Detection of Network Security Component Misconfiguration by Rewriting and Correlation

Frederic Cuppens, Nora Boulahia Cuppens, Joaquin Garcia-alfaro

Detection of Network Security Component Misconfiguration by Rewriting and Correlation

Frédéric Cuppens[†], Nora Cuppens-Boulahia[†], and Joaquín García-Alfaro[†,‡]

[†] *GET/ENST-Bretagne, 02, rue de la Châtaigneraie, 35576 Cesson Sévigné - France*

[‡] *DEIC/UAB, Edifici Q, Campus de Bellaterra, 08193, Bellaterra, Barcelona - Spain*

The use of firewalls is the dominant method to guarantee network access control, leading to distributed access control scenarios where the access control role is assigned to more than one component. Firewalls are network security components which provide means to filter traffic within corporate networks, as well as to police incoming and outcoming interaction with the Internet. For this purpose, it is necessary to configure firewalls with a set of filtering rules. Nevertheless, the existence of anomalies between rules, particularly in distributed access control scenarios, is very likely to degrade the network security policy. The discovering and removal of these anomalies is a serious and complex problem to solve. In this paper, we present a set of algorithms for such a management. Our approach is based on the analysis of relationships between filtering rules, in order to detect anomalies, as well as propose policy changes within both single or multi-firewall scenarios.

Keywords: Network Security, Distributed Firewalls, Filtering Rules, Policy Anomalies

1 Introduction

Many companies use firewalls to filter traffic between *trusted* and *untrusted* zones of corporate networks, as well as to police their incoming and outcoming interaction with the Internet. Firewalls are security components, with several interfaces associated with the different zones of the network. A company may partition, for instance, its network into three different zones: a demilitarized zone, a private network and a zone for security administration. In this case, one may use a single firewall setup, with three interfaces associated with these three zones, as well as a multi firewall setup, with a firewall protecting each zone.

To apply the filtering policy, it is necessary to configure each firewall with a set of filtering rules. Each filtering rule typically specifies a *decision* (e.g., *accept* or *deny*) that applies over a set of *condition* attributes, such as protocol, source, destination, and so on. For our work, we define a filtering rule as follows:

$$R_i : \{condition_i\} \rightarrow decision_i \tag{1}$$

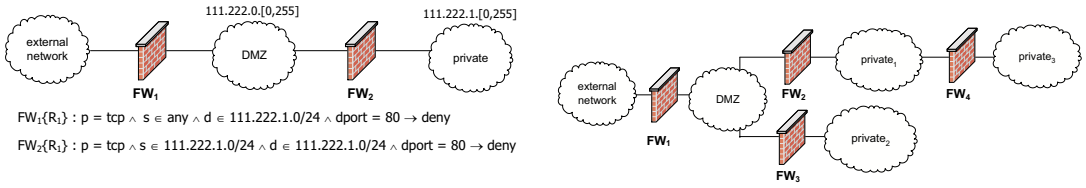
where i is the relative position of the rule within the set of rules, $decision_i$ is a boolean expression in $\{accept, deny\}$, and $\{condition_i\}$ is a conjunctive set of condition attributes such that $\{condition_i\}$ equals $A_1 \wedge A_2 \wedge ... \wedge A_p$, and p is the number of condition attributes of the given filtering rules.

In a single firewall scenario (cf. Figure 1(a)), conflicts due to rule overlaps, i.e., the same rule matching more than one filtering rule, can occur. To solve these conflicts, most firewall implementations use a *first matching* strategy through the ordering of rules. This way, each packet processed by the firewall is mapped to the decision of the rule with highest priority. This strategy introduces, however, new configuration errors, often referred in the literature as *intra-firewall anomalies*. In multi-firewall setups (cf. Figure 1(b)), on the other hand, different firewalls within the same path may perform different decision to the same network traffic. This problem is often referred in the literature as *inter-firewall anomalies*.

The discovering and removal of both intra and inter-firewall anomalies is a serious problem which must be solved since, a misconfigured policy, if not handled correctly, is very likely to cause packets to be subject to the wrong actions, and to lead to a weak security policy.

In [6], we presented an audit process to manage intra-firewall policy anomalies, in order to detect and remove anomalies within the set of rules of a given firewall. This audit process is based on the existence of relationships between the condition attributes of the filtering rules, such as coincidence, disjunction, and inclusion, and proposes a transformation process which derives from an initial set of rules – with potential policy anomalies – to an equivalent one which is completely free of errors. Furthermore, the resulting rules are completely disjoint, i.e., the ordering of rules is no longer relevant. In this paper, we extend our proposal of detecting and removing intra-firewall policy anomalies [6], to a distributed firewall setup. Our extended approach is based on the hypothesis that not only one firewall ensures the network access control. We assume that this role is assigned to more than one network security component, i.e., a distributed access control. Our main objective is the following. Given a specific distributed access control setup, we want to analyze the existing firewall configurations to check whether there are errors in such a configuration regarding the policy set up of the rest of firewalls which match the same traffic.

