A Unified Firewall Model for Web Security

Grzegorz J. Nalepa[1]

Institute of Automatics, AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland, gjn@agh.edu.pl

Summary. The paper presents a new formalization for firewall systems, called the *Unified Firewall Model* (UFM). It offers an abstraction over firewall implementations, and uses formal concepts of Rule-Based Systems to describe firewall syntax and semantics. It is backed by the XTT/ARD design methods. It allows for improving system quality, by introducing a formal verification during the design stage.

1 Introduction

Security issues play an important role in the development of real life web systems. These issues include problems of: access control, privacy, system monitoring, and data protection. The focus of this paper is on access control, provided by the firewall systems. The design of such systems remains a challenge, due to complex security requirements, and number of different incompatible implementations. There are no standard design approaches, or methods. Quality issues are even more complex than the design.[1]

This paper continues the line of research started in [3]. In that paper an idea of using Rule-Based Systems (RBS) formalism, to the design and implementation of firewall systems was put forward. Strong logical foundations [2] of RBS allow for introducing a *formal* design and analysis into the firewall design process. These ideas were presented in [5], with the application of the XTT knowledge representation method. Since then, the design process has been extended by the conceptual design method ARD [4], which allows for formalization of RBS requirements. Using this approach, this paper presents a new formalization for firewall systems. It is called the *Unified Firewall Model*, since it offers an abstraction over common firewall implementations. It uses formal RBS concepts in order to describe the syntax and semantics of the most common firewall systems. It is backed by the XTT and ARD.

The paper is organized as follows: in Sect. 2 the architecture of web firewalls is discussed; next, in Sect. 3, the formal XTT design process is briefly

[1] The paper is supported by the *HEKATE* Research Project.

discussed; then in Sect. 4 the UFM is introduced. A practical example is presented in Sect. 5. Directions for future work are presented in the Sect. 6.

2 Firewalls for Web Security

The practical definition of *the firewall system* has been changing, along with the changes in both technology and security requirements. The first firewalls were just simple network packet filters, working on the IP protocol level. Later on, the statefull inspection, involving the analysis of the TCP sessions, and network translation technology has been introduced. Recently, the technology has been extended by the application-level firewalls at the HTTP level.

The general idea behind this paper is to consider a hybrid network and web application level firewall model. In this model a statefull network firewall serves as a gateway to a demilitarized zone (DMZ), where the main web server is located. The web server itself is integrated an the application-level firewall, in this approach an open implementation, called *ModSecurity* (`www.modsecurity.org`), available for the well known opensource *Apache2* (`httpd.apache.org`) web server is considered. It provides protection from a range of attacks and intrusions against web applications (including server-side solutions, such as PHP). It allows for HTTP traffic monitoring and analysis.

Ultimately, the application of the UFM/XTT approach presented in this paper should develop into an integrated design methodology, combining both network-level and application-level firewall. However, in this short paper only the network-level statefull firewall design using the UFM/XTT is discussed.

3 Application of the XTT-based process

The UFM has been developed, with XTT, the design methodology for RBS, in mind. In this case, the design process of a RBS (the firewall system) consists of several stages [4, 2]: 1) attribute specification, with explicit domains, 2) the conceptual design with *Attribute-Relationship Diagrams* (ARD) that model functional dependencies between system attributes [4], 3) the logical system design with *EXtended Tabular Trees* (XTT), that includes full firewall rule specification, 4) on-line formal analysis, using a dynamically generated Prolog-based description of the XTT logical structure [6], 5) the physical design, in this case providing the rule translation to particular firewall target languages.

4 The Unified Firewall Model

In the solution proposed herein, the *Unified Firewall Model*[2] (UFM) [1] is introduced as a middle layer in firewall design process, enabling logical analysis

[2] The first description of the UFM has been given by Michał Budzowski, in [1].

of the created firewall. It is a formal, implementation-free firewall model, build on top of the XTT methodology, providing a unified attribute specification for representing network firewalls. Generation of target language code for specific firewall implementation is achieved by defining translation rules transforming abstract firewall model into a specific implementation.

