A model-driven approach for the extraction of network access-control policies

Salvador Martínez, Jordi Cabot
ATLANMOD, & Ecole des Mines de Nantes,
INRIA, LINA
Nantes, France
{salvador.martinez _perez,
jordi.cabot}@inria.fr

Joaquin Garcia-Alfaro,
Frédéric Cuppens, Nora
Cuppens-Boulahia
Télécom Bretagne ; LUSSI Department
Université Européenne de Bretagne
Cesson Sévigné, France
forename.surname@telecom-
bretagne.eu

ABSTRACT

Network security constitutes a critical concern when developing and maintaining nowadays corporate information systems. Firewalls are a key element of network security by filtering the traffic of the network in compliance with a number of access control rules that enforce a given security policy. Unfortunately, once implemented, and due to the complexity of firewall configuration languages and the underlying network topology, knowing which security policy is actually being enforced by the network system is a complex and time consuming task that requires low-level and, often, vendor-specific expertise. In an always-evolving context, where security policies are often updated to respond to new security requirements, this discovery phase becomes critical since it could hamper the proper evolution of the system and compromise its security. To tackle this problem, our approach generates an abstract model of the firewall configurations in a network that facilitates the understanding and evolution of network security policies.

Keywords

Model-driven, security, reverse-engineering

1. INTRODUCTION

Network security policies constitute a critical concern when developing and maintaining corporate information systems. Among them, access control policies which are often the mechanism of choice to guarantee the confidentiality and integrity of resources within a network, present, due its relative simplicity with respect to other techniques, like cryptography, an special interest.

Access-control policies indicates which subjects are allowed to interact with a given object. In the networking environment, they indicate which hosts can send, under given conditions (i.e., which protocol or port is used, etc) , messages to other host. In this context, firewalls, designed to filter the traffic of a network with respect to a given number of filtering rules, are key elements in the enforcement of security policies.

Although there exist approaches to derive firewall configurations from high-level network policy specifications, this process is still normally done by hand, using low level and, often, vendor-specific rule filtering languages. Moreover, the network topology, that may include several firewalls, may impose the necessity of splitting the enforcement of the global security policy among several elements. All this complexity, could potentially introduce differences between the implemented policy and the desired one.

Therefore, once implemented, knowing which access control policy is actually being enforced by the firewalls in the network system is a complex and time consuming task that requires, again, low-level and vendor-specific expertise. Given a network system consisting in several firewalls configured with hundreds of rules, the feasibility of this manual approach could be seriously questioned. As a consequence, in an always-evolving context, where security policies are often updated to respond to new security requirements, this policy discovering task becomes critical as it could hamper the proper evolution of the system and compromise its security.

We believe that, in order to tackle this problem, a first step is to raise the level of abstraction of the information contained in the firewall configurations files so that the access control policy they implement is easier to understand, analyze and manipulate.

Thus, we propose here a model-driven approach aimed at extracting a model of the access control policy enforced by the firewalls within a network system. Our approach produces a model that abstracts the information from the low-level firewall configurations so that low-level, concrete language and vendor-specific expertise is not longer needed to understand which access control policy is being implemented. Moreover, complexities due to the topology are also hidden as the information contributed by each firewall is combined into a single model that only shows hosts, services and permissions for connections that are relevant for the global access control

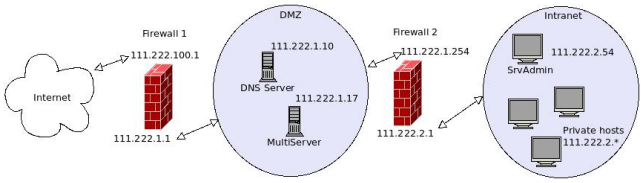

Figure 1: Network example

policy of the network, disregarding information only locally relevant (i.e., information that is only relevant for the policy of a single firewall but not for the global network). Once this model is available, reasoning about the policy implemented on the network is accessible for non-technical security experts. Moreover, a plethora of Model Driven Development (MDD) tools, like query engines, graphic editors, etc., will become automatically available.

We demonstrate the feasibility of our approach by providing a prototype implementation working for firewalls using the netfilter iptables rule filtering language.

The rest of the paper is organized as follows. Section 2 presents a motivating and running example. In section 3 we present and describe our approach whereas in section 4 we discuss a prototype implementation of it. Section 5 discuss related works and finally the paper finish in section 6 with some conclusions and future works.

