Automatic Detection of Anomalies for Efficient Firewall Policies

Jalal M. Al-Frihat

Ministry of Education P.O.Box 1646, Amman, Jordan.
Email: jl_frihat@yahoo.com

Abstract—Firewalls are safety-critical systems that secure most private networks. An error in a firewall either leaks secret information from its network or disrupts legitimate communication between its network and the rest of the Internet. Firewall filtering rules have to be carefully written and organized in order to correctly implement the security policy. In this paper, advanced techniques that provide automatic discovery of firewall policy anomalies and anomaly-free policy editing for rule insertion and modification are presented. These techniques significantly simplifies the management of firewall policy written as filtering rules, while minimizing network vulnerability due to wrong configurations of the firewall rules.

1. INTRODUCTION

With the global Internet connection, network security has gained significant attention in the research and industrial communities. Due to the increasing threat of network attacks, firewalls have become important integrated elements not only in enterprise networks but also in small-size and home networks. A firewall is a security guard placed at the point of entry between a private network and the outside Internet so that all incoming and outgoing packets have to pass through it. A packet can be viewed as a tuple with a finite number of fields; examples of these fields are source/destination IP address, source/destination port number, and protocol type. By examining the values of these fields for each incoming and outgoing packet, a firewall accepts legitimate packets and discards illegitimate ones according to its configuration. A firewall configuration defines which packets are legitimate and which are illegitimate. An error in a firewall configuration means a wrong definition of being legitimate or illegitimate for some packets, which will either allow unauthorized access from the outside Internet to the private network, or disable some legitimate communication between the private network and the outside Internet. Neither case is desirable. How to design a correct firewall configuration is therefore an important security issue. Firewalls have been the frontier defense for secure networks against attacks and unauthorized traffic by filtering out unwanted network traffic coming into or going from the secured network. The filtering decision is taken according to a set of ordered filtering rules written based on predefined security policy requirements. Although deployment of firewall technology is an important step toward securing our networks, the complexity of managing firewall policy might limit the effectiveness of firewall security. A firewall policy may include anomalies, where a packet may match with two or more different filtering rules. When the filtering rules are defined, serious attention has to be given to rule relations and interactions in order to determine the proper rule ordering and guarantee correct security policy semantics. As the number of filtering rules increases, the difficulty of writing a new rule or modifying an existing one also increases. It is very likely, in this case, to introduce conflicting rules such as one general rule shadowing another specific rule, or correlated rules whose relative ordering determines different actions for the same packet. In addition, a typical large-scale enterprise network might involve hundreds of rules that might be written by different administrators in various times. This significantly increases the potential of anomaly occurrence in the firewall policy, jeopardizing the security of the protected network [1]. Therefore, the effectiveness of firewall security is dependent on providing policy management techniques and tools that enable network administrators to analyze...
and verify the correctness of written firewall legacy rules.

In this paper, a formal model for firewall rule relations and their filtering representation is defined. This model is used to develop an anomaly discovery algorithm to report any anomaly that may exist among the filtering rules. Although firewall security has been given strong attention in the research community, the emphasis was mostly on the filtering performance issues [2]–[4]. On the other hand, a few related works [5], [6] attempt to address only one of the conflict problems which is the rule correlation in filtering policies. Other approaches [7]–[9] propose using a high-level policy language to define and analyze firewall policies and then map this language to filtering rules. Although using such high-level languages might avoid rule anomalies, they are not practical for the most widely used firewalls that contain low level filtering rules. This paper is organized as follows. In Section 2 an introduction to firewall operation is presented. In Section 3 the formalization of filtering rule relations is described. In Section 4 the firewall policy anomalies are defined and classified and then the anomaly discovery algorithms are presented. Section 5 contains the conclusions of this paper.

2. FIREWALL FUNDAMENTALS

A firewall is a network element that controls the traversal of packets across the boundaries of a secured network based on a specific security policy. A firewall security policy is a list of ordered rules that define the actions performed on network packets based on specific filtering conditions. A rule is composed of a set of filtering fields (also called network fields) such as protocol type, source and destination IP addresses and ports, as well as an action field. The filtering fields of a rule represent the possible values of the corresponding fields in actual network traffic that matches this rule. Each network field could be a single value or range of values. Filtering actions are either to accept, which permits the packet into or from the secure network, or to deny, which causes the packet to be blocked. A “deny” default policy action is assumed. The most commonly used matching fields are: protocol type, source IP address, source port, destination IP address and destination port [12], [13].

