FIREMAN

A Toolkit for
FIREwall Modeling and ANalysis

Problem

Firewalls examine each rule sequentially and accept (or reject) a packet as soon as it matches a rule. Hence, a preceding rule may shadow subsequent rules. It is difficult to analyze the interactions among a large number of rules.

Firewall rules are written in platform-specific, low-level languages, which makes it difficult to analyze whether these rules have implemented a network's high-level security policies accurately.

In large enterprises, firewalls are deployed on multiple network components. Due to dynamic routing, a packet from the same source to the same destination may be examined by a different set of firewalls at different times. We cannot reason that all these sets of firewalls satisfy the end-to-end security policies of the enterprise.

What does FIREMAN do?

FIREMAN uses Static Analysis

To discover and remove serious vulnerabilities before firewalls are deployed.

To discover all the instances of many known types of misconfigurations, because it can examine every path in the firewall efficiently.

To discover vulnerabilities resulting from interaction among firewalls in complex netwrok topologieswithout the need to configure any routers.

Why Static Analysis and not Testing?

Impractical to process all possible packets because address space of packets is enormous.

Inspiration

Al-Shaer and Hamed proposed an algorithm to detect common pairwise inconsistencies in firewall rules. FIREMAN is inspired by them, but can detect a much wider class of misconfigurations.

What does FIREMAN do?

FIREMAN discovers 2 classes of misconfigurations

(1) Violations of user-specified security policies
(2) Inconsistencies and inefficiencies among firewall rules

FIREMAN's Major Contributions

Comprehensive classification of firewall misconfigurations for single and distributed firewalls.

Static analysis algorithm to examine firewall rules for policy violations and inconsistencies at intra-firewall, inter-firewall and cross-path levels.

Implementation of the above algorithm based on BDDs.

Modelling Firewalls

Individual Firewalls

Firewall interfaces are configured with several access control lists (ACLs). Each ACL consists of a list of rules. Individual rules can be interpreted in the form <P; action>, where P is a predicate describing what packets are matched by this rule and action describing the corresponding action performed on the matched packets.

1

Simple List Model --> Only “accept” or “drop” actions.

2

Complex Chain Model --> In addition to “accept” and “drop”, also supports calling upon another user-defined chain or “return.”

Modelling Firewalls

Network of Firewalls

A network depends on the correct configuration of all related firewalls to achieve the desired end-to-end security behaviour.
There exist multiple paths from the Internet to the internal network. Care has to be taken to ensure consistency among all such paths to the same destination.

What are misconfigurations?

Policy Violation

Administrators have a high-level policy describing what should be prohibited (blacklists) or ensured (whitelist) access to the network. Firewall configurations must exactly reflect the security policy. Nonconforming configurations may result in undesired blocking, unauthorized access, or even the potential for an unauthorized person to alter security configurations.

Inconsistency

Firewall configurations represent the administrator’s intention, which should be consistent. Inconsistencies are often good indicator of misconfigurations. Checking for inconsistencies is solely based on the configuration files and does not need external input. Inconsistencies happen at three levels: intra-firewall, inter-firewall, and cross-path.

Inefficiency

An efficient firewall configuration should require the minimum number of rules, use the least amount of memory, and incur the least amount of computational load while achieving the same filtering goals. A faster and more efficient firewall will encourage firewall deployment and therefore makes the network safer. Efficiency also determines a network’s responsiveness to Denial-of-Service (DoS) attacks.

Inconsistency

1 Intra-firewall Inconsistency

- Shadowing
  refers to the case where all the packets one rule intends to drop (accept) have been accepted (dropped) by preceding rules. This often reveals a misconfiguration and is considered an “error.”

- Generalization
  refers to the case where a subset of the packets matched to this rule has been excluded by preceding rules. It is the opposite of shadowing and happens when a preceding rule matches a subset of this rule but takes a different action.

- Correlation
  refers to the case where the current rule intersects with preceding rules but specifies a different action. The predicates2 of these correlated rules intersect, but are not related by the superset or subset relations. The decision for packets in the intersection will rely on the order of the rules.

Inconsistency

2 Inter-firewall Inconsistency

When firewalls are chained together, a packet has to survive the filtering action of all the firewalls on its path to reach its destination. A downstream firewall may rely on upstream firewall to achieve policy conformance and can be configured loosely. But a downstream firewall at the inner perimeter needs a tighter security policy.

Without consulting the administrator, the only inter-firewall inconsistency a tool writer can classify as an “error” is shadowed accept rules wherein a packets permitted at a downstream firewall have already been filtered out at an upstream firewall. To the downstream users, this may manifest as a connectivity problem.

Inconsistency

3

Cross-path Inconsistency

Cross-path inconsistency refers to the case where some packets dropped on one path are accepted through another path.

