J Ambient Intell Human Comput (2013) 4:223–234
DOI 10.1007/s12652-011-0069-8

ORIGINAL RESEARCH

An intelligent security architecture for distributed firewalling environments

Alfredo De Santis · Aniello Castiglione · Ugo Fiore · Francesco Palmieri

Received: 2 April 2011 / Accepted: 8 September 2011 / Published online: 30 September 2011

Abstract Due to the increasing threat of attacks and malicious activities, the use of firewall technology is an important milestone toward making networks of any complexity and size secure. Unfortunately, the inherent difficulties in designing and managing firewall policies within modern highly distributed, dynamic and heterogeneous environments might greatly limit the effectiveness of firewall security. It is therefore desirable to automate as much as possible the firewall configuration process. Accordingly, this work presents a new more active and scalable firewalling architecture based on dynamic and adaptive policy management facilities, thus enabling the automatic generation of new rules and policies to ensure a timely response in detecting unusual traffic activity as well as identify unknown potential attacks (zero-day). The proposed scheme, with a multi-stage modular structure, can be easily applied to a distributed security environment and does not depend on any specific security solutions or hardware/software packages.

keywords Distributed security · Multi-firewall systems · Automated firewall management · Adaptive rule and rule generation

A. De Santis · A. Castiglione
Dipartimento di Informatica "R.M. Capocelli",
Università degli Studi di Salerno, Via Ponte don Melillo,
84084 Fisciano, SA, Italy
e-mail: ads@dia.unisa.it

A. Castiglione
e-mail: castiglione@ieee.org; castiglione@acm.org

U. Fiore
Information Services Center, Università degli Studi di Napoli
"Federico II", Via Cinthia 5, 80126 Naples, Italy
e-mail: ugo.fiore@unina.it

F. Palmieri (✉)
Dipartimento di Ingegneria dell'Informazione, Seconda
Università degli Studi di Napoli, Via Roma 29,
81031 Aversa, CE, Italy
e-mail: francesco.palmieri@unina.it

1 Introduction

Firewall systems, often consisting of several devices distributed across the network, filter out unwanted or unauthorized outgoing or incoming traffic in accordance with rule sets related to domain-specific security policies and requirements. Security policies define what is permitted and what is prohibited during normal operation, by identifying limitations and constraints on data management and communications. In a multifaceted and rapidly evolving network environment, the increasing complexity of these security policies makes their design, implementation and maintenance more problematic and error-prone. Security devices and polices risk losing effectiveness.

A poorly created set of rules can become a performance bottleneck, for example when improper rule ordering causes rarely triggered rules to be unnecessarily frequently checked. On the other hand, the whole security system is as effective as it is able to adapt to the current characteristics and volumes of the traffic flowing through the network. The analysis of the current traffic trends may reveal that some rules are outdated or have not been used for a long time. In addition, classic access control lists can often result in conflicts between policies and could originate security holes. Finally, the examination of server and network logs may be used to assess the degree of consistency and adequacy of the current firewall policy rules with respect to the actual network

services and traffic profiles, in compliance with the associated security objectives. Nevertheless, as the number of rules increases beyond the amount reasonably manageable by a manual process, the task of managing complex firewalling policies becomes extremely difficult and time-consuming. As a consequence, enhanced policy management techniques and tools that almost totally automate the generation, validation and optimization of firewall rules become essential for the effective administration of complex security systems.

This work consolidates and extends the previous one by Castiglione et al. (2010), where an adaptive and scalable firewalling architecture was proposed, aimed at modifying policies through the automatic generation of new effective rules and, at the same time, keeping security policies efficient and up-to-date through the reorganization and optimization of rule sets. The resulting multistage modular security system is independent from specific hardware/software packages and is amenable to a distributed implementation. A dynamic, automatic definition of new security rules adapted to observed network events increases the system effectiveness against unknown (zero-day) security outbreaks.

The main contribution of this paper is to refine and improve the integrated architecture introduced by Castiglione et al. (2010), with the addition of a new inference mechanism as well as an entirely new module: the first aims to improve the analysis of observed data, while the second takes care of communication towards external systems and networks. The effective communication of interesting events assists administrators with a quicker and more effective anomaly analysis, avoidance and prevention.

2 Related works

The challenges in firewall technologies have created a substantial amount of attention among the research community. Individual security aspects discussed in this work, such as the gap between access control requirements and rule sets, the high complexity of rule set design and management, rule set consistency and redundancy, have been dealt with in many papers.

The FANG system (Mayer et al. 2000) can derive a policy model by reverse engineering firewall configurations. A recent work presented by Abedin et al. (2010) focuses on the generation of firewall rules as a result of the application of data mining techniques on firewall log files. These rules were generalized, according to an appropriately created model, with an algorithm for the discovery of potential anomalies being consecutively applied to the rules.

An online clustering mechanism whose aim is to generate traffic-driven firewall policies is proposed in Samak and Al-Shaer (2010). In Samak et al. (2009), a probabilistic learning approach is used, where domain information will be learned as transition probabilities for a specific policy grammar supported by the target security device.

This work differs from the others in many aspects: the framework, being based on an abstract model, is more general with respect to the specific firewall used. At the same time, emphasizing system modularity, the data categories to be analyzed are extended, also including system log files and warnings raised by external IDS/IPS. On the other hand, many research groups have proposed models and languages to represent access control policies, with the aim of simplifying the syntax, abstracting from the details of low-level firewall languages, and of separating the security policy from the network topology completely. A good review of these languages can be found in (di Vimercati et al. 2007). Most works introducing models and languages include components dedicated to isolating and identifying inconsistencies and redundancies. However, they lack distributed conflict removal capabilities.

In addition, there are graphical tools that aim at simplifying the creation of rule sets. One of the most complete ones is Firewall Builder from NetCitadel LLC (2010), which creates an object-oriented firewall model, compiling it into many low-level firewall languages. The problem of firewall ACL consistency has been addressed by many studies, which propose algorithms that work directly with rule sets. Hamed and Al-Shaer (2006) defined a complete inconsistency model for firewall rule sets. However, their approach can only detect and diagnose inconsistencies between pairs of rules and does not analyze problems with a combination of more than two rules.

