Review of Security Policy Implementations

Richard Macfarlane, Prof William Buchanan, Dr Elias Ekonomou, Omair Uthmani, Dr Lu Fan and Owen Lo

Centre for Distributed Computing and Security,
Edinburgh Napier University
{r.macfarlane,w.buchanan,e.ekonomou,o.uthmani,l.fan,o.lo}@napier.ac.uk
http://cdcs.napier.ac.uk

Abstract. Network security should be based around security policies. From high-level natural language, non-technical, policies created by management, down to device and vendor specific policies, or configurations, written by network system administrators. There exists a multitude of research into policy-based network systems which has been undertaken. This paper provides an overview of the different type of policies relating to security in networks, and a taxonomy of the research into systems which have been proposed to support the network administrators in difficult tasks of creating, managing and deploying these policies.

1 Security Policies

High-level security policy documents should be written by upper management and should be the 'what' of security in the organisation. Without this definition of the security goals, it is difficult to use security mechanisms effectively [1]. The implementation, or technical policies, are then created from the overall high-level policy. This is the 'how' and it is used to enforce the security policy. The term policy is used in the literature to describe both the high-level policies, as well as the low-level implemented rules. The processes of security policy creation and implementation are shown in figure 1, and a good definition of an overall security policy is taken from The Site Security handbook - RFC2196 [1]:

> "A security policy is a formal statement of the rules by which people who are given access to an organization's technology and information assets must abide." [1].

RFC2196 [1] provides an excellent reference for network system administrators and management-level decision makers, when creating network security policies. At its core is a five-step iterative process detailing the Security Policy creation and maintenance process. A key element of this is that the definition of a security policy is an ongoing process, with regular reviews and auditing of security policies and mechanisms, providing feedback to improve it. This matches two fundamental concepts: security being an integral part of the design and systems being an ongoing process - rather than simply the implementation of

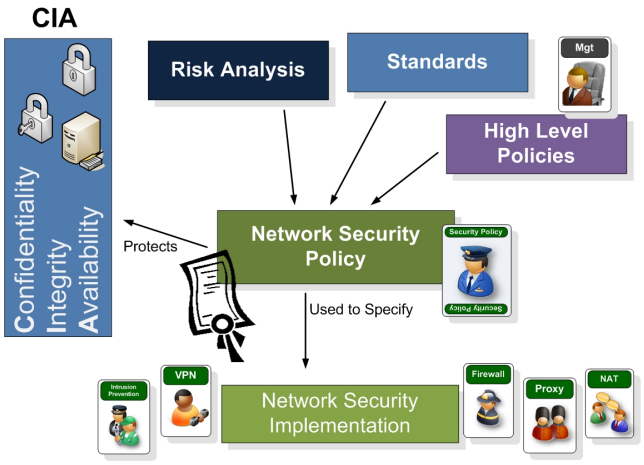

Fig. 1. Network Security Policy

security products - which are both common throughout security literature [2, 3]. The Site Security Handbook does not put forward a formal method of specifying the policies, but defines:

1. Identify what you are trying to protect.
2. Determine what you are trying to protect it from.
3. Determine how likely the threats are.
4. Implement measures which will protect your assets in a cost effective manner.
5. Review the process continuously and make improvements each time a weakness is found.

Industry recognised standards frameworks can be used to help create the security policy, based on industry best practices (as shown in Figure 1). Currently, the two best known frameworks are COBIT and the ISO27002 Code of Practice for Information Security Management (previously ISO 17799). These provide detailed industry standards in IT security management and audit compliance [4].

It is important to involve users in the implementation of a security policy and the understanding of security problems by users, and giving them clear and easy to follow rules, can be a key factor in the successful implementation of the policy [5]. Danchev calls this the "Security Awareness program" and emphasises that the latest technical security measures, such as firewalls and IDPSs (Intrusion Detection/Prevention), can be rendered useless by careless, or badly informed, end-users. The User's Security Handbook [6], the companion guide to the Site Security Handbook [1], can be used as a guide on how to educate users about the dangers of networked systems and how to keep data and communications safe.

Although management should have a good deal of input into the high-level security policies, they may also need technical input due to the nature of the

services they describe. Ideally they should be created by staff with the executive power of the CEO of the organisation, and the technical ability of a system administrator. Typically administrators will help create the policies and the management will make the final decisions, but sometimes business plans will trump security policy decisions [2].

2 Enforcing Policies

Security policies protect the confidentiality, integrity, and availability of the assets of an organisation. To enforce this security services should to be deployed, such as authentication, encryption, antivirus software, and firewalls. To do this the security policy documents are often used to create technical security procedures, and guidelines, which can then be implemented in the network. Types of procedures include: identification or authentication; access control or authorization; and accountability or auditing procedures. Procedures and guidelines could be created for each of the different type of security mechanisms to be used to enforce the policy. These technical mechanisms include authentication systems, firewalls, proxy servers, IDPSs, VPNs, and access control systems [7]. The guidelines are best practice suggestions for each, and the procedures are specifications for implementation of the specific security measures. This process is shown in Figure 2.

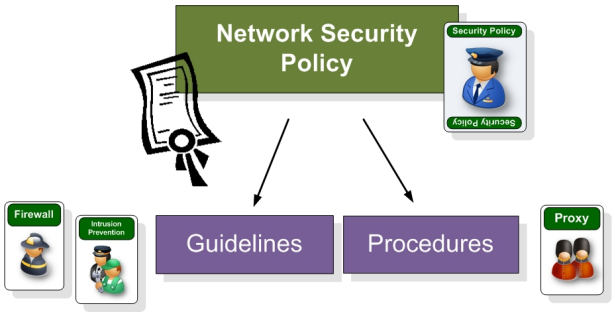

Fig. 2. Enforcing Security Policy

Security Policies can be split into several components, which combine to achieve the security goals of the organisation. These components are enforced by various security mechanisms or procedures. The authentication policy could be enforced using user names and passwords, software tokens, and/or VPNs. The accountability policy may use IDPs and firewalls to enforce auditing and incident response capabilities. The Access Control Policy, which this work focuses on, would typically use firewalls, VPNs and authorization systems to enforce access to resources [1].

The access control part of the security policy deals with making sure that authorized individuals can perform the tasks they are authorized to, and that

others cannot. It is typically referred to as the 'access control policy' [7]. Access control makes sure that requests to access a specific resource are only granted if the request agrees with the security policy definition. In terms of networks, the most commonly used access control mechanisms are firewalls and filtering routers [8]. Firewalls control access to resources by filtering network traffic, only allowing access that is specified by the security policy.

The network access control policies, defining which traffic can cross network boundaries, are implemented as policies on network devices which have access control functionality, such as traffic filtering capabilities. The system administrator is typically tasked with manually creating these low-level policies, or configurations. In order to determine whether to grant an access request, access control mechanisms uses a number of criteria. The primary criterion being the network address of the machine from which the traffic originates. Other criteria, which can decide whether access is granted or not, would include network service and destination of the traffic [8]. The most common technique used to by firewalls to filter traffic is known as Packet Filtering.

Implementation of security mechanisms can be based on best practices, and several sets of guidelines exist. The NSA and NIST publish configuration guidelines for implementing security controls. The Cisco SAFE framework provides detailed guidelines for the implementation of network security mechanisms for various different sizes of networks and different site specific setups [9]. A compilation of the most common policy errors are detailed by Wool in his review of firewall configuration problems [10], and provide guidelines on what to avoid when implementing firewall policies.

2.1 Policy Enforcement Problems

If the high-level policies are not defined correctly, the implementation cannot provide the security protection need by the organisation. As described by Wool in his review of firewall configuration problems [10], and as a major motivation for the firewall auditing and testing system described in the latest research by Mayer, Wool, and Ziskind [11]:

> "the protection that these firewalls provide, is only as good as the policy they are configured to implement" – [11]

The policy should be clear, concise, and easy for the administrator to follow. If a policy is not well designed, then it will not be enforced properly and the security goals will not be met [12].

Conversely, polices are only as good as the configurations which enforce them [13, 14]. The enforcement of policies is not always an easy task. Policy management can be difficult as policies grow and become increasingly complex [15, 16, 10]. Blakley makes the following statement in [15].

> "Policies do not scale well and their complexity quickly increases as systems grow and diverge, which makes them unmanageable" – [15]

Madigan et al. categorises violations of security policies, and shows that violations from network issues were by far the largest type reported [12]. The author also states that the network violations were among the most time consuming to correct. This work was based on real security policy violations, on two university campuses, over a two year period. This could be a direct result of policy enforcement problems, such as policy configuration errors and anomalies. This is not surprising, as it is generally accepted by security experts that firewalls and other traffic filtering devices are poorly configured [10].

The configuration of a firewall is probably the most important factor in terms of the security a firewall provides [17], but are often configured incorrectly [10]. Firewall policies are made up of rule sets, and these rule sets are ever expanding due to new rules continually being added and very few removed, so device access policies tend to be large and always increasing in size [10, 18]. It follows that the management of these policies at the network device level can be extremely complex, error-prone and expensive as the policies expand [19]. The configurations are typically hand crafted and bespoke for each individual system by network administrators, which can be error prone work [20]. This is a serious problem as errors in the firewall policies mean that the intended security policy will not be enforced.