2. MOTIVATING EXAMPLE

In order to illustrate the problem and simplify the discussion, we introduce here a running example that will be used all along the present work.

The example consist in a network system and the corresponding firewall configurations. The network system is based in the well-known network system described in [2] and depicted in figure 1. This network system contains the following elements:

- An intranet composed by a number of private hosts where one of the private hosts acts as an administrator of certain services provided by the network system.
- A DMZ (demilitarized zone) that contains two servers. A DNS server and a multiserver providing the following services: HTTP/HTTPS (web), FTP, SMTP (email) and SSH.
- Two firewalls controlling the traffic towards and from the DMZ. The first firewall controls the traffic between the public hosts (the Internet) and the services provided by the DMZ. The second firewall controls the traffic between the intranet and the DMZ.

The firewalls in the exemplary network control the traffic from and towards the services provided in the DMZ. In the following, we show the configuration excerpts of these firewalls with respect to two of the provided services, HTTP and SMTP. These sample configurations use the netfilter iptables [12] rule language. Note that although these configurations files are written using the iptables custom chain feature, which allow the user to define exclusions to rules without using drop or deny rules, the concrete rule language and coding style does not affect our approach as we will show in section 3.

First, we will consider here that the global policy for both firewalls states the rejection of any connection not explicitly allowed as shown by the rules in listing 1.

Listing 1: Global policy netfilter configuration

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

Listing 2: Firewall 1 netfilter configuration

```
iptables -N Out_SMTP
iptables -A FORWARD -s 111.222.1.17 -d 0.0.0.0/0 -p tcp --
    dport 25 -j Out_SMTP
iptables -A Out_SMTP -d 111.222.0.0/16 -j RETURN
iptables -A Out_SMTP -j ACCEPT

iptables -N In_SMPT
iptables -A FORWARD -s 0.0.0.0/0 -d 111.222.1.17 -p tcp --
    dport 25 -j Out_SMTP
iptables -A Out_SMTP -s 111.222.0.0/16 -j RETURN
iptables -A Out_SMTP -j ACCEPT

iptables -N NetWeb_HTTP
iptables -A FORWARD -s 0.0.0.0/0 -d 111.222.1.17 -p tcp --
    dport 80 -j NetWeb_HTTP
iptables -A NetWeb_HTTP -s 111.222.0.0/16 -j RETURN
iptables -A NetWeb_HTTP -j ACCEPT
```

The firewall number 1 controls the traffic from the public hosts to the services provided in the DMZ. The listing 2 shows the rules that control the access to the SMTP and HTTP services.

The first chain controls the outcoming SMTP messages towards the public host. It allows them for every hosts but for the host in the local network. The second chain controls the incoming SMPT messages to the server. If the request is done through one machine belonging to the local network, it is rejected while it is allowed for any other machine. The third rule controls the HTTP requests from the public hosts. Again, connections are allowed for any host but for the local ones.

Listing 3: Firewall 2 netfilter configuration

```
iptables -N IntraMail_SMTP
iptables -A FORWARD -s 111.222.2.0/24 -d 111.222.1.17 -p
    tcp --dport 25 -j IntraMail_SMTP
iptables -A IntraMail_SMTP -s 111.222.2.1 -j RETURN
iptables -A IntraMail_SMTP -s 111.222.2.54 -j RETURN
iptables -A IntraMail_SMTP -j ACCEPT

iptables -N IntraWeb_HTTP
iptables -A FORWARD -s 111.222.2.0/24 -d 111.222.1.17 -p
    tcp --dport 80 -j IntraWeb_HTTP
iptables -A IntraWeb_HTTP -s 111.222.2.1 -j RETURN
iptables -A IntraWeb_HTTP -s 111.222.2.54 -j RETURN
iptables -A IntraWeb_HTTP -j ACCEPT
```

The firewall number 2 controls the traffic from the private hosts to the services provided in the DMZ. Listing 3 shows the rules that control the access to the SMTP and HTTP services.

