Handling Stateful Firewall Anomalies

Frédéric Cuppens[1], Nora Cuppens-Boulahia[1,2],
Joaquin Garcia-Alfaro[1], Tarik Moataz[1], and Xavier Rimasson[1]

[1] Institut Télécom, Télécom Bretagne,
CS 17607, 35576 Cesson-Sévigné, France
forename.surname@telecom-bretagne.eu
[2] Swid Web Performance Service
Rennes, France

Abstract. A security policy consists of a set of rules designed to protect an information system. To ensure this protection, the rules must be deployed on security components in a consistent and non-redundant manner. Unfortunately, an empirical approach is often adopted by network administrators, to the detriment of theoretical validation. While the literature on the analysis of configurations of first generation (stateless) firewalls is now rich, this is not the case for second and third generation firewalls, also known as stateful firewalls. In this paper, we address this limitation, and provide solutions to analyze and handle stateful firewall anomalies and misconfiguration.

1 Introduction

Firewalls aim at optimizing the degree of security deployed over an information system. Their configuration is, however, very complex and error-prone. It is based on the distribution of several packages of security rules that define properties such as acceptance and rejection of traffic. The assembly of all these properties must be consistent, addressing always the same decisions under equivalent conditions, and avoiding conflicts or redundancies. Otherwise, the existence of anomalies and misconfiguration will lead to weak security architectures, potentially easy to be evaded by unauthorized parties. Approaches based on formal refinement techniques, e.g., using abstract machines grounded on the use of set theory and first order logic, ensures, by construction, cohesion, completeness and optimal deployment [1]. Unfortunately, these approaches have not always a wide follow. Network policies are often empirically deployed over firewalls based on security administrator expertise and flair. It is then relevant to analyze these deployed configurations in order to detect and correct errors, known in the literature as configuration anomaly discovery. Several research works exist to directly manage the discovery and correction of *stateless* firewall configuration anomalies [2–5]. By stateless firewall configurations we refer to the security policies of first generation firewalls, mostly packet filtering devices working only on the lower layers of the OSI reference model. However, little work has been done to address the case of second and third generation firewalls peeking into the transport and upper layers. In this paper, we are particularly interested in addressing such a problem.

The main goal of a firewall is to control network traffic flowing across different areas of a given local network. It must provide either hardware or software means to block unwanted traffic, or to re-route packets towards other components for further analysis. First generation firewalls only allow a *stateless* filtering of network traffic. The filtering actions, such as accepting or rejecting packet flows, are taken according to a set of static configuration rules that only pay attention to information contained in the packet itself, such as packet's addresses (source

and destination), ports and protocol. Their main advantage is the speed of filtering operations. However, since they do not keep track of state connection data, they fail at handling some vulnerabilities that benefit from the position of a packet within existing streams of traffic. Stateful firewalls solve this problem and improve packet filtering by keeping track of connection status. Indeed, they can block those packages that are not meeting the state machine of a protocol over the transport layer. As with stateless packet filtering, stateful filtering intercepts the packets at the network layer and verifies if they match previously defined security rules. Moreover, stateful firewalls keep track of each connection in an internal state table. Although the entries in this table varies according to the manufacturer of every product, they typically include source and destination IP addresses, port numbers and information about the connection status.

Most methods that have been proposed to detect anomalies in the configuration of firewalls, such as [2–5], are limited to the stateless case. The detection of anomalies in stateful firewalls is still an unexplored research problem. Existing literature is still very limited. Some approaches aim at describing stateful firewall models [7], while others simply provide straightforward adaptations of management processes previously designed for stateless firewalls [8]. In this paper, we propose to extend the algorithms defined in [2] to include the management of stateful firewall anomalies as well. The principle of our approach is based on the specification of general automata. Such automata describe the different states that traffic packages can take throughout the filtering process. We then extend the taxonomy of anomalies defined in [2] in order to uncover new configuration anomalies for the case of stateful firewalls. Some anomalies occur on rule sets which only contain stateful rules, denoted hereinafter as *intra-state rule anomalies*. Other anomalies may affect those rule sets holding both stateful and stateless rules, denoted in our work as *inter-state rule anomalies*. We define algorithmic solutions to automatically detect and correct these new types of anomalies. The algorithms that we present also provide an extension of MIRAGE [9], a firewall audit tool implemented in the Java language, that allows the automatic detection and correction of stateless firewall configuration anomalies. The extension aims at covering the management of stateful firewalls as well.