Mapping the high-level security policies to the lower level implementation can be extremely difficult [7]. High-level policies are written in a natural language, and describe security aims in terms of entities of an organisation, such as networks, users and resources. Enforcement policies are in terms of the points of enforcement, or devices. Firewalls and other devices have their own vendor specific languages and tools, which tend to be very low-level. Many researchers have used the simile that these configuration languages are 'like programming in assembly languages' [19]. A conceptual gap exists between the two, and administrators can find it very difficult to map from one to the other correctly (to bridge the gap) [21, 20, 22]. GUIs are provided by some vendors, but most require the administrator to click through several windows, simple to understand a single rule fully [11].

Administrators are typically tasked with the creation of the low-level device policies, which implement the security policy of the organization. In terms of firewalls, these are the firewall rule sets. The administrators will add, delete, and change the rules to match changes to the high-level security requirements. For example, when new web servers are added to the organisation's network, new rules would be added to the perimeter firewalls to allow appropriate access to them from outside and inside the organisation's network. The complexity of the rule sets increase as they increase in size, but also the complexity can change depending on the rules used within them. For example, OSI layer 3 packet filtering is not as challenging to understand as OSI layer 4 or layer 7 filtering due to less filtering fields in each filtering rule. The filtering rules can be based on source address only or source and destination, as well as various other traffic attributes within a rule. Wool created a classification system for complexity of rule sets, based on the number of rules, the objects (traffic filtering

parameters) and the interfaces rules could be applied to. In [10] the following rule set complexity measure is defined:

> Rule Complexity = Number of Rules + Network Objects + Number of Interfaces(Number of Interfaces -1) / 2 – [10]

For most firewalls the ordering of the rules in a rule set are important, as in the common 'first match' filtering mechanism, the position of the rules in the rule set dictate if they are matched against traffic or not. The earlier in the rule set the higher the priority the rule has when matching against traffic [23]. Thus filtering rule sets, that use first match semantics, are complicated to create and amend due to this dependency on the rule ordering. As the size and complexity of the rule set increases it becomes more difficult for the administrator to predict the impact of a rule on the overall rule set. This makes rule sets extremely difficult to manage [10].

Effectiveness of security policies can be compromised due to poor policy management, especially when enforcing a security policy across a range of devices around a network [24]. Security policies can be spread over a range of different security devices [25]. Packets can take multiple paths through a network, with multiple filtering devices on each different path. An administrator needs to understand the interaction of combinations of these devices for each traffic path [11]. The low-level configurations, which implement the security policy, can span heterogeneous networks, and may be spread over many network devices. Different vendors implement different algorithms and low-level languages for configuring their devices. Thus, the scope of the deployment in terms of devices, can also increase the difficulty of the task of policy enforcement for the system administrators [20]. If hundreds of network devices have to be coordinated to enforce a network security policy, manual translation of the policy into device configurations, can become extremely complex and error prone. If multiple devices such as firewalls, from different vendors, have to be used to create overlapping enforcement policies, in order to enforce a single global security policy, this adds even more complexity.

2.2 Policy Enforcement Solutions

Abrams [26] describes a layered approach which can be taken with security policies. This can help the management, administrators, and users understand the policy and its implementation, as the mapping between the high-level policy and the low-level deployment is more clearly defined. Different individuals in an organisation will have a different view of the security policy. At a management level, an enterprise wide view concerning overall security objectives, such as protect the company's assets, will be taken. This is the highest level of abstraction of the policy, and it should be easy to understand, but does not contain guidance in how to carry out the security requirements. To do this policies at a lower level of abstraction need to be introduced. At the users level of abstraction, the policy is seen as rules relating to access to resources and data, such as systems

and applications [26]. The high-level policy is translated into this level of policy abstraction, providing guidance for users [6]. This level might be defined in terms of finite-state machines, or as access matrixes [27, 7]. The lowest level of abstraction is the network implementation level, were the security mechanisms are deployed to satisfy the higher level specifications.

The policy can be shown in three different ways: first as a high-level policy described in a natural language, secondly as formal statements to describe a model of the policy, or thirdly as a technical implementation [7, 26]. Natural language is prone to ambiguities, thus the high-level policy may also have ambiguities in terms of the security requirements. If the policy is modeled in a formal language, the ambiguities can be reduced, or even removed completely. The formal model can also be used to audit the security procedures for compliance with the requirements, and it provides a specification free of any specific implementation methods, such as vendor specific tools and languages. The downside is that specialists are needed to work with such models, especially if they are mathematical models. Otherwise administrators, who implement the security mechanisms, would have to learn such modeling techniques. A formal model which is non mathematical, such as a more formal natural language, can be a good compromise. The ambiguities from the high-level language can be reduced, and the modeling language would be understood by a wider range of individuals. Tasking specialised individuals with such work, such as firewall administrators, is preferable to asking general system administrators to carry out such tasks [3].

Samarati [7] outline how the different levels of abstraction provide the benefits of separating the requirements and design, from the implementation. The security requirements can be dealt with, regardless of how are they are to be implemented, and different methods of implementation can be compared against each other for the same requirements. The formal model could also be used to prove the security proposed is sound, and possibly even to automate the implementation of the security mechanisms, by creating device configurations from the model definition [28].

Various policy management systems have been proposed to help the network administrators with the creation, management, analysis, and auditing of these complex policies. The next section presents a taxonomy of research into these systems and tools, along with comparisons between the different approaches. Packet filtering firewalls are one of the most important mechanisms used by organisations to implement their security policies, and in the past 10 years, there has been a great deal of research into the areas of firewall policy management. The taxonomy and review of the literature is therefore focused on research into firewall policy-based systems.

3 Firewall Policy Management Systems

3.1 Introduction

Systems which can assist administrators in the creation and management of network access policies, can be used in the various stages of security development

and operations. Network security should be a continual process, built around the network security policy, and it should be integrated into all stages of the SDLC [2]. Many definitions exist for the SDLC, with possibly the best known being the Waterfall Model, in which the development process is described by several phases in a downward flow. The phases for general development are typically Requirements Analysis, Design, Implementation, and Maintenance. More specifically, for security system development, the NIST process definition is defined in special publications 800-14 [29] and 800-64 [30]. This includes security system Initiation, Development, Implementation, Operation and Maintenance, and Disposal.

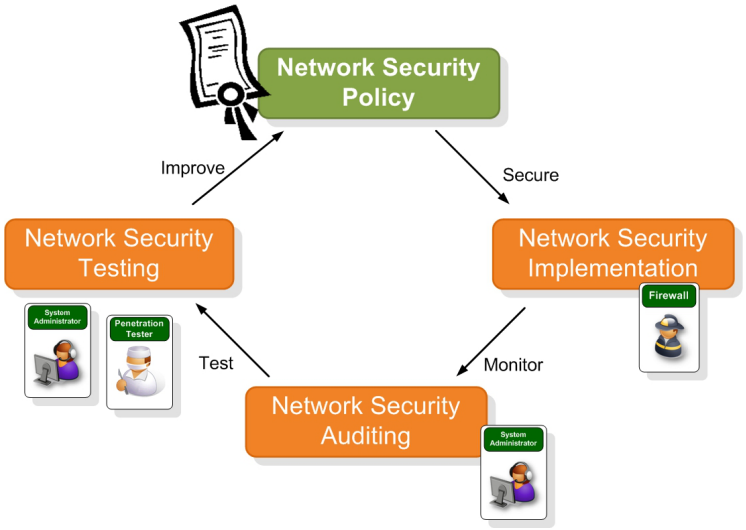

Fig. 3. System Life Cycle

In the NIST definition the initiation phase is concerned with security planning, including documenting the need and high-level requirements for the system. The Development phase includes the design, purchasing and creation of the system. Implementation involves the testing, and subsequent installation of the system. The Operation phase takes place once the system is in production, and includes the monitoring and auditing of the security mechanisms, and any configuration management.

The policy management systems, discussed in this section, can be categorised by which phase of the development life cycle they operate in. A simplified security development life cycle, showing the stages which contain the most relevant literature, is shown in Figure 3.

3.2 Factors used to Compare Systems

The main factors which are used to compare systems include:

- **Modeling Technique** high-level, abstract, languages can be used to describe the policy model. Some systems introduce new high-level languages to describe firewall policies. These can be graphical languages, or abstract textual languages similar to high-level programming languages. These high-level languages can then be translated into the vendor specific configuration language and be implemented on firewalls or routers. Some systems are based around the low-level vendor specific languages, which the system administrators are already familiar with. Another method is to keep the language technically similar to the low-level vendor languages, but abstract just enough to be used across a heterogeneous network, such as a XML based language. Systems can also be based around formal models, such as BDDs, bipartite graphs, or relational databases.
- **Top-Down or Bottom-Up** - Systems can be regarded as having a bottom-up approach if the starting point, or input to the system, is from device policy. These systems may create an abstract policy model, analyse, or aggregate the policy into a higher level policy, but always start with a concrete policy from a network device. Top-down systems start from high-level descriptions, possibly a security policy. They typically create an abstract model of a policy, which then can be compiled into the low-level technical policies and then deployed on devices.
- **Scope** Some of the systems describe a single network device policy, and some allow the description of entire network security policy which maps to policies for multiple firewalls.
- **User Interface** - The systems, which have been implemented as tools, are typically split between having either a CLI, or a GUI, which is an important consideration, depending on which tasks the system performs and which type of user the system is aimed at [31].
- **Administration tool design issues** Issues raised on the subject of administration and security tool design highlight factors such as flexibility, customisability, usability, and error reporting techniques [32].