Recently, commercial vendor-independent tools have been made available, including those from SkyBox Inc. (2011) and RedSeal Inc. (2011). Skybox Firewall Assurance and Red Seal Network Advisor derive the security topology from routing, comparing firewall rules against either an abstractly specified policy or some recommended best practices. The best ideas from the aforementioned schemes and models were taken into consideration and combined into a uniform consistent firewall security framework, by proposing an integrated multi-stage architecture that takes advantage of all the benefits of automatic generation, optimization as well as deployment.

3 New developments in firewall solutions

The need to effectively compete against network attacks has brought about noteworthy advances in firewall systems. A firewall is a network element whose purpose is to selectively control over flows crossing the boundaries of a secured network, by using several collections of filtering

rules, implementing specific security policies. A filtering rule can be formally defined as:

$$R_i : \{C_i\} \rightarrow D_i \quad (1)$$

where i indicates the rule position into the ordered list of rules $R_1, \ldots R_n$, D_i is a boolean expression referring to the associated traffic filtering actions, and $\{C_i\}$ is a conjunctive set of condition attributes, combined into specific conditional matching expressions E_k so that

$$\{C_i\} = E_1 \cap E_2 \cap \cdots \cap E_{i_n}, \quad (2)$$

with i_n being the number of conditional expressions filtering rule i.

A list of ordered filtering rules, therefore, states the actions to be carried out on the flows satisfying the specific conditions set in the matching expressions. The set of matching expressions E_i (with $1 \leq i \leq n$) in a generic rule is made up of a set of traffic attributes, which may include source and destination IP addresses and ports, header flags and protocol type. They indicate the values, or value ranges, which the corresponding fields in the network packets should have in order for the rule to be applicable. When all the conditions of a rule are met, the filtering actions of that rule define what to do with the flow under analysis. The action can either be to "allow" traffic transit, with the packets being forwarded, or to "deny" it, with the packets being discarded. Should all the clauses in the matching part not be satisfied, the following rule is not disregarded until either a matching rule is found or a default action, usually denial, is carried out. Rule ordering therefore has a vital role.

3.1 Multi-firewall systems

Several firewalls can work together, under a common control, within the same enterprise network. Network administrators take advantage from this type of architecture, since it guarantees local control for each domain according to its specific security requirements and application needs. For example, a specific type of traffic (e.g. multicast) can be explicitly blocked in several parts of the network, while considered essential elsewhere.

Let F be the set of firewalls collaborating within the context of a common security domain. Each firewall f_i, belonging to F, partitions the whole network into a set of security zones $Z = \{z_1, \ldots, z_n\}$, with each one being related to the address space associated to the filtering interfaces of the corresponding firewall.

Two firewalls f_i and f_j are connected when there is at least one interface directly (passing through no other firewalls) or indirectly connecting firewall f_i to firewall f_j. Similarly, a security zone z_i is defined as adjacent to the firewall f_i when it is directly interfaced to firewall f_i. A pair of zones in Z can be considered as mutually disjoint when, if z_i and z_j (with $i \neq j$) belong to Z, then $z_i \cap z_j = 0$.

From this assumption, the overall security domain can therefore be represented as a graph $G = (V, E)$, where V is the set of firewalls and zones ($V = F \cup Z$) and E the set of links, where a link between two nodes $u, v \in V\exists$ only if u is connected to v or vice-versa.

In more detail, all the zones will be edge nodes of the graph G while the firewalls in F will be the intermediate nodes. It is therefore possible to define a set of paths P on the graph G associated to all the possible zones within the security domain whose address spaces are potentially reachable from each other. It can be assumed that all the traffic streams flowing between the address spaces associated to the zone z_x as well as zone z_y passes across multiple firewalls cascaded onto the network path between the two zones. A firewall, on this path, preceding another one in the direction of the traffic flow, is called an upstream firewall, whereas another firewall following it is indicated as a downstream one.

3.2 Detailed traffic analysis

There are various types of firewall, which are classified according to their capabilities as well as the protocol layer at which they act. The first type is known as a packet filtering firewall. In day-to-day operations, packet filtering techniques are used to make the majority of the network communication infrastructures and protocols safer, since they can be simply implemented within almost any network company framework. However, packet filtering cannot inspect any upper-layer payload. Moreover, stateful inspection firewalls include the classical firewall functions of intelligent protocol-layer information management. These firewalls have the same limitations and strengths of packet filtering ones. Notwithstanding the ability of a stateful inspection firewall to add new assessment functions to the transport-layer of a network, it can only statically process packets. Firewalls do not generally disregard the content of packets across open ports.

In order to prevent malicious self-propagating worms and virus attacks from entering into intranets, the dynamic and application-aware filtering of data packets is compulsory. Consequently, the use of deep packet inspection technology (DPI) inside stateful firewalls is considered a key innovation along with both intrusion detection systems (IDS) as well as intrusion prevention system (IPS) which are currently being used in traditional firewall technology. DPI refers to the ability of a firewall to look inside either the payload of a packet or within a traffic data flow and make decisions depending on the content of the data inspected. DPI implementations rely on a combination of

signature analysis and heuristic-matching technologies. Despite the fact that DPI seems highly interesting, it is quite complex to achieve in practice. Firewalls capable of DPI functions must keep track of all the network connections while simultaneously maintaining the status of the application that is in charge with that communication channel. The possibility to move the control of the data content into a network firewall gives the network administrator better tools with which to protect the networks from malicious data flow and related attacks. DPI is a significant support in contrasting these types of attack, with the changing of the identification place as well as the reaction in the firewall, by implementing the countermeasure of closing the communication with the attacker in a given network point. Unfortunately, while sufficiently granular in their traffic analysis capabilities and efficient in real-time response, integrated security systems based only on the previously mentioned technologies are impractical when exposed to either completely novel attacks, or even slight modifications of already known ones where the attack pattern does not match stored signatures or known communication behavior.

Furthermore, other attacks (hidden or stealth activities) have been explicitly conceived to present traffic patterns that are totally undistinguishable from normal ones and can therefore be considered as a structural part of the overall network traffic. In order to cope with these threats, new features, ideas and components have to be introduced into security architectures, aiming at effectively detecting the "manifestations" of each attack or suspicious network activity rather than the explicit mechanism behind it, which is clearly unknown for zero-day attacks.