The remainder of the paper is organized as follows. Section 2 presents in more detail our motivation and problem domain. Section 3 surveys related work. Section 4 presents a classification of stateful firewall intra-state rule anomalies, and defines an algorithmic solution to handle them. Section 5 extends the approach to the case of inter-state rule anomalies. Section 6 concludes the paper.

2 Motivation

Nowadays, packet filtering requires more than a passive solution to stop malicious traffic. Filtering is evolving into a dynamic process that depends on the protocol states. Consequently, new policies tend to use more and more the stateful features of next generation firewalls. Stateless inspection consists only on a static verification of the five first flag attributes of the IP (internet protocol) layer with an existing ACL (Access Control List) table. Stateful inspection goes further, and provides the firewall with the means to analyze packets not only by using the matching attribute correspondence, but also by linking each packet to the related connection. Finally, stateful firewalls provide better fine-grained filtering capabilities, and protect against more complex attacks, such as denial of service (DOS) and IP Spoofing. The main disadvantages of stateful filtering are a more complex implementation and greater use of memory to keep the trace of the previous sessions.

3

In our work, we consider stateful filtering for connection-oriented protocols, such as TCP, DCCP, ATM, Frame Relay, TIPC, SCTP, and IPX/SPX. All these protocols are based on a common principle: sending a tagged stream belonging to a particular session. In the case of TCP and SCTP, this is respectively called a session connection and an association. A stateful inspection can be performed on any of these protocols by defining a protocol automaton. In the literature, there is no formal model that describes a general behavior of firewalls in states with all specifications required and coverage of all protocols. Gouda and Liu [7] proposed in their work a modeling of a state inspection. This inspection integrates a combination of the *state* phase detection then the *free state* one. They proposed the addition of the attribute *state* depending on the protocol used for storing packets belonging to the connection initialization, and one *tag* to check the membership of new packets for this connection. In this paper the attribute state is for a state protocol itself, which does not necessarily imply the initialization of a connection-oriented session (e.g., the ICMP protocol can be modeled by the states resulting from the exchange of messages *echo-request* and *echo-reply*, even if the protocol is not connection-oriented). This approach corresponds to the model proposed for stateful firewalls, except that we can give other models satisfying the same functionality as the simultaneous approach explained above, based on stateful and stateless rules at the same time. Specifically, instead of proposing a filtering series through *state*, then the attributes presented in the stateless model, we consider a model of the filter including rules for states and stateless in parallel. Regardless of the model, the purpose of firewalls is to filter states packets according to the membership to a given session. Thus, according to the definition of *session*, modeling and representation of rules in the tables, the specification and identification of anomalies is different. In this paper, we focus, initially, on detecting anomalies on those configurations that contain only stateful rules. Specifically, we define and analyze new types of anomalies depending on the protocol transitions. The protocol chosen to illustrate the approach is TCP as a connection-oriented protocol reference. Then, we extend the approach in order to address the case in which stateful and stateless rules coexist in a given configuration rule set.

3 Related Work

Traditional research work on the design of firewalls, essentially stateless firewalls, mainly address the construction of high level languages for the specification of firewall configurations. This includes functional languages [10], rule-based languages [11] and higher abstract models that allow capturing some further aspects such as network topologies [12]. Such languages allow security administrators to free themselves from the technical complexity and specificity of proprietary firewalls languages. Some of them allow, moreover, the automatic derivation of concrete access control rules to configure specific firewalls through a translation process. At the same time, research and development work in this context may allow the verification of consistency (i.e., absence of conflicts), completeness (all the expected requirements are covered), and compactness (none of the rules are redundant or unnecessary) [1]. Refinement approaches may also take into account the functionality offered by every firewall manufacturer (stateless, stateful, management of virtual private networking, etc.) [13] to ensure the effective distribution of tasks between a decision module and the eventual filtering (enforcement) components.