3.3 SDLC Development Phase

In this phase of the development life cycle, a risk assessment is carried out, which specifies the security requirements necessary to protect an organisation's network-based assets. Security requirements analysis are then carried out to define what is required to secure the network. This typically includes the creation of the high-level security policy documents. Which security mechanisms are needed can then be planned for, and specific security controls - such as firewalls and filtering routers - can be developed and implemented [30].

Tsoumas and Tryfonas suggests a system to automate some of the development phase of the development life cycle [33]. Their system is top-down, and

takes a natural language description of a policy, such as the recommendations from a risk analysis, and creates a formal model of a security policy. This is an attempt to fill the gap between high-level policy statements, risk analysis output and standards - shown in Figure 1 - and the network security policy definition. The typical approach is to task experienced administrators, along with management input, to translate the high-level security requirements into a network security policy. Their research did not produce a prototype system and is left as theory, but it is an interesting idea, as potentially the system could relieve the administrators of the time consuming and error prone policy creation task [11, 19]. The authors suggest that their system, in conjunction with the output from a Risk Analysis tool, could produce a security policy and recommendations for administrators concerning deployment issues. They also suggest that a formal model could be created using a high-level policy language such as Ponder [34] or FLIP [35]. This would assist the administrators, as they would not have to learn the high-level policy language used to describe the formal policy model. The output from this system could be generated in the syntax of an existing formal policy language. The system, would in this way, interface with high-level policy language based systems described next.

High-level Policy Languages Currently, there is no generally accepted high-level model, which is commonly used for policy configuration [36]. The following quotes back this up:

> "The thing to note here is that there is no fixed terminology for the description of firewalls." – [1]

> "generic data models and high-level languages for router configuration do not exist and are likely to remain elusive for some time" – [18]

Research by Guttman, which was funded by the NSA, led to a system called NPT. The system can define an overall access policy in a high-level policy language, and verify that packet filtering specifications, which it generates, enforce the policy [37]. They suggest a high-level access policy language to describe which packets can get where in an organisation's network. This is an abstract language, above and independent of, device configuration languages. The system deals with a global policy, covering the entire network access policy, not just the filtering at a single network boundary. Logical 'filtering postures', which define packet filtering at each device, are generated by the system. The motivation for the system was this creation of multiple filters to enforce the global access policy. The filters do not define the configuration of firewalls, but only the logical access policy. This assists the network administrators in delegating the filtering to various devices around the network, to enforce the overall policy, but the administrators would have to then manually create the device configurations.

As the system describes the relationship between devices, it needs a description of the networks topology. This is modelled using a bipartite graph, and

specified using a policy specification language. The areas of the network are represented by nodes, as are the filtering devices which route traffic between the areas. The (undirected) edges between the nodes represent the interfaces of the devices connected to the different areas. Each interface can have an associated 'filtering posture' created, in both an inbound and outbound directions. These postures are abstract representation of the bi-directional packet filtering that is enforced by routing and firewall devices.

A Lisp type language is used to describe the access policy and the network topology, and the filtering posture NPT generates. An example of the language is shown in Listing 1.1. This consists of source and destination hosts and areas, and the traffic which is allowed to flow between them. This system can be categorised as top-down, as the network administrator would have to manually create the abstract access policies in the modeling language, as well as the network topology information. The interface to the system is text based, and the administrator would have to learn the lisp type policy specification language and become familiar with this way of modeling the access policy. This is not the most complex of the high-level policy languages encountered in the review of this literature, but it would still be a challenge for administrators to learn and use.

Listing 1.1. NPT Language Code Snippet [37]

```
(defined-host-sets          ; define some host sets
  (internal                 ; new name
   ((areas engineering      ; two areas
           financial)))
```

Guttmans NPT system also can also audit the filters it generates, by comparing them to the global policy specification. This verifies that the filters created, correctly implement the overall policy. This conformance checking could be useful if the administrator makes changes to the filters, and wants to validate them against the overall access control policy. However, this auditing facility could only be used in the design stage of the SDLC, as once the filters have been implemented on devices, the validation would no longer provide any assurance of the implemented security measures. To use this in the operations phase, a mechanism would have to be added to reverse engineer the high-level language policy from the configured devices, before running the auditing facility.

Later research by Guttman documents an evaluation of the NPT system, by generating filtering postures from a realistic sized network. They use a dozen filtering devices connecting sixteen network areas and their evaluation metric consisted of timings of the system runs. Their performance evaluation produced results which seem usable, with runs only taking seconds. Some problems were encountered with their system. These were mainly concerning the usability of the system. Specifically, administrators had difficulty when creating the network and policy definitions in the abstract language, and also when trying to translate the filtering postures into concrete device configurations, such as Cisco router ACLs [38]. The administrators found it hard to work with the abstract policy

representations, particularly when trying to create a representation of an existing network policy [38]. Another tool, the Atomizer, was created to assist the administrators in this task [39]. It can take Cisco ACL configurations (Cisco filtering language used on routers and firewalls) as input and generates abstract NPT policy specifications, combining common traffic filtering behaviors. The tool uses BDDs to model the traffic filtering policy generated from the device configurations, and 'atomizes' the sets of traffic within the filters together. The output of the NPT system, however, is still in the difficult to use and does not provide automatic generation of firewall rule sets.

The work of Bartal, Mayer, Nissim, and Wool started with the Firmato Firewall Management Toolkit [40] in 1999, which is another top-down, configuration generation system. A high-level policy language is used to create a, vendor independent global policy, which can be compiled into individual vendor specific device configurations. The high-level policy definition language is used to manually specify the network topology and the high-level security policy. This is then translated into an entity-relationship model which is a role based model of the access policy and its relationship to the network topology.

One of the aims of the Firmato system was to separate the network topology and the high-level policy definitions, which is an improvement on the work of Guttman, as changes to the topology does not mean that the policy has to be reworked. Other motivations behind the system were to abstract the policy away from low-level languages, enabling vendor independent management of firewall configurations, and to automatically generate configurations, across multiple filtering devices, from the abstract global policy.

The high-level policy language used to describe the abstract policy is called MDL and is used to specify both the policy, and the network topology. An example of the MDL language is shown in Listing 1.2. Again, like the first generation system from [38], the administrator has to manually create these definitions.

Listing 1.2. MDL Code Snippet [40]

```
corp_gw =
 {
   I_dmz_corp : { addr=ether0, INVIS,
                  file=``RULES_I_dmz_corp'' }
   I_corp_in  : { addr=ether1, INVIS,
                  file=``RULES_I_corp_in'' }
   I_admin    : { addr=ether2,
                  ip = 111.222.3.1, file=``RULES_I_admin''}
 } : LMF
```

The language is very different from the configuration languages of firewalls, and administrators generally find this type of language problematic to use [38]. In the testing of the Firmato system, existing firewall rules were converted into MDL manually, but rule sets with under 50 rules were used. The authors recognised that any more rules would have necessitated an automated mechanism for translating the rules into MDL. The system does improve on the work by

[38], solving one of their systems main problems, in that the administrator does not now have to translate abstract filter definitions into the low-level device configurations, as this can performed automatically. A limitation of both this system and Guttman's is that it models a closed firewall with only pass rules, and a single drop 'all other traffic' rule at the end of the rule set. This means the rule set is conflict free, but may have more rules than necessary, and may be more difficult to understand for the administrator. The interface provided to the administrator is textual, using text files, and no GUI was provided. This was regarded as a surprise success of the system, and an important feature for end-users [40]. However, as a survey carried out by [31] shows, this is not so surprising. The survey found that a majority of system administrators preferred a text based CLI, due to CLIs being more reliable, faster, and more robust.

Around the same time, research at Cisco Systems by [21] produced a policy-based management system. This can model abstract filtering device polices, and automatically create the low-level device configurations, in the form of Cisco ACLs. The system can create filtering device policies, for multiple devices, from global policy rules, as well as performing some basic rule set analysis. The motivation for the system is summarised in the title of their research paper 'Policy-Based Management: Bridging the Gap' [21]. The gap between device configurations and the high-level, natural language, security policy is difficult to bridge for system administrators, and the research introduces a functional language to express the high-level policy. This is done in a precise way, using nested sets of conditional statements. An example of the language, taken from [21], is shown in Listing 3.3. This language has to be manually created by the system administrator, which is a drawback of the system, as the administrator would have to learn this abstract high-level language as well as the low-level device languages.

Listing 1.3. Code Snippet of an HTTP Policy [21]

```
corp_gw =
 {
   If Service is HTTP
     If Destination is S
        If Source is H
           Service level is premium
           Permit
        Else If Source is N1 or N4
           If Source is N4
              Use encrypting tunnel
           Permit
```

The system was implemented as part of the CSPM tool [41], and its functionality has since been incorporated into the Cisco's latest security management tool, Cisco Security Manager [42]. The CSPM interface is a GUI which provides the administrator with a tree view to implement both the policy and the network topology information. The topology tree represents the enforcement devices (firewalls and filtering routers) and the network areas between the devices. The

policy tree contains a policy structure with the individual policies, defined in the abstract policy language. These can then be applied to the enforcement devices in the topology tree to enforce policies between the network zones. Figure 4, taken from [42],shows the interface to the CSPM tool. Note for the two trees, the top represents the policy, and the bottom the network topology. Note also that the inner security policy has been applied to the PIX2 firewall device, which seems to be segmenting a specific internal network from the general internal network.

