Firewall Policy Modeling, Analysis and Simulation: a Survey

Vadim Zaliva, lord@crocodile.org

May 9, 2008

Abstract

Computer firewalls are widely used for security policy enforcement and access control. Current firewalls use various processing models and are configured using their own policy description languages.

In this paper we will try to survey research efforts in the area of formalization of firwall operational sematnics and policy description languages and applications of such formal models and languages for firewall simulation, policy optimization, detection of configuration errors and enterprise security policy comliance testing.

1 The Scope

The range of network security is very broad. It roots in low-level security enforcement mechanism on the level of IP packets (packet filtering). If we go beyond individual packets analysis, we are now dealing with technique called "stateful inspection".

The next level of network security is high-level enterprise security policies like RBAC [17], and their application to network firewalls[25].

Finally, taking into account network topology, with multiple perimeters of policy enforcement and possbily internal sectioning into security zones, we are now dealing with notion of distributed firwalls and distributed Intrusion Detection Systems (IDS).

There is ongoing research in the areas mentioned above. However in this paper, we will deal mostly with the most fundamental level - packet filtering (for broader survey of higher level securitiy-related formal languages and models see [12]).

A packet filtering engine lies at the core of most network security mechanisms. Having a formal model of this fundamental building block is essential in developement of more complex, higher-level security models.

2 Packet Filtering Defined

2.1 Processing Model

Packet filtering is a core functionality of network firewalls. The main idea is that the firewall resides on a network Node (Host or Router) and inspects all network traffic. Inspection is performed in accordance to network security policy (which we will discuss in detail later). Based on this policy, the firewall makes a decision regarding what action to perform on a given packet. The most commonly performed actions are:

Accept the packet is permitted to pass through

Deny/Drop the packet is silently dropped

Some firewalls allow additional actions, which does not necessarily affect the packet's traversal of the firewall, but are invoked for side effects. Common examples are:

Accounting the packet counter associated with this rule is incremented

Reject the packet is rejected, notyfing the server via ICMP message

The low-level implemenation of packet matching and the algorithms for doing this efficiently are commonly refered to as *packet classification problem*. It is mostly dealing with performance and resource usage constraints. There is a significant body of research on this subject (For example see: [16], [20],[19]) and although it is closely related to the subject of this article, we will try not to dwell too much on actual matching algorihtms, but rather concentrate on formalization of firewall behavior and the subsequent application of such formalization.

2.2 Policy

The firewll's behavior is controlled by the "Policy". Policy consists of "Rules" (in context of packet routing they also often reffered as "filters"). Each rule constist of *condition* and *action*.

Condition describes the criteria used to match individual packets. Action describes the activity to be performed if matches have been made.

Basic conditions consitst of tests, matching individual fields of the packet such as source address, destination address, packet type, etc. In the case of *stateful inspection* (e.g. via *ip_conntrack* module in *iptables*), connection-related variables like connection state ("established", "related", "new") could be checked. Finally, various system state variables like current time of day, CPU load, or system-wide configuration parameters could be taken into account.

The condition could be viewed as a predicate. Usually, for a packet to match a condition, all tests must be satisfied (logical conjunction).

The sequence of rules processing differs significantly between varous firewall implementations. There are two common matching strategies:

"single trigger" processing means that an action of the first matching rule will be performed.

"multi-trigger" processing means that all rules will be matched and an action from the last matching rule will be peformed.

Some firewalls like *ipfilter* support "multi-trigger" policy by default, but allow individual rules to specify *quick* option which signifies that no further processing should be done on matched packet.

Some firewall like *iptables* have even more complex processing logic, which allows for branching by organizing rules in into *chains* and providing special actions to redirect control from from one chain to another.

Hari [23] mentions another interesting strategy, where each of the filter fields are assigned priorities and the filter with the most specific matching field with the highest priority is selected. This allows, for example, in packet matching to give a highest priority to match based on the source IP rather than on destination IP.

3 Formal Models

One direction of research is the definition of special high-level languages (sometimes graphical) to describe firewall policy. In such languages, the policy representation is translated to the native policy description language of an actual firewall platform. Examples are: Firewall Builder [35], HLFL [1], FLIP[36], Firmato[7],INSPECT[2],XACML[18]. Some of these languages allow you to describe the policy of a single firewall, while others allow you to define an organization security policy which is translated to policy files for multiple firewalls.