For the already deployed firewall configurations, the aforementioned approaches do not solve redundancy or configuration conflicts that might have been introduced due to periodic, often manual, updates. Several studies have been conducted toward audit mechanisms that analyze already deployed configurations, with the goal of signaling inconsistencies and fixing the

discovered anomalies. We can classify them into three categories: (I) those that are oriented towards directly querying the firewall itself [14–16], (II) those targeting conflict management [17, 18] and (III) those focusing on the detection of anomalies [19, 4, 2, 20, 21]. In category I, the analysis problem is relayed towards a process of information retrieval by directly querying the firewall. This requires having highly structured configurations and specific query languages for processing them, as well as for generating complete and effective data queries. The results are, moreover, prone to both false negatives and false positives, since no track from previous filtering matches are taken into account during the audit process. Category II is concerned with packet classification algorithms, mostly for packet filtering routers, and that rely on optimized data structures to speed up the matching process between incoming flows of packets and filtering rules. Then, the goal is to verify that there are no conflicting situations in which several rules with different actions (e.g., accept or reject the traffic) apply to the same traffic. Examples in this category include the use of techniques such as *grid-of-tries* classification [22] and *bit vector aggregation* [17]. Class III improves the detection offered by solutions in class II, by: (1) characterizing in more detail the set of anomalies, e.g., redundancy is also addressed; (2) transforming the rule sets in such a way that the ordering of rules is no longer relevant; (3) considering combinations of rules instead of simply comparing rules two by two as proposed by Al-Shaer *et al.* [19], which enables the detection of a combination of rules that conflict with another rule [2]; and (4) extending the process to distributed setups with multi-firewall scenarios, in order to detect situations in which different firewalls within interconnected paths may perform different actions to the same network traffic.

None of the above surveyed techniques consider the case of stateful firewalls. So far, little work in the stateful case has been conducted. Buttyán *et al.* proposed in [8] an early approach that heads towards this research line. Nevertheless, their solution is limited to the adaptation of existing anomaly detection techniques for stateless firewalls to those that are stateful. Therefore, their work does not take into account anomalies that may impact, for instance, the tracking of connections or the management of the internal firewall state memory table. In a different vein, Fitzgerald *et al.* propose in [23] an approach based on semantic web technologies to model both stateless and stateful firewalls. Although the generality of their proposed representation is interesting enough, the work fails at characterizing the precise types of errors that would be necessary to handle by the detection process in the stateful case. The approach only represents those good practices that must be followed when configuring a given firewall.

4 Handling Anomalies on Stateful Firewall Intra-state Rules

Like in the case of stateless firewalls, stateful firewall configurations may be affected by the following basic anomalies (cf. algorithmic solutions in [2] and citations thereof, for the discovery and correction of these two anomalies):

- *Shadowing*: Let R be a set of filtering rules. Then, rule R_i is shadowed iff it never applies because all the packets that R_i may match, are previously matched by another rule, or combination of rules, with higher priority in order;
- *Redundancy*: Let R be a set of filtering rules. Then, rule R_i is redundant iff (1) R_i is not shadowed by any other rule (or combination of rules); and (2) when removing R_i from R, the filtering result does not change.

In the stateful case, we can also identify a third type of anomalies, hereinafter denoted as *intra-state protocol anomalies*, in the sense of misconfiguration that may put in risk the inner

logic of transport layer protocol states. For instance, in protocols like TCP, we may distinguish the following operations for the establishment of a connection between a server and a client:

- the client sends a SYN packet to a server (i.e., a packet with the SYN flag set);
- the server replies with a SYN+ACK;
- the client sends an ACK back to the server.

Then, it comes a phase of data transfer. Finally, the connection ends with a termination phase. When the client leads the termination phase, the following handshake takes place (still, assuming the case of TCP):

- the client sends a FIN packet to a server (i.e., a packet with the FIN flag set);
- the server replies with an ACK, then with a FIN;
- the client sends an ACK back to the server.

Given such a rationale, the following two anomaly scenarios arise:

1. the client succeeds to start the three-way handshake connection establishment with a server, while the firewall is configured in a way that either the second or third steps of the establishment operations are rejected;
2. the client starts the connection termination, but the firewall rejects, at least, one of the remainder termination operations;

In the sequel, we present an algorithmic solution to handle and correct this third type of anomalies that affects the establishment and termination of transport layer connections.