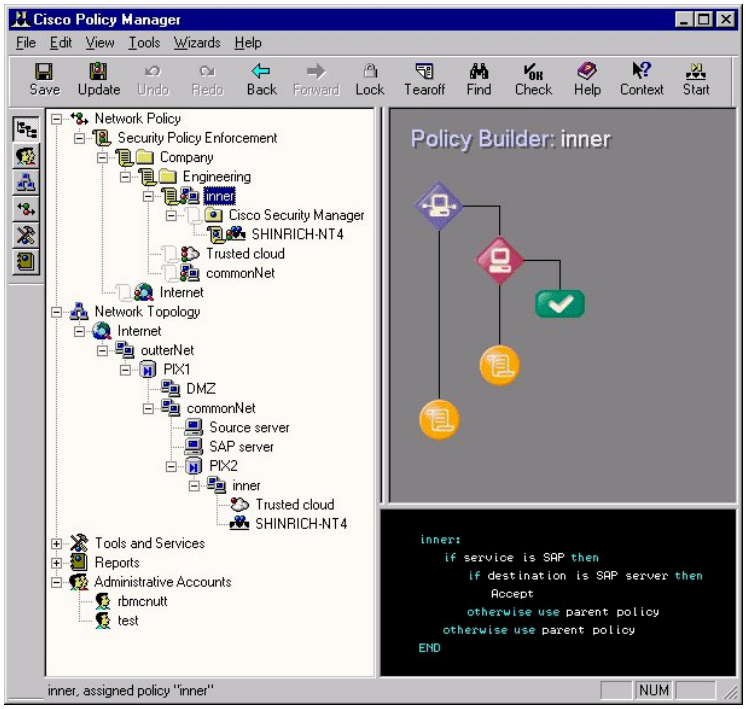

Fig. 4. Cisco Security Policy Manager GUI [42]

The generation of the filtering device configurations consists of four steps. Firstly, the high-level policy is applied to device and the policy is distilled into rules which are applicable to the individual devices. The authors call this process pruning and involves a mechanism to detect if policy rules specify traffic which could pass through each device. In this way, devices only receive rules which are needed on the device. This is an improvement on, and in contrast with, the systems discussed previously which either apply all rules to all devices, or perform very minimal pruning. The second step in the process is some simple rule set analysis. This checks that the device has resources to carry out the policy, such as the memory available for filtering rules. It also performs some rule set anomaly analysis, checking for conflicting, or overlapping, filtering rules. An example of

conflicting rules would be if the filter has the same source and destination address and the same service, but different actions. The administrator is warned about any analysis problems and the administrator would have to decide on the solution, such as which of the two conflicting rules would be used in the rule set. The next step is to generate the device filters. The tool creates an intermediate, abstract, filter rule set for each device and stores them in a database. This stores the semantics of the device configurations to be created, but leaves the creation to an agent which reads the generic filter rules and creates low-level rules in the device configuration language. This means filtering devices could be interchanged and the fourth stage repeated to create configurations for the new devices. Although this is a Cisco specific closed system, it could in theory be used to create other vendor device configurations, if agents were added to do this.

The system is MS Windows-based, and provides only a GUI for the network administrator. There is no support for the CLI and this means the tool, even in its latest form, cannot be scripted. This is something that is sought after by administrators, so the tool can be combined with other tools, and provides customisation in their work practices [32]. Small configuration changes could be time consuming to perform in this type of system, as the high-level policies would have to be studied and amended, and the fourth stage process run again. This type of all or nothing configuration generation may not be very useable for the system administrator [43].

Several other similar approaches, using high-level policy languages and creating low-level device configurations from them, appear in the literature. [44] describes a similar system to Guttman's, with the addition of modeling and configuring NIDS as well as firewalls. PRESTO is a configuration management system for routers [45]. It is interesting, in the way it extends the low-level configuration language, creating a hybrid scripting language rather than introducing an entirely new language. Templates, or configlets, are used with information embedded into the low-level configuration language. These are then used at runtime to generate the low-level configurations, with a database providing the content of the templates.

The Ponder policy specification language can be used to specify the entire high-level security policy, including access control policies, user authorisation policies, and traffic filtering policies [34]. Ponder seems more complex than is needed to specify device filtering policies. The high-level firewall policy modeling language, FLIP [35], can also be compiled into low-level device configurations. The scope of the FLIP system is global and can manage firewalls across an entire network. FLIP generates conflict free rules automatically, by performing conflict analysis as it generates the low-level device configurations. This improves on most of the systems described, which would need to be analysed separately for rule conflicts, although tying the functionality together like this means less flexibility [32]. Another high-level policy language, AFPL [46, 47], resulted from work at the University of Seville. This, again, intends to fill the gap between high-level network security policies and low-level firewall languages. It has been designed to

be simpler than some of the preceding high-level languages while still retaining the functionality needed to describe filtering policies and can also be automatically compiled into leading vendor firewall filtering languages. A similar proposal by Hinrichs et al. in [48] is the Flow-based Management Language (FML). FML is a formal, high-level network policy specification language designed to replace the low-level policy rules by controlling the 'flows' of data within the network, regardless of the physical devices that they flow through. After specifying a policy, FML can be translated to low-level network hardware configuration rule sets automatically, using tools provided by the authors. This work contributes a novel, extensible, adaptable and efficient mechanism for controlling network access. High-level policy languages, nevertheless, face a common hurdle in acceptance from administrators, who generally do not welcome the overheads involved with learning and using them [19].

3.4 SDLC Implementation Phase

Policy Testing Systems Testing of packet filtering rule sets was explored by Hazelhurst et al. in the late 1990's [49]. One of their main motivations for their system was the analysis of low-level rule sets to understand the policy they implement. This ended up being the main focus of research, with a query-based system being developed to analyse rule sets. BDDs were used to represent firewall and router access policies. Each rule in the rule set can be converted into a Boolean expression, and the Boolean expressions combined in a BDD. Queries are used to pose questions about the rules, which can then be answered using the BDDs. An example of these 'what if?' queries might be 'which destination addresses can packets reach from a source address and a certain port?'. The user can analyse, and so test, a policy by querying the rule set in various ways. This can be used to validate and explore the policy before deploying the rule set onto filtering devices. The language used to specify the queries is a functional language 'FL', and the output is a textual representation of the query answer. The FL query language is low-level and is difficult to use, for the system administrator when creating queries. This was recognised, and the system was improved to include a GUI for easier querying of the policy [50]. The authors also recognised that the best interface was a GUI for visualising important information about the rule set, and for basic querying, but a textual interface was better suited for an advanced user to develop more powerful queries [50]. The scope of the system only extends to a single rule set, but later research expanded query-based systems to cover entire networks, some of which are covered later in this paper. The primary analysis mechanism is the manual query and answer system, but a basic rule set conflict analysis process was also developed. This automated the task of detecting redundant rules in the rules set prior to deployment. This seems to be one of the first systems to perform conflict analysis within rule sets and is cited by most research in the area.

The research of Mayer, Wool, and Ziskind continued with the creation of the FANG system [51]. This was built from earlier research into the Firmato system [40], and FANG is actually an analysis engine which runs on top of the same

Firmato policy model. It can be used in the Implementation phase, to test a policy before it is deployed, and could also be use in the Operations phase to audit a deployed policy. It has functionality to build the model from existing filtering configurations, so it is classed as a bottom-up system. It can take Cisco Router configuration files or Lucent Firewall files as input to create the policy model. It uses a separate parser module for each filtering device it supports. This is a good system as it support extensibility. The system works on multiple filtering devices, and so a global policy can be tested. A network topology has to be entered, and this is still done manually using the MDL language, the same way as described for the Firmato system. The queries which can be performed on the FANG system are based around a triple of source host group (source network address range), a destination host group (destination network address range) and a network service. Queries can be created such as (*, web_servers, http_services) to find the answers to questions, such as 'which systems have access to the organisations web servers'. A GUI was created to perform queries, and drop down menus implement the query triples. An example, taken from [40], showing the result of a query asking 'which services can get from the internal network to the DMZ network', is shown in figure 5.