The research in this area is fragmented. A single, generaly accepted mathematical model describing firewall policies is yet to emerge. Belowe we hightlight some of the work in this area:

Ehab S. Al-Shaer and Hazem H. Hamed [4],[5] use fixed rule structure, they call "5-tuple filter": *order,protocol,src_ip,src_port,dst_ip,dst_port,action*

In order to formally model firewall policy, these researchers start by defining the relationship between rules in the policy. Then they define the following relations between two rules: "completely disjoined", "exactly matched", "inclusively matched", "partially disjoing", "correlated".

Next Al-Shaer and Hamed prove that these relaionships are distinct and that their union represents the universal set of relaions between any two k-tuple filters in a firewall policy.

The policy is represented as a single-rooted tree, where each node represents a field of a filtering rule and each branch at this node represents a possible value of the associated tree. An example of such a tree taken from [4] is shown at Figure 1

Hari et al. [23] consider a much simpler packet filtering model, where each filter is k-tuple $(F[1], F[2], \ldots, F[k])$ and where each field $F[i]$ is a prefix bit

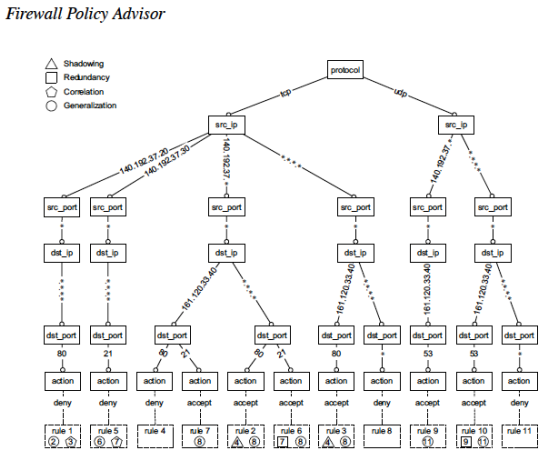

Figure 2. The policy tree for the firewall policy in Figure 1.

Figure 1: Example of policy representation as a tree

string. This model could be used not only in firewalls, but also for routing. Note that all matching is done only by matching prefix bit strings. However, as shown in [30] it is always possible to represent a subrange of $[0, 2^k]$ as at most $2k$ prefixes.

The chosen models have an interesting property, on which they base their algorithms:

> If filter fields are prefix fields, then each field of a filter is either a strict subset of, or equal to, or a strict superset of, or completely disjoint from the corresponding field in any other filter. In other words, it is not possible to have partial overlaps of fields. Partial overlaps can only occur when the fields are arbitrary ranges, not prefixes.[23]

Using that property, they propose to solve a filter conflict problem by reordering as cycle elimiation problem in directed graph.

There are some efforts related to analysis of firewall plolicies, using machine reasoning techniques. In particular, in [15] describes Expert System built using Constraint Logic Programming (CLP). Considering each rule as 6-tuple or ranges along with action taken ("permit" or "deny"), the system represents them as constraints on the 6-dimensional packet space. Each rule is a 6-dimensional hypercube.

Capretta et al. in [9] using Coq[8] proof assistant to detect conflicts in firewall policies. Their "conflict" definition is two rules for which exists a request for which they give an opposit action (only *accept* and *deny* actions are considered). Then they proceed to proove formally soundness and completeness to establish the correctness of their algorithm.

Another approach to policy modeling is using geometric interpretation. For example, Eppstein[14] suggests that each rule could be represented as a collection of d-dimensional ranges $[l_i^1, r_i^1] \times \ldots \times [l_i^d, r_i^d]$, an action A_i and priority p_i. Similarly, each packet can be viewed as a d-dimensional vector of values $[P_1, \ldots, P_d]$. A filter i applies to packet if $P_j \in [l_i^j, r_i^j]$. Epspstein proceeds to formally define *packet classification problem* and *filter conflict detection problem* using this geometrical abstraction and suggests algorithms for solving them.

The *multidimensional range searching* problem from computational geometry is related to the filter conflict detection problem. Multiple algorithms exist to solve this problem, surveyed in [26]. In particular, as mentioned in [23], Edelsbrunner [11] has proposed an algorithm which in the worst case can solve this problem in $O((log(N))^{2k-1} + R)$ where N is number or k-dimensional rectangle boxes and R is number of boxes intersecting the query box.