4.1 Discovery and Elimination of Intra-state Protocol Anomalies

Definitions: The communication between two nodes with a transport layer protocol consists of (1) establishment phase and (2) termination phase. We model such a protocol as two deterministic finite-state automata that correspond, respectively, to the establishment (A_1) and termination phase (A_2). We characterize these two automata by giving some examples for TCP:

- Σ is the alphabet of the automaton, which contains the series of state *flags* of a protocol. For instance, for TCP, $\Sigma = \{SYN, SYN + ACK, ACK, FIN, FIN + ACK\}$;
- Q contains the series of states of the protocols, such as the CLOSED, LISTEN, and SYN SENT states in the case of TCP;
- δ defines the transition function, such that $\delta : Q \times \Sigma \rightarrow Q$;
- q_0 defines the initial state. For TCP, this includes LISTEN for A_1 and ESTABLISHED for A_2.
- q_f defines the final state. For TCP, A_1 equals ESTABLISHED; and A_2 equals CLOSED.

Figure 1 depicts the state automaton of TCP. It includes both A_1 and A_2, which share the ESTABLISHED state.

The following functions are used for the construction of our algorithms:

- *Adjacent(*A, $State_i$, $State_j$*)*: boolean function that holds true iff there exists within automaton A a transition from $State_i$ to $State_j$. Equivalently, the expression is true iff there exists $a \in \Sigma$ such that $\delta(State_i, a) = State_j$

6

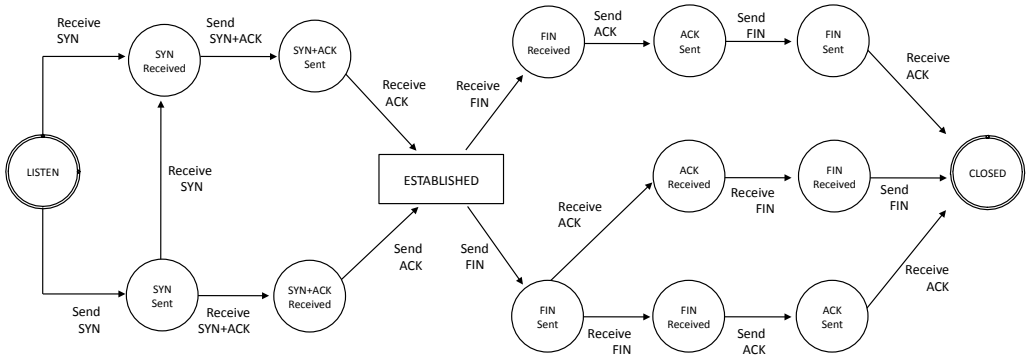

Fig. 1. TCP state automaton.

– *Next(*$A, State, Symbol$*)*: if $\delta(State, Symbol) = State'$ exists, the function returns $State'$. Otherwise, it returns the empty set.

Finally, we consider a rule set R of size n containing R_i rules ($i \leq n$). Each rule R_i typically specifies an $Action$ (e.g., ACCEPT or DENY) that applies to a set of condition attributes, such as $SourceAddr$, $DestAddr$, $SrcPort$, $DestPort$, $Protocol$, $Flag$, $State$.

Algorithms: We assume a rule subset R whose protocol ($R_i[protocol]$ attribute) stands for a given protocol. Our algorithm uses the establishment (A_1) and termination (A_2) automata of the protocol.

If an ordered pair of rules stands for two adjacent states of the automaton, we check if the related flags can be used in order to switch from the first state to the second. If so, and if the actions of the two rules are different, then we raise an anomaly. First, we apply the algorithm for anomaly corrections on the rules which are related to the establishment of transport layer connection; secondly we do it for the termination. In order to correct the anomalies of establishment connection, we change the action of the rules to DENY. Therefore we avoid connection failures. Regarding the anomalies of termination connection, we set the action of the rules to ACCEPT in order to avoid termination failure. Algorithm 1 sums up the corrections; it uses Algorithm 2 which analyzes the rules, detects the anomalies and then carries out the corrections.