Accordingly, the identification of circumstances where network behavior deviates from its operational normality, known as anomaly detection, is also creating considerable interest among the security community. Anomaly detection schemes strive to define, by using several flavors of machine learning techniques, precisely what normal behavior should look like, and then detect variations from the baseline and compute their extent. Advanced methods for anomaly detection rely on techniques including principal component analysis (Lakhina et al. 2004), entropy estimation (Gu et al. 2005) and nonlinear methods (Palmieri and Fiore 2010). All these schemes are either implemented into advanced IDS/IPS systems or as additional passive sensor devices with advanced network tapping capabilities, communicating with firewall devices.

Honeypots are other passive sensing systems which are designed to look like unpatched, vulnerable hosts, while in reality they are closely monitored. Their purpose is to be probed and compromised, and as such they carry fictitious services, only apparently related to the real ones in the organization. Generally implemented as virtual machines, honeypots log and signal scanning and exploiting activities. They are very useful to identify compromised hosts used to launch attacks as well as to produce forewarnings about new attacks.

In conclusion, other more traditional network security devices such as application-proxy firewalls and gateways can be integrated into the security architectures in order to provide additional capabilities such as smart logging and user authentication. They are generally more suitable to control spoofing attacks. However, this application-proxies are not able to handle real-time and high-bandwidth usage.

4 The reference architecture

The reference architecture introduced by Castiglione et al. (2010) has been expanded and is now structured into six separate modules, operating in pipeline (as shown in Fig. 1). Each module implements a specific task and communicates the results to other modules, which treat these, besides other sources of information, as input. In detail, the analyzer module (see Sect. 4.1) has the role of extracting, by means of data mining techniques, significant alert information from network traffic traces and log files, and integrating such information with the other ones coming from companion systems such as IDS/IPS, anomaly detectors and honeypots/honeynets. It also should allow network operators to manually introduce specific input to the following module associated with security alerts. The Analyzer's results are input to the rule generator module (see Sect. 4.2) which uses them to generate, on the firewall devices cooperating within the security domain, the filtering rules that are necessary to cope with the supplied security alerts. The third module (the optimizer module as described in Sect. 4.3) optimizes the generated rules, whereas the fourth module (the conflict remover module in Sect. 4.4) detects and removes any resulting timing conflicts within the rules, preparing the translation and deployment in a distributed and heterogeneous network environment performed by the fifth module, the deployer module (presented in Sect. 4.5). The final module (the first position in Fig. 1) is the communicator module, ensuring the interaction of the whole distributed environment with other cooperating domains, for the sake of sharing alerts and containment strategies (in the form of meta rules) generated by neighbor analyzer modules. In detail, it interfaces the analyzer module of a domain with the rule generator of another and vice-versa.

The benefits of a modular approach include the possibility to independently implement and tune the separate components that carry out the required functions. The architecture modularity, together with the use of abstraction to isolate, whenever possible, the inner workings of the

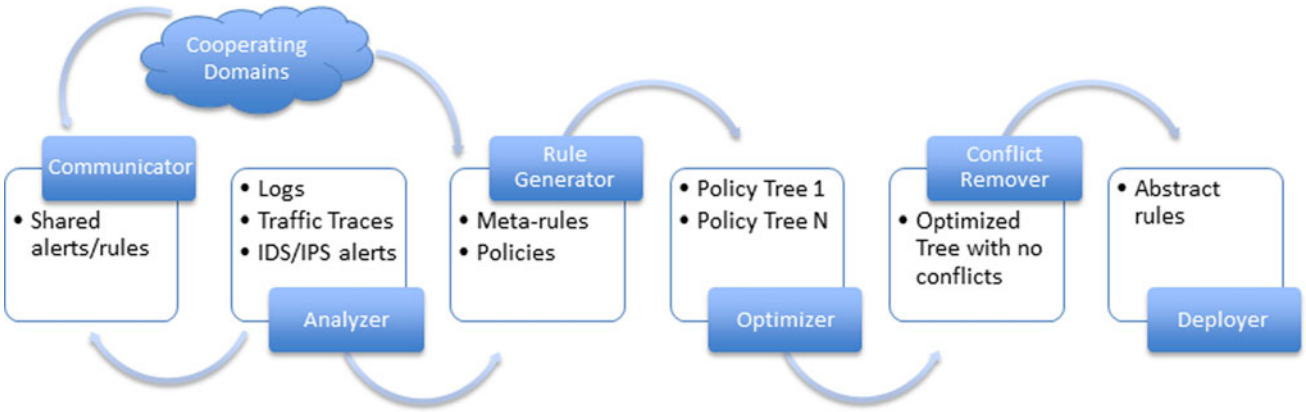

Fig. 1 The pipelined security system architecture

single modules and of the firewalls, bring about scalability, easing the application to large, complex, and diversified security environments. In addition, keeping in mind that some activities in the complete security management lifecycle are much more expensive than others—in particular requiring more computational time—modular design helps desynchronize the various activities between themselves. Separate thresholds can be set up for the various modules, effectively allowing the system to be fine-tuned to the characteristics, requirements, and policies of the operating environment. Another key aspect is the possibility to leverage upon multiple sources of information. In particular, it is believed that the operators must have the possibility to simply specify particular events, or behaviors that should be monitored. This information may result, for example, from security bulletins or similar sources, leaving the possibility to integrate the architecture with modules which handle the automatic broadcasting of such information. Ideally, the resultant module chain should be cross-platform and capable of running on Unix-like systems. Main target systems and their corresponding firewall solutions can be, for example, ipfw and pf on FreeBSD, iptables and ipchains on Linux and ACL on Cisco-like devices.