The advantages of our proposal are twofold. First, when performing our proposed intra-firewall discovery of anomalies, and after rewriting the rules, one can verify that the resulting configuration of each firewall in the network is free of errors. Each anomalous rule – considered as useless during the audit process – will be removed from the set of filtering rules of each given firewall. Second, when applying our proposed inter-firewall discovery of anomalies, as well as when performing the intra-firewall proposal, the discovering process will provide an evidence of error to the administration console. This way, the security officer in charge of the network can check the network policy, in order to verify the correctness of the whole process, and perform the proper policy modifications to avoid such anomalies.



(a) Single-firewall setup

(b) Multi-firewall setup

Fig. 1: Example of some firewall setups

The rest of this paper is organized as follows. Section 2 starts with an analysis of some related work. Sections 3 and 4 present, respectively, our intra and inter-firewall algorithms. Section 5 overviews the performance of our proposed algorithms, and Section 6 closes the paper.

2 Related Work

A first approach to get an access control policy free of errors is by applying a formal security model to express the network policy. In [5], for example, we presented a formal model with this purpose. This way, a set of filtering rules, whose syntax is specific to a given firewall, may be generated using a transformation language.

Some other proposals, such as [1, 7, 8, 3], provide means to directly manage the discovery of anomalies from the components' configuration. For instance, the authors in [1] consider that, in a configuration set, two rules are in conflict when the first rule in order matches some packets that match the second rule, and the second rule also matches some of the packets that match the first rule. This approach is very limited since it just detects a particular case of wrongly defined rules in a single firewall configuration, i.e., just ambiguity within a intra-firewall configurations could be detected. It does not provide, furthermore, detection on more complex scenarios, i.e., inter-firewall configurations, where more than one component is intended to perform network access control.

In [7], two new cases of anomalies are considered. First, a rule R_j is defined as backward redundant iff there exists another rule R_i with higher priority in order such that all the packets that match rule R_j also

match rule R_i. Second, a rule R_i is defined as forward redundant iff there exists another rule R_j with the same decision and less priority in order such that the following conditions hold: (1) all the packets that match R_i also match R_j; (2) for each rule R_k between R_i and R_j, and that matches all the packets that also match rule R_i, R_k has the same decision as R_i. Although this approach seems to head in the right direction, we consider it as incomplete, since it does not detect all the possible cases of anomalies (as we define in sections 3 and 4). For instance, given the set of rules shown in Figure 2(a), since R_2 comes after R_1, rule R_2 only applies over the interval $[51,70]$ – i.e., R_2 is not necessary, since, if we remove this rule from the configuration, the filtering policy does not change. The detection proposal, as defined in [7], cannot detect the redundancy of rule R_2 within the configuration of such a given firewall.

$R_1 : s \in [10,50] \rightarrow deny$
$R_2 : s \in [40,70] \rightarrow accept$
$R_3 : s \in [50,80] \rightarrow accept$
(a) Set of rules A

$R_1 : s \in [10,50] \rightarrow accept$
$R_2 : s \in [40,90] \rightarrow accept$
$R_3 : s \in [30,80] \rightarrow deny$
(b) Set of rules B

Fig. 2: Example of some firewall configurations

To our best knowledge, the *firewall policy advisor* [3] propose the most efficient set of techniques and algorithms to detect policy anomalies in both single and multi-firewall configuration setups. In addition to the discovery process, their approach also attempts an optimal insertion of arbitrary rules into an existing configuration, through a tree based representation of the filtering criteria. Nonetheless, and even though the efficiency of their proposed discovering algorithms and techniques is very promising, we also consider this approach as incomplete.

First, their intra-firewall discovery approach is too weak since, given a misconfigured firewall, their detection algorithms could not detect all the possible errors. For example, given the set of rules shown in Figure 2(b) the approach defined in [3] cannot detect that rule R_3 will be never applied due to the union of rules R_1 and R_2. Second, the authors do not cover an automatic rewriting of rules, as our intra-firewall approach does, to correct the discovered errors. This way, they intentionally point this work to be performed by the security officer, once the discovery process will finish. Third, their inter-firewall discovery approach considers anomalies some situations that, from our point of view, must be suited to avoid inconsistent decisions between firewalls used in the same policy to control the access to different zones. For instance, given the scenario shown in Figure 1(a), their algorithms will wrongly report a redundancy anomaly between filtering rules $FW_1\{R_1\}$ and $FW_2\{R_1\}$. This is because rule $FW_1\{R_1\}$ matches every packet that also $FW_2\{R_1\}$ does. As a consequence, [3] considers rule $FW_2\{R_1\}$ as redundant since packets denied by this rule are already denied by rule $FW_1\{R_1\}$. However, this conclusion is wrong because rule $FW_1\{R_1\}$ applies to packets from the external zone to the private zone whereas rule $FW_2\{R_1\}$ applies to packets from the DMZ zone to the private zone. So, rule $FW_2\{R_1\}$ is useful and cannot be removed. Summing up, [3] is unable to draw this right conclusion because it does not properly model, as we do (cf. Section 4.1), which traffic flows through a given firewall.

3 Intra-Firewall Analysis

The main objective of the intra-firewall algorithms we proposed in [6] is the discovering and removal of both redundancy and shadowing anomalies inside an initial set of filtering rules R. These two main intra-firewall anomalies are defined as follows.