The first chain controls the SMTP requests to the server. They are all allowed for the hosts in the private zone discarding only the administrator host, identified by the IP ad-

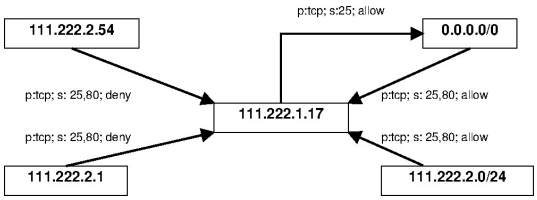

Figure 2: Access-control model

dress 111.222.2.54, and for the firewall interface, identified by IP address 111.222.2.1. The second chain does the same for the HTTP requests. Again, they are allowed for all the hosts in the private zone discarding only the administrator host and the firewall interface.

Example evaluation:

The task of extracting the global access control policy enforced by the set of rules in these two firewalls (that are just minimal excerpts of what a full configuration policy would be) requires expert knowledge about netfilter iptables. Its syntax along with its execution semantics would have to be mastered to properly interpret the meaning of the configuration files. Moreover, the information from the two configuration files and the global policy for the default iptables chains would have to be combined as they collaborate to enforce the global policy and can not be regarded as isolated.

In networks composed by several firewalls, configured by hundreds of rules and potentially from different vendors using different configuration languages and execution semantics, the task of manually extracting the enforced access control policy would become too complex and expensive to be feasible.

Conversely, after applying our approach, a security expert willing to analyze the security policy would face a simple model, showing the hosts and their allowed connections, i.e., directly the access control model of the network. In the diagram in figure 2, we show such a network access control model for the examplary firewall configurations. Note that the connection to and from the host 111.222.0.0/16 does not appear, as they are only relevant in the local context of the firewall 1. Indeed, the derivation of global to local exceptions is a highly error-prone task [6]. Our approach aims at easing and handling this problem as well.

3. EXTRACTION APPROACH

We believe the level of abstraction of firewall configurations has to be raised in order to facilitate the understanding and manipulation of the access control policy of a network system. Thus, we describe here a model-driven approach that extracts a model of the global access control policy from the set of configuration files of the firewalls enforcing that policy. The approach, depicted in figure 3 consist in the following steps:

1. Parsing/injecting firewall configuration files into platform specific models (PSMs).

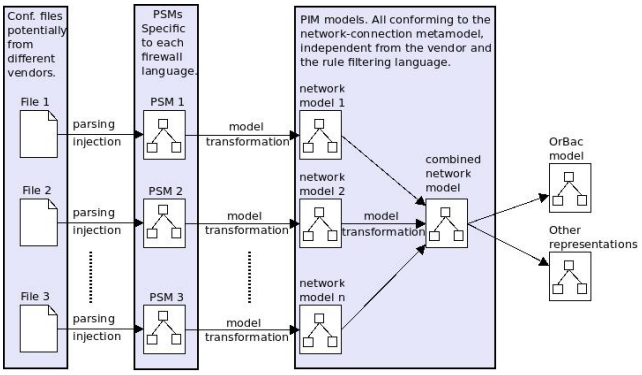

Figure 3: Model-driven process

2. Transforming the obtained PSMs into Platform independent models (PIMs), conforming to a metamodel able to represent network access-control policies.

3. Aggregating the individual PIM models (all conforming to the same metamodel) into a global model conforming to the same metamodel.

4. Additionally, a further step aiming to reach formalisms focused in general access control and not network access-control can be performed by transforming the aggregated model into access-control models like RBAC[13] and OrBac [1].

Before starting a detailed description of the approach, we have to remark that firewall configuration files may contain anomalies and inconsistencies. *Rule shadowing*, *rule redundancy* and *rule irrelevance* are well known problems that may appear in firewall configuration files. However, in the present work we consider either the configuration files are correct and free from this kind of errors or that algorithms to detect and correct such errors, as the ones presented in [5] and [14], has been applied before starting the extraction process. Discussing whether to perform this step directly on the models is better or worse than doing it in the configuration files falls out of the scope of the present paper. Note however that model checking techniques have been proved useful to perform this rule fixing step as mentioned in [9].

Step1. Parsing and Injection.

The first step of our approach constitutes a mere translation between technical spaces where the information in the configuration files is expressed in terms of models. A platform-specific metamodel (PSM) and a parser recognizing the grammar of each concrete firewall rule-filtering language involved in the process (as the one presented in section 4 for Iptables) is required. Note that this step is performed without loosing any information and that the obtained models rest at the same abstraction level as the configuration files.

