by monitoring the set of events collected by the security components of the network security architecture. They are, therefore, treated in a management process by a monitoring system. Figure 3 depicts a simplified scheme of the main processes for the implementation of this fourth functionality: monitoring, policy update, and re-deployment. We detail these processes next.

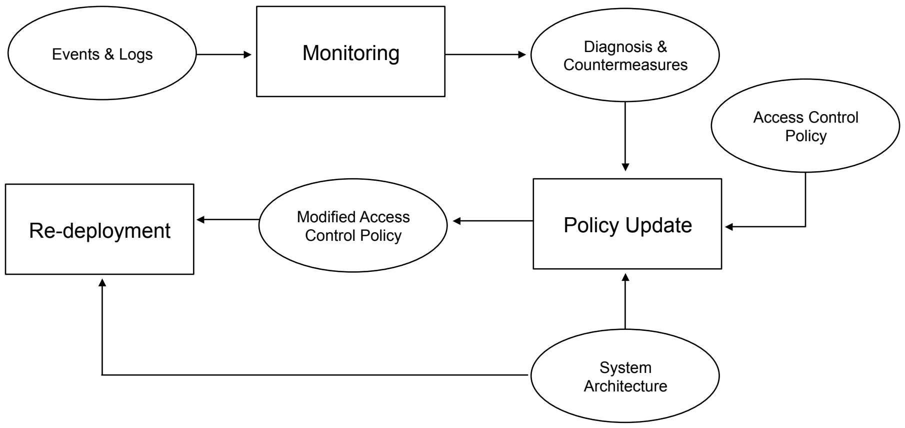

Fig. 4. Reaction and Re-deployment of Configurations.

Monitoring — This process must provide real time diagnosis of the system state to detect whether the system has been victim of an attack. We assume that the goal of a monitoring process is to provide to either the administrator or the management platform the appropriate set of concrete counter-measures. We also assume that these counter-measures where designed on the basis of a response mechanism that will eventually update the global security policy by activating, for example, conditions already defined within the policy (e.g., specified in the abstract model as logical predicates that can either be enabled or disabled). There are several proposals defined in the literature in order to implement this process. The analysis of this proposals is beyond the scope of this paper. We recommend the reader references like [8, 12, 13], and citations thereof, for more information.

Policy Update — This second process must offer the necessary mechanisms to make effective the update of the global policy. This function establishes the link between the policy and the counter-measures raised by the monitoring process — with the purpose of reacting against policy violations as a dynamic policy enforcement. This policy update aims at short-term reactions. For example, to block the access to those services or resources that are the target of the detected attacks — while expecting that the administrator will later enable some kind of long-term reaction. Existing solutions in the literature provide the means for implementing this process. For example, the proposals presented in [17, 23] provide a network filtering policy update based on the processing of intrusion detection alerts. A centralized server that controls the policy and receives the intrusion alerts, is in charge of deciding the necessary modifications in order to react to the detected threats. Unfortunately, these solutions do not provide a clear indication about how the response strategy or the mapping from alerts to counter-measures should later be handled. Two more complete solutions are presented in [12, 13]. In [13], the mapping from alerts to counter-measures is appropriately defined as a mapping between concrete alerts (expressed in the IDMEF model [11]) and OrBAC contexts [1]. The OrBAC contexts represent the necessary reaction that must be enforced when the IDMEF alerts are detected by the monitoring system. In [12], the authors provide, moreover, the specific instantiation process for updating the global policy as a result of the alerts provided by the monitoring process.

Re-deployement — This last process must offer the administrator the necessary mechanisms to re-enforce the configuration of the security components, according to the previously updated security policy. We assume that this function is equivalent to the deployment function presented in Section 2.2.

3 Available Network Management Tools in Today's Market

We survey in this section state-of-the-art solutions taken by the industry that might be useful for the implementation of the platform functionalities.

The current market of support tools for the management of network security architectures is dominated by the following two industrial leaders: Cisco and Check Point. Their main products are the Cisco Security Manager and the Check Point SmartCenter. These support tools may assist the administrator in a, relatively, low level language. These two state-of-the-art tools provide a very complete set of networking management features. The core functionality relays on the systematization of intrusion detection alert management, deployment of component configurations, and management of VPN routers. They offer additional features for network discovery, such as collection of technical configurations and collection of topology network cartography at the

OSI levels 2 and 3. They also provide, in addition to the traditional security management operations, historical storage of software/firmware update and modifications of technical configurations, management of fault-tolerance support, centralization of fault-tolerance procedures, and management of workflow assistance for the enforcement of security and fault tolerance technologies. We can, however, point out some limitations when using these tools. First, and most important, these solutions are mono-constructor. They only provide the above mentioned set of operations in order to assist the administrations of, respectively, Cisco and Check Point products. Second, they do not offer a semantic model rich enough to express a complete global security policy that might integrate notions such as contexts and hierarchies. Although it is possible to define some variables, and thus to define access rules involving such variables, the administration tasks are not much simplified. The administrator always needs a global view of the topology in order to correctly assign each rule to network devices; then, there is no automatic discovery of security devices that optimally implement an access rule involving an IP source and a destination. Furthermore, the lack of a real downward deployment approach is partially replaced by other tools (e.g., the Cisco conflict discovery tools) that need the security officer's assistance and that unfortunately only guarantee conflict resolution for local configurations.