The following is the common format of packet filtering rules in a firewall policy. The order of the rule determines its position relative to other filtering rules in the policy. IP addresses can be a host (e.g. 140.191.35.120), or a network address (e.g. 140.191.35.*). Ports can be either a single specific port number, or any port number indicated by “any.” Some firewall implementations allow the usage of non-wildcard ranges in specifying source and destination addresses or ports. However, it is always possible to split a filtering rule with a multi-value field into several rules each with a single-value field [2]. An example of typical firewall rules is shown in Table 1.

<table>
<thead>
<tr>
<th>order</th>
<th>Protocol</th>
<th>s_ip</th>
<th>s_port</th>
<th>d_ip</th>
<th>d_port</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tcp</td>
<td>140.191.35.20</td>
<td>any</td>
<td><em>.</em>.<em>.</em></td>
<td>80</td>
<td>deny</td>
</tr>
<tr>
<td>2</td>
<td>Tcp</td>
<td>140.191.35.20</td>
<td>any</td>
<td><em>.</em>.<em>.</em></td>
<td>80</td>
<td>accept</td>
</tr>
<tr>
<td>3</td>
<td>Tcp</td>
<td><em>.</em>.<em>.</em></td>
<td>any</td>
<td>161.122.31.40</td>
<td>80</td>
<td>accept</td>
</tr>
<tr>
<td>4</td>
<td>Tcp</td>
<td>140.191.35.20</td>
<td>any</td>
<td>161.122.31.40</td>
<td>80</td>
<td>deny</td>
</tr>
<tr>
<td>5</td>
<td>tcp</td>
<td>140.191.35.30</td>
<td>any</td>
<td><em>.</em>.<em>.</em></td>
<td>21</td>
<td>deny</td>
</tr>
<tr>
<td>6</td>
<td>tcp</td>
<td>140.191.35.30</td>
<td>any</td>
<td><em>.</em>.<em>.</em></td>
<td>21</td>
<td>accept</td>
</tr>
<tr>
<td>7</td>
<td>tcp</td>
<td>140.191.35.30</td>
<td>any</td>
<td>161.122.31.40</td>
<td>53</td>
<td>accept</td>
</tr>
<tr>
<td>8</td>
<td>udp</td>
<td>140.191.35.30</td>
<td>any</td>
<td>161.122.31.40</td>
<td>53</td>
<td>accept</td>
</tr>
<tr>
<td>9</td>
<td>udp</td>
<td><em>.</em>.<em>.</em></td>
<td>any</td>
<td>161.122.31.40</td>
<td>53</td>
<td>accept</td>
</tr>
</tbody>
</table>

3. FIREWALL POLICY MODELING

Modeling of firewall rule relations is necessary for analyzing the firewall policy and designing management techniques such as anomaly discovery and policy editing. In this section, the model of firewall rule relations is described. To be able to build a useful model for filtering rules, it is necessary to determine all the relations that may relate packet filters. In this section we define all the possible relations that may exist between filtering rules, and we show that no other relation exists. We determine these relations based on comparing the network fields of filtering rules, independent of the rule actions. In the next section, we consider these relations as well as rule actions in our study of firewall rule conflicts.
Definition 1. Rules $R_x$ and $R_y$ are completely disjoint if every field in $R_x$ is not a subset nor a superset nor equal to the corresponding field in $R_y$. $R_x \not\supseteq R_y$ iff $\forall i, R_x[i] \not\supseteq R_y[i]$, where $\not\supseteq \in \{\subset, \supseteq, =\}$.

Definition 2: Rules $R_x$ and $R_y$ are exactly matching if every field in $R_x$ is equal to the corresponding field in $R_y$. $R_x \equiv R_y$ iff $R_x[i] = R_y[i]$.

where $i \in \{\text{protocol}, \text{s_ip}, \text{s_port}, \text{d_ip}, \text{d_port}\}$

Definition 3: Rules $R_x$ and $R_y$ are inclusively matching if they do not exactly match and if every field in $R_x$ is a subset or equal to the corresponding field in $R_y$. $R_x \supseteq R_y$ iff $\forall i, R_x[i] \subseteq R_y[i]$.

where $i \in \{\text{protocol}, \text{s_ip}, \text{s_port}, \text{d_ip}, \text{d_port}\}$. For example, in Table 1, Rule 1 inclusively matches Rule 2. Rule 1 is the subset match while Rule 2 is the superset match.