It depends on the underlying routing table whether these anomalies are exploitable. Cross-path inconsistencies may also manifest as intermittently disruptive services. Packets originally reachable to the network may switch over to another path that drops such packets because of routing changes.

Inefficiency

- Redundancy
  refers to the case where if a rule is removed, the firewall does not change its action on any packets. Reducing redundancy can reduce the total number of rules, and hence also reduces memory consumption and packet classification time.
  Redundant accept or drop rules are “errors” within the same firewall. This is, however, not true in distributed firewalls. A packet must be accepted on all the firewalls on its path to reach the destination. Redundant drop rules on different firewalls are unnecessary, but are often considered good practice to enhance security. This redundancy provides an additional line of defense if the outer-perimeter is compromised.

- Verbosity
  refers to the case where a set of rules may be summarized into a smaller number of rules.

Step 1

Parsing and Flow Graph Analysis

Step 2

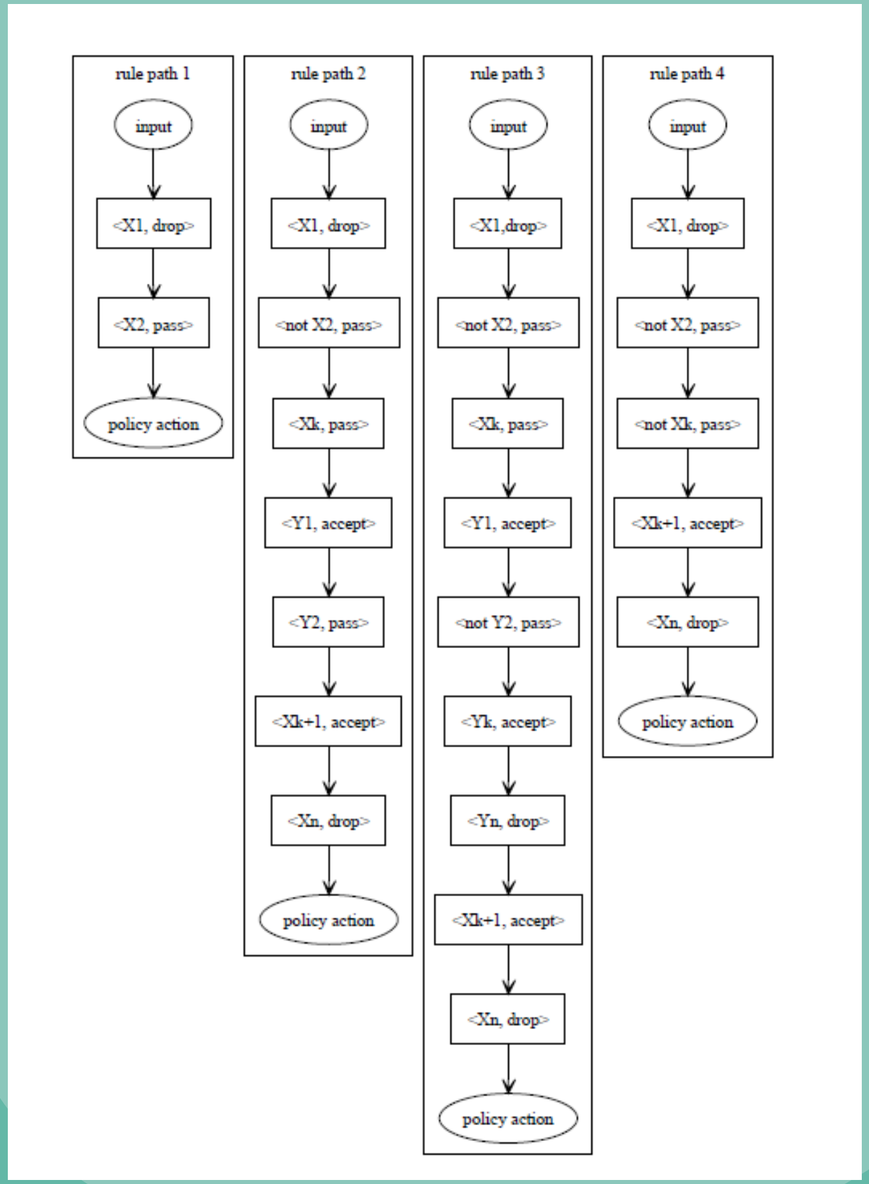
Checking for Anomalies

Step 3

Formal Properties and Discussions

Problem Formulation and Checking Algorithm

STEP 1: Parsing and Flow Graph Analysis



1. Parse and translate firewall configuration files originally written in their own languages into a uniform internal representation.
2. Based on the configuration, network topology and routing information, we perform control-flow analysis to find all possible rule paths packets may go through. Each path represents a list of filtering operations packets may receive.