The following vendors also offer pertinent solutions: IBM, LogLogic, Juniper Networks, and Tufin. They provide some more general purpose administration tools, like the IBM Proventia Desktop Endpoint Security; the LogLogic Security Change Manager, formerly known as Solsoft Policy Server and Network Security Policy Server; the Juniper Network and Security Manager, and the Tufin SecureTrack and SecureChange. The main advantage of these four solutions is that they can interact with third party network security components (e.g., firewalls, VPN routers and intrusion detection systems manufactured by well-known companies like Cisco and Check Point). Compared with the above mentioned solutions by Cisco and Check Point, these other tools try to integrate more open standards — specially standards based on XML and SOAP — in order to ease the integration of future third party products. While the definition of policies by the solution of IBM and Juniper offers RBAC-like policy definitions, the remainder solutions by LogLogic and Tufin still relays on the use of relatively low-level technical languages, too close to the original syntax and semantics of the components intended to be administered. Some other extended features offered by these solutions are: handling of operating systems, firmware and antivirus updates; analysis of threats; management of workflows; and management of rollbacks (e.g., aiming at managing the restoration of previous architecture configurations).

Finally, it is worth mentioning the existence of some relevant open source solutions, like the product Firewall Builder. Indeed, this solution comes with a GPL Open Source license for its use on GPL-based Unix-like systems (such as GNU/linux and freebsd) and a commercial license for its use on commercials systems like MS-Windows and Apple Mac OS. Although the Firewall Builder only allows the configuration of firewalls, it allows the management of different vendors' solutions and the deployment of large configurations on heterogeneous networks. The main limitation of this solution is, however, the definition of a centralized security policy in a relatively poor policy definition language.

4 Limitations and Perspectives

We have already mentioned in the previous sections some problems related to the state-of-the-art solutions in today's commercial market and academia. We group in this section some of the limitations, in terms of relevance, and summarize several areas of study that we envision as important in order to move towards the ideal management platform addressed in our paper.

The first, and most important limitation, is the lack of semantics for the expression of the global security policy. Indeed, the lack of an appropriate language, rich enough to express all the semantics that must be included in global policy, limits the power of the functionalities we presented in Section 2. Indeed, support tools for the management of large network security architectures depending on a poor semantic abstract model to represent the global properties of the security policy will hardly help the administrator to escape the specificity of the components that are supposed to be managed. The second limitation we want to stress is the lack of stateful analysis of configurations. The discovery functions of the surveyed solutions are mostly done at a very low-level. They lack of an appropriate development to perform the analysis of misconfiguration at a more abstracted level which would help, indeed, on the analysis of stateful firewall configurations — an area that is still to be explored by most commercial applications. Some research solutions has recently been presented by Fitzgerald et al. in [14]. We do not find, however, appropriate solutions for doing this kind of analysis on commercial products. Finally, the third limitation we want to address is the high dependency of some important solutions (specially those manufactured by Cisco and Check Point vendors) to mono-constructor platforms. This limitation leads to an analysis of technical configurations that remains very close to the original syntax of the configuration rules that must be processed. This problem limits the interoperability of most commercial solutions and make difficult a full integration of third party components.

Before closing this section, we consider two interesting perspectives that are worth exploring in order to highly improve the efficiency of the surveyed state-of-the-art commercial solutions presented in Section 3. A very unexplored area in most of the surveyed solutions is the integration of a guided process to complement the upward analysis of technical configurations (cf. Section 2.3) with an automatic discovery of roles associated to the different security components already deployed in the system. The use of role mining techniques [18], for example, would highly benefit the administration tasks by discovering access control roles associated to the components — to derive, after the analysis, the appropriate rules of the global configuration. Another interesting perspective is the inclusion of workflow management. Large-scale architectures will highly benefit for some kind of automatic management to enable a more efficient enforcement of daily administration activities that should include inclusion of new rules, removal of unnecessary conditions, and so on. The challenge is finding an effective link between workflow management and policy deployment, in order to optimize the enforcement of administrative operations as a whole.

5 Conclusion

The aim of a platform for the management of a network security architecture is to simplify the work of the administrator that is in charge of the whole architecture and its security components. The platform has to provide the following functions: (1) assistance to the conception (expression) of the architecture; (2) assistance to the automatic configuration (deployment) of the components of the architecture; (3) detection and removal of misconfiguration; and (4) supervision and assistance to activation of countermeasures for a proper update and redeployment of the policy when malicious activities and policy violations are detected. The study of state-of-the-art support tools in today's industrial market allows us to claim the following limitations: lack of appropriate semantics for the description of policies; lack of stateful analysis of configurations; and high dependency of mono-constructor solutions. We have envisioned the following perspectives: integration of a role mining process to complement the upward analysis of technical configurations; and inclusion of workflow management to enable a more efficient enforcement of daily administration activities.

