Usage of Iterated Local Search to improve Firewall Evolvability

Author: Geert Haerens

Presenter: Geert Haerens
  Enterprise IT Architect @ Engie
  PhD student at Antwerp University
  geert.haerens@engie.com
Geert Haerens holds a degree in industrial engineering (electricity and automation) and civil engineering (computer science and mechatronics). After having worked for 4 years at the NMBS and 8 years at AB Inbev, he started working for Engie/Electrabel as an IT Architect. In his pursuit for professionalizing the work of the IT Architect, he became a certified EA at the University of Carnegie Mellon and got his Master in Enterprise IT Architect at the Antwerp Management School. In addition to his job at Engie, he is currently doing research at the University of Antwerp on the applicability of the Normalized Systems theory on IT Infrastructure systems.
Content

- Introduction: The TCP/IP Firewall and Normalized Systems
- Problem: Relationships between rules
- Artifact: Previous work and requirements
- Artifact: Construction
- Demonstration
- Evaluation
- Conclusion
Introduction
# Introduction: The TCP/IP Firewall

- **Source:** 1.1.1.1
- **Destination:** 1.1.2.1
- **SSH Service**

**Rule Base**

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Service</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.5.3</td>
<td>1.1.2.1</td>
<td>TCP/22</td>
<td>Deny</td>
</tr>
<tr>
<td>1.1.1.1...</td>
<td>1.1.2.1</td>
<td>TCP/22</td>
<td>Allow</td>
</tr>
<tr>
<td>1.1.1.20</td>
<td>1.1.2.1</td>
<td>TCP/22</td>
<td></td>
</tr>
<tr>
<td>1.1.1.1</td>
<td>1.1.2.1</td>
<td>UDP/10-25</td>
<td>Deny</td>
</tr>
</tbody>
</table>

**Port TCP/22**

- **Source:** 1.1.1.1
- **Destination:** 1.1.2.1
Introducing Normalized Systems
What is it about?

- Studies the evolvability of modular software systems.
- Defines 4 theorems as the necessary conditions a modular structure must adhere to, for evolvability.
- A systems is considered evolvable when it is stable under change.
- Stable under changes = Bounded input leads to Bounded output.

- A limited functional change (bounded input) must lead to a limited change in software modules (bounded output).
- If not, a Combinatorial Effect is observed: change is proportional to the system itself.
The Problem
Problem: Relationships between rules

- Completely disjoint rules
- Inclusively matched rules
- Partially disjoint (or partially matched) rules
- Correlated rules
The Artifact
Artifact: Previous work and requirements

A “Green Field” Artifact:
Enforce disjointness of service definitions – use destination definitions that represent host/service combinations.
Provides and evolvable rule base with respect to anticipated changes.

<table>
<thead>
<tr>
<th>ADD</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a rule</td>
<td>no CE</td>
</tr>
<tr>
<td>new service</td>
<td>no CE</td>
</tr>
<tr>
<td>new host with existing service</td>
<td>no CE</td>
</tr>
<tr>
<td>new host with new service</td>
<td>no CE</td>
</tr>
<tr>
<td>a client</td>
<td>no CE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REMOVE</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a rule</td>
<td>no CE</td>
</tr>
<tr>
<td>a service</td>
<td>no CE</td>
</tr>
<tr>
<td>a host</td>
<td>no CE</td>
</tr>
<tr>
<td>service from a host</td>
<td>no CE</td>
</tr>
<tr>
<td>a client</td>
<td>CE at client level</td>
</tr>
</tbody>
</table>

A “Brown Field” Artifact:
Convert an existing rule base into an evolvable rule base
Necessary condition (not sufficient): disjoint Service Definitions
Artifact: Previous work and requirements

Break Relationships - Disentangling Services

SERVICE LIST

\[ S_1 \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \]

SERVICE LIST

\[ S_1 \begin{array}{c} 1 \\ 2 \\ 3a \begin{array}{c} 4 \\ 3b \begin{array}{c} 5 \\ 4 \end{array} \\ 6 \end{array} \begin{array}{c} 7 \\ 8 \end{array} \end{array} \]

Apply algorithm
Artifact: Iterated Local Search Metaheuristic

REPEAT:
  Do a local search until a local optimum is reached
  Perform a perturbation
UNTILL (Stop Condition)
Artifact: Algorithm components

- **Port Frequencies**: how many times is a port used in service definitions.
- **Disjointness Index**: sum of all Port Frequencies of a service definition, divided by the number of ports.

- **Initial Solution**: a give rule base – the service definitions
- **Neighbourhood**: DI of all service definitions
- **Objective Function**: Sum of the DI’s of all service definitions in the solution
- **Move Type**: Split a service – carve out all existing subgroups
- **Move Strategy**: Split service with highest DI
- **Perturbation**: Split service – according to overlap
- **Stop Conditions**: full neighbourhood searched, full disjointness reached
Artifact: Move – subgroup carve out

Always improves (= decent) the Objective Function = SUM of DI
Artifact: Perturbation – intersect carve out

Sometimes improves (= decent) the Objective Function = SUM of DL
Artifact: Iterated Local Search

While (end conditions not reached)

While (there are still subgroups)

Do a full cave out

Make a perturbation

result
Demonstration
Demonstration

3 Data sets are used:

• A Demo set: including a lot of exceptions

• A Tractebel set: operational firewall connecting a branch office to the company network

• A Engie IT DC set: operational firewall connecting tooling and management systems to client systems
Demonstration – demo set

Start: OF = 110, services = 28
Stop: OF = 34, services = 34
Demonstration – Tractebel set

Engie Tractebel set

Start: OF = 278, services = 79
Stop: OF = 62, services = 62
Demonstration – Engie IT set

Start: OF = 3876, services = 459
Stop: OF = 418, services = 418
5 Evaluation
Evaluation

• Big O = $n^3$

• Splitting services = impacting rules → how much extra rules?

• Essential building block for evolvable rule base creator
  • Destination splitting to be developed.

• Potential Improvement
  • Memory, performance optimizations.

• Global Optimum?
Conclusion
Conclusion

• Splitting Services = applying SoC.
• Resulting in fine grained rule base – fine grained modular structure with low coupling.

• The algorithm works

• The algorithm needs extension:
  • adjust rules → already done – rule base increases with an order of magnitude
  • Redefine destinations → to be done
THANK YOU

gleert.haerens@engie.com

engie.com