Definition 4: Rules $R_x$ and $R_y$ are partially disjoint (or partially matching) if there is at least one field in $R_x$ that is a subset or a superset or equal to the corresponding field in $R_y$, and there is at least one field in $R_x$ that is not a subset and not a superset and not equal to the corresponding field in $R_y$.

$R_x \not\supseteq R_y$ iff $\exists i, j s.t. R_x[i] \not\supseteq R_y[i]$ and $R_x[j] \not\supseteq R_y[j]$.

where $\not\supseteq \in \{\subset, \supseteq, =\}$.

For example, Rule 2 and Rule 6 in Table 1 are partially disjoint (or partially matching).

Definition 5: Rules $R_x$ and $R_y$ are correlated if some fields in $R_x$ are subsets or equal to the corresponding fields in $R_y$, and the rest of the fields in $R_x$ are supersets of the corresponding fields in $R_y$. $R_x \supseteq R_y$ iff $\forall i, R_x[i] \supseteq R_y[i]$ and

$\exists j, k s.t. R_x[j] \subseteq R_y[j]$ and $R_x[k] \supseteq R_y[k]$.

where $\supseteq \in \{\subset, \supseteq, =\}$.

$i, j \in \{\text{protocol}, \text{s_ip}, \text{s_port}, \text{d_ip}, \text{d_port}\}$. For example, Rule 1 and Rule 3 in Fig. 1 are correlated.

The following theorems show that these relations are distinct, i.e. only one relation can relate $R_x$ and $R_y$, and complete, i.e. there is no other relation between $R_x$ and $R_y$ could exist.
Theorem 1: Any two $k$-tuple filters in a firewall policy are related by one and only one of the defined relations.

Theorem 2: The union of these relations represents the universal set of relations between any two $k$-tuple filters in a firewall policy.

The firewall policy is represented by a single-rooted tree called the policy tree. The tree model provides a simple representation of the filtering rules and at the same time allows for easy discovery of relations and anomalies among these rules. Each node in a policy tree represents a network field, and each branch at this node represents a possible value of the associated field. Every tree path starting at the root and ending at a leaf represents a rule in the policy and vice versa. Rules that have the same field value at a specific node will share the same branch representing that value.

Fig. 2 illustrates the policy tree model of the filtering policy given in Table 1. Every rule should have an action leaf in the tree.

The basic idea for building the policy tree is to insert the filtering rule in the correct tree path. When a rule field is inserted at any tree node, the rule branch is determined based on matching the field value with the existing branches. If a branch exactly matches the field value at a specific node will share the same branch representing that value.

4. FIREWALL ANOMALY DISCOVERY

The ordering of filtering rules in a centralized firewall policy is crucial in determining the filtering policy within this firewall. This is because the packet filtering process is performed by sequentially matching the packet against filtering rules until a match is found. If filtering rules are disjoint, the ordering of the rules is insignificant. However, it is very common to have filtering rules that are inter-related. In this case, if the related rules are not carefully ordered, some rules may never be used because of other rules, resulting in an incorrect policy. Moreover, when the policy contains a large number of filtering rules, the possibility of writing conflicting or redundant rules is relatively high.

An firewall policy anomaly is defined as the existence of two or more filtering rules that may match the same packet or the existence of a rule that can never match any packet on the network paths that cross the firewall. Different anomalies that may exist among filtering rules in one firewall could be classified as follows:

Here, some possible firewall policy anomalies are described.