4.1 The analyzer module

Any security system is effective to the extent to which it is capable of reflecting the characteristics of its operating environment and adapting to them. In a complex and distributed environment, in which multiple firewall and IDS/IPS devices cooperate within a common security domain, it is very difficult to review/monitor all the logging information generated by the security devices and to interpret it in order to discover any real threats and deploy the necessary countermeasures. Accordingly, the analyzer module is meant to create a connection between traffic monitoring and security-relevant scenarios, by automating, as much as possible, the administrators efforts in log files analysis, dynamically identifying attack patterns and objectives as well as producing simple response strategies in the form of filtering rules. In Castiglione et al. (2010), this is accomplished by applying data-mining techniques to traffic traces and network/systems logs (see Bashah et al. (2005), Frigault and Wang (2008), and Vaarandi and Podins (2010)). Meta-rules specify the information sources and translation rules in order to achieve a common format (normalized alert information), as well as the interesting patterns to be looked for. A priori analysis produces association rules which uncover correlations between network measurements and activity profiles, which are the basis for building a classifier that distinguishes important alerts from irrelevant ones.

This work proposes the addition of a new supplementary strategy for the analysis and correlation of raw data, such as log file excerpts and IDS/IPS/honeypots signals, based on Bayesian networks. Current IDS/IPS system warnings contain a degree of uncertainty, which will show up in terms of false positives and false negatives. Nevertheless, any firewall architecture aiming at being useful in practice may not afford to disregard information about possible causal relationships between observed patterns. The challenge is, therefore, to model uncertainty so that it can be appropriately managed. Bayesian networks have been used by Frigault and Wang (2008) to model uncertainty within the context of security systems. A Bayesian network is a directed acyclic graph (DAG) whose nodes symbolize variables of interest, and whose directed links account for the cause-and-effect rapport between them.

The simple Bayesian network in Fig. 2 depicts the casual relationship between three variables, X_i ($i \in \{1, 2, 3\}$). Variables X_1 and X_2 are independent, i.e. $Pr(X_1|X_2 = Pr(X_1)$, whereas X_3 is dependent on both X_1 and X_2.

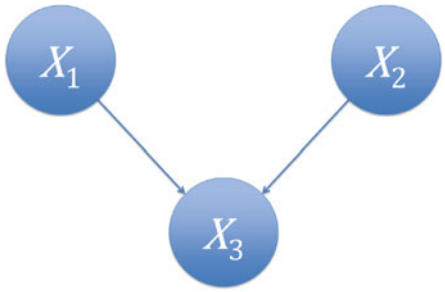

Fig. 2 A very simple Bayesian network

X_1		X_2		X_3			
T	F	T	F	X_1	X_2	T	F
0.2	0.8	0.4	0.6	F	F	0	1
				F	T	0.3	0.7
				T	F	0.3	0.7
				T	T	0.3	0.7

Associated with the above links, conditional probability tables (CPT) will express the uncertainty of the underlying causal connection. Given a node X_k and its predecessors in the DAG, X_i and X_j, the CPT will report the conditional probability of X_k given X_i and X_j, that is $Pr\ (X_k \mid X_i, X_j)$.

The joint probability distribution for all the random variables can be written as:

$$Pr(X_i, \ldots, X_n) = \prod_{i=1}^{n} Pr(X_i | Parents(X_i)) \quad (3)$$

where $Parents(X_i)$ denotes the set of nodes in the DAG that are predecessors of X_i.

Bayesian networks can be used in two fundamental ways: inferring the values of unobserved variables, given the values of observed ones; learning the conditional probabilities that characterize the model or, if the structure of the network is unknown, learning it from available data. In practice, building a Bayesian network is not trivial because specifying the CPT values is not simple. In order to account for the inaccuracy in estimating the CPT values, care must be taken to the fact that the Bayesian network analysis is not highly sensitive to the CPT values. Nodes in the network are aggregated output from IDS, i.e. correlated alerts. The reliability of those alerts is reflected in the CPT. A simplifying approach is called for in order to keep complexity under control. This is a challenging problem and, for the time being, the proposed scheme has relied upon the estimation of the conditional probabilities coupled with simplification and aggregation of raw data controlled through meta-rules. A fixed set of alert patterns, and their corresponding Bayesian networks, will enable to compute, given a set of "significant" raw alerts, the likelihood that the observed data are matching an alert pattern.

For example, attacks against a Web server which are consistently preceded by anomalous activity on some non-standard TCP ports, may suggest that the control channel of some botnet could use that particular port. The derived associations, together with their degree of reliability, should be transformed into tentative rules that must be verified against, and made coherent with, the global security policies. It is preferable that the associations be as general as possible. Some associations may, in fact, be restricted to single IP addresses rather than a particular traffic type (e.g. protocol/port). This raises two issues: firstly, after the host is no longer a threat, stale IP addresses have to be cleared. Secondly, a malicious attacker could create a DoS attack against a specific IP address by spoofing packets with that address and waiting for the system to block it. A practical difficulty is that traffic is highly variable across different environments and changes significantly over time. In order to meet this challenge, systems should have some fairly loose thresholds, ensuring tolerance of anomalous behaviors, as well as adapt their reference values during their operations.

Clearly, the module footprint on memory and computational resources will depend on the breadth and depth of this analysis. It should also be kept in mind that an excessively detailed analysis might require as much time as to make the response lag to significant events unacceptable. Another important factor which has not been given enough attention until now is the speed of the update process. Updates to configuration rules and security policies are rarely required in traditional stateless firewall environments so that the consequent impact on performance may be considered as negligible. On the other hand, when working with stateful firewalls, rule updates are significantly more frequent. In worst-case scenarios, frequent in personal firewall environments as well as restricted applications based on a supervised configuration paradigm, a new rule may be required for every new connection, resulting in hundreds of rules added every day. Finally, the level and granularity of information that should be reported to the local administrators is also an important parameter the can be used to adapt the framework better to each operating environment. After human verification, which is expected to be still unavoidable at the present state of research, generated outputs are passed to the rule generator module. A simplified functional scheme of the analyzer module is reported in Fig. 3.