Rule graph of individual ACLs

There is no possibility of branching in the simple list model and the rule graph is the same list. In the complex chain model, branching can be caused by calling “chain Y” and “return” from it. To handle such branching, we introduce <P, pass> to indicate that only packets matching this predicate will remain in this path. For a <P, chain Y> rule, we insert <P, pass> before going to “chain Y”. We also insert <~P, pass> for the path that does not jump to “chain Y”.

Rule graph of individual ACLs

For an ACL using the complex chain model, the rule graph may give n rule paths from the input to the output. For each of the n rule paths, we traverse the path to collect information.

I = input to an ACL, i.e. the collection of packets that can arrive at this access list. For the jth rule <Pj, actionj> in this rule path, we define the current state as

<Aj, Dj, Fj>

where Aj and Dj denote the network traffic accepted and denied before the jth rule, respectively and Fj denotes the set of packets that have been diverted to other data paths. We use Rj to denote the collection of the remaining traffic that can possibly arrive at the jth rule. Rj can always be found using the input I and the current state information, as shown.

$$R_j = I \cap \neg(A_j \cup D_j \cup F_j)$$

For the first rule of an ACL, we have the initial value of A1 = D1 = F1 = 0; and R1 = I. After reading each rule, we update the state according to the state transformation defined in the above equation until the end of each rule path.

$$\begin{aligned} A_{ACL} &= \bigcup_{i \in path} A_{path_i} \\ D_{ACL} &= \bigcup_{i \in path} D_{path_i} \\ A_{ACL} \cup D_{ACL} &= I_{ACL} \\ R_{ACL} &= \emptyset \end{aligned}$$

$$\begin{aligned} \langle A, D, F\rangle, \quad \langle P,\ \text{accept}\rangle \quad &\vdash \langle A \cup (R \cap P),\ D,\ F\rangle \\ \langle A, D, F\rangle, \quad \langle P,\ \text{drop}\rangle \quad &\vdash \langle A,\ D \cup (R \cap P),\ F\rangle \\ \langle A, D, F\rangle, \quad \langle P,\ \text{pass}\rangle \quad &\vdash \langle A,\ D,\ F \cup (R \cap \neg P)\rangle \end{aligned}$$

ACL graph of Distributed Firewalls

Here, a packet will go through a series of ACLs to reach the destination and has to survive the filtering of all the ACLs on the path. As a result of dynamic routing a packet could traverse different ACL paths at different times.

The topology can be represented as a directed graph, to determine all possible paths from one node to another. A tree of ACLs can thus be built, rooted at a destination using either DFS or BFS algorithms. For the portions of network that are not involved in packet filtering, and therefore do not interfere with the firewall configurations, abstract virtual nodes are used as representations.

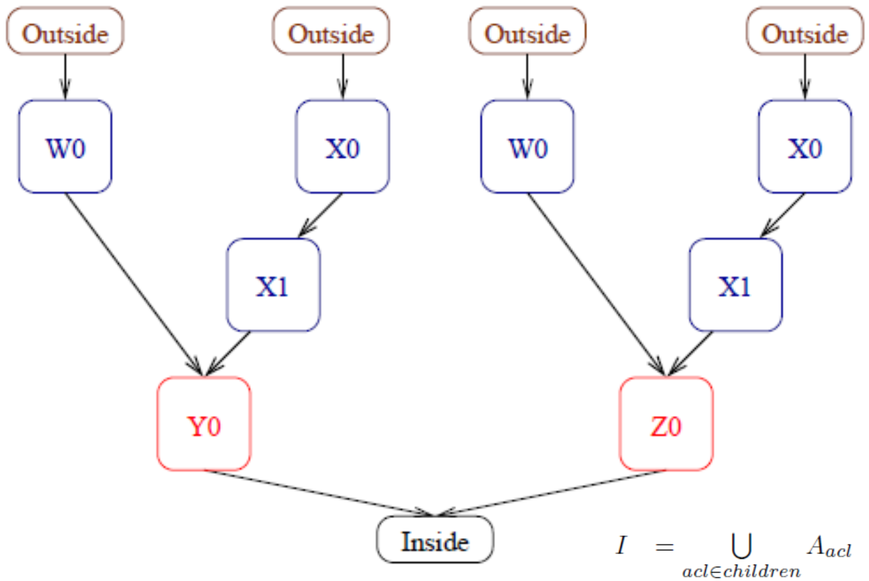


Fig. 4: ACL tree

Firewalls in Series:

$$A = \bigcap_{acl \in n} A_{acl}$$
$$D = \bigcup_{acl \in n} D_{acl}$$

Firewalls in Parallel:

$$A = \bigcup_{acl \in n} A_{acl}$$
$$D = \bigcap_{acl \in n} D_{acl}$$

STEP 2: Checking for Anomalies

Local Check for Individual Firewalls

Since a firewall can rely on the filtering action of other firewalls to achieve policy conformance, local checks focus on checking inconsistency and inefficiency. The local check is performed after parsing each rule, and just before updating the state