1) Shadowing anomaly. A rule $R_y$ is shadowed by rule $R_x$ if one of the following conditions holds:

$$R_x[\text{ord}] < R_y[\text{ord}] \quad R_x \mathcal{R}_c R_y \quad R_x[\text{action}] \cdot R_y[\text{action}]$$

2) Correlation anomaly: A rule $R_x$ and rule $R_y$ have a correlation anomaly if the following condition holds:

$$R_x \mathcal{R}_c R_y \quad \text{and} \quad R_x[\text{action}] \cdot R_y[\text{action}]$$

3) Generalization anomaly: A rule is a generalization of a preceding rule if they have different actions, and if the first rule can match all the packets that match the second rule. The rule $R_y$ is a generalization of rule $R_x$ if the following condition holds:

$$R_x[\text{ord}] < R_y[\text{ord}] \quad R_x \mathcal{R}_g R_y \quad R_x[\text{action}] \cdot R_y[\text{action}]$$

Rule 1 is in correlation with Rule 3 in Table 1. The two rules with this ordering imply that all HTTP traffic that is coming from 140.191.35.20 and going to 161.122.31.40 is denied. However, if their order is reversed, the same traffic will be accepted. Correlation is considered an anomaly warning because the correlated rules imply an action that is not explicitly stated by the filtering rules. Therefore, in order to resolve this conflict, we point out the correlation between the rules and prompt the user to choose the proper order that complies with the security policy requirements.

Rule 2 is a generalization of Rule 1 in Fig. 1. Generalization is often used to exclude a specific part of the traffic from a general filtering action. It is considered only an anomaly warning because the specific rule makes an exception of the general rule.
This might cause an accepted traffic to be blocked or a denied traffic to be permitted, and thus it is important to highlight its action to the administrator for confirmation.

4) Redundancy anomaly: A rule is redundant if there is another rule that produces the same matching and action such that if the redundant rule is removed, the security policy will not be affected. Formally, rule $R_y$ is redundant to rule $R_x$ if one of the following conditions holds:

$$R_x[\text{ord}] < R_y[\text{ord}] \land R_y[\text{protocol}] \supseteq R_x[\text{protocol}] \land R_x[\text{action}] = R_y[\text{action}]$$

$$R_x[\text{ord}] < R_y[\text{ord}] \land R_y[\text{protocol}] \supseteq R_x[\text{protocol}] \land R_x[\text{action}] = R_y[\text{action}]$$

Referring to Fig. 1, Rule 7 is redundant to Rule 6, and Rule 9 is redundant to Rule 8. Redundancy is considered an error in the firewall policy because a redundant rule adds to the size of the filtering rule list, and therefore increases the search time and space requirements of the packet filtering process [15]. It is important to discover redundant rules so that the administrator can decide whether to keep these rules, modify their filtering actions, or remove them from the policy.

5) Irrelevance anomaly: A filtering rule in a firewall is irrelevant if this rule does match any traffic that may flow through this firewall. This rule has no effect on the filtering outcome of this firewall. Formally, rule $R_x$ in firewall $F$ is irrelevant if:

$$F \not\ni n : s a n o d e o n a p a t h f r o m R_x[\text{src}] t o R_x[\text{dest}]$$

Irrelevance is considered an anomaly because it adds unnecessary overhead to the filtering process and it does not contribute to the policy semantics.

It is assumed that any two rules, $R_x$ and $R_y$, are in the same firewall and $R_y$ follows $R_x$. For simplicity, the address and port fields are integrated in one field for both the source and destination. Initially no relationship is assumed. Each field in $R_y$ is compared to the corresponding field in $R_x$ starting with the protocol, then source address and port, and finally destination address and port. The relationship between the two rules is determined based on the result of subsequent comparisons. If every field of $R_y$ is a subset or equal to the corresponding field in $R_x$ and both rules have the same action, $R_y$ is redundant to $R_x$. While if the actions are different, $R_y$ is shadowed by $R_x$.

If every field of $R_y$ is a superset or equal to the corresponding field in $R_x$ and both rules have the same action, $R_x$ is potentially redundant to $R_y$, while if the actions are different, $R_y$ is a generalization of $R_x$. If some fields of $R_x$ are subsets or equal to the corresponding fields in $R_y$, and some fields of $R_x$ are supersets to the corresponding fields in $R_y$, and their actions are different, then $R_x$ is in correlation with $R_y$. Irrelevance anomalies can be discovered simply by verifying that each rule in the policy matches a source and a destination address that lie on a path controlled by the firewall. If none of the preceding cases occur, then the two rules do not involve any anomalies.