A full list of conditional firewall attributes, corresponding to the information found in packets header, is specified. In UFM some attributes that are not related to packet header, but rather to firewall design will also be distinguished. Following attributes are considered: Source/Destination IP address, Input/Output interface, Source/Destination group, Protocol, Destination port, Service, ICMP type and error code. The following firewall decision attributes are considered: Accept, Reject, Snat, Dnat. The `nat` decisions refer to the network translation technique present in all advanced firewalls.

Domains for UFM attributes are presented in the Table 1. The specification is given in the Table 2, where each attribute is specified with: *Name*, *Symbol*, *Subset* (the position in inference process, specifying whether attribute is *input*, *output* or its value is defined during inference process – *middle*), *Atomicity* (specifying whether attribute takes only *atomic* values from specified domain or also *sets* or *ranges* of these values). Attributes *aSGR* and *aDGR* define groups, that organize network traffic and facilitate specifying decisions.

Domain	**Type**	**Constrains**
Ipaddr	integer	$< 0, 2^{32} >$
Port	integer	$< 0, 2^{16} >$
Protocol	enum, symbolic	{tcp, udp, icmp }
Interface	enum, symbolic	int$i$, $i \in Integer$
Icmptype	enum, symbolic	{echoreq, ... maskrep }
Icmperrcode	enum, symbolic	{net-unr, ... host-unk}
Tcpflags	enum, symbolic	flag/mask, flag, mask $\subset$ {fin, syn, rst, psh, ack}
Service	enum, symbolic	{www, ssh, dns, smtp, icmp, other$i$}, $i \in Integer$
Group	enum, symbolic	$\{fw$i, neti, neti$_host$j, neti$_host$k$\}$, $i, j, k \in Integer$
Action	enum, symbolic	{dnat, accept, reject, log, snat}

Table 1. UFM attribute domains.

In order to construct practical firewall implementation, it is necessary to provide a formal translation from the unified model to particular implementations. Two open firewall implementations have been considered: Linux NetFilter [8] and OpenBSD PacketFilter (PF) [7]. Full translation contained in [1] is long and detailed, and is out of scope of this paper.

5 Formal Hierarchical Design Example

The conceptual design of the firewall is conducted using ARD. The logical structure, including rules is based on XTT. The ARD/XTT design, and a formal Prolog-based analysis [6], is supported by the prototype Mirella tool.

Name	Symbol	Subset	Domain	Atomic
Source/Destination IP	aSIP/aDIP	input	Ipaddr	*set*
Protocol	aPROTO	input	Protocol	*set*
Destination port	aPORT	input	Port	*atomic*
Input/Output interface	aIINT/aOINT	input	Interface	*atomic*
ICMP type	aICMPT	input	Icmptype	*atomic*
ICMP error code	aICMPC	input	Icmperrcode	*atomic*
TCP flags	aTCPF	input	Tcpflags	*atomic*
Service	aSERV	middle	Service	*set*
Source/Destination group	aSGR	middle	Group	*set*
Action	aACT	output	Action	*atomic*

Table 2. UFM attribute specification.

In this case a non-trivial network firewall is considered. It is discussed in detail in [1]. It consists of: the firewall host and connections to four networks: the Internet, DMZ1, DMZ2, and LAN. The goal of the system is to monitor the traffic between networks and accept, reject or forward packets between them basing on the characteristics of the packets traversing the networks.

Decision taken by the firewall system can be defined if *source*, *destination* and *protocol* of the packet are known. This dependency can be denoted with ARD diagram of level 0, where `Source`, `Destination` and `Protocol` are *conceptual* variables. Later in design process these conceptual variables are replaced with one or more *physical* attributes present in firewall rules. Decision taken by the firewall is defined with `aACT`, attribute also alerting source and destination is possible, so conceptual variable. Decision will be replaced with `aACT`, `aSIP`, `aDIP`, and `aPORT` UFM attributes. A three-level ARD diagram (Fig. 1) models the functional dependencies between firewall attributes.