- For $\langle P,$ accept$\rangle$ rules:
  1) $P_j \subseteq R_j \Rightarrow$ good: This is a good rule. It defines an action for a new set of packets, and it does not overlap with any preceding rules.
  2) $P_j \cap R_j = \emptyset \Rightarrow$ masked rule: This is an "error". This rule will not match any packets and action defined here will never be taken.

  a) $P_j \subseteq D_j \Rightarrow$ shadowing: This rule intended to accept some packets which have been dropped by preceding rules. This contradiction reveals a misconfiguration.
  b) $P_j \cap D_j = \emptyset \Rightarrow$ redundancy: All the packets have been accepted by preceding rules or will not take this path.
  c) else $\Rightarrow$ redundancy and correlation: Part of the packets for this rule have been dropped. Others are either accepted or will not take this path. Rule j itself is redundant since it will not match any packets. Some preceding rule has correlation with rule j also.

3) $P_j \nsubseteq R_j$ and $P_j \cap R_j \neq \emptyset \Rightarrow$ partially masked rule
   a) $P_j \cap D_j \neq \emptyset \Rightarrow$ correlation: Part of the packets intend to be accepted by this rule have been dropped by preceding rules. This raises a "warning".
   b) $\forall\ x < j, \exists\ \langle P_x,$ drop$\rangle$ s.t. $P_x \subseteq P_j \Rightarrow$ generalization: Rule j is a generalization of rule x since rule x matches a subset of the current rule j but defined a different action. This is a "warning".
   c) $P_j \cap A_j \neq \emptyset$ and $\forall\ x < j, \exists \langle P_x,$ accept$\rangle$ s.t. $P_x \subseteq P_j \Rightarrow$ redundancy: If rule $\langle P_x,$ accept$\rangle$ is removed, all the packets that match P_x can still be accepted to the current $\langle P_j,$ accept$\rangle$. Therefore, rule $\langle P_x,$ accept$\rangle$ is redundant. This is an "error".

STEP 2: Checking for Anomalies

Checks for Distributed Firewalls

After passing the local checks, FIREMAN performs distributed checks for network of firewalls based on the ACL-tree derived from the network. Starting at the top level ACLs of the tree, FIREMAN traverses the tree downwards level by level.

- For $\langle P,$ accept$\rangle$ rules:
  1) $P \subseteq I \Rightarrow$ *good*: This is not a redundancy as in the case of local checks. A packet need to be permitted by *all* firewall on its path to reach destination.
  2) $P \subseteq \neg I \Rightarrow$ shadowing: This rule is shadowed by upstream ACLs. It tries to accept some packets that are blocked by *all* upstream firewalls. This kind of inconsistency can manifest as connectivity problems which are difficult to troubleshoot manually.
- For $\langle P,$ drop$\rangle$ rules:
  1) $P \subseteq I \Rightarrow$ raised security level?: This probably reveals a raised security level. In the case of Figure 2, certain packets might be allowed to access the DMZ but not the internal network. Therefore, ACLs $W0$, $X1$ and $X0$ will permit these packets but ACL $Y0$ will drop them.
  2) $P \subseteq \neg I \Rightarrow$ redundancy?: This is probably a redundancy since the packets to be dropped will not reach this ACL anyway. However, multiple lines of defense are often encouraged in practice to increase overall security level. This should be performed with caution by the administrator.

Checks at the Root of the ACL Tree

The root of the ACL tree is the destination, which is also the network we want to secure. We want to ensure that all the inputs to the root are the same. Otherwise, this is a "cross-path inconsistency".

Assume the root has m children, and child j gives input Ij to the root.

$$\forall j \in m, I_j = I$$

- $I \cap$blacklist $\neq \emptyset \Rightarrow$ policy violation: The firewalls permit some packets forbidden by the stated policy. This is a security violation.
- whitelist $\not\subseteq I \Rightarrow$ policy violation: The firewalls drop some protected packets. This causes disrupted service.

Formal Properties and Discussions

Theorem 1 (Soundness and Completeness)

FIREMAN's checking algorithm is both sound and complete:

1. If the algorithm detects no misconfigurations, then there will not be any misconfigurations (soundness).
2. Any misconfiguration detected by the algorithm is real misconfiguration (completeness).