4.2 The rule generator module

The automated generation process becomes essential when there is nobody available with sufficient knowledge to manually inspect the data. The automatic generation of rules is required where it is important to evaluate and

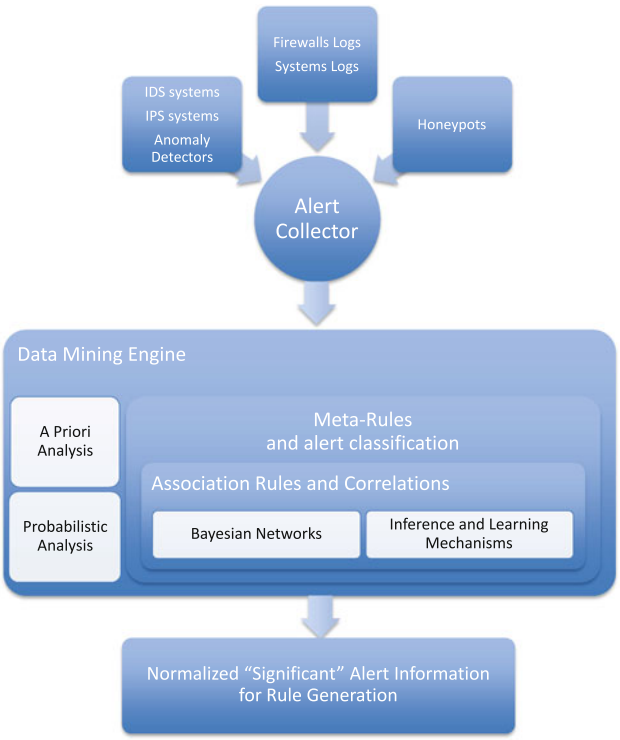

Fig. 3 The analyzer module

validate expert knowledge in a quick and trustworthy manner, especially in applications where the lack of reliability is dangerous. Alert-level output from a local or remote (operating within another security domain) analyzer module, non-ambiguously indicating that malicious activity is taking place, can be directed into this module, due to these data already being significant and needing no further confirmation.

In order to automatically generate a rule set, the network needs to be divided in two parts: "inside the wall" and "outside the wall". Initially, both sides begin with the least possible privileges ("deny all"). All the incoming flows targeted at commonly known services are subsequently permitted. Flows targeting high port numbers are only allowed as a response to outgoing flows. This less restrictive basic configuration is then refined by the administrator by either individually allowing or denying flows or specifying wildcards on an IP, protocol or port level. The number of rules influences the difficulty of writing and modifying a rule set.

Inserting new rules within a global security framework is usually carried out in three steps. The first one is to determine the security device (i.e. the firewall(s)) where the new rule should be deployed. This step is important due to it establishing where to apply the new filtering rule without adding incongruity among firewalls, applying it only to the appropriate part of the network topology. This is done by working on the firewalls and zones described in the above graph G, determining all the possible paths between the zones corresponding to the source and destination addresses associated with the traffic profiles/activities which are at the origin of the new rule. A modified breadth first search (BFS) algorithm (Knuth 1997) can be used which, while crossing the graph and finding all the paths from the starting node/zone, keeps all the paths that reach the required destination zone by using an appropriately designed predecessor function within the BFS scheme. All the inner nodes within the above paths are the firewall devices which are eligible for placing the new rule. In detail, any allowing rule should be set up on every firewall on the paths from the source and destination zones, whereas a prohibition rule can be only located on the most external upstream firewalls (i.e., the closest one to the source) on all the available paths between the two zones.

The next step is the choice of the security attributes to be checked in implementing the requested filtering action. The attributes associated with this action may vary from the direction of the flow (incoming, outgoing) to the traditional packet/traffic features and finally to the action to be carried out on the traffic flows ("allow" or "deny") involved. The use of specifically designed data structures which can streamline the representation and processing within the syntactical structure of rules and policies, are therefore required.

The third and final step is to establish the correct order of every new rule in each of the involved firewalls so that no intra-firewall inconsistencies are introduced. In the second step, a local check is carried out between each new rule and the existing ones. A new rule should be placed prior to any other rule that is a superset match and following any other rule that is a subset match for the rule it refers to. It is also possible to use the above policy representation structures in order to accomplish this task without any difficulties.

Every single rule within the firewall policy can be represented with a single rooted tree [see Al-Shaer and Hamed (2004)] named "policy tree", as shown in Fig. 4. This abstract representation gives a simple, straightforward modeling of the filtering rules while enabling an easy detection of any anomalies and relations among all the rules involved. Every node in this representation can be seen as a field in the filtering rule, with every branch starting from that node indicating a possible value of the associated field. Each tree starts with a special node, the root, which represents the protocol field and continues with the leaf nodes that can be seen as the action field. The intermediate nodes are 5-tuple representing the other filter nodes in a natural order. It is therefore possible to consider that an entire rule in the policy is represented with a full path in the tree starting from the root node and ending at a leaf. For a specific node, rules having the same value in a

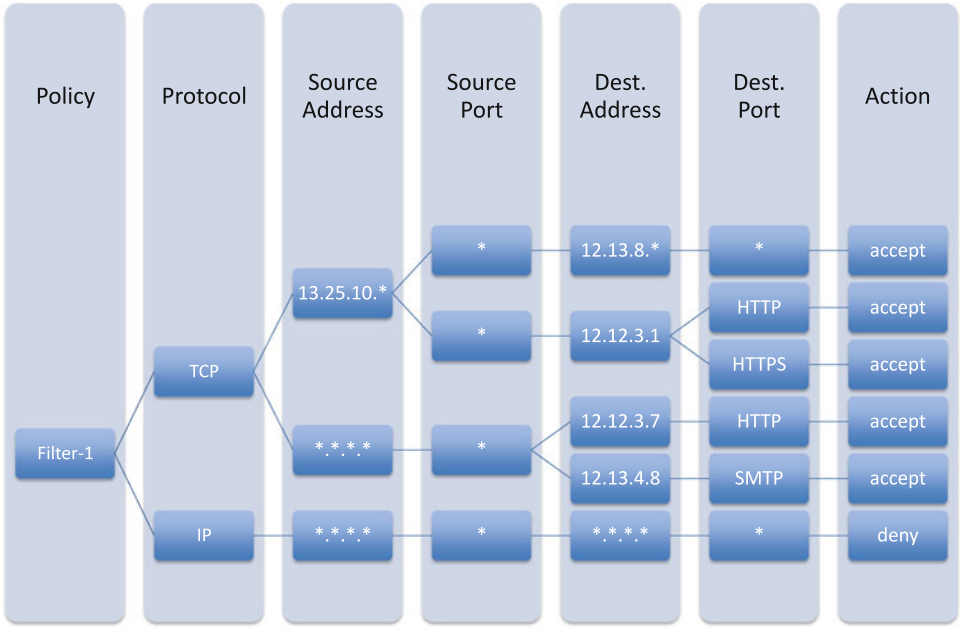

Fig. 4 Policy tree example

given node will share the entire branch which represents that value. Each rule will have an action at the end of the tree (i.e. at the leaf node).

