Fast and Scalable Method for Resolving Anomalies in Firewall Policies

Hassan Gobjua
Verizon

Kamal Ahmat
City University of New York
Introduction

- Firewalls
- Types of Anomalies
- Related Work
- Data Structure and Algorithm
- Experimental Results
- Conclusion
Firewalls

- Firewall
  - System acting as an interface of a network to one or more external networks.
  - Implements the security policy of the network
  - By deciding which packets to let through
    - Based on rules defined by the network administrator.
### Example

<table>
<thead>
<tr>
<th>Id</th>
<th>Protocol</th>
<th>Source IP address</th>
<th>Source Port</th>
<th>Destination IP address</th>
<th>Destination Port</th>
<th>Action</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>TCP</td>
<td>71.123.10.*</td>
<td>any</td>
<td>10.0.0.1</td>
<td>21</td>
<td>permit</td>
<td>0.3</td>
</tr>
<tr>
<td>r₂</td>
<td>TCP</td>
<td><em>.</em>.<em>.</em></td>
<td>any</td>
<td>10.0.0.1</td>
<td>21-23</td>
<td>deny</td>
<td>0.25</td>
</tr>
<tr>
<td>r₃</td>
<td>TCP</td>
<td>71.<em>.</em>.*</td>
<td>any</td>
<td>10.0.0.1</td>
<td>21</td>
<td>permit</td>
<td>0.15</td>
</tr>
<tr>
<td>r₄</td>
<td>TCP</td>
<td>71.123.<em>.</em></td>
<td>any</td>
<td>10.0.0.1</td>
<td>21</td>
<td>deny</td>
<td>0.1</td>
</tr>
<tr>
<td>r₅</td>
<td>TCP</td>
<td><em>.</em>.<em>.</em></td>
<td>any</td>
<td>10.0.0.1</td>
<td>23-25</td>
<td>permit</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Protection Methods

- Firewalls – Firewall policy rules should be designed carefully!

Challenges

- Rules are created by multiple people
- Rules are created over extended period of time
- Number of rules in a firewall policy can be 5K+!
- Rules are dynamic!
Relationships Between Rules - Disjoint Rules

Two rules $r$ and $s$ are disjoint if they have at least one criterion for which they have completely disjoint values.

Example:
- $<\text{IN}, \text{TCP}, 64.233.179.104, 80, 192.168.20.*$, ANY, ACCEPT$>
- $<\text{IN}, \text{TCP}, 64.233.179.104, 80, 172.16.20.*$, ANY, REJECT$>
Two rules $r$ and $s$ are exactly matched if each criterion of the rules match exactly.

Example:
- `<IN, TCP, 64.233.179.104, 80, 192.168.20.*, ANY, ACCEPT>`
- `<IN, TCP, 64.233.179.104, 80, 192.168.20.*, ANY, ACCEPT>`
Two rules \( r \) is a subset, or inclusively matched of another rule \( s \) if there exists at least one criterion for which \( r \)'s value is a subset of \( s \)'s value and for the rest of the attributes \( r \)'s value is equal to \( s \)'s values.

Example:
- \(<\text{IN, TCP, 64.233.179.104, 80, 192.168.20.3, ANY, ACCEPT}>\)
- \(<\text{IN, TCP, 64.233.179.104, ANY, 192.168.20.*, ANY, ACCEPT}>\)
Two rules $r$ and $s$ are correlated if $r$ and $s$ are not disjoint, but neither is the subset of the other.

Example:
- $<\text{IN, TCP, 64.233.179.104, ANY, 192.168.20.3, ANY, ACCEPT}>$
- $<\text{IN, TCP, 64.233.179.104, 80, 192.168.20.*, ANY, REJECT}>$
Existing Work

- E. W. Fulp – O(n^3) algorithm to order rules in a given policy; it doesn't discover correlated ones.
- E. Al-Saher *et al.* – Method for selecting rules based on their probability.
- A. Liu – Method to discover and remove redundant rules (Exact matching).
Our Approach

- We aim at removing few troublesome rules from given policy to resolve anomalies.
- Design a data structure to represent dependencies among rules.
- Remove troublesome rules.
- Return a subset of consistent rules and correlated rules (for editing).
Our Approach