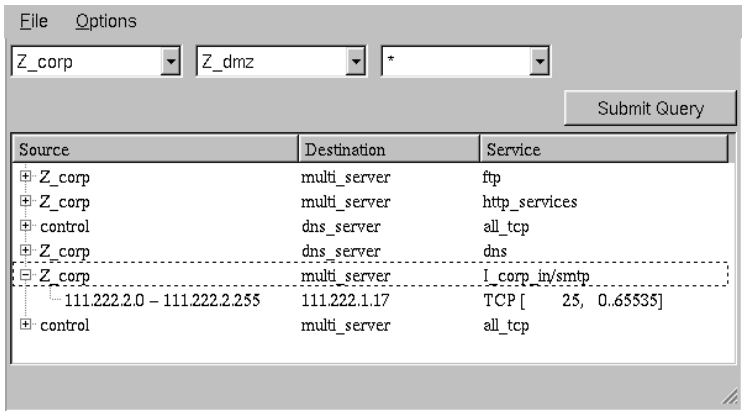

Fig. 5. The FANG systems GUI showing results of a query [40]

A similar query-based system, which was created for the Linux iptables firewall, is ITVal [52]. It uses MDDs rather then BDDs, but operates in much the same way. Their main motivation was to provide a simple query tool to aid in firewall configuration, so an administrator could test a firewall configuration before deploying it. The query language is designed to be simple and natural language based. A single iptables rule set can be read in by the tool and a MDDs model built. The queries are created in the English-based query language and return a simple textual answer, in a similar way as the FANG system does.

The main problem with analysis systems using queries, is that they have to be created by the administrator. The system administrator has to learn another query language, as well as knowing which queries to perform. The onus is on the administrator to work out what, and when, to query.

These off-line passive testing systems have advantages over active testing systems, such as vulnerability testing or penetration testing, as they can be performed before policies are deployed. With active testing the policy has to be deployed before testing, and if problems are found, the production network is vulnerable until a solution can be deployed [51].

Policy Deployment The SNMP protocol [53] was developed by the IETF in the 1980's as a comprehensive network management system. It was intended to be a standard way of managing increasing numbers of devices, across heterogeneous networks. Its functionality includes the ability to configure managed devices remotely. It is implemented as 'agents' on network devices, and the configuration is modeled using MIBs which store information on each specific device. In a workshop held by the IETF in 2002, attended by protocol developers and network administrators, it was found that SNMP was not being used to configure network devices as much as they originally intended. One of the main issues raised was device manufacturers have not implemented all the parts of the protocol needed to read and write configurations to and from devices. As the following, from the 2003 workshop [54] on the subject, states:

> "There is too little deployment of writable MIB modules. While there are some notable exceptions in areas, such as cable modems where writable MIB modules are essential, it appears that router equipment is usually not fully configurable via SNMP" – [54]

> "It is usually not possible to retrieve complete device configurations via SNMP so that they can be compared with previous configurations or checked for consistency across devices. There is usually only incomplete coverage of device features via the SNMP interface, and there is a lack of differentiation between configuration data and operational state data for many features." – [54]

It seems network device vendors are reluctant to fully implement the SNMP agents on their devices, thus their own proprietary languages are needed to manage the device configurations. SNMP is widely supported by vendors, but mostly the monitoring part of the protocol is implemented, rather than the configuration part [55]. This could be a good business decision for vendors, as rather than the administrator using the same IETF protocol on any device, they have to become familiar with that vendor's low-level device specific language, and perhaps have to take some training courses from that vendor. For example, Cisco OSs have subtle differences across their range of network products, and different certifications are needed to manage the different devices. Investment in training administrators in a vendor's products can make it difficult to justify

using any other vendor's products [18]. Vendors typically provide GUI-based products to perform some management of the devices, but these are normally aimed at simple initial setups and inexperiences users performing basic tasks [54]. Network administrators invariably have to use the CLI to perform more complicated management work. In the workshop reported in [54] the CLI was found to be the administrator's favorite way of configuring devices.

Apart from SNMP, several other systems were also discussed at the workshop including the CIM [56], the COPS [57], and using the device CLI and SSH, for configuration management. It was decided that neither CIM or COPS addressed the needs of the users, and that a new vendor independent configuration management protocol was needed. The XML based Netconf protocol [58] was born out of the recommendations from this workshop. Netconf was designed to be a simple mechanism through which device configurations can be managed, using an XML-based system [59]. The entire, or partial, XML encoded configurations, of a Netconf enabled device can be retrieved, updated, and deployed back to the device by remote management applications. The protocols control messages are encoded in XML, as well as the data being sent. Major network device vendors, such as Cisco and Juniper Networks, now have XML-based agents in their latest products and are participating in Netconf standardisation [60]. Cisco Netconf configuration is detailed in [61], and Juniper in [62]. XML has many advantages over SNMP, such as XML is human readable, there are many standard tools and libraries for parsing and processing XML, and XML-based languages are extensible by the user [63].

3.5 SDLC Operation and Maintenance Phase

The operations phase of the SDLC includes configuration management of deployed security systems. This could range from small changes to a single device, to implementation of an new security policy over the entire network. This phase also includes monitoring and auditing of deployed systems, checking for compliance with policies and procedures. This can be done by comparing the implemented mechanisms with the higher level policy definitions. System Administrators can also use these auditing systems to help with their understanding of deployed policy implementations. For example to create high-level policies for management where none currently exist, or during the process of making changes to device configurations. Understanding deployed policies can be extremely challenging [64], especially if the policy is being enforced across many devices over an entire network. Periodic testing of security mechanisms is also recommended in this phase of the life cycle, to check the systems for, and alert administrators to, the latest security vulnerabilities. The auditing and testing processes can be carried out in parallel, and both aim to highlight any weaknesses in the current security solution, and possibly to suggest improvements. This would then feed back into the development phase of the SDLC, as part of a continual process, to improve the security policies and implementation [65, 4]. This continual security process is shown in Figure 3, and summarised nicely by the following quote from [1].

"Implement measures which will protect your assets in a cost effective manner. Review the process continuously and make improvements each time a weakness is found." Fraser et al. [1]

Policy Auditing Systems Some of the systems discussed previously can also be used for testing or auditing of deployed access control policies. Such as the FANG system [51]. It can reverse engineer a model of a policy from firewall configurations. The administrator can then query the policy, to become familiar with it before changes are made, or to check it matches the overall security requirements. This auditing system has been improved on, by the creation of the Lumeta Firewall Analyser [64, 11]. This improved on the Fang system by automatically creating the queries needed to analyse the firewall policy model. Lumeta generates what the authors describe as the most interesting queries, and then displays the answers to these queries. This tries to highlight possible risks in the firewall policy, and to limit the need for user input to the system. The authors recognised the fact that one of the problems with their earlier system was that the administrator had to decide which queries to ask the system, and then create the queries manually. This is described as a major usability problem with FANG [64]. In the FANG system, the administrator has to enter the network topology description manually using the Firmato MDL language. Using this language was raised as a problem by beta testers and in the new Lumeta system, the routing table is used to create this automatically. The GUI to FANG, shown in Figure 5, was also replaced as it was deemed difficult to use by testers. This has been replaced by a batch process which performs a comprehensive simulation of traffic through the firewall policy and reports on this. This is interesting as the administrator users found the original Firmato CLI interface easy to use, and yet it was replaced with a GUI. This shows the design was perhaps not tailored to the type of user correctly [32]. The output from the system is now a report in the form of web pages, with the ability to drill down into more detail if the users needs to. Using HTML to provide this type of flexible reporting is described as an ideal mechanism for security analysis tools [32]. The FANG system can only translate Lucent Managed Firewall, which does not have a large market share. The Lumeta system added parser modules for CheckPoint firewall and Cisco ACL configurations, so heterogeneous networks could be modeled, and therefore the product would be useful to a wider audience. The low-level configurations are abstracted to the Lucent Managed Firewall based language used by Firmato and FANG, and the analysis query engine uses this as input. The Lucent Managed Firewall language was used as it is contains high-level constructs and is easy to parse. The Lumeta architecture, taken from [11], is shown in 6.

The Lumeta system has since been developed into a commercial product, the Algosec Firewall Analyser from Algorithmic Security Inc. [66], which provides multi-vendor firewall analysis, monitoring and auditing. The Firewall Analyser system supports the major enterprise firewall platforms: CheckPoint Firewall-1 software firewall (which runs on various hardware platforms), Cisco PIX, ASA and Router firewalls, and Juniper Netscreen.

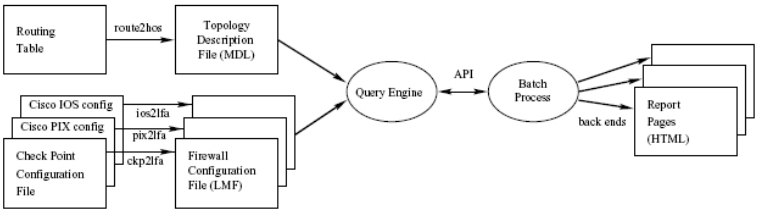

Fig. 6. The Lumeta system architecture [11]

DePaul University in Chicago has contributed greatly to research in the areas of firewall policy modeling and analysis. Al-Shaer and Hamed have been the main contributors, with over a dozen publications between them. Their first research into firewall policy analysis was concerned with auditing legacy firewall policies to automatically discover conflicts and anomalies in firewall policies. This anomaly checking can also assist administrators when editing the deployed policies [16, 67]. The rule set conflict detection aims to highlight possible problems in a rule set based on the order of the rules, and the dependencies between rules due to the ordering. For example a more general rule before a more specific rule in a rule set, would mean the more specific rule would never be reached. The more specific rule is classified as 'shadowed' by the first rule if the filtering actions taken are different (pass and drop), or 'redundant' if the actions are the same (for example, both rules pass the packet). These are classed, by the authors, as rule set 'anomalies' [67]. The FPA tool was created, based around a formal model of the firewall rules and the relationships between them. Modeling of the filtering policies is done using BDDs and algorithms to detect anomalies in the rule set model, have been created. To prove the concept they demonstrate a five tuple filtering syntax, which is used to describe the filtering rules used as input to the system. This maps directly from current low-level filtering languages, such as Cisco ACLs. The format of the five tuple filtering rule is shown in Listing 1.4. The literature only shows examples of these commonly used filtering fields, but the authors state this could easily be extended to include any other filtering fields from low-level languages [16]. This could be extended to include the filtering options available in modern low-level filtering languages, such as Cisco ACLs or Linux IP Tables. Note that the filters used in the examples only use classful ranges of IP addresses, and classless ranges would need a more sophisticated wildcard specification. Al-Shaer and Hamed define all the possible relationships between rules, which are then proved mathematically to be the union of all possible relations [16].