References

1. A. Abou el Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel and G. Trouessin. Organization Based Access Control. IEEE 4th Intl. Workshop on Policies for Distributed Systems and Networks, pp. 120–131, Lake Come, Italy, 2003.
2. H. Adiseshu, S. Suri, and G. Parulkar. Detecting and resolving packet filter conflicts. *Joint Conference of the IEEE Computer and Communications Societies*, pp. 1203–1212, 2000.
3. E. S. Al-Shaer and H. H. Hamed. Firewall Policy Advisor for Anomaly Discovery and Rule Editing. In *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003)*, pp. 17–30, 2003.
4. E. S. Al-Shaer and H. H. Hamed. Taxonomy of Conflicts in Network Security Policies. In *IEEE Communications Magazine*, 44(3), March, 2006.
5. S. Baek, M. Jeong, J. Park, T. Chung. Policy based Hybrid Management Architecture for IP-based VPN. *In Network Operations and Management Symposium*, NOMS 2000.
6. Y. Bartal, A. Mayer, K. Nissim, A. Wool. Firmato: A novel firewall management toolkit. In *20th IEEE Symposium on Security and Privacy*, Oakland, California, 1999.
7. F. Clemente, G. Lopez, G. Martinez, and A. Gomez-Skarmeta. Deployment of a Policy-Based Management System for the Dynamic Provision of IPsec-Based VPNs in IPv6 Networks. In *2005 Symposium on Applications and the Internet Workshops*, pp.10–13, 2005.
8. F. Cuppens, F. Autrel, Y. Bouzida, J. Garcia-Alfaro, S. Gombault, and T. Sans. Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework. *Annals of Telecommunications*, 61(1-2):192–217, 2006.

9. F. Cuppens, N. Cuppens, T. Sans, and A. Miège. A formal approach to specify and deploy a network security policy *Second Workshop on Formal Aspects in Security and Trust*, Toulouse, France, August 2004, pp. 203–218.
10. F. Cuppens, N. Cuppens, and J. Garcia-Alfaro. Misconfiguration management of network security components. *7th International Symposium on System and Information Security (SSI 2005)*, Sao Paulo, Brazil, November 2005, pp. 1–10.
11. H. Debar, D. Curry, and B. Feinstein. Intrusion detection message exchange format data model and extensible markup language. *Request for Comments 4765*, March 2007.
12. H. Debar, Y. Thomas, F. Cuppens, and N. Cuppens-Boulahia. Enabling Automated Threat Response through the Use of a Dynamic Security Policy. In *Journal in Computer Virology (JCV)*, 3(3):195-210, 2007.
13. H. Debar, Y. Thomas, F. Cuppens, and N. Cuppens-Boulahia. Using contextual security policies for threat response. In *Third GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA)*, Berlin, Germany, 2006.
14. W. M. Fitzgerald, M. O. Foghlu, and S. N. Foley. Network access control configuration management using semantic web techniques. *Journal of Research and Practice in Information Technology*, 41(2):99–118, 2009.
15. T. Franco, W. Lima, G. Silvestrin, R. Pereira, M. Almeida, L. Tarouco, L. Granville, A. Beller, E. Jamhour, and M. Fonseca. Substituting COPS-PR: An Evaluation of NETCONF and SOAP for Policy Provisioning. In *7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pp. 195–204, USA, 2006.
16. Z. Fu, et al. IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution. *International Policy Workshop*. January 2001.
17. K. Hwang and M. Gangadhran. Micro-Firewalls for Dynamic Network Security with Distributed Intrusion Detection. In *International Symp. on Network Computing and Applications*, 2001.
18. M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining-revealing business roles for security administration using data mining technology In *Eighth ACM symposium on Access control models and technologies*, 2003.
19. J. Garcia-Alfaro, F. Cuppens, and N. Cuppens-Boulahia. Towards Filtering and Alerting Rule Rewriting on Single-Component Policies. In *Intl. Conference on Computer Safety, Reliability, and Security (Safecomp 2006)*, pp. 182–194, Gdansk, Poland, 2006.
20. J. Garcia-Alfaro, F. Cuppens, and N. Cuppens-Boulahia. Complete Analysis of Configuration Rules to Guarantee Reliable Network Security Policies. *International Journal of Information Security*, Springer, 7(2):103–122, April 2008.
21. S. Preda, F. Cuppens, N. Cuppens-Boulahia, J. Garcia-Alfaro, L. Toutain, and Y. Elrakaiby. A Semantic Context Aware Security Policy Deployment. *ACM Symposium on Information, Computer and Communications Security*, pp. 251–261, Sydney, Australia, March 2009.
22. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.
23. F. Xian, H. Jin, K. Liu, and Z. Han. A Mobile-Agent based Distributed Dynamic μFirewall Architecture. In *9th International Conf. on Parallel and Distributed Systems*, pp. 431-436, 2002.
24. L. Yuan, J. Mai, S. Su, H. Chen, C. Chuah, and P. Mohapatra. FIREMAN: a toolkit for FIREwall Modeling and ANalysis. In *IEEE Symposium on Security and Privacy*, pp. 199–213, Oakland, California, 2006.