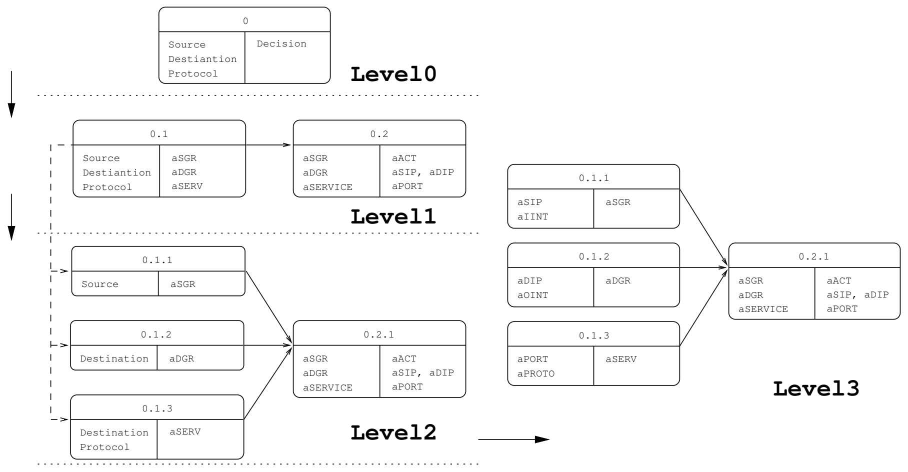

Fig. 1. The ARD diagram for the firewall.

Following conceptual design and the ARD diagram, the XTT tables are designed. There are four XTT tables: `source` table, a *root table* checking source group, `destination` table checking destination group, `service` table inferring the type of service, `decision` table specifying firewall decision.

Each table is created based on final the ARD diagram. Left part corresponds to Precondition part in XTT table, right part corresponds to Assert/Retract or Decision part. Firewall policy is implemented by filling XTT tables with rules. Ctrl part of XTT table specifies control algorithm. In the `source` table packet is classified to one of the source groups basing on its source IP address and interface it is coming from. Similar operations are conducted in the `destination` table, but the packet is classified to one of the destination groups. The same control algorithm is implemented, but the next table is the `service` table. The `decision` table is the final table where the actual decision is inferred from the values of *middle* attributes: *aSGR*, *aDGR*, *aSERV*. For brevity only the last, `decision` table is shown in Tab. 3

4. decision												
Info	Prec			Retract			Assert			Dec	Ctrl	
I	aSGR	aDGR	aSERV	aSIP	aDIP	aPort	aSIP	aDIP	aPort	aACT	N	E
1	inet	fw_inet	www		*f_inet*	80		*d_3w*	8080	dnat	2.1	4.3
2	inet	dmz1_www	www							accept	1.1	4.3
3	dmz1_dns	inet	dns							accept	4.4	4.5
4	dmz1_dns	inet	dns	*d_dns*			*f_inet*			snat	1.1	4.5
5	sDNS	dmz1_dns	dns							accept	1.1	4.6
6	employee	dmz1_mail	smtp							accept	1.1	4.7
7	dmz2_proxy	inet	www							accept	4.8	4.9
8	dmz2_proxy	inet	www	*d_prox*			*f_inet*			snat	1.1	4.9
9	admin	sMACHINE	ssh							accept	1.1	4.10
10	employee	inet	www	–		80		*d_prox*	8080	dnat	2.1	4.12
11	employee	dmz2_proxy	forward							accept	1.1	4.12
12	–	–	icmp							accept	1.1	4.13
13	–	–	–							reject	1.1	1.1

Table 3. XTT decision table.