- Design a data structure to represent dependencies among rules.
- Graph D is directed, and U is undirected.
  - Each node in U represents a rule
  - Two nodes are connected in U if there is *shadowing* or *correlation* relationship between these two rules.
- Graph D describes dependency among rules.
Our Approach

- Select a rule that doesn’t depend on any other rule (terminal node) from D.
- Remove corresponding links from U and links/nodes from D.
- If graph U is disconnected and new component formed, continue, else there is correlation
- If there is correlation, choose the rule with highest probability.
### Example

<table>
<thead>
<tr>
<th>Id</th>
<th>Protocol</th>
<th>Source IP address</th>
<th>Port</th>
<th>Destination IP address</th>
<th>Port</th>
<th>Action</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>TCP</td>
<td>71.123.10.*</td>
<td>any</td>
<td>10.0.0.1</td>
<td>21</td>
<td>permit</td>
<td>0.3</td>
</tr>
<tr>
<td>$r_2$</td>
<td>TCP</td>
<td><em>.</em>.<em>.</em></td>
<td>any</td>
<td>10.0.0.1</td>
<td>21-23</td>
<td>deny</td>
<td>0.25</td>
</tr>
<tr>
<td>$r_3$</td>
<td>TCP</td>
<td>71.<em>.</em>.*</td>
<td>any</td>
<td>10.0.0.1</td>
<td>21</td>
<td>permit</td>
<td>0.15</td>
</tr>
<tr>
<td>$r_4$</td>
<td>TCP</td>
<td>71.123.<em>.</em></td>
<td>any</td>
<td>10.0.0.1</td>
<td>21</td>
<td>deny</td>
<td>0.1</td>
</tr>
<tr>
<td>$r_5$</td>
<td>TCP</td>
<td><em>.</em>.<em>.</em></td>
<td>any</td>
<td>10.0.0.1</td>
<td>23-25</td>
<td>permit</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Example – Our Approach
Complexity

- \(O(n^2)\) to construct graphs \(D\) and \(U\)
- \(O(2\log n)\) to discover dependencies
- Algorithm complexity \(O(n^2 \log n)\)
Experimental Results

Two sets of test experiments executed:

- Real-life tests: five policies of size 107, 361, 647, 881, and 1385 over a month period on Verizon firewall using the original (non-improved) approach.
- Tests done over the same period using improved approach.

- Five test sets have been executed on synthetic policies of sizes 10K – 30K.
### Experimental Results – Real-Life Policies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>107</td>
<td>43.1</td>
<td>2.7</td>
<td>8.5</td>
<td>63.3%</td>
</tr>
<tr>
<td>2</td>
<td>361</td>
<td>87.2</td>
<td>1.4</td>
<td>2.2</td>
<td>47.2%</td>
</tr>
<tr>
<td>3</td>
<td>647</td>
<td>381.1</td>
<td>3.1</td>
<td>7.9</td>
<td>62.7%</td>
</tr>
<tr>
<td>4</td>
<td>881</td>
<td>341.6</td>
<td>3.3</td>
<td>6.4</td>
<td>71.2%</td>
</tr>
<tr>
<td>5</td>
<td>1385</td>
<td>715.3</td>
<td>3.8</td>
<td>6.7</td>
<td>74.8%</td>
</tr>
</tbody>
</table>
### Experimental Results – Synthetic Policies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10K</td>
<td>4224</td>
<td>121.3</td>
<td>13.5</td>
<td>68.6 %</td>
</tr>
<tr>
<td>2</td>
<td>12.5K</td>
<td>5584</td>
<td>389.6</td>
<td>11.6</td>
<td>40.5 %</td>
</tr>
<tr>
<td>3</td>
<td>15K</td>
<td>8054</td>
<td>274.7</td>
<td>12.0</td>
<td>76.4 %</td>
</tr>
<tr>
<td>4</td>
<td>25.5K</td>
<td>14263</td>
<td>649.2</td>
<td>15.2</td>
<td>79.3 %</td>
</tr>
<tr>
<td>5</td>
<td>30K</td>
<td>17714</td>
<td>712.4</td>
<td>20.7</td>
<td>87.6 %</td>
</tr>
</tbody>
</table>
Current & Future Work

- Find exact minimum number of rules to eliminate all anomalies from policy.
- Modify algorithm to handle dynamic-policies.
- Improve the algorithm performance.
Thank You All!

Questions?