When building a policy tree, it is important to consider the correct place of the path within the tree where to insert the filtering rule. In the case of a new rule field being incorporated, regardless of its position on the tree, the rule branch is set considering the matching between the field value and the existing branches. If a given branch suits the field value, the rules are collocated in the branch with a new branch being created. As a consequence of the tree representation, a rule, in order to preserve the relations among the policy rules, also divides the branches on the tree into superset and subset branches. If two rules coincide in their policy tree paths, there is a potential anomaly. The policy tree representation may be advantageous when it is required to keep track of the right order of every new added rule. A good starting point may be looking for the right rule collocation within the tree by comparing the attributes of the new rule to be added with the matching values on each branch of the tree. If the attribute value is contained within the branch (i.e. it is a subset of that branch), then, the order of the new rule is smaller than the minimum order of all the other rules already in this branch. Whereas, if the attribute value is a superset of the branch (i.e. it fully contains the branch), the order of the new rules will be greater than the maximum order of all the other rules already in this branch.

Finally, when the rule is totally disjointed from the existing ones in the tree, it can then be given any order in the policy. Correspondingly, the visiting of the tree can continue by analyzing the next field in the rule proceeding in a recursive manner until the attribute value exactly matches an entire branch or a subset of it. If the action field is met, then the new rule is added, with it being given the order established in the previous browsing phase. Therefore, the last step in adding a new rule implies that the corresponding policy tree instances have to be passed to the optimizer module.

4.3 The optimizer module

Tools designed for policy analysis are extremely decisive to administrators where human errors and misconfigurations need to be identified and resolved. In this third phase, optimization on the sets of rules configured on all the component devices will be carried out. The optimization process has two aims: (a) restricting the number of clauses within every rule set without changing the external devices behavior, and (b) optimizing the packet control/filtering performance.

The same behavior, in terms of allowed and forbidden traffic, may result from rule sets widely differing in the number of rules. For example, many contiguous IP addresses involved with a similar action can be expressed either using a rule for each address or, more synthetically, as a number of rules related to address ranges. Moreover, performance improvements can also be achieved by moving the most frequently matched rules to the top, since rule processing generally stops when a matching rule is found.

Optimization may be implemented during several phases/moments of the packet control activity. The first opportunity is when new rules are added to the firewall configuration. This is a rare event (when compared to the packet filtering activity), with it using more resources. In

addition, the normal packet forwarding and filtering operations cannot be interrupted for too long. A second fundamental optimization step takes place during the check for the configured rules. Upon arrival of every packet, several heuristic algorithms are used in order to check all the rules available in the firewalls involved. The optimization algorithms should exhibit quick runtime features so that the implied firewalls can sustain the current traffic demands.

What are considered as rule optimization methods in current literature can be divided into three groups. Methods belonging to the first group are used once, only when some rules change. The first group carries the algorithms which try to limit useless rules and order these rules in an optimal way. The second deals with methods and algorithms for real-time packet checking, while the third uses special algorithms in order to discover which kind of traffic flow crosses the network and reorders the rules based on that evidence.

In this work, all these aspects are taken into account. However in this section, only the first group methods are considered for optimization. Since in this architecture most of the activities related to the third group are carried out in the first module, it is assumed that the firewall operation and rule lookup methods have to be considered as static parameters, also for the sake of focusing on vendor independence, as well as immediate applicability on regular, commercially available solutions. Significant performance gains can be achieved in every case by reducing the number of rules. This type of rearrangement may be highly useful only if the algorithm results in many multiple rule comparisons.

These algorithms have a tendency to find the minimal set of rules to be compared since this operation is significantly expensive. While the volume and frequency analysis of traffic would give valuable information that could be useful in generating efficient matching rules, this type of analysis would also have the disadvantage of being significantly time/resource consuming. All the traffic must be examined, since there is no information about traffic that is authorized or not. However, placing this analysis at the optimization stage, thus acting on active firewall rules only, reduces the data size and therefore gains efficiency. It is recommended that fully dynamic optimization should not be carried out due to the computational effort being impractical, and adaptively reacting too quickly to extemporaneous traffic conditions may not be such a good idea.

Real-world traffic changes often and quite unpredictably, thus the benefits of dynamic optimization would not be adequate to compensate for the computation required. In addition, this type of scheme would be exposed to a DoS attack consisting of a sequence of apparently regular traffic flows which have the intent of altering the parameters, triggering extremely frequent updates. A "dampened" dynamic approach has been proposed, where rule firing frequency information is available to the optimizer module, and separate thresholds govern the triggering of the rule generator and optimizer module(s). When the rule generator module determines the need for a new rule, it creates and inserts it at the lowest-ordered feasible position in the rule set. As long as the new rule is fired, counters will reflect its application frequency—and thus its importance—with the optimizer module possibly deciding, when an independent threshold is exceeded, to reorganize the rule space to reflect the changes. The most frequently fired rules, will "bubble up" in the rule space. Simultaneously, the optimizer module will downgrade the less frequently fired clauses. Eventually, the rules that are not used for a long time (according to another threshold determined by the meta-policy), and may possibly be considered as unnecessary, can be removed, thus drastically reducing the rule space dimension and, therefore, the memory footprint.

4.4 The conflict remover module

The security policies configured in a distributed firewall environment may be subject to periodical updates, consisting in the modification, insertion or removal of any rules, with the aim of dynamically adapting the overall security strategy to network topology changes or new requirements due to emerging threats or changed operating conditions. Accordingly, rules should be periodically checked against the characteristics of network traffic in order to verify that they are still useful, well organized, and consistent with the current traffic shape and volume parameters.