Intra-Firewall Redundancy Let R be a set of filtering rules. Then R has redundancy if and only if there exists at least one filtering rule, R_i in R, such that when removing R_i from R, the filtering result, i.e., the security policy, does not change.

Intra-Firewall Shadowing Let R be a set of filtering rules. Then R has shadowing if and only if there exists at least one filtering rule, R_i in R, which 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.

Algorithm 1: exclusion(B,A)

```
1  begin
2      C[condition] ← ∅;
3      C[decision] ← B[decision];
4      C[shadowing] ← false;
5      C[redundancy] ← false;
6      forall A[condition] and B[condition] do
7          if ((A1 ∩ B1) ≠ ∅ and (A2 ∩ B2) ≠ ∅ and ...
           and (Ap ∩ Bp) ≠ ∅) then
8              C[condition] ← C[condition] ∪
9              {(B1 − A1) ∧ B2 ∧ ... ∧ Bp,
10             (A1 ∩ B1) ∧ (B2 − A2) ∧ ... ∧ Bp,
11             (A1 ∩ B1) ∧ (A2 ∩ B2) ∧ (B3 − A3) ∧ ... ∧ Bp,
12             ...
13             (A1 ∩ B1) ∧ ... ∧ (Ap−1 ∩ Bp−1) ∧ (Bp − Ap)};
14         else
15             C[condition] ←
16              (C[condition] ∪ B[condition]);
17     return C;
18 end
```

Algorithm 2: testRedundancy(R,r)

```
1  begin
2      test ← false;
3      i ← 1;
4      temp ← r;
5      while ¬test and (i ≤ count(R)) do
6          temp ← exclusion(temp, Ri);
7          if temp[condition] = ∅ then
8              test ← true;
9          i ← (i + 1);
10     return test;
11 end
```

Algorithm 3: intra-firewall-audit(R)

```
1  begin
2      n ← count(R);
3      /*Phase 1*/
4      for i ← 1 to (n − 1) do
5          for j ← (i + 1) to n do
6              if Ri[decision] ≠ Rj[decision] then
7                  Rj ← exclusion (Rj,Ri);
8                  if Rj[condition] = ∅ then
9                  Rj[shadowing] ← true;
10     /*Phase 2*/
11     for i ← 1 to (n − 1) do
12         Ra ← {rk ∈ R | n ≥ k > i and
13             rk[decision] = ri[decision]};
14         if testRedundancy (Ra,Ri) then
15             Ri[condition] ← ∅;
               Ri[redundancy] ← true;
16         else
17             for j ← (i + 1) to n do
18                 if Ri[decision]=Rj[decision] then
19                     Rj ←exclusion (Rj,Ri);
20                     if (¬Rj[redundancy] and
21                     Rj[condition] = ∅) then
22                         Rj[shadowing] ← true;
23 end
```

Our proposed audit process is a way to alert the security officer in charge of the network about these configuration errors, as well as to remove all the useless rules in the initial firewall configuration. The data to be used for the detection process is the following. A set of rules *R* as a list of initial size *n*, where *n* equals *count*(R), and where each element is an associative array with the strings *condition*, *decision*, *shadowing*, and *redundancy* as keys to access each necessary value.

To simplify the algorithms, we assume one can access a linked-list through the operator R_i, where *i* is the relative position regarding the initial list size – *count*(R). We also assume one can add new values to the list as any other normal variable does (*element* ← *value*), as well as to remove elements through the addition of an empty set (*element* ← ∅). The internal order of elements from the linked-list *R* keeps with the relative ordering of rules. In turn, each element $R_i[condition]$ is an indexed array of size *p* containing the set of conditions of each rule; each element $R_i[decision]$ is a boolean variable whose values are in $\{accept, deny\}$; each element $R_i[shadowing]$ is a boolean variable in $\{true, false\}$; each element $R_i[redundancy]$ is another boolean variable in $\{true, false\}$. These variables are initialized to $false$ by default.

For reasons of clarity, we split the whole process in three different algorithms. The first algorithm is an auxiliary function whose input is two rules, *A* and *B*. Once executed, this auxiliary function returns a further rule, *C*, whose set of condition attributes is the exclusion of the set of conditions from *A* over *B*. In order to simplify the representation of this algorithm (cf. Algorithm 1), we use the notation A_i as an abbreviation of the variable $A[condition][i]$, and the notation B_i as an abbreviation of the variable $B[condition][i]$ – where *i* in $[1, p]$. The second algorithm is a boolean function in $\{true, false\}$ which, in turn, applies the transformation *exclusion* over a set of filtering rules to check whether the rule obtained as a parameter is potentially redundant.

The third algorithm performs the whole process of detecting and removing both redundancy and shadowing, and is also split in two different phases. During the first phase, a set of shadowing rules are detected and removed, by iteratively applying Algorithm 1 – when the decision field of the two rules is different. Let us

notice that this stage of detecting and removing shadowed rules is applied before the detection and removal of proper redundant rules. The resulting set of rules is then used when applying the second phase. This stage is performed to detect and remove proper redundant rules, as well as to detect and remove all the further shadowed rules resulting during the latter process.

