FIREMAN: A Toolkit for FIREwall Modeling and ANalysis

Michael Lin
All about firewalls

• A firewall is only as good as its configuration

• Big deal, it should be easy to configure a firewall, right?

• Basically... no. “A Quantitative Study of Firewall Configuration Errors” showed that 100% of firewalls in a study were vulnerable due to bad configuration

• Firewall rules are easy to manage for simple rulesets, but as the rulesets grow, the configurations grow too
What can we do about this?

• Manually analyzing firewall rules is tedious and error-prone...

• Static analysis to the rescue!

• Firewall rules are ideal for static analysis: simple and logic-based

• FIREMAN is a system for doing static analysis on firewall rulesets and automatically finding configuration errors
What are configuration errors?

- 3 types of “errors”
  - Policy violation
    - The configuration violates the high-level semantics of the rules
  - Inconsistency
  - Inefficiency
Inconsistencies

• Shadowing
  
  • An earlier rule makes a later rule impossible, ie. “deny all” “accept tcp 192.168.0.0/16 any”

• Generalization
  
  • An earlier rule is a subset of a later rule which does the opposite thing

• Correlation
  
  • Two rules cover the same packets with different actions

• Generalization and correlation can be intentional, so they are classified as warnings rather than errors
Analyzing firewall rules

• How do we formally represent a firewall ACL?

• Rule graphs
  • Good for individual firewalls, but for networks with multiple firewalls we need a little more information

• ACL trees
  • Abstracted representations of all relevant nodes in a network
Formalizing correctness

- Important variables:
  - $A_j$ - All packets accepted before the $j$th rule
  - $D_j$ - All packets denied before the $j$th rule
  - $F_j$ - All packets sent to a different rule path before the $j$th rule
  - $R_j$ - All packets not covered by the above by the $j$th rule
  - $I$ - input
Formalizing correctness

• Using the previously defined variables and errors, we can make formal logic statements about firewall rules

• Some examples for <P, accept> rules:

  • \( P_j \subseteq R_j \Rightarrow \text{good} \) - This is a “good” rule

  • \( P_j \subseteq D_j \Rightarrow \text{shadowing} \)

  • \( P_j \cap D_j \neq \emptyset \Rightarrow \text{correlation} \)
Example

- $P_1$: accept tcp 192.168.0.0/16 any
  $P_2$: deny tcp 192.168.0.3/32 any

- $A_1 = 192.168.0.0/16$
  $D_1 = \emptyset$
  $R_1 = \text{All packets} - 192.168.0.0/16$

- $A_2 = 192.168.0.0/16$
  $D_2 = 192.168.0.3/32$
  $R_2 = \text{All packets} - 192.168.0.0/16$

- $P_2 \subseteq A_2 \Rightarrow \text{shadowing!}$
Bringing it all together...

• In more complex firewalled networks, there might be separate logic trees for each firewall

• We need some way to ensure that configurations are policy-consistent across the network...

• \( \forall j \in m, l_j = l \) - for a network with \( m \) firewalls, the input from the \( j \)th firewall should match all other inputs

• \( l \cap \text{blacklist} \neq \emptyset \Rightarrow \text{policy violation} \)

• \( \text{whitelist} \not\subset l \Rightarrow \text{policy violation} \)
So how well does it work?

- Represented firewall rule graphs using binary decision diagrams, implemented and tested on 3 (only?) real-life firewalls

- Found previously unnoticed policy violations in all 3, inconsistencies and inefficiencies in some

- Performance isn’t a huge issue since this is an offline analysis, but it turns out to not be too bad: $O(n)$

- So FIREMAN does a good job of analyzing firewalls, and static analysis is demonstrably effective at finding firewall configuration errors
Take away

FIREMAN rocks!
Really, though...

- Avi Wool’s study covered 37 firewalls and found problems in every one of them

- This covered 3 and found problems in every one of them

- A lot of work in this area is actually about *managing* firewalls...
Take away

Ease of use is just as important as technical soundness