Figure 2 gives an example of a correction on a subset of stateful rules by using Algorithm 1. In the initial configuration of the example, a first rule allows packets with the SYN flag and the LISTEN state. In accordance with the automaton of TCP on 1, the rule actually allows the first

Algorithm 1: `HandleAnomalies`(R, A_1, A_2)

```
/*Handle, first, the connection
  establishment automaton */
R ← HandleRules(R, A1, DENY)
/*Then, handle the connection
  termination automaton */
R ← HandleRules(R, A2, ACCEPT)
return R
```

7

Algorithm 2: `HandleRules`$(R, A, Action)$

```
n ← size(R);
for i ← 1 to n − 1 do
    for j ← i + 1 to n do
        if R_i[SourceAddr] = R_j[DestAddr]
           ∧ R_i[SrcPort] = R_j[DestPort]
           ∧ R_i[Action] ≠ R_j[Action] then
            if Adjacent(A, R_i[State], R_j[State]) then
                if (R_j[State] ∈ Next(A, R_i[State],
                R_i[Flag]) ∧ Next(A, R_j[State], R_j[Flag])
                ≠ ∅) ∨ (R_i[State] ∈ Next(A, R_j[State],
                R_j[Flag]) ∧ Next(A, R_i[State], R_i[Flag])
                ≠ ∅) then
                    R_i[Action] ← Action
                    R_j[Action] ← Action
                end
            end
        end
    end
end
return R
```

step of a TCP connection establishment (i.e. the shift from LISTEN state to the SYN Received state). However the second rule forbids the acknowledgment. Then the connection cannot be established. In that case, the algorithm corrects the first rule in order to deny the connection.

SourceAddr	*DestAddr*	*SrcPort*	*DestPort*	*Protocol*	*Flag*	*State*	*Action*
41.1.1.1	193.1.1.2	8080	2011	TCP	SYN	LISTEN	~~ACCEPT~~ DENY
193.1.1.2	41.1.1.1	2011	8080	TCP	SYN+ACK	SYN RCVD	DENY

Fig. 2. Example of an intra-state protocol anomaly in the rule set of a stateful firewall.

5 Handling Inter-state Rule Anomalies

We have previously addressed the case of intra-state rule anomalies, in which a set of stateful rules, at the transport layer, contains anomalies that may put in risk the inner logic of transport layer protocol states. The use of both stateful and stateless rules may be also found in a firewall configuration. For instance, a network administrator may add a rule in order to handle TCP connections which are used to transferring data in a FTP session. For instance, for a Netfilter firewall, we can manage this situation by adding a rule with the RELATED state parameter. This rule will be inserted in the other stateless rules which have been previously defined by the administrator. We then search for anomalies between stateful and stateless rules based on the specification of a transport layer protocol. Some application layer protocols use several TCP connections (or other transport layer protocol) during a session between two nodes. This applies for FTP, IRC or VoIP protocols which use related connections if needed. Let us further analyze the case of FTP. A typical FTP session consists in two steps:

8

1. The client begins the session with the FTP server on port 21; a TCP connection – the control connection – is established;
2. When the client wants to transfer data (file transfer, directory listing, etc.), two cases may occur:
   - After a control connection negotiation, the server initiates a new TCP connection for the transfer, from the port 20 to a client's given port. This is the active mode.
   - Or the transfer connection is initiated by the client to a FTP server's given port. This is the passive mode.

The FTP server's firewall configuration may contain:

- A stateless rule for allowing TCP packets with the destination port 21;
- A stateful rule for allowing packets whose associated TCP connection is marked with a related connection in a FTP session. The destination port will be either 20 (active mode) or greater than 1024 (passive mode).

In this example, one issue consists in correctly handling the related TCP connections between two nodes which use an application layer protocol. We note the firewall shall understand the given application layer protocol which is concerned by the rules in order to identify related connection packets. Netfilter especially has a module which allows FTP sessions tracking. Therefore a packet which belongs to a TCP transfer connection (according to the FTP terminology) has the RELATED status. This tracking of the application layer context allows the administrator to define stateful rules.

To automatically identify such anomalies for a given protocol, we assume knowing a full specification of possible scenarios of TCP connection used between two nodes during a session for the protocol. This specification explains how to initiate the connection and how it deals with the related connections during the session (order, number, ports, etc.).

The first step consists in searching the stateless rules which stand for the establishment of the protocol connection. In the case of FTP, we search a rule which matches the TCP packets with the destination port 21. If such rules are found, we consider the three following cases:

1. Stateful rules exist in the configuration to handle the possible related connections that may be used by the application layer protocol;
2. Stateless rules exist to handle these connexions;
3. No rule is defined to handle the related connexions.