3.1 Correctness of the Intra-Firewall Algorithms

Theorem 3.1 *Let R be a set of filtering rules and let $Tr(R)$ be the resulting filtering rules obtained by applying Algorithm 3 to R. Then the following statements hold: (1) R and $Tr(R)$ are equivalent; (2) Ordering the rules in $Tr(R)$ is no longer relevant; (3) $Tr(R)$ is free from both shadowing and redundancy*[†].

3.2 Default policies

Each firewall implements an open or closed default policy. In the open policy, the default policy is to accept a packet when no filtering rule applies. By contrast, the closed policy will reject a packet when no rule applies. After rewriting the rules with the intra-firewall-audit algorithm (cf. Algorithm 3), we can actually remove every rule whose decision is accept if the default policy of this firewall is open (else this rule is redundant with the default policy) and similarly we can remove every rule whose decision is deny if the default policy is closed. Thus, we can consider that our intra-firewall algorithm generates a configuration that only contains accept rules if the firewall default policy is closed and deny rules if the default policy is open.

4 Inter-Firewall Analysis

The objective of the inter-firewall audit algorithms is the complete discovering of policy anomalies that could exist in a multi-firewall scenario, i.e., to discover and warn the security officer about potential anomalies between policies of different firewalls.

The main hypotheses to deploy our algorithms hold the following: (1) An upstream network traffic flows away from the closest firewall to the origin of this traffic (i.e., the most-upstream firewall) towards the closest firewall to the remote destination (i.e., the most-downstream firewall); (2) Every firewall policy in the network has been rewritten through the algorithms defined in Section 3, i.e., it does not contain intra-firewall anomalies and the rules within such a policy are completely independent between them.

4.1 Network Model

The purpose of our network model is to determine which firewalls are crossed by a given packet knowing its source and destination. It is defined as follows. First, and concerning the traffic flowing from two different zones of the distributed access control scenario, we may determine the set of firewalls that are crossed by this flow. Regarding the scenario shown in Figure 1(b), for example, the set of firewalls crossed by the network traffic flowing from zone *external network* to zone $private_3$ equals [FW_1,FW_2,FW_4], and the set of firewalls crossed by the network traffic flowing from zone $private_3$ to zone $private_2$ equals [FW_4,FW_2,FW_3].

Let F be a set of firewalls and let Z be a set of zones. We assume that each pair of zones in Z are mutually disjoint, i.e., if $z_i \in Z$ and $z_j \in Z$ then $z_i \cap z_j = \emptyset$. We then define the predicate $connected(f_1, f_2)$ as a symmetric and anti-reflexive function which becomes *true* whether there exists, at least, one interface connecting firewall f_1 to firewall f_2. On the other hand, we define the predicate $adjacent(f,z)$ as a relation between firewalls and zones which becomes *true* whether the zone z is interfaced to firewall f. Referring to Figure 1(b), we can verify that predicates $connected(FW_1,FW_2)$ and $connected(FW_1,FW_3)$, for example, become *true*, as well as predicates $adjacent(FW_1,DMZ)$, $adjacent(FW_2,private_1)$, $adjacent(FW_3,DMZ)$, and so on, also do. We then define the set of paths, P, as follows. If $f \in F$ then $[f] \in P$ is an atomic path. Similarly, if $[p.f_1] \in P$ (be "." a concatenation functor) and $f_2 \in F$, such that $f_2 \notin p$ and $connected(f_1, f_2)$,

[†] A set of proofs to validate Theorem 3.1, as well as a complexity analysis for the intra-firewall algorithms, is provided in [6].

then $[p.f_1.f_2] \in P$. This way, we can notice that, concerning Figure 1(b), $[FW_1, FW_2, FW_4] \in P$ and $[FW_1, FW_3] \in P$.

Let us now define the functions $first$, $last$, and the order functor between paths. We first define function $first$ from P in F such that if p is a path, then $first(p)$ corresponds to the first firewall in the path. Conversely, we define function $last$ from P in F such that if p is a path, then $last(p)$ corresponds to the last firewall in the path. We then define the order functor between paths as $p_1 \leq p_2$, such that path p_1 is shorter than p_2, and where all the firewalls within p_1 are also within p_2.

Finally, let us conclude this section by defining the functions *route* and *minimal_route*. We define function *route* from Z to Z, i.e., $\{route(z_1, z_2) : Z \times Z \; in \; 2^P\}$, such that $p \in route(z_1, z_2)$ iff path p connects zone z_1 to zone z_2. Formally, $p \in route(z_1, z_2)$ iff $adjacent(first(p), z_1)$ and $adjacent(last(p), z_2)$. Similarly, we define the function *minimal_route* from Z to Z, i.e., $\{minimal\_route(z_1, z_2) : Z \times Z \; in \; 2^P\}$, such that $p \in minimal\_route(z_1, z_2)$ iff the following conditions hold: (1) $p \in route(z_1, z_2)$; (2) There does not exist $p' \in route(z_1, z_2)$ such that $p' < p$. Thus, and regarding Figure 1(b), we can verify that the *minimal_route* from zone $private_3$ to zone $private_2$ equals $[FW_4, FW_2, FW_3]$, i.e., $minimal\_route(private_3, private_2) = \{[FW_4, FW_2, FW_3]\}$.