Even if a new rule may not affect every network security zone, it should therefore be correctly inserted into the appropriate firewall configuration in order to avoid any unwanted traffic processing and filtering effects. Security components presenting inconsistencies or errors in their configuration may be the origin of weak traffic control policies that can be easily exploited by unauthorized malicious parties. Nonetheless, every new rule must be inserted in the right order within the existing configured policies in order to avoid the generation of anomalies. When a rule is changed or removed, conflict analysis is required due to the whole policy logic being affected in its behavior. Intra-firewall anomalies (see Hamed and Al-Shaer (2006) as well as Abbes et al. (2008)) occur within a single firewall, when the same flow matches more than one local filtering rule. This often results in conflicts between policies, which may in turn provoke security flaws. These conflicts can often be very difficult to manually find when inspecting a large number of rules that may have been written by different people at various times. For example, if it is noted that several rules have not been recently used,

this may lead to considering rule reordering, reaggregation, or even removal.

Shadowing is a common intra-firewall anomaly. It occurs when a rule is never applied due to its matching conditions always being covered by other rules occurring before it, and thus taken into consideration earlier. On the other hand, a rule is said to be redundant, when it is not shadowed by other rules and has no effect in the sense that removing it does not change the policy. However, when working in a distributed environment, inter-firewall anomalies may result from the removal of some rules or rule clauses from a participating firewall element. In detail, the removal of a "deny" rule clause from an upstream firewall, can only result in some illegitimate traffic flowing downstream. Conversely, the removal of an "allow" rule upstream in the firewall chain can cause relevant traffic to be blocked with the consequent shadowing of all or part of the related downstream rules. When removing a rule from a firewall participating in the overall distributed security scheme, the first and fundamental action to be carried out is to identify in advance all the possible impacts in the associated (source or destination) security zones as a consequence of the operation. This can be achieved by using the same procedure described in the rule generation process in order to determine the paths on the firewalls/zones graph between every source-destination zone pair involved in this rule. In order to discover anomalies, it is possible to check in every probable pair of rules the existence of common elements in their policy tree paths (Al-Shaer and Hamed 2004). In presence of two-rule policy tree paths which coincide in a point, a potential (redundancy or matching) anomaly can be determined between the rules involved, according to the previous definitions of anomalies. When rule paths do not match, it is certain that the rules involved are completely disjointed, with no anomalies being associated to them. When either a new rule is introduced or an existing rule modified, by simply changing its order within the policy, the corresponding policy tree should be matched pairwise with all the other existing instances in order to discover any anomalous situation that occurred as a consequence of the action of the previous modules.

In a second step, it is possible to carry out the definitive removal or modification of the rule from the configured policies on all the involved firewall, according to the following criteria:

- it can be directly removed or modified from the firewalls involved in all paths from source to destination in the presence of an "allow" rule. However, it is necessary to check if any shadowing and/or spuriousness anomalies are generated on each local operation on the upstream or down-stream security devices with respect to a specific firewall;
- it is necessary to merely suppress it from the configuration of the directly impacted firewalls because its removal does not generate any anomaly on the other firewalls participating to the security scheme in presence of a "deny" rule.

The conflict remover output should result in the final rules expressed according to a firewall-independent abstract modeling language with the expressive power of existing firewall-specific languages, but with significantly less complexity and specificity. The model represented by this abstract language will then be automatically translated into any of the existing low-level firewall languages by the deployer module. The final output of the conflict remover will therefore be expressed in an abstract language such as the AFPL (Pozo et al. 2008) or the FLIP (Zhang et al. 2007) so that the next module down the pipeline will be able to carry out its task on these results. These languages can consistently express stateful and stateless rules, positive and negative rules, overlap-pings, exceptions, by keeping the inherent complexities hidden, which can be easily compiled into several market-leader firewall languages.

4.5 The deployer module

Once access control rules have been extrapolated, specified, generated, optimized and checked against potential conflicts, they must be actualized. Rules, therefore, have to be translated from the abstract AFPL or FLIP syntax into the appropriate low-level firewall languages of the destination device and then deployed to the device itself. There are noticeable differences between both commercial and open source firewall platforms. These differences range from small variations in the number, type, and syntax of the various parts which compose a filtering rule, to huge dissimilarities in rule-processing algorithms which can affect the design of the entire filter set. However, the vast majority of filtering policies can be expressed with any of the filtering languages and platforms, provided that it is adapted to the number and syntax of the required rules.

The deployer module will translate the already generated and optimized rule sets into the specific languages of the involved firewalls, adapting the rules to the specific conventions, limitations and characteristics of the target devices. The protocol for the actual loading of the rules onto the devices depends on the device type. The appropriate interface (CLI, SNMP or specific APIs) can be used to securely communicate with the network devices. This type of communication needs to be bidirectional. Rule configuration and updates will go in a direction, while statistical data to be collected and analyzed go in the opposite one.

4.6 The communicator module

Some of the security alerts or meta-rules generated by the analyzer module will have a scope which may either be local to the security domain considered, or so general as to cross the boundaries between security domains. In order to save time and effort, security administrators may then elect to share information with administrators of neighboring domains. Communication and collaboration between domains has the potential to be mutually beneficial to the parties involved, since it reduces the effort of duplication and increases the timeliness of security alerts propagation.

The clean, standardized, and simplified way in which they are represented within the modules, encourages an information exchange that is as automated as possible. This approach would avoid or limit the impact of issues related to conversion and adaptation of different formats and contents. This process may leverage on techniques intended for the cooperative distribution of traffic filters such as the ones introduced by Kao and Shiue (2009) and by Palmieri and Fiore (2008). This is true even when other neighbors cooperating domains are using a security architecture different from the one outlined here. Nevertheless, administrators are usually reluctant to share complete information about rules and policies, since a full disclosure of these might expose too many details about the internal organization of their network and security assets, with them preferring to keep these details private. The communicator module has, therefore, the task of filtering out any unwanted detail, while simultaneously preserving, for security reasons, the validity and usefulness of the information meant to be shared. This is accomplished by using a virtualized data exchange model, namely the one specified in RFC 4765 by Debar et al. (2007), along with the Intrusion Detection Message Exchange Format in the RFC 4767 by Feinstein and Matthews (2007). The flow of alerts will be fed into the analyzer module(s) of the communicating parties, where local conditions will guide the generation of appropriate filtering rules.