Listing 1.4. Format of filtering rules, used as input to the FPA tool [16]

```
<order> <protocol><src_ip><src_port><dst_ip><dst_port> <action>
```

The firewall rule set, and the relations within, are then modeled as a BDD. This is then represented as a policy tree, with nodes on the tree representing

filtering fields, and branches being the values. Each path through the tree represents a rule in the input rule set. This model was chosen, as the rule set and the anomalies can be visualised by the users. An example of this type of tree, taken from [68], is shown in Figure 7.

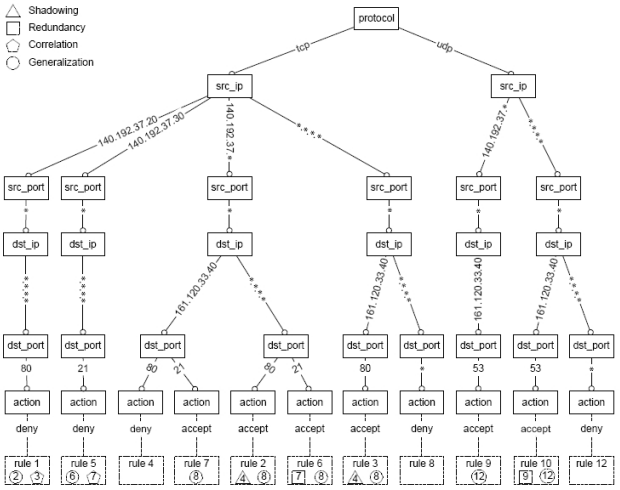

Fig. 7. FPA tool BDD Tree model of a filtering rule set [68]

Algorithms to detect any anomalies within the rule set are then run, and can be displayed to the user in the FPA tool interface. The interface provided by the system is a GUI, which can show the policy tree, and any rule set anomalies. The interface is shown in Figure 8, which is taken from [16]. The policy tree is shown in the left hand pane, and the discovered anomalies in the right hand pane. The rule set is shown in the bottom pane, with the conflicts highlighted. The FPA tool also allows the user to maintain the rule set, inserting, modifying and deleting rules. As the user edits the rules, the tool provides feedback on any conflicts that may be introduced by these actions.

Although the use of system is described as analysis of legacy firewall policies, by the authors, no automatic importing of these policies is described. The system seems to need an administrator to manually translate the low-level rule set into the five tuple syntax. This would classify the systems top-down, however, the translation of deployed policies is not difficult as it is a direct mapping from most firewall configuration languages. Fixed format configurations from devices can be easily parsed using a scripting language, such as Perl [64].

The authors regarded this system as a step forward from the query-based analysis systems such as FANG [51] and Lumeta [64]. The main reason for this is that it takes much more effort to redefine deployed rules sets into the high-

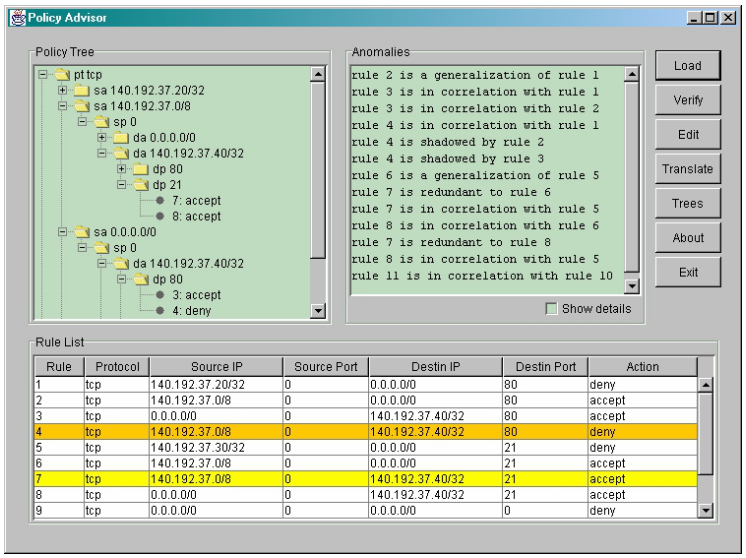

Fig. 8. FPA tool GUI [16]

level policy specification languages used by these systems, rather than analysing the rule sets directly with the FPA system [68].

The scope of the FPA system was a single device and rule set, but this was extended to anomaly detection across multiple firewalls [24, 69]. Further research extended the system to include other filtering devices, including IDS, and VPN gateways, and the inter-device anomaly analysis, across the global policy [70].

The FIREMAN Firewall Analysis system [71] can perform analysis of firewall filtering, detecting redundant and conflicting rules, which may point to misconfigurations. Again, BDDs are used to model one or more firewalls. The analysis can discover mis-configurations, similar to anomalies defined in the FPA system, as well as detecting policy violations in the policy. The policy violations are based on a blacklist or whitelist input by the user, as well as a general policy for all rule sets based on the twelve common firewall configuration errors identified by [10]. The model is created with a bottom-up approach, by parsing device configuration files. An evaluation of the system was carried out, by creating artificial rule sets of up to 800 rules. Performance evaluation was carried out based on timing metrics. The experimental conditions used, were rule sets increasing size. The system can analyse up to 800 rules in less than three seconds.

Research at AT&T labs produced a configuration auditing and management system called EDGE [18]. It creates an abstract model of the networks routing information by reading in and processing entire device configuration files. The network is modeled using an entity relationship model stored in a database of network services described in the configuration files. Off-line analysis of the network configuration is then provided, and reports are generated. These can be

used to identify problems such as inconsistent routing configurations on devices. The administrator can decide whether to correct these highlighted problems, before the system rewrites the configurations back out to the devices. This is an example of a pragmatic, bottom-up, approach to modeling from network configurations. The motivation for the system is similar to many of the firewall management systems. The configuration languages used to configure these complex routing systems are difficult for the administrator to get right. Automation provisioning of configurations is proposed as a solution in this case, with the database model of the current network being used to facilitate this. The interface to the system is via the EDGE web site. EDGE users have access to their own networks data. Various network visualisations are available to help the administrator understand the policies, such as network topology and routing information. Various web-based reports are also available on request. This system has been successfully commercialised, and runs daily on thousands of enterprise networks, helping administrators manage complex networks [18].

One of the main problem with systems which convert to and from vendor specific configurations, such as the systems discussed in this section, is that the configuration languages tend to change rapidly. For example, Cisco's ACL filtering language has had many additional features added to it since the first version in the mid 1990s. This means that the parser modules have to be continually reviewed, to keep them up to date with the latest language changes [18].

"Command line interfaces often lack proper version control for the syntax and the semantics. It is therefore time consuming and error prone to maintain programs or scripts that interface with different versions of a command line interface." – [54]

As part of another configuration management system being developed by Caldwell et al., some interesting research into Cisco Router IOS parsing and modeling was carried out at AT&T Labs. In [72], a learning system and adaptive parser are described. The overall system will be able to process and extract information from existing network configurations, much as described in their previous work with the EDGE system [18], but the parser can adapt to changes in the configuration language being parsed. The parser is automatically generated from valid configurations, which are fed into the learning system. The system, if perfected, would not need manual changes to be made whenever the configuration language version had new features introduced. This would overcome one of the major problems with the bottom-up approach to any type of policy modeling, especially when dealing with rapidly changing configuration languages such as Cisco device operating systems.

Reverse Engineering Security Policies Al-Shaer and Hamed, at DePaul University seem to have produced the first research into reverse engineering of natural language, high-level policies from existing device configurations in 2002 [16, 73]. They aggregate network services together, using their FPA system model as a base, and produce a basic text based abstract policy description from a filtering rule set. They take their BDD model of the filtering rule set, and

create another BDD based on network services from that. Services can then be aggregated together, changing it into a form which can be presented to the user in a natural language. The interface for the tool was a GUI showing the network services based tree and the textual representation of the policy. Figure 9, taken from [16], shows the network service-based tree and the high-level policy which has been inferred.