4.2 Inter-Firewall Anomalies Classification

We classify in this section the complete set of anomalies that can occur within a multi-firewall setup. Our classification is based on the network model presented in Section 4.1. An example for each anomaly will be illustrated through the distributed access control setup shown in Figure 3. Referring to this figure, we assume that a network traffic flows from the most-upstream firewall (which is the closest firewall to the flow source's zone) to the most-downstream firewall (which is the closest firewall to the flow destination's zone).

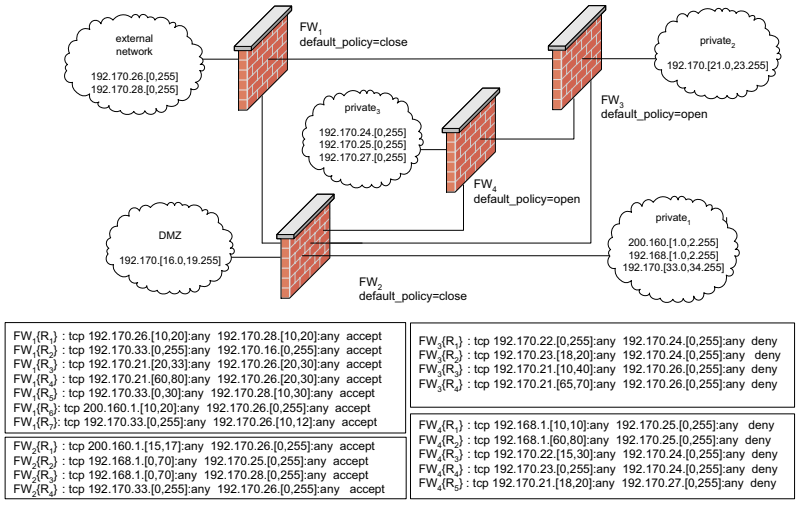


Fig. 3: Example of a distributed network policy setup

Inter-Firewall Reflexivity A filtering rule in a multi-firewall setup is reflexive if both source and destination address are within the same zone, and its decision is accept. Thus, the reflexive rule will never match network traffic since it does not flow through this firewall. For instance, referring to Figure 3, rule $FW_1\{R_1\}$ is reflexive since the source of this address, *external network*, as well as its destination, is the same.

Inter-Firewall Irrelevance An irrelevance anomaly occurs when a firewall is not within the minimal route that connects the source zone, concerning the irrelevant rule which causes the anomaly, to the destination zone. Hence, the rule is irrelevant since it matches traffic which does not flow through this firewall. Rule $FW_1\{R_2\}$, for example, is irrelevant since firewall FW_1 is not in the path which corresponds to the minimal route between the source zone $private_1$ to the destination zone DMZ.

Inter-Firewall Spuriousness A spuriousness anomaly occurs if the most-upstream firewall denies, completely or partially, network traffic that, in turn, is accepted by a downstream firewall. The first case, *full spuriousness anomaly*, is when the most-upstream firewall denies all the traffic. As we can see in Figure 3, rule $FW_4\{R_1\}$ shows an example of full spuriousness anomaly with rule $FW_2\{R_2\}$. The second case, *partial spuriousness anomaly*, happens when the most-upstream firewall denies just a part of the traffic that is accepted by the firewall where the anomaly is detected. Rules $FW_4\{R_2\}$ and $FW_2\{R_2\}$ show an example of partial spuriousness anomaly.

Inter-Firewall Redundancy A redundancy anomaly occurs if a downstream firewall completely or partially blocks traffic that is already blocked by the most-upstream firewall. Rules $FW_4\{R_3\}$ and $FW_3\{R_1\}$ show a proper example of full redundancy, whereas rules $FW_4\{R_4\}$ and $FW_3\{R_2\}$ show an example of *partial redundancy*. Sometimes, this kind of redundancy is expressly introduced by network administrators to guarantee the forbidden traffic will not reach the destination. Nonetheless, it is important to discover it since, if such a rule is applied, we may conclude that at least one of the redundant firewalls is wrongly working.

Inter-Firewall Shadowing A *full shadowing anomaly* occurs if the most-upstream firewall completely blocks network traffic that, in turn, is permitted by a downstream firewall. Rule $FW_1\{R_3\}$ shows an example of full shadowing anomaly with rule $FW_3\{R_3\}$. On the other hand, a *partial shadowing anomaly* occurs if the most-downstream firewall denies just some network traffic that, in turn, is permitted by an upstream firewall. Rule $FW_1\{R_4\}$ is a proper example of partial shadowing anomaly with rule $FW_3\{R_4\}$.