The case 2 is too general because it does not take into account the inner logic of the protocol. An attacker may be able to initiate a TCP connection on a port which will be used only for a related connection of an application session. For example, a FTP connection on the server's port p will be allowed only if the server has previously initiated a FTP transfer on passive mode with a client on the port p. In the case 3, the application session may fail because the firewall will probably deny the related connections. The case 1 solves the encountered problem with the other ones and complies with the protocol specification. In a Netfilter firewall, such rules may have the RELATED state.

Definitions: Our algorithm aims at assisting the system administrator to detect and fix cases 2 and 3 of the aforementioned anomaly. We first provide the following definitions that will be used in the algorithm definition:

- L: set of stateless rules, such that every rule L_i (where i is a natural integer) is characterized by the following conditions $L_i[SourceAddr]$, $L_i[DestAddr]$, $L_i[SrcPort$, $L_i[DestPort]$ and $L_i[Protocol]$ (such as TCP, UDP, or any other transport layer protocol).
- F: set of stateful rules, such that every rule F_i is characterized by the same conditions as the rules in L, plus the condition attribute $F_i[State]$. It is important to consider $F_i[Protocol]$ since the transport protocol of a given connection could be different from the protocol of the main connection. For instance, in VoIP scenarios, data transfer might be carried upon UDP, while the main connection is relayed via TCP.
- A: deterministic finite automaton that describes an application layer protocol. We rely on the use of the alphabet Σ of A, containing the set of operations that can be exchanged between hosts, e.g., remainder set of operations once the main connection of two FTP entities has been established. Q is the set of states, from which we identify the subset Q_2. The elements q of Q_2 represent establishment of adjacent connections (such as TCP connections or from any other protocol type). The elements are characterized by the same set of conditions as the one in the rules (i.e., $q[SrcAddr]$, $q[DestAddr]$, etc.). Let us observe that $q[State]$ will highly rely on the specific firewall vendor (cf. following function definition, in which we define the way to link the specific state attribute of the automaton to the corresponding firewall device). Notice that $R_i[State]$ (i.e., the state defined in a given rule R_i) corresponds to the specific state as it is represented by the underlying firewall that contains the rule, not the state attribute of the automaton. In Netfilter, for instance, relevant connections are identified by the state attribute RELATED. If necessary, we can rely on extended features of Netfilter to provide a more fine-grained state management of some application layer protocols. $Q_1 = Q - Q_2$ contains the set of states that are independent from related connections, and for which the element $q[State]$ is not defined. Finally, the initial state q_0 of the automaton holds the following condition attributes: $q_0[SrcPort]$, $q_0[DestPort]$ and $q_0[Protocol]$ (corresponding to the transport layer protocol). Figure 3 depicts the example of an automaton based on our construction, for the FTP protocol.
- $stateFirewall(q)$: function that links a given state $q \in Q_2$ of the corresponding state automaton to the firewall. For instance, in the case of the FTP protocol and a firewall based

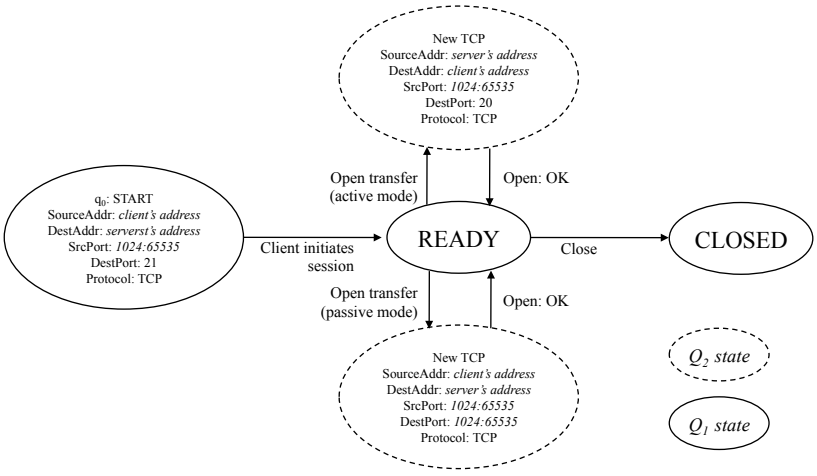

Fig. 3. Suggested automaton for the application layer protocol FTP.

on Netfilter, this function returns RELATED for those states in which the establishment of connections is called.