Step2. PIM2PSM.

In the second step, the PSMs obtained in the first step are translated into PIM models able to represent access-control policies. For this purpose, we propose here a simple network-

connection metamodel that is further explained in the last part of this section.

At this point, the information is already independent from the concrete firewall vendor, filtering language and rule organization (e.g., in iptables we can use negative logic to explicitly drop and deny connections or do the same by using custom chains that will accept certain connections and simply let not accepted ones be managed by other chains or by the custom policy. This two coding styles would lead to the same PIM model). Each model will represent the local access-control policy enforced by a firewall so that it would be possible to be analyzed easily than using the configuration file.

Step3. PIM aggregation.

The first two steps have helped to obtain a higher-level view of the filtering rules in a firewall configuration file. However, a network system usually consists in more than one firewall so that having the network-connection model of one unique firewall only provides a partial view of the access control policy. As an example, in the network system presented as a motivating example is section 2, two firewalls are present, one managing the connections from the intranet and the other managing connection from the Internet. In that schema, a host can be accessed using same port and same protocol from any of this two zones. Thus, to really represent the access control policy of a network system in an easy to read and manage way, the individual network-connection models of each firewall would have to be combined in a single network-connection model. This model combination is performed in the third step giving as a result the desired global network-connection model (note that this combination process could include operations to eliminate information only locally relevant, as it was done for the example sin section 2 and shown in figure 2).

With this third step we have succeeded to obtain a higher-level, less complex to understand and manage representation of the information provided by the firewall configuration files. This representation isolates us from the specificities of the firewall vendor and languages and even from some aspects of the topology as only access-control information is shown.

Step4. Translation to other formalisms.

The fourth and last step consist in transforming our network-connection model into a higher-level or less network specific model representation suitable to be analysed by security policy experts without network knowledge. This step depends of the target formalism the process is targeting so that no further discussion is pertinent for the present work.

Network-connection metamodel

To represent the information contained in the firewall configuration files in a higher abstraction and less complex level we propose here a simple network-connection metamodel that keeps all the relevant information contained in the configuration files while eliminating the redundancy and readability problems that low level filter rule languages present.

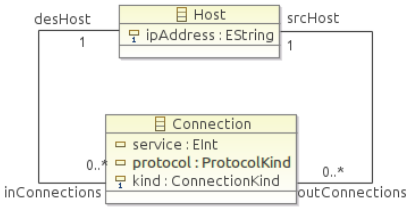

Figure 4: Network-connection metamodel

Our proposed metamodel contains only two entities, Host, that represent network host as they are represented in the configuration files, i.e., as IP addresses and IP ranges and Connection that represent connections between hosts specifying the port used to make the connection, the protocol and if the connection is allowed or denied (depending of the global policy set for the firewall, all the possible connections that do not appear in the model are either all forbidden or all allowed). This metamodel is depicted in figure 4.

4. IMPLEMENTATION

In order to demonstrate the feasibility of our approach, a prototype tool has been developed under the Eclipse[1] environment focused in extracting access-control information out of one of the most popular firewall languages, the netfilter iptables.

As our approach states, the first step implies parsing the firewall configuration files to inject their information into a PSM. To perform this step we have used Xtext [2], an eclipse tool for building domain specific languages. First, we have written the simple yet usable iptables grammar that is summarized (only the rules relevant for the example provided in section 2 are shown) in listing 4. Note that as we consider we are parsing configuration files of working firewalls, the grammar can be simplified as no language constraints (e.g, checking that a chain name in a rule target has been defined) need to be defined.

This grammar has been then used as an input for the Xtext tool that generates from it an ecore metamodel for iptables, depicted in figure 5, a parser and an editor. All these generated artifacts allow us to inject iptables configuration files into iptables models.