In many practical contexts some alerts can be regarded as sensitive information, since they reveal that some traffic or services are allowed or denied in the network by local access control policies that should be kept confidential. For example, these policies may closely reflect the structure of internal business processes, or liaisons with other entities. In addition, not every communicating partner will have the same trustworthiness level. Accordingly, some meta-rules will determine, for each partner, the type of alerts intended to be shared, their scope and threshold as well as the completeness of details.

Finally, a foreseeable use of the communicator module is the realization of a feedback cycle to the monitoring and logging facilities. This significantly improves adaptivity, since the granularity of logging and monitoring can be increased when an event having security implications is in progress.

5 Conclusions

While the use of firewalls is still a fundamental step in achieving secure networks, the complexity of designing and managing firewall policies within the next generation highly heterogeneous networks might significantly limit the effectiveness of firewall security. Starting from these considerations, this work proposes an integrated, dynamic, and adaptive architectural solution which integrates techniques and facilities present in different security systems in order to achieve the objectives of scalability and flexibility. Although some of the constituents already exist, either they work as standalone components or they may achieve only a subset of the required tasks. The proposed architecture aims at addressing most of the actual challenges in the multiple-firewall scenarios by allowing the semi-automated implementation of conflict-free policies. It simultaneously avoids security inconsistencies, and optimizes the effectiveness and performance of the traffic filtering facilities, in order to make the above security systems safer against denial of service and fast propagating menaces.

References

Abbes T, Bouhoula A, Rusinowitch M (2008) An inference system for detecting firewall filtering rules anomalies. In Roger L. Wainwright and Hisham Haddad, editors, SAC 08 Proceedings of the 2008 ACM symposium on Applied computing, pages 2122–2128. ACM. ISBN 978-1-59593-753-7

Abedin M, Nessa S, Khan L, Al-Shaer E (2010) Analysis of firewall policy rules using traffic mining techniques. International Journal of Internet Protocol Technology 5(1-2):3–22

Al-Shaer E, Hamed H (2004) Discovery of policy anomalies in distributed firewalls. In INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies 4:2605 –2616

Bashah N, Bharanidharan Shanmugam I, Ahmed A (2005) Hybrid Intelligent Intrusion Detection System. Transactions on Engineering, Computing and Technology 6:291–294

Castiglione A, De Santis A, Fiore U, Palmieri F (2010) An enhanced firewall scheme for dynamic and adaptive containment of emerging security threats. In Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on :475–481

De Capitani di Vimercati S, Foresti S, Jajodia S, Samarati P (2007) Access control policies and languages in open environments. In Secure Data Management in Decentralized Systems, volume 33 of Advances in Information Security, pages 21–58. Springer. ISBN 978-0-387-27694-6

Debar H, Curry DA, Feinstein BS (2007) The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765, March 2007. http://www.faqs.org/rfcs/rfc4765.htm..

Feinstein BS, Matthews GA (2007) The Intrusion Detection Exchange Protocol (IDXP). RFC 4767, March 2007 .http://www.faqs.org/rfcs/rfc4767.html

Frigault M, Wang L (2008) Measuring network security using bayesian network-based attack graphs. In Computer Software and Applications, 2008. COMPSAC 08. 32nd Annual IEEE International Conference on, pages 698–703. IEEE Computer Society. ISBN 978-0-7695-3262-2

Gu Y, McCallum A, Towsley D (2005) Detecting anomalies in network traffic using maximum entropy estimation. In Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, IMC '05, pages 32–32, Berkeley, CA, USA. USENIX Association.

Hamed H, Al-Shaer E (2006) Taxonomy of conflicts in network security policies. Communications Magazine, IEEE 44(3):134–141, march 2006. ISSN 0163-6804

Kao S, Shiue L (2009) Security management of mutually trusted domains through cooperation of defensive technologies. Int. Journal of Network Management 19(3):183–201

Knuth DE (1997) The Art of Computer Programming, Volume I: Fundamental Algorithms, 3rd Edition. Addison-Wesley

Lakhina A, Crovella M, Diot C (2004) Diagnosing network-wide traffic anomalies. SIGCOMM Comput. Commun. Rev. 34:219–230, August 2004. ISSN 0146-4833

Mayer A, Wool A, Ziskind E (2000) Fang: a firewall analysis engine. In Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on :177–187

NetCitadel LLC (2010) http://www.fwbuilder.org/

Palmieri F, Fiore U (2008) Containing large-scale worm spreading in the internet by cooperative distribution of traffic filtering policies. Computers & Security 27(1-2):48–62

Palmieri F, Fiore U (2010) Network anomaly detection through nonlinear analysis. Computers & Security 29(7):737–755

Pozo S, Ceballos R, Gasca RM (2008) Afpl, an abstract language model for firewall acls. In Proceedings of the international conference on Computational Science and Its Applications, Part II, ICCSA '08, pages 468–483, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-69840-1.

RedSeal Inc (2011) http://www.redseal.net/products/redseal-networkadvisor, March 2011

Samak T, Al-Shaer E (2010) Synthetic security policy generation via network traffic clustering. In Proceedings of the 3rd ACM workshop on Artificial intelligence and security, AISec '10, pages 45–53, New York, NY, USA .ACM. ISBN 978-1-4503-0088-9

Samak T, El-Atawy A, Al-Shaer E Towards network security policy generation for configuration analysis and testing. In Proceedings of the 2nd ACM workshop on Assurable and usable security configuration, SafeConfig '09, pages 45–52, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-778-3

SkyBox Inc (2011) http://http://www.skyboxsecurity.com/, March 2011

Vaarandi R, Podins K (2010) Network ids alert classification with frequent itemset mining and data clustering. In Network and Service Management (CNSM), 2010 International Conference on, pages 451 –456. IEEE, oct

Zhang B, Al-Shaer E, Jagadeesan R, Riely J, Pitcher C (2007) Specifications of a high-level conflict-free firewall policy language for multi-domain networks. In Proceedings of the 12th ACM symposium on Access control models and technologies, pages 185–194. ISBN 978-1-59593-745-2