Physical design of the system consists of the translation high-level rules in XTT tables to target firewall language of Linux NetFilter (iptables) or OpenBSD PF. Let us show how rules translation is performed for the rule 4.1 from the table `decision`. The firewall rule for the NetFilter is:

```
iptables -t nat -A PREROUTING -s 0/0 -i eth0 -d 83.29.224.129
   -p tcp --dport 80 -j DNAT --to-destination 192.168.2.2:8080
```

The PF form allows for observing differences in the target firewall language:

```
rdr on eth0 proto tcp from any to 83.29.224.129 port 80->192.168.2.2 8080
```

This difference is even more important in case of rule 4.4. While the NetFilter translation requires the use of two rules:

```
iptables -t nat -A POSTROUTING -s 192.168.2.4 -d 0/0 -o eth0
   -p tcp --dport 53 -j SNAT --to-source 89.29.224.129
iptables -t nat -A POSTROUTING -s 192.168.2.4 -d 0/0 -o eth0
   -p udp --dport 53 -j SNAT --to-source 89.29.224.129
```

the PF form is more compact:

```
nat on eth0 proto {tcp,udp} from 192.168.2.4 to any port 53->83.29.224.129
```

The full translation is discussed in [1]. The expressiveness of the UFM is high, so it is closer to the more expressive target language. However, all of the UFM syntactic structures can be translated to any firewall language, provided that the the implementation has the features represented by the UFM.

The XTT approach offers a possibility of automatic, on-line formal analysis of the firewall structure *during* the logical design. The analysis is accomplished by an automatic transformation of the XTT model into a corresponding code in Prolog. Some important features of the firewall system can be analyzed, including completeness of the system, or its determinism. Unfortunately, these issues are out of scope of this short paper. They have been discussed in [6].

6 Future Work

The original contribution of this paper is the new formalization of the firewall model, the *the Unified Firewall Model*. The research is considered a work in progress. Future work includes: a new version of Mirella, using the *Eclipse Modelling Framework*, and UFM application to *ModSecurity* and IDS. The UFM is being developed within the *HEKATE* project. The approach allows for improving system quality, by introducing the formal verification during the design. It offers an abstract layer over common firewall implementations.

References

1. Michał Budzowski. Analysis of rule-based mechanisms in computer security systems. formulation of generalized model for firewall systems. Master's thesis, AGH-UST, 2006. Supervisor: G. J. Nalepa, Ph. D.
2. Antoni Ligęza. *Logical Foundations for Rule-Based Systems*. Springer-Verlag, Berlin, Heidelberg, 2006.
3. Grzegorz J. Nalepa and Antoni Ligęza. Designing reliable web security systems using rule-based systems approach. In Ernestina Menasalvas, Javier Segovia, and Piotr S.Szczepaniak, editors, *Advances in Web Intelligence. AWIC 2003*, volume LNAI 2663, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
4. Grzegorz J. Nalepa and Antoni Ligęza. Conceptual modelling and automated implementation of rule-based systems. In Tomasz Szmuc Krzysztof Zieliński, editor, *Software engineering : evolution and emerging technologies*, volume 130 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2005.
5. Grzegorz J. Nalepa and Antoni Ligęza. Security systems design and analysis using an integrated rule-based systems approach. In Piotr Szczepaniak, Janusz Kacprzyk, and Adam Niewiadomski, editors, *Advances in Web Intelligence: AWIC 2005*, volume LNAI 3528. Springer-Verlag, 2005.
6. Grzegorz J. Nalepa and Antoni Ligęza. Prolog-based analysis of tabular rule-based systems with the xtt approach. In Geoffrey C. J. Sutcliffe and Randy G. Goebel, editors, *FLAIRS 2006*, FLAIRS. - Menlo Park, 2006. Florida Artificial Intelligence Research Society, AAAI Press.
7. OpenBSD Project. *PF: The OpenBSD Packet Filter*, 2006.
8. Rusty Russell. *Linux 2.4 Packet Filtering HOWTO*. NetFilter Project, 2002.