Listing 4: Iptables sample grammar

```
Model: rules += Rule*;

Rule: declaration=ChainDeclaration | filter=
     FilterDeclaration
;

ChainDeclaration: 'iptables' '-N' ChainName
;

ChainName: name=ID
;

FilterDeclaration: filter=FilteringSpec
;

FilteringSpec: FilterSpec | StateFilterSpec
;
```

[1]http://www.eclipse.org/
[2]http://www.eclipse.org/Xtext/

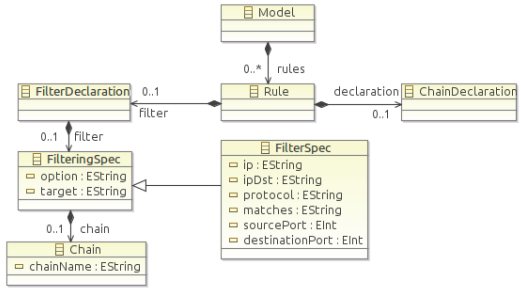

Figure 5: iptables metamodel

```
FilterSpec:
    'iptables' option=('-A' | '-D' | '-P') chain=Chain
        (('-src' | '-s') ip=IPExpr)? ('-i' interface=
        Interface)? ('-d' ipDst=IPExpr)? ('-p' protocol=
        Protocol)? ('-m' matches=Protocol)? ('--sport'
        sourcePort=INT)? ('--dport' destinationPort=INT)?
         ('-j')? target=Target
;

IPExpr:
    ipByteExpr '.' ipByteExpr '.' ipByteExpr '.'
        ipByteExpr (IpRangeExpr)?
;

ipByteExpr: INT
;

IpRangeExpr: '/' INT
;
```

For the second step the implementation is as follows. The network-connection metamodel has been implemented as an ecore model. Then, for the transformation from iptables models to models conforming to this metamodel, we have defined ATL [7] model-to-model transformations. As the semantic distance between the iptables models and the network-connection models is big, the transformation has been decomposed in two steps, to simplify its definition.

In listing 5, we show the ATL rule that transforms the exceptions in a iptables chain into deny connections in a network-connection model.

Listing 5: ATL transformation example

```
rule createDestinationConnectionFromChain{
    from
        chainInit:IPTABLES!FilterSpec,
        chainFollow:IPTABLES!FilterSpec (chainInit.target
            = chainFollow.chain.chainName and (not
            chainFollow.ipDst.oclIsUndefined()) and
            chainFollow.target = 'RETURN')
    to
        connection:NETWORK!Connection (
        desHost <- thisModule.createHost(thisModule.
            findHost(chainFollow.ipDst.debug('dst'))),
        srcHost <- thisModule.createHost(thisModule.
            findHost(chainInit.ip.debug('src'))),
        service <- chainInit.destinationPort,
        kind <- #deny
        )
}
```

The third step, the combination of the individual PIMs obtained in the previous step has been also implemented as ATL transformations. Again, decomposed in steps so that the definition is easier.

Finally, some operations needed to be defined on the network-connection global model, as the one that finds and eliminates only locally-relevant information. For that, an in-place transformation, e.g., a transformation where the changes are directly performed on the input model, has been implemented in ATL.

5. RELATED WORKS

Regarding the extraction of security policies out of firewalls and networks, there are already several works tackling the problem. However, these works do not reach a high abstraction level or not provide an usable policy representation as they are more focused in the automatic analysis of the policy. [15] proposes a technique that aims to infer the high-level security policy from the rules on firewalls using a merging algorithm to extract classes (types) of services hosts and protocols. The result however is a set of more abstract rules, still too near to the implementation level. Moreover, it takes only one firewall into account for the process. [8] proposes a method and tool to discover and test the global firewall policy. It collects and reads all the relevant configuration files, and builds an internal representation of the implied policy and network topology that can be known by the user through queries. Unfortunately, no explicit model of the recovered security policy is provided so that the extracted policy can be globally inspected. [3] proposes a bi-directional method to enforced and reverse engineer firewall configurations. Basically it promotes the use of an intermediate policy representation but does not provide a model for such representation nor specific processes to perform the enforcement and the discovery tasks. In other efforts more focused in the automatic analysis, [14] represent single firewall rules as trees in order to detect anomalies and check conflicts during rule insertion whereas [16] present a framework that represent rules from multi-firewall systems as binary decision trees to detect miscofigurations. Both approaches remain quite close to the implementation details instead of providing a less complex and higher level representation of the policy to the security expert.

The other way round, i.e., producing firewall rules from high level representations has been also studied. In [10] the authors present a firewall PIM and PSM for IPtables. Then, an ATL transformation PIM2PSM and a model-to-text transformation to configuration files are provided. Out of the modelling community, [2] presents a firewall management toolkit consisting in a global policy representation if the form of entity-relationship paradigm and a compiler to generate firewall configuration files. Similar to this approach, in [4, 11], firewall configuration rules are derived from network security policies expressed in OrBac by using xslt transformations. These approaches could be complemented by the approach we are proposing in the present work.