> Unfortunately, the filter conflict problem is somewhat different from the multi-dimensional rectangle intersection problem. While a filter contained inside another filter (a filter that is more specific than the other in all fields) is not a conflict, the corresponding rectangles are considered intersecting in the geometric framework. Thus, the number of rectangle intersections R can be much bigger than the number of filter conflicts C. Secondly, even for modest values of N and k , the worst-case time and space bound guaranteed by this data structure are hopelessly bad. For instance, when $N = 10,000$ and $k = 4$, the algorithm guarantees a worst-case search cost of $13^7 = 62748517$, meaning that it is no better than a linear search through the filters.[23]

Guttman[21] et al. describe group of network security related problems and modeing frameworks that lead to their solutions:

> We focus the modeling work on representing behavior as a function of configurations, and predicting the consequences of interactions among differently configured devices.[21]

While Guttman et al. cover both packet filtering firewalls and IPSec gateways, Uribe et al.[31] build upon their work, extending it by including specifications and requirements for Network Itrusion Detection Systems (NIDSs).

Yuan et al. in FIREMAN[34] are one of few researches who go beyond a simple linear policy model and consider what they call *Complex Chain Model*, covering more complex policy organization similar to one implemented in popular Linux firewall Netfilter. They also introduce the notion of *ACL Graph*, formed by a combination of multiple ACLs accross the trajectory of the packet. Using this graph they provide some analysis of anomalies in distributed firewall configuration.

4 Applications

4.1 Shadowing, Redundancy and Anomaly Detection

Some studies[19] show that 15% of rules in real-life policies might be redundant. More formal definition of shadowing and redudancy, from [10] are:

> **Definition 1.1** *Let R be a set of filtering rules. Then R has shadowing iff there exists at least one filtering rule, Ri in R, which never applies because all the packets that Ri may match, are previously matched by another rule, or combination of rules, with higher priority in order.*
>
> **Definition 1.2** *Let R be a set of filtering rules. Then R has redundancy iff there exists at least one filtering rule, Ri in R, such that the following conditions hold: (1) Ri is not shadowed by any other rule; (2) when removing Ri from R, the filtering result does not change.*

In [10] the authors present (with formal proofs of correctness) algorithms for shadowed and redundant rules detection and removal. However, their algorithms are using simplified firewall model with only "single trigger" processing strategy and just two possible actions: *accept* and *deny*. The authors do not go into exact semantics of rule conditions matching, treating them as an conjunctive set of opaque condition attributes.

In [4] the authors identify four firewall policy anomalies: "shadowing anomaly", "correlational anomaly", "generalization anomaly" and "redundancy anomaly" along with the algorithm to detect any of these anomalies.

In [5] they extend their anomaly-detection algorithms to configuration, consisting of multiple firewalls. They provide format definition of various *Inter-Firewall Anomalies* and propose algorithms for their detection.

Baboescu [6] suggest an optimized conflict detection algorithm, which while based on a known Bit Vector approach, is showing order of magnitude improvement compared to previous work.

Qian et al. in [29] introduce ACLA framework which includes algorithms, allowing it to:

> detect and remove redundant rules, discover and repair inconsistent rules, merge overlapping or adjacent rules, map an ACL with complex interleaving permit/deny rules to a more readable form consisting of all permits or denies, and finally compute a meta-ACL profile based on all ACLs along a network path.[29]

They present set of formal *rule relation* definitions: "intersect", "contain", "overlap", "disjoin", "adjacent", "inconsistent" and "redundant".

Gouda and Liu in [27] analyze rule redundancy problem. They introduce the notion of *upward redundant rules* and *downward redundant rules* (with formal definition). They offer algorithms for identification of redundant rules using *firewall decision tree*.

4.2 Firewall Simulation

Being able to simulate firewall behavior is important for security audit, testing and policy debugging. In order to simulate a physical firewall one needs to implement a model of security policies provided by this firewall and put it in the context of the (simulated) network environment (network topology,routing tables, addresses, etc.). Firewall simulation requires building well-defined formal model of the firewall and seeding it with actual policy files. Some work in this direction was done in context of *Fang*[28] and *Lumeta Firewall Analyzer*[33] projects.

Expert System described in [15] allows one to analyze policy by asking queries like "From which sources are packets to this destination are permitted?" or "What rules permit packets from this network?". The knowledge base could be easily extended with new rules.