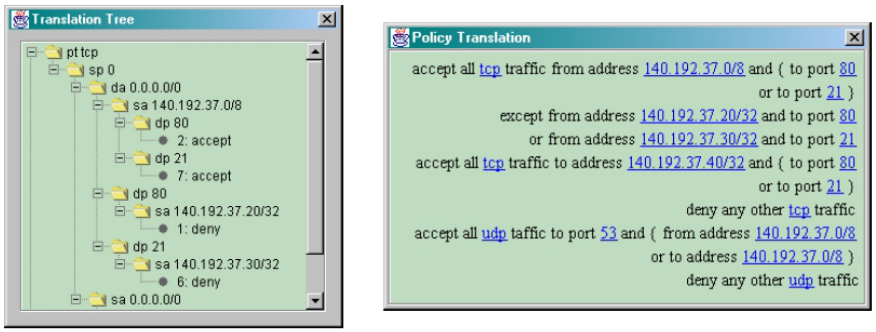

Fig. 9. Policy Inference Tool GUI [16]

The creation of an abstract high-level policy from low-level device configurations is a fairly new subject for research. The 2007 paper by Tongaonkar, Inamdar, and Sekar describes a technique to extract the high-level security policy from analysis of the low-level representation of a rule set [74]. Their system attempts to flatten the rule set, by aggregating overlapping rules together, thus eliminating the dependency on the order of the rules. This also reduces the size and complexity of the rule set, hopefully giving the administrator a better understanding of the policy it represents. This is a similar technique to the [16] system.

In [22], Bishop and Peisert developed a system of reverse engineering access control policies from Linux file system access control configurations. This doesn't involve firewall policies however, only user access policies from Linux password files. Around the same time Golnabi et al. have also worked on the problem of inferring high-level policies [75], but their approach is based around an active system, and the data mining of the device logs, and not by static analysis of the device configurations. Similarly, Abedin et al. in [76] describe techniques used to mine firewall log files to regenerate effective firewall rules. Their method uses algorithms to reduce the data set through mining of firewall network traffic logs using packet frequencies. These are then used to calculate the occurrence of each attribute of a log record in a firewall log file, thus generating primitive rules. Firewall rules are then regenerated from these primitive rules using aggregation and a set of heuristics. Anomalies in the original rule set and defects

in the firewall implementation are then identified through a comparison of the regenerated rules with the original, manually-defined rules.

Policy Visualisation Another way to analyse or validate a policy is to graphically represent the rule set in some form. This technique has been used in several tools to some extent, for example the FPA system Al-Shaer and Hamed uses visualisation of BDDs tree to represent the firewall policy. This was regarded as an important design feature. The choice of a BDD as a model was used over, faster modeling techniques for simplicity and ease of visualisation to the user [67].

[77] describes a firewall packet filter visualisation tool, PolicyVis. A GUI provides the administrator with a visualisation of packets passed or blocked by the firewall, based on a set of specified query parameters. The administrator can enter queries, much like the Firmato or FANG systems, and instead of textual answers, a visualisation of the filtered traffic is generated. Like the query-based systems reviewed earlier in this paper, this leaves the administrator to create the queries.

[19] discusses a system, to compliment the CLI, visualising the most complex parts of the policy. Their system was based around improving the usability of network administrators current systems, by complimenting them with visualisations, but not replacing them. Their approach is different from the clean-slate approaches taken in some of the systems reviewed in this section, such as with new high-level policy specification languages. They assume that the administrator prefers the low-level CLI based tools, they are familiar with, and attempt to give additional usablility through visaulisation rather than replacing the CLI tools. This assumption is made, based on research into the network administration community and the context of the work administrators perform [31, 78].

4 Conclusions

This paper explored literature in the areas of network security, security policies, policy enforcement, and firewall policy management systems. A taxonomy of systems which aid the administrator in the management of firewall policies, was also carried out. It has been shown that security policies play a central and important role in network security. High-level policies are created with input from management, and are typically written in a natural language. These policies then have to be mapped to the technical policies which network devices enforce. The process of mapping these high-level policies to low-level device configurations is normally performed manually by network administrators. This task is difficult and error prone as there is a large conceptual gap between the high-level policies and the configurations. The low-level firewall policies themselves are difficult to understand, reverse engineer, and maintain due to the complexity in large rule sets, and the fact that overall policies are sometime enforced over multiple firewalls and filtering devices. Many systems to help with the understanding and management of these policies have been proposed in the literature. Much

research into policy-based systems has been carried out, including high-level abstract policy languages, query-based rule set testing systems, and firewall rule set auditing systems. High-level languages, used to bridge the gap between high-level policies and low-level firewall configurations have been developed, but have some problems. A common problem with abstract policy languages is that they require configuration by specialists. The administrator typically does not want to, or have the time to learn these new languages. Some of the languages also have limitations with the features they support. Query-based analysis systems have been produced specifically for firewall auditing, and analyse the firewall rule set by simulating network traffic passing through the firewall, based on queries input by the user. Some have overcome the problem of abstract language based systems, as they use models hidden from the user, which can be created from device configurations automatically. Some query-based systems have problems also, as they put the onus on the administrator to create the queries, and sometimes to learn a new query language which leads to the same problems as high-level modeling languages. Other off-line auditing systems were reviewed, including rule set anomaly detection systems, visualisation systems, and high-level policy reverse engineering systems - which are still in their infancy. From the review, it was highlighted that off-line configuration analysis is the area were some of these systems have become commercialised, such as automated firewall configuration analysis, and routing configuration analysis. These off-line passive systems have advantages over active testing systems, such as vulnerability testing, as they can be performed before policies are deployed. Another issue raised in the research more than once, was that of which interface systems had. Network and security administrators prefer a low-level, textual based CLI interface over an abstract interface, such as a GUI for most tasks.

Bibliography

[1] B. Fraser, J. P. Aronson, N. Brownlee, and F. Byrum, "Site security handbook (rfc 2196)," Sep 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2196.txt?number=2196

[2] B. Schneier, *Secrets and Lies - Digital Security in a Networked World*. Wiley, 2000.

[3] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin, *Firewalls and Internet Security: Repelling the Wiley Hacker, 2nd Edition*. Addison-Wesley, Feb 2003.

[4] Y. Bhaiji, *CCIE Professional Development - Network Security Technologies and Solutions*. Cisco Press, 2008.

[5] D. Danchev, "Building and implementing a successful information security policy," online at www.windowsecurity.com, 2003.

[6] E. Guttman, L. Leong, and G. Malkin, "Users' security handbook (rfc 2504)," Feb 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2504.txt?number=2504

[7] P. Samarati and S. C. de Vimercati, "Access control: Policies, models, and mechanisms," *Lecture Notes in Computer Science*, vol. 2171, pp. 137–196, 2000.

[8] T. Corbitt, "Protect your computer system with a security policy," *Management Services*, vol. 46(5), pp. 20–21, 2002. [Online]. Available: http://findarticles.com/p/articles/mi_qa5428/is_200205/ai_n21313131/pg_2?tag=artBody;col1

[9] Cisco Systems, "Cisco safe," 2009. [Online]. Available: http://www.cisco.com/en/US/netsol/ns954/index.html

[10] A. Wool, "A quantitative study of firewall configuration errors," *Computer*, vol. 37, no. 6, pp. 62–67, 2004.

[11] A. Mayer, A. Wool, and E. Ziskind, "Offline firewall analysis," *International Journal of Information Security*, vol. 5, no. 3, pp. 125–144, 2006.

[12] E. M. Madigan, C. Petrulich, and K. Motuk, "The cost of non-compliance: when policies fail," in *SIGUCCS '04: Proceedings of the 32nd annual ACM SIGUCCS conference on User services*. New York, NY, USA: ACM, 2004, pp. 47–51.

[13] S. Bellovin and W. Cheswick, "Network firewalls," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 50–57, 1994.

[14] F. B. Schneider, "Enforceable security policies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 1, pp. 30–50, 2000.

[15] B. Blakley, "The emperor's old armor," in *Proceedings of the 1996 workshop on New security paradigms*. ACM Press New York, NY, USA, 1996, pp. 2–16.

[16] E. Al-Shaer and H. Hamed, "Design and implementation of firewall policy advisor tools," DePaul University, CTI, Tech. Rep., 2002.

[17] A. D. Rubin, D. Geer, and M. J. Ranum, *Web Security Sourcebook*. Wiley, 1997.

[18] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford, "The cutting edge of ip router configuration," in *Proceedings of 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)*, 2003.

[19] T. Wong, "On the usability of firewall configuration," in *Symposium on Usable Privacy and Security*, 2008.

[20] F. Cuppens, N. Cuppens-Boulahia, T. Sans, and A. Miege, "A formal approach to specify and deploy a network security policy," in *Formal Aspects in Security and Trust*, ser. IFIP International Federation for Information Processing, vol. 173/2005. Springer Boston, 2004, p. 203218.

[21] S. Hinrichs, "Policy-based management: Bridging the gap," in *Computer Security Applications Conference, Annual*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 1999, pp. 209–218.

[22] M. Bishop and S. Peisert, "Your security policy is what," University of California at Davis, Tech. Rep., 2006.

[23] A. Wool, *Packet Filtering and Stateful Firewalls*. Wiley, 2006, ch. Firewall Architectures, p. 526.

[24] E. Al-Shaer and H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, 2004.

[25] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith, "Implementing a distributed firewall," in *Proceedings of the 7th ACM conference on Computer and communications security*. ACM New York, NY, USA, 2000, pp. 190–199.

[26] M. Abrams and D. Bailey, "Abstraction and refinement of layered security policy," in *Information Security: An Integrated Collection of Essays*. IEEE Computer Society Press, 1995, pp. 126–136.