- $ruleExists(R, q)$: boolean function. R is a set of either stateless or stateful rules (but not both), q represents a state of the automaton A which belongs to Q_2 (the state corresponding to the establishment of related connections). If R contains stateless rules, then $ruleExists(R, q)$ is true iff there exists exactly one rule $R_i \in R$, such that $q[SourceAddr] \in R_i[SourceAddr]$, $q[DestAddr] \in R_i[DestAddr]$, $q[SrcPort] \in R_i[SrcPort]$, $q[DestPort] \in R_i[DestPort]$, and $q[protocol] = R_i[protocol]$. If R contains stateful rules, then $ruleExists(R, q)$ is true iff the previous conditions also hold and, moreover, $stateFirewall(q[State]) = R_i[State]$.
- $ruleExists(L, q_0)$: boolean function. q_0 contains the initial state of the protocol, and L is a set of stateless rules. The functions is true iff there exists a rule $L_i \in L$, such that $q_0[SrcPort] \in L_i[SrcPort]$, $q_0[DestPort] \in L_i[DestPort]$, $q_0[Protocol] = L_i[Protocol]$.

Algorithms: Algorithm 3 enables the verification of every state Q_2 of an automaton associated with a given protocol, in order to find rules that can be correlated. The algorithm specifies the appropriate corrections in accordance to the detection of anomalies, and following the three cases mentioned above (absence of rules, or misconfigured stateless or stateful rules). $A[Q_2]$ points out to the Q_2 set of the automaton.

Algorithm 4 allows detection and correction of anomalies between stateless and stateful rules, provided that a library of application layer protocols is given as input. Such a library must contain the corresponding automata for the protocols. Then, it verifies whether the firewall handles each of them, by looking at the initial state attribute q_0 of the corresponding automaton.

Algorithm 3: `HandleInterRuleAnomalies`(L, F, A)

```
/*A[Q2]: Q2 states for automaton A */
forall q ∈ A[Q2] do
    if ruleExists(F, q) then
        /*Move to following state */
        continue;
    end
    if ruleExists(L, q) then
        warning("stateless rule for state q of protocol A")
    else
        warning("missing rule for state q of protocol A")
    end
end
```

Algorithm 4: `HandleAllProtocols`$(L, F, Library)$

```
/*Library: automata library, containing
  the list of supported application-layer
  protocols */
forall A ∈ Library do
    if ruleExists(L, A[q0]) then
        HandleInterRuleAnomalies(L, F, A)
    end
end
```

11

In such a case, Algorithm 3 processes the specific anomalies associated with that protocol. $A[q_0]$ points out the initial state q_0 of every automaton.

The following example presents an extract from a Netfilter-based configuration. We can look at three rules that aim at granting authorization to FTP services, both in active and passive mode:

R_1: iptables –A FORWARD -s $ANY -d $SERVERS –sport 1024:65535 –dport 21 -j ACCEPT

...

R_2: iptables –A FORWARD -d $ANY -s $SERVERS –dport 1024:65535 –sport 20 -j ACCEPT

R_3: iptables –A FORWARD -s $ANY -d $SERVERS –sport 1024:65535 –dport 1024:65535 -j ACCEPT

Notice that the sample contains two inter-state anomalies. Rule R_1 is a stateless authorization to control incoming higher port TCP connections targeting a range of FTP servers listening on port 21. Then, rules R_2 are R_3 expected to grant authorization access to the data connection counterpart, i.e., outcoming TCP connection from servers to clients. However, these last two rules are stateless. They grant access to any connection targeting a TCP high port (i.e., the whole range 1024:65535). If we apply Algorithm 4 to the previous configuration, it will detect such a situation and suggest the administrator to handle it (e.g., by adding the Netfilter parameter `--state RELATED` to both rules).

6 Conclusion

Stateful firewalls are the predominant solution to guarantee network security. They provide an effective enforcement of access control rules at both network and transport layers, in order to protect incoming and outcoming interaction with the Internet. Nevertheless, the existence of anomalies in their configuration is very likely to degrade such a protection. While some anomalies may occur in rule sets that only contain stateful rules (*intra-state rule anomalies*), others affect rule sets that contain both stateful and stateless rules (*inter-state rule anomalies*). In this paper, we have presented new types of anomalies for each of these two categories, and have provided algorithmic solutions to handle them. General automata describing the stateful nature of the filtering process drive the discovering and correction functionality of our solutions. Perspectives for further work include the extension of our solutions towards multi-firewall scenarios, to handle distributed policy control.