The basic idea for discovering anomalies is to determine if any two rules coincide in their policy tree paths. If the path of a rule coincides with the path of another rule, there is a potential anomaly that can be determined based on the firewall anomaly definitions. If rule paths do not coincide, then these rules are disjoint and they have no anomalies.

The algorithm can be divided into two phases: the state transition phase and the state termination phase. The transition routine is invoked upon inserting every rule in the policy tree. If the field of the current rule matches an already existing rule branch, then the next relation state is determined. The algorithm is executed iteratively to let the rule propagate in existing branches and check the remaining fields. As the rule propagates, the relation state is updated until the final state is reached. If there is no match for a field value, the relation state is set to disjoint.

The termination routine is activated once all the rule fields have been matched and the action field is reached. If the rule action coincides with the action of another rule on the tree, an anomaly is discovered. At that point the final anomaly state is determined and any anomalies are reported together with the rules involved.

Applying the algorithm on the rules from Table 1, the discovered anomalies are marked in the dotted boxes at the bottom of the policy tree in Figure 2. Shadowed rules are marked with a triangle, redundant rules with a square, correlated rules with a pentagon and generalization rules with a circle.

The firewall anomaly discovery algorithm has as inputs rule and branch and as output the anomaly and has the following structure:

```plaintext
for each field \( \in \text{rule.fields} \) do
  if field \( \neq \text{ACTION} \) then
    { find transition states algorithm }
  else
    { find termination state algorithm }
end for
```
The transition states algorithm is as follows:

\[
\text{relation} \leftarrow \text{UNDETERMINED}
\]

if \( \text{branch} = \text{field} \) then \{exact match\}

if \( \text{relation} = \text{UNDETERMINED} \) then
\[
\text{relation} \leftarrow \text{EXACT}
\]
end if

else if \( \text{field} \supset \text{branch} \) then \{superset match\}
if \( \text{relation} \in \{\text{SUBSET}, \text{CORRELATED}\} \) then
\[
\text{relation} \leftarrow \text{CORRELATED}
\]
else if \( \text{relation} \neq \text{DISJOINT} \) then
\[
\text{relation} \leftarrow \text{SUPERSET}
\]
end if

else if \( \text{field} \in \text{branch} \) then \{subset match\}
if \( \text{relation} \in \{\text{SUPERSET}, \text{CORRELATED}\} \) then
\[
\text{relation} \leftarrow \text{CORRELATED}
\]
else if \( \text{relation} \neq \text{DISJOINT} \) then
\[
\text{relation} \leftarrow \text{SUBSET}
\]
end if
else / not matching\}
\[
\text{relation} \leftarrow \text{DISJOINT}
\]
end if

\[
\text{branch} \leftarrow \text{branch.next}
\]

The termination state algorithm has the following structure:

\[
\text{anomaly} \leftarrow \text{NOANOMALY}
\]

if \( \text{relation} \neq \text{DISJOINT} \) then

if \( \text{relation} \in \{\text{CORRELATED} \text{ and field} \neq \text{branch}\} \) then
\[
\text{anomaly} \leftarrow \text{CORRELATION}
\]
else if \( \text{relation} = \text{SUPERSET} \) then

if \( \text{field} = \text{branch} \) then \{similar actions\}
\[
\text{anomaly} \leftarrow \text{REDUNDANCY}
\]
else \{different actions\}
\[
\text{anomaly} \leftarrow \text{GENERALIZATION}
\]
end if

else if \( \text{relation} \in \{\text{EXACT}, \text{SUBSET}\} \) then

if \( \text{field} = \text{branch} \) then \{similar actions\}
\[
\text{anomaly} \leftarrow \text{REDUNDANCY}
\]
else \{different actions\}
\[
\text{anomaly} \leftarrow \text{SHADOWING}
\]
end if
end if
end if
end if

5. CONCLUSIONS

Firewall security, like any other technology, requires proper management to provide the proper security service. Thus, just having a firewall on the boundary of a network may not necessarily make the network any secure. One reason of this is the complexity of managing firewall rules and the potential network vulnerability due to rule conflicts.

In this paper all possible firewall rule relations were defined and used to classify firewall policy anomalies. The firewall rule information and relations were modelled in a tree-based representation. Based on this model and formalization, were developed algorithms for anomalies detection. The future work includes extending the proposed anomaly discovery techniques to handle distributed firewall policies.

REFERENCES