[27] R. Sandhu, "The typed access matrix model," in *1992 IEEE Computer Society Symposium on Research in Security and Privacy, 1992. Proceedings.*, 1992, pp. 122–136.

[28] A. El-Atawy, T. Samak, Z. Wali, E. Al-Shaer, F. Lin, C. Pham, and S. Li, "An automated framework for validating firewall policy enforcement," in *POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 151–160.

[29] NIST, "Generally accepted principles and practices for securing information technology systems," NIST, Tech. Rep., 1996.

[30] ——, "Special publication 900-64 - security considerations in the system development life cycle," NIST, Tech. Rep., 2008.

[31] E. M. Haber and J. Bailey, "Design guidelines for system administration tools developed through ethnographic field studies," in *CHIMIT '07: Proceedings of the 2007 symposium on Computer human interaction for the management of information technology*. New York, NY, USA: ACM, 2007, p. 1.

[32] P. Jaferian, D. Botta, F. Raja, K. Hawkey, and K. Beznosov, "Guidelines for designing it security management tools," in *CHiMiT '08: Proceedings of the 2nd ACM Symposium on Computer Human Interaction for Management of Information Technology*. New York, NY, USA: ACM, 2008, pp. 1–10.

[33] V. Tsoumas and T. Tryfonas, "From risk analysis to effective security management: towards an automated approach," *Information Management & Computer Security*, vol. 12, pp. 91–101, 2004.

[34] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *Workshop on Policies for Distributed Systems and Networks*, 2001.

[35] B. Zhang, E. Al-Shaer, R. Jagadeesan, J. Riely, and C. Pitcher, "Specifications of a high-level conflict-free firewall policy language for multi-domain networks," in *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2007, pp. 185–194.

[36] V. Zaliva, "Firewall policy modeling, analysis and simulation: a survey," SourceForge, Tech. Rep., 2008.

[37] J. D. Guttman, "Filtering postures: local enforcement for global policies," vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 1997, p. 0120.

[38] ——, "Rigorous automated network security management," *International Journal of Information Security*, vol. 4, pp. 29–48, 2005.

[39] ——, "Security goals: Packet trajectories and strand spaces," *Lecture Notes in Computer Science*, pp. 197–261, 2001.

[40] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, "Firmato: A novel firewall management toolkit," *IEEE Symposium on Security and Privacy*, vol. 0, pp. 17–31, 1999.

[41] Cisco Systems, "Cisco secure policy manager." [Online]. Available: http://www.cisco.com/en/US/products/sw/secursw/ps2133/prod_technical_reference09186a00800a9ebc.html

[42] ——, "Cisco security manager." [Online]. Available: http://www.cisco.com/en/US/products/ps6498/index.html

[43] S. Lee, T. Wong, and Kim, "To automate or not to automate: On the complexity of network configuration," in *IEEE International Conference on Communications (ICC)*, 2008, pp. 5726 – 5731.

[44] T. Uribe and S. Cheung, "Automatic analysis of firewall and network intrusion detection system configurations," *Journal of Computer Security*, vol. 15, no. 6, pp. 691–715, 2007.

[45] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, S. Rao, and W. Aiello, "Configuration management at massive scale: system design and experience," in *2007 USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–14.

[46] S. Pozo, R. Ceballos, and R. M. Gasca, "Afpl, an abstract language model for firewall acls," in *ICCSA '08: Proceedings of the international conference on Computational Science and Its Applications, Part II*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 468–483.

[47] S. Pozo, A. J. Varela-Vaca, and R. M. Gasca, "Afpl2, an abstract language for firewall acls with nat support," *International Conference on Dependability*, pp. 52–59, 2009.

[48] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, "Practical declarative network management," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*, ser. WREN '09. New York, NY, USA: ACM, 2009, pp. 1–10.

[49] S. Hazelhurst, A. Fatti, and A. Henwood, "Binary decision diagram representations of firewall and router access lists," Department of Computer Science, University of the Witwatersrand, Tech. Rep., 1998.

[50] S. Hazelhurst, "Algorithms for analysing firewall and router access lists," University of theWitwatersrand, Tech. Rep., July 1999.

[51] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," *Security and Privacy, IEEE Symposium on*, vol. 0, p. 0177, 2000.

[52] R. Marmorstein and P. Kearns, "A tool for automated iptables firewall analysis," in *Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC)*. Berkeley, CA, USA: USENIX Association, 2005, pp. 44–44.

[53] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (snmp) (rfc 1157)," May 1990. [Online]. Available: http://tools.ietf.org/html/rfc1157

[54] J. Schoenwaelder, "2002 iab network management workshop (rfc 3535)," 2003.

[55] C. Ehret, "From snmp deception to verinecs cisco service," Master's thesis, University of Fribourg, Switzerland, 2005.

[56] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy core information model - version 1 specification (rfc 3460)," 2001. [Online]. Available: http://tools.ietf.org/html/rfc3460

[57] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, "The cops (common open policy service) protocol (rfc2748)," Jan 2000. [Online]. Available: http://tools.ietf.org/html/rfc2748

[58] A. Bierman, K. Crozier, R. Enns, T. Goddard, E. Lear, P. Shafer, S. Waldbusser, and M. Wasserman, "Netconf configuration protocol (rfc 4741)," Dec 2006. [Online]. Available: http://tools.ietf.org/html/rfc4741

[59] S. Halle, R. Deca, O. Cherkaoui, R. Villemaire, and D. Puche, "A formal validation model for the netconf protocol," *15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2004)*, pp. 147–158, 2004.

[60] M. Choi, H. Choi, J. Hong, H. Ju, and S. POSTECH, "Xml-based configuration management for ip network devices," *Communications Magazine, IEEE*, vol. 42, no. 7, pp. 84–91, 2004.

[61] Cisco Systems, "Network configuration protocol," Jun 2009. [Online]. Available: http://www.cisco.com/en/US/docs/ios/netmgmt/configuration/guide/nm_cns_netconf.pdf

[62] Juniper Networks, "Netconf api guide," 2008. [Online]. Available: http://www.juniper.net/techpubs/software/junos/junos91/netconf-guide/netconf-guide.pdf

[63] G. Munz, A. Antony, F. Dressler, and G. Carle, "Using netconf for configuring monitoring probes," in *IEEE/IFIP Network Operations & Management Symposium (IEEE/IFIP NOMS 2006), Poster Session, Vancouver, Canada, Apr*, 2006.

[64] A. Wool, "Architecting the lumeta firewall analyzer," in *Proceedings of the 10th conference on USENIX Security Symposium*, USENIX Association Berkeley, CA, USA. USENIX Association, 2001, pp. 7–7.

[65] L. W. Wai, "Sans security life cycle," 2001. [Online]. Available: http://www.sans.org/reading_room/whitepapers/testing/security_life_cycle_1_diy_assessment_260?show=260.php&cat=testing

[66] Algosec, "Algosec firewall analyser," 2009. [Online]. Available: http://www.algosec.com/en/products/firewall_analyzer.php

[67] E. Al-Shaer and H. Hamed, "Firewall policy advisor for anomaly discovery and rule editing," in *Integrated Network Management*, 2003, p. 1730.

[68] ——, "Modeling and management of firewall policies," *IEEE Transactions on Network and Service Management*, vol. 1-1, pp. 2–10, 2004.

[69] ——, "Taxonomy of conflicts in network security policies," *IEEE Communications Magazine*, vol. 44, no. 3, pp. 134– 141, 2006.

[70] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. ElBadawi, "Towards global verification and analysis of network access control configuration," DePaul University, Chicago, IL, USA, Tech. Rep., 2008.

[71] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra, "Fireman: A toolkit for firewall modeling and analysis," *Security and Privacy, IEEE Symposium on*, vol. 0, pp. 199–213, 2006.

[72] D. Caldwell, S. Lee, and Y. Mandelbaum, "Learning to talk cisco ios: Inferring the ios command language from router configuration data," AT&T, Tech. Rep., 2007.

[73] E. Al-Shaer and H. Hamed, "Management and translation of filtering security policies," in *2003 IEEE International Conference on Communications*. IEEE Press, 2003.

[74] A. Tongaonkar, N. Inamdar, and R. Sekar, "Inferring higher level policies from firewall rules," in *LISA'07: Proceedings of the 21st conference on Large Installation System Administration Conference*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–10.

[75] K. Golnabi, R. Min, L. Khan, and E. Al-Shaer, "Analysis of firewall policy rules using data mining techniques," in *10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, 2006.

[76] M. Abedin, S. Nessa, L. Khan, E. Al-Shaer, and M. Awad, "Analysis of firewall policy rules using traffic mining techniques," *International Journal of Internet Protocol Technology*, vol. 5, pp. 3–22, April 2010.

[77] T. Tran, E. Al-Shaer, and R. Boutaba, "Policyvis: firewall security policy visualization and inspection," in *LISA'07: Proceedings of the 21st conference on Large Installation System Administration Conference*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–16.

[78] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher, "Towards understanding it security professionals and their tools,"

in *SOUPS '07: Proceedings of the 3rd symposium on Usable privacy and security.* New York, NY, USA: ACM, 2007, pp. 100–111.