Inter-Firewall Accessibility To guarantee the flow of network traffic may reach the destination, all upstream firewalls must permit, explicitly with a rule, or implicitly through the default open policy, any traffic that is also permitted along the downstream firewalls chain. Otherwise, a completely or partially accessibility anomaly will occur if one of the following conditions holds:

(1) The traffic is permitted by a downstream firewall, but not permitted – explicitly or implicitly – by the most-upstream firewall. Rules $FW_1\{R_5\}$ and $FW_1\{R_6\}$ show an example of this first case of *full* and *partial accessibility anomaly*, respectively, with the upstream firewall FW_2.
(2) The traffic is permitted by an upstream firewall, but not permitted – explicitly or implicitly – by the following downstream firewall. Rules $FW_2\{R_3\}$ and $FW_2\{R_4\}$ show an example of this second case of *full* and *partial accessibility anomaly*, respectively, with the downstream firewall FW_1.

Inter-Firewall Misconnection A filtering rule in a distributed firewall setup is misconnected if this rule blocks traffic which is not explicitly blocked by the most-upstream firewall and, at the same time, the default policy of such a firewall is open. A proper example of this anomaly is rule $FW_4\{R_5\}$, since it blocks traffic which is not explicitly specified within the firewall FW_3 – which, in turn, has an open default policy.

4.3 Inter-Firewall Analysis Algorithms

For reasons of clarity, we split the whole analysis process in four different algorithms. The input for the first algorithm (cf. Algorithm 4) is the set of firewalls F, such that for all $fw \in F$, we note $fw[rules]$ as the set of filtering rules of firewall fw, and $fw[policy] \in \{open, close\}$ as the default policy of such a firewall fw. In turn, each rule $r \in fw[rules]$ consists of a boolean expression over the attributes *szone* (source zone), *dzone* (destination zone), *sport* (source port), *dport* (destination port), *protocol*, and *decision* (accept or deny).

We then define the functions $source(r) = szone$ and $dest(r) = dzone$. Thus, we compute for each firewall $fw \in F$ and for each rule $r \in fw[rules]$, each one of the source zones $z_1 \in Z_s$ and destination zones $z_2 \in Z_d$ – whose intersection with respectively *szone* and *dzone* is not empty – which become, together with a reference to each firewall fw and each rule r, the input for the second algorithm (i.e., Algorithm 5).

The first verification Algorithm 5 does is to check whether both z_1 and z_2 are the same. If this case occurs, and the decision of this rule is pointing to *accept*, it warns the security officer about the occurrence of a *reflexivity anomaly* (cf. Section 4.2). Otherwise, it computes the minimal route of firewalls that connects

zone z_1 to z_2, i.e., $[FW_1, FW_2, \ldots, FW_n] \in \mathit{minimal\_route}(z_1, z_2)$. Once computed the set of paths, the second verification of Algorithm 5 is to check whether current firewall fw is not within such a path. If so, it warns the security officer about the occurrence of an *irrelevance anomaly* (cf. Section 4.2). Otherwise, it decomposes the set of firewalls inside each path in upstream path ($path_u$) and downstream path ($path_d$). To do so, we use the implicit functions *head* and *tail*. Then, the first firewall $fw_d \in path_d$, and the last firewall $fw_u \in path_u$ are passed, respectively, as argument to the last two algorithms (i.e., Algorithm 6 and Algorithm 7) in order to conclude the set of necessary checks that guarantee the audit process[‡].

Algorithm 4: `inter-firewall-audit`(F)

```
1  foreach fw ∈ F do
2      foreach r ∈ fw[rules] do
3          Z_s ← {z ∈ Z | z ∩ source(r) ≠ ∅};
4          Z_d ← {z ∈ Z | z ∩ dest(r) ≠ ∅};
5          foreach z_1 ∈ Z_s do
6              foreach z_2 ∈ Z_d do
7                  audit(fw,r,z_1,z_2);
```

Algorithm 5: `audit`(fw,r,z_1,z_2)

```
1  if (z_1 = z_2) and (r[decision] ="accept") then
2      warning("Reflexivity");
3  else if z_1 ≠ z_2 then
4      foreach p ∈ minimal_route(z_1,z_2) do
5          if fw ∉ p and r[decision] ="accept" then
6              warning("Irrelevance");
7          else if fw ∈ p then
8              path_d ← tail(p,fw);
9              path_u ← header(p,fw);
10             if path_d ≠ ∅ and r[decision] ="accept"
               then
11             fw_d ← first(path_d);
12             downstream(r,fw,fw_d);
13             if path_u ≠ ∅
14             then
15             fw_u ← last(path_u);
16             upstream(r,fw,fw_u);
```

Algorithm 6: `downstream`(r,fw,fw_d)

```
1  if fw_d[policy] = close then
2      R_da ← {r_d ∈ fw_d | r_d ⇝ r ∧ r_d[decision] = accept};
3      if R_da = ∅ then warning("Full Accessibility");
4      else if ¬ testRedundancy(R_da,r) then
5          warning("Partial Accessibility");
```