ACLA [29] also have simulation capabilities which allow us to answer queries like "What are all the permitted traffic from src=X to dest=Y?" or "Will traffic flow with src=X, dest=Y, protocol=TCP, port=80 be permitted?".

FIREMAN[34] is a static analysis toolkit for firewall modeling and analysis using static analysis and symbolic model checking techniques. The policy modelling is implemented using Binary Decision Diagrams (BDDs).

4.3 Policy Optimization

Policy optimizaton is important area of research. While individual rules are quite simple, they have to match huge amounts of packets in real time and performance becomes key. Some researchers suggest that the distrbution of traffic volume produced by different rules is very uneven (20% of the flows last 5 seconds or more and carry about 60% of total traffic)[22]. So optimizing rule matching order based on rule sequencing promises very substantial perfomrance benefits. In [22] Hamid et. al. the authors prove that optimal firewall rule ordering is an *NP-complete* problem.

There are multiple algorithms and techniques suggested in the literature (e.g. Content Addressable Memory, Tuple Space Search, Fat Inverted Segment Trees, Recursive Flow Classification, etc), see [19] for a survey.

4.4 Testing

Vigna [32] describes the formal foundation of firewall "white-box" testing based on formal models. Jurgens has also done some related work [24] using CASE tools. The main challenge of this kind of testing is that in addition to firewall policy, one needs to have a model of related network topology to generate and run meaningful tests.

In [13] the authors present an automated testing toolkit to test firewall implementations which automatically generates both policy and test data.

5 Future Research Directions

While the generic packet classficiation problem is well studied, most models consider a simple ordered set of rules, usually with "single trigger" semantics. For practical applications, models must be able to deal with more complex processing models implemented in real firewall producs. This includes the "multi-trigger" processing model, as well as more complex, branching models like the "chains"-based one used in NetFilter. Perhaps some of these models could be transformed to a more simple "ordered set of rules" model, but such transformations have yet to be formally defined.

Most policy optimization and anomaly detection algorithms described in the literature only consider "accept" and "reject" actions. However some rules might have actions producing side effects and such rules (along with their triggering order) must be preserved during policy transformation/optimiziation process.

Most models make some assumtions about packet attributes. Some assume that they are bit strings which could be matched via prefixes, some assume them to be continuous numeric ranges, and some assume them to be just a predicates. However, real firewalls provide support for the attributes which can not be easily converted to any of these simple representations. For example, some firwalls allow you check for certain days of week or hours of a day in rules.

References

[1] High level firewall language. http://www.hlfl.org/.

[2] Inspect language reference. http://www.security-gurus.de/docs.php.

[3] Netspoc: a network security policy compiler. http://netspoc.berlios.de/.

[4] Al-Shaer, E., and Hamed, H. Firewall Policy Advisor for anomaly discovery and rule editing. *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on* (2003), 17–30.

[5] Al-Shaer, E., and Hamed, H. Discovery of policy anomalies in distributed firewalls. *IEEE INFOCOM 4* (2004), 2605–2616.

[6] Baboescu, F., and Varghese, G. Fast and scalable conflict detection for packet classifiers. *Computer Networks 42*, 6 (2003), 717–735.

[7] Bartal, Y., Mayer, A., Nissim, K., and Wool, A. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems (TOCS) 22*, 4 (2004), 381–420.

[8] Bertot, Y., and Castéran, P. *Interactive Theorem Proving and Program Development: Coq'Art: the Calculus of Inductive Constructions*. Springer, 2004.

[9] Capretta, V., Stepien, B., Felty, A., and Matwin, S. Formal correctness of conflict detection for firewalls. *Proceedings of the 2007 ACM workshop on Formal methods in security engineering* (2007), 22–30.

[10] Cuppens, F., Cuppens-Boulahia, N., and Garcia-Alfaro, J. Detection and Removal of Firewall Misconfiguration. *Proceedings of the 2005 IASTED International Conference on Communication, Network and Information Security 1* (2005), 154–162.

[11] Edelsbrunner, H. A new approach to rectangle intersections part I. *International Journal of Computer Mathematics 13*, 3 (1983), 209–219.

[12] El-Atawy, A. Survey on the Use of Formal Languages/Models for the Specification, Verification, and Enforcement of Network Access-lists. *School of Computer Science, Telecommunication, and Information Systems, DePaul University, Chicago, Illinois 60604*.