6. CONCLUSIONS AND FUTURE WORK

We have presented a model-driven approach to extract access-control policy models out of the configuration files of the firewalls enforcing that policy in a network. We have shown that this process isolates the policy from the low-level implementation details facilitating its understanding and manipulation. We have then implemented a prototype of the approach for the Netfilter Iptables language. As further works we envision to investigate the following challenges:

- Our network access-control metamodel shows hosts and connections. Hosts could play different roles within a

network. Automatically identifying the role of a host (a host can be a server, an interface, etc.) through analysing the firewall configuration files would facilitate the understanding of the policy by non-technical personnel. We would work in an extended version of our approach and implementation to tackle this problem.

- Firewalls are the main component used to enforce access-control policies in network systems. However, other elements can also contribute to this enforcement. Thus, we plan to extend our extraction approach to include other network components such as MPLS routers, VPN tunnels, intrusion detection systems, etc.

- Our extraction approach produces network access-control models that conform to our own defined metamodel. Nevertheless, the actual trend in access-control policies is to use contextual models like RBAC and OrBAC as they facilitate the administration of the policy. We plan to study the translations from our model to such models.

7. REFERENCES

[1] A. Abou El Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03)*, pages 120 – 131, June 2003.

[2] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Trans. Comput. Syst.*, 22(4):381–420, Nov. 2004.

[3] M. Bishop and S. Peisert. Your security policy is what?? Technical report, 2006.

[4] F. Cuppens, N. Cuppens-Boulahia, T. Sans, and A. Miège. A formal approach to specify and deploy a network security policy. In *Formal Aspects in Security and Trust'04*, pages 203–218, 2004.

[5] J. Garcia-Alfaro, N. Boulahia-Cuppens, and F. Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. *Int. J. Inf. Secur.*, 7(2):103–122, Mar. 2008.

[6] J. Garcia-Alfaro, F. Cuppens, and N. Cuppens-Boulahia. Management of exceptions on access control policies. In H. S. Venter, M. M. Eloff, L. Labuschagne, J. H. P. Eloff, and R. von Solms, editors, *SEC*, volume 232 of *IFIP*, pages 97–108. Springer, 2007.

[7] F. Jouault and I. Kurtev. Transforming models with atl. In *MoDELS Satellite Events*, pages 128–138, 2005.

[8] A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, pages 177–, Washington, DC, USA, 2000. IEEE Computer Society.

[9] S. Pozo, R. Ceballos, and R. M. Gasca. Model-based development of firewall rule sets: Diagnosing model inconsistencies. *Inf. Softw. Technol.*, 51(5):894–915, May 2009.

[10] S. Pozo, R. Gasca, A. Reina-Quintero, and A. Varela-Vaca. Confiddent: A model-driven consistent and non-redundant layer-3 firewall acl design, development and maintenance framework. *Journal of Systems and Software*, 85(2):425 – 457, 2012.

[11] S. Preda, F. Cuppens, N. Cuppens-Boulahia, J. Alfaro, L. Toutain, and Y. Elrakaiby. Semantic context aware security policy deployment. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 251–261. ACM, 2009.

[12] R. Russell. Linux 2.4 packet filtering howto. http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html, 2002.

[13] R. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for role-based access control: towards a unified standard. In *Proceedings of the fifth ACM workshop on Role-based access control*, RBAC '00, pages 47–63, New York, NY, USA, 2000. ACM.

[14] E. A. Shaer and H. Hamed. Modeling and management of firewall policies. *IEEE Trans. Network and Service Management*, pages 2 – 10, Apr. 2004.

[15] A. Tongaonkar, N. Inamdar, and R. Sekar. Inferring higher level policies from firewall rules. In *Proceedings of the 21st conference on Large Installation System Administration Conference*, LISA'07, pages 2:1–2:10, Berkeley, CA, USA, 2007. USENIX Association.

[16] L. Yuan and H. Chen. Fireman: a toolkit for firewall modeling and analysis. In *In Proceedings of IEEE Symposium on Security and Privacy*, pages 199–213, 2006.