Algorithm 7: `upstream`(r,fw,fw_u)

```
1  R_ua ← {r_u ∈ fw_u | r_u ⇝ r ∧ r_u[decision] = accept};
2  R_ud ← {r_u ∈ fw_u | r_u ⇝ r ∧ r_u[decision] = deny};
3  if r[decision] ="deny" then
4      if testRedundancy(R_ua,r) then
5          warning("Full Spurious");
6      else if R_ua ≠ ∅ then
7          warning("Partial Spurious");
8      else if testRedundancy(R_ud,r) then
9          warning("Full Redundancy");
10     else if R_ud ≠ ∅ then
11         warning("Partial Redundancy");
12     else if R_ua = ∅ and R_ud = ∅ and
       fw_u[policy] = open then
13         warning("Misconnection");
14 else
15     if testRedundancy(R_ud,r) then
16         warning("Full Shadowing");
17     else if R_ud ≠ ∅) then
18         warning("Partial Shadowing");
19     else if R_ua = ∅ and fw_u[policy] = close then
20         warning("Full Accessibility");
21     else if ¬ testRedundancy(R_ua,r) and
       fw_u[policy] = close then
22         warning("Partial Accessibility");
```

Algorithm 8: `deployment`(r, z_1, z_2)

```
1  if r[decision] = accept then
2      foreach fw ∈ minimal_route(z_1, z_2) do
3          if fw[policy] = close then
           fw[rules] ← fw[rules] ∪ r'
4  else
5      fw ← first(minimal_route(z_1, z_2))
6      if fw[policy] = open then
       fw[rules] ← fw[rules] ∪ r'
```

Let us conclude this section by giving an outlook to the set of warnings send to the security officer after the execution of Algorithm 4 over the scenario of Figure 3:

(**Reflexivity**) on $FW_1\{R_1\}$
(**Irrelevance**) on $FW_1\{R_2\}$
(**Full Shadowing**) on $FW_1\{R_3\}$ with $FW_3\{R_3\}$
(**Partial Shadowing**) on $FW_1\{R_4\}$ with $FW_3\{R_4\}$
(**Full Accessibility**) on $FW_1\{R_5\}$ with FW_2
(**Partial Accessibility**) on $FW_1\{R_6\}$ with $FW_2\{R_1\}$
(**Full Accessibility**) on $FW_2\{R_3\}$ with FW_1
(**Partial Accessibility**) on $FW_2\{R_4\}$ with $FW_1\{R_7\}$
(**Full Spurious**) on $FW_4\{R_1\}$ with $FW_2\{R_2\}$
(**Partial Spurious**) on $FW_4\{R_2\}$ with $FW_2\{R_2\}$
(**Full Redundancy**) on $FW_4\{R_3\}$ with $FW_3\{R_1\}$
(**Partial Redundancy**) on $FW_4\{R_4\}$ with $FW_3\{R_2\}$
(**Misconnection**) on $FW_4\{R_5\}$ with FW_3

‡ The operator "$\backsim$" within algorithms 6 and 7 denotes that two rules r_i and r_j are correlated if every attribute in R_i has a non empty intersection with the corresponding attribute in R_j.

4.4 Correctness of the Inter-Firewall Algorithms

To prove the correctness of the inter-firewall algorithms, we first define what is a deployment without anomalies for a set of filtering rules. For this purpose, let us consider a set R of filtering rules to be deployed over a set F of firewalls that partitions a network into a set Z of zones. We also assume that F has be rewritten by applying the intra-firewall-audit algorithm showin in Section 3.

Let us now consider a rule $r \in R$ and let us assume that r applies to a source zone z_1 and a destination zone z_2, i.e., $s = z_1 \cap source(r) \neq \emptyset$ and $d = z_2 \cap dest(r) \neq \emptyset$. Let r' be a rule identical to r except that $source(r') = s$ and $dest(r') = d$. Finally, let us assume that $[FW_1, FW_2, ..., FW_k] \in minimal\_route(z_1, z_2)$. By keeping with all these statements, and based on our deployment algorithm (cf. Algorithm 8), we can now prove the following theorem.

Theorem 4.1 *Let F be a set of firewalls. The inter-firewall algorithms presented in Section 4 do not detect any anomaly in the configurations of F iff there is a set R of filtering rules such that configurations of F are obtained by applying the above deployment algorithm*[§].

5 Performance Evaluation

In this section, we present an evaluation of the performance of MIRAGE (which stands for MIsconfiguRAtion manaGEr), a software prototype that implements the intra and inter-firewall algorithms presented in sections 3 and 4. MIRAGE has been developed using PHP language. MIRAGE can be locally or remotely executed by using a HTTP server (e.g., Apache server over UNIX or Windows setups) and a web browser.

We evaluated our approach through the following experiments[¶]. We first measured the memory and time processing needed to perform Algorithm 3 over several sets of filtering policies for a first IPv4 network, according to the three following security officer profiles: beginner, intermediate, and expert – where the probability to have overlaps between rules increases from 5% to 90%. The results of these measurements are plotted in Figure 4(a) and Figure 4(b). We conducted, in a second phase, similar experiments to measure the performance and scalability of Algorithm 4 through a progressive increment of rules, firewalls and zones for a second IPv4 network. The results of these measurements are plotted in Figure 5(a) and Figure 5(b).