[13] El-Atawy, A., Samak, T., Wali, Z., Al-Shaer, E., Lin, F., Pham, C., and Li, S. An Automated Framework for Validating Firewall Policy Enforcement. *Policies for Distributed Systems and Networks, 2007. POLICY'07. Eighth IEEE International Workshop on* (2007), 151–160.

[14] Eppstein, D., and Muthukrishnan, S. Internet packet filter management and rectangle geometry. *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms* (2001), 827–835.

[15] Eronen, P., and Zitting, J. An expert system for analyzing firewall rules. *Proceedings of the 6th Nordic Workshop on Secure IT Systems* (2001), 100–107.

[16] Feldman, A., and Muthukrishnan, S. Tradeoffs for packet classification. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE 3* (2000).

[17] Ferraiolo, D., Cugini, J., and Kuhn, D. Role-Based Access Control (RBAC): Features and Motivations. *Proceedings of 11th Annual Computer Security Application Conference* (1995), 11–15.

[18] Godik, S., and Moses, T. eXtensible access control markup language (XACML), version 1.0, oasis-xamcl-1.0. pdf, 2003.

[19] Gupta, P. *Algorithms for Routing Lookups and Packet Classification*. PhD thesis, Stanford University, 2000.

[20] Gupta, P., and McKeown, N. Algorithms for packet classification. *Network, IEEE 15*, 2 (2001), 24–32.

[21] Guttman, J., and Herzog, A. Rigorous automated network security management. *International Journal of Information Security 4*, 1 (2005), 29–48.

[22] Hamed, H., and Al-Shaer, E. Dynamic rule-ordering optimization for high-speed firewall filtering. *Proceedings of the 2006 ACM Symposium on Information, computer and communications security* (2006), 332–342.

[23] Hari, A., Suri, S., and Parulkar, G. Detecting and resolving packet filter conflicts. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE 3* (2000).

[24] Jurjens, J., and Wimmel, G. Specification-based testing of firewalls. *Andrei Ershov 4th International Conference Perspectives of System Informatics(PSI01)*, 308–316.

[25] Laborde, R., Nasser, B., Grasset, F., Barrère, F., and Benzekri, A. Network Security Management: A Formal Evaluation Tool based on RBAC Policies. *IFIP NetCon2004*, 0–387.

[26] Lin, M., and Gottschalk, S. Collision detection between geometric models: A survey. *Proc. of IMA Conference on Mathematics of Surfaces 1* (1998), 602–608.

[27] Liu, A., and Gouda, M. Complete redundancy detection in firewalls. *Proc. 19th Annual IFIP Conference on Data and Applications Security* (2005).

[28] Mayer, A., Wool, A., and Ziskind, E. Fang: A firewall analysis engine.

[29] Qian, J., Hinrichs, S., and Nahrstedt, K. ACLA: A Framework for Access Control List (ACL) Analysis and Optimization. *Communications and Multimedia Security Issues of the New Century* (2001).

[30] Srinivasan, V., Varghese, G., Suri, S., and Waldvogel, M. Fast and scalable layer four switching. *ACM SIGCOMM Computer Communication Review 28*, 4 (1998), 191–202.

[31] Uribe, T., and Cheung, S. Automatic analysis of firewall and network intrusion detection system configurations. *Journal of Computer Security 15*, 6 (2007), 691–715.

[32] Vigna, G. A formal model for firewall testing. *Dipartamento di Elettronica Politecnico di Milano*.

[33] Wool, A. Architecting the Lumeta firewall analyzer. *Proceedings of the 10th conference on USENIX Security Symposium-Volume 10 table of contents* (2001), 7–7.

[34] Yuan, L., Mai, J., Su, Z., Chen, H., Chuah, C., and Mohapatra, P. FIREMAN: A Toolkit for FIREwall Modeling and ANalysis. *IEEE Symposium on Security and Privacy* (2006), 199–213.

[35] Zaliva, V. Platform-independent firewall policy representation, 2007.

[36] Zhang, B., Al-Shaer, E., Jagadeesan, R., Riely, J., and Pitcher, C. Specifications of a high-level conflict-free firewall policy language for multi-domain networks. *Proceedings of the 12th ACM symposium on Access control models and technologies* (2007), 185–194.