Acknowledgements: This research was partially supported by the European Commission, in the framework of the ITEA2 Predykot project (Grant agreement no. 10035).

References

1. S. Preda, N. Cuppens-Boulahia, F. Cuppens, J. Garcia-Alfaro, and L. Toutain, “Model-Driven Security Policy Deployment: Property Oriented Approach,” in *ESSoS*, Pisa, Italy, 2010, pp. 123–139.

12

2. J. Garcia-Alfaro, N. Boulahia-Cuppens, and F. Cuppens, "Complete analysis of configuration rules to guarantee reliable network security policies," *Int. J. Inf. Sec.*, vol. 7, no. 2, pp. 103–122, 2008.
3. H. Adiseshu, S. Suri, and G. Parulkar, "Detecting and Resolving Packet Filter Conflicts," in *INFOCOM*, Tel Aviv, Israel, 2000, pp. 1203–1212.
4. E. Al-Shaer and H. Hamed, "Discovery of Policy Anomalies in Distributed Firewalls," in *INFOCOM*, Hong Kong, China, 2004.
5. L. Yuan, J. Mai, Z. Su, H. Chen, C. Chuah, and P. Mohapatra, "FIREMAN: A Toolkit for FIREwall Modeling and ANalysis," in *IEEE Symposium on Security and Privacy*, Berkeley, California, USA, 2006, pp. 199–213.
6. W. Cheswick, S. Bellovin, and A. Rubin, *Firewalls and Internet Security: Repelling the Wily Hacker, 2nd edn.* Addison-Wesley, 2003.
7. M. Gouda and A. Liu, "A model of stateful firewalls and its properties," in *DSN*, Yokohama, Japan, 2005, pp. 128–137.
8. L. Buttyan, G. Pék, and T. V. Thong, "Consistency verification of stateful firewalls is not harder than the stateless case," *Infocommunications Journal*, vol. LXIV, 2009.
9. J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, and S. Preda, "MIRAGE: A Management Tool for the Analysis and Deployment of Network Security Policies," in *DPM/SETOP*, Athens, Greece, 2010, pp. 203–215.
10. J. Guttman, "Filtering postures: Local enforcement for global policies," in *In Proceedings, 1997 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1997, pp. 120–129.
11. Y. Bartal, A. Mayer, K. Nissim, and A. Wool, "Firmato: A novel firewall management toolkit," 1999.
12. 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*, 2004, pp. 203–218.
13. S. Preda, F. Cuppens, N. Cuppens-Boulahia, J. Garcia-Alfaro, and L. Toutain, "Dynamic deployment of context-aware access control policies for constrained security devices," *Journal of Systems and Software*, vol. 84, no. 7, pp. 1144–1159, 2011.
14. S. Hazelhurst, A. Attar, and R. Sinnappan, "Algorithms for improving the dependability of firewall and filter rule lists," in *DSN*, 2000, pp. 576–585.
15. A. Liu, M. Gouda, H. Ma, and A. Ngu, "Firewall queries," in *In Proceedings of the 8th International Conference on Principles of Distributed Systems, LNCS 3544, T. Higashino Ed.* Springer-Verlag, 2004, pp. 124–139.
16. A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *IEEE Symposium on Security and Privacy*, 2000, pp. 177–187.
17. F. Baboescu and G. Varghese, "Scalable packet classification," in *In ACM SIGCOMM*, 2001, pp. 199–210.
18. D. Eppstein and S. Muthukrishnan, "Internet packet filter management and rectangle geometry," 2001, pp. 827–835.
19. E. Al-Shaer and H. Hamed, "Firewall policy advisor for anomaly discovery and rule editing," in *Integrated Network Management*, 2003, pp. 17–30.
20. J. Garcia-Alfaro, F. Cuppens, and N. Cuppens-Boulahia, "Analysis of policy anomalies on distributed network security setups," in *ESORICS*, 2006, pp. 496–511.
21. ——, "Management of exceptions on access control policies," in *SEC*, 2007, pp. 97–108.
22. V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in *In Proc. of SIGCOMM*, 1999, pp. 135–146.
23. W. Fitzgerald, S. Foley, and M. Ó. Foghlú, "Network access control interoperation using semantic web techniques," in *WOSIS*, 2008, pp. 26–37.