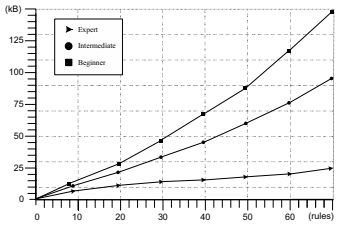


Fig.4(a). Memory space evaluation for the intra-firewall algorithms

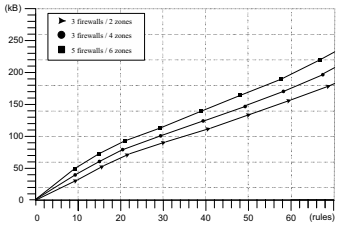


Fig.5(a). Memory space evaluation for the inter-firewall algorithms

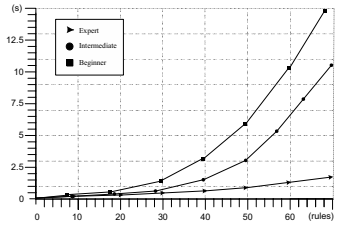


Fig.4(b). Processing time evaluation for the intra-firewall algorithms

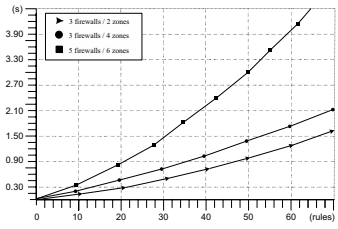


Fig.5(b). Processing time evaluation for the inter-firewall algorithms

§ This theorem guarantees the correctness of the anomalies detected by our inter-firewall algorithms.

¶ The whole of these experiments were carried out on an Intel-Pentium M 1.4 GHz processor with 512 MB RAM, running Debian GNU/Linux 2.6.8, and using Apache/1.3 with PHP/4.3 configured.

6 Conclusions

In this paper we presented an audit process to set a distributed access control policy free of anomalies. Our audit process has been presented in two main blocks. We first presented, in Section 3, a set of algorithms for intra-firewall analysis, according to the discovering and removal of policy anomalies over single-firewall environments. The detection process is based on the existence of relationships between the condition attributes of the filtering rules, such as coincidence, disjunction, and inclusion. Then, our proposal uses a transformation process which derives from an initial set of rules – potentially misconfigured – to an equivalent one which is completely free of misconfiguration. We then presented, in Section 4, a set of algorithms for inter-firewall analysis, in order to detect and warn the security officer about the complete existence of anomalies over a multi-firewall environment.

Some advantages of our approach are the following. First, our intra-firewall transformation process verifies that the resulting rules are completely independent between them. Otherwise, each redundant or shadowed rule considered as useless during the process is removed from the configuration. Second, both intra and inter-firewall discovering processes provide an evidence of error to the administration console. This way, the security officer can check whether the security policy is consistent, in order to verify the correctness of the whole process.

We presented in Section 4.1 a network model to determine which firewalls are crossed by a given packet knowing its source and destination. Thanks to this model, our approach better defines all the set of anomalies studied in the related work, and it reports, moreover, three new anomalies (*reflexivity*, *misconnection*, and *irrelevance*) not reported, as defined in this paper, in none of the other approaches. Furthermore, and as pointed out in Section 2, the lack of this model in [3] leads to wrong decisions. Finally, the implementation of our approach in a software prototype demonstrates the practicability of our work. We shortly discussed this implementation, and presented an evaluation of its performance. Although these results show that our algorithms have strong requirements, we believe that these requirements are reasonable for off-line analysis, since it is not part of the critical performance of a firewall.

Acknowledgements

This work was supported by funding from the French ministry of research, under the *ACI DESIRS* project, the Spanish Government project *TIC2003-02041*, and the Catalan Government grant *2003FI126*.

References

[1] Adiseshu, H., Suri, S., and Parulkar, G. (2000). Detecting and Resolving Packet Filter Conflicts. *In 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, 1203–1212.

[2] Al-Shaer, E. S., Hamed, H. H. (2004). Discovery of Policy Anomalies in Distributed Firewalls. In *IEEE INFOCOM'04*, March.

[3] Al-Shaer, E. S., Hamed, H. H., and Masum, H. (2005). Conflict Classification and Analysis of Distributed Firewall Policies. In *IEEE Journal on Selected Areas in Communications*, 23(10).

[4] Al-Shaer, E. S., Hamed, H. H., and Marrero, W. (2005). Modeling and Verification of IPSec and VPN Security Policies. In *IEEE ICNP'2005*, November.

[5] Cuppens, F., Cuppens-Boulahia, N., Sans, T. and Miège, A. (2004). A formal approach to specify and deploy a network security policy. In *Formal Aspects in Security and Trust*, 203–218.

[6] Cuppens, F., Cuppens-Boulahia, N., and García-Alfaro, J. (2005). Detection and Removal of Firewall Misconfiguration. In *Intl. Conf. on Communication, Network and Information Security*, 154–162.

[7] Gupta, P. (2000). *Algorithms for Routing Lookups and Packet Classification*. PhD Thesis, Department of Computer Science, Stanford University.

[8] Liu, A. X. and Gouda, M. G. (2005). Complete Redundancy Detection in Firewalls. In *Proceedings of 19th Annual IFIP Conference on Data and Applications Security*, 196–209.