Improving Firewall Evolvability with an Iterated Local Search Algorithm

Geert Haerens

Department of Management Information Systems
Faculty of Business and Economics
University of Antwerp, Belgium
and Engie IT — Dir. Digital & Consulting
Email: geert.haerens@engie.be

Herwig Mannaert

Department of Management Information Systems
Faculty of Business and Economics
University of Antwerp, Belgium
Email: herwig.mannaert@uantwerp.be

***Abstract*—The Transmission Control Protocol/Internet Protocol (TCP/IP) based firewall is a notorious non-evolvable system. Changes to the firewall often result in unforeseen side effects, resulting in the unavailability of network resources. The root cause of these issues lies in the order sensitivity of the rule base and hidden relationships between rules. It is not only essential to define the correct rule. The rule must be placed at the right location in the rule base. As the rule base becomes more extensive, the problem increases. According to Normalized Systems, this is a Combinatorial Effect. In previous research, an artifact has been proposed to build a rule base from scratch in such a way that the rules will be disjoint from each other. Having disjoint rules is the necessary condition to eliminate the order sensitivity and thus the evolvability issues. In this paper, an algorithm, based on the Iterated Local Search metaheuristic, will be presented that will disentangle the service component in an existing rule base into disjoint service definitions. Such disentanglement is a necessary condition to transform a non-disjoint rule base into a disjoint rule base. The math behind the algorithm is presented, a demonstration using multiple firewall exports from a real operational environment is provided and the implications of the artifact are discussed.**

***Index Terms*—Firewall; Rule Base; Evolvability; Metaheuristic; Iterated Local Search.**

I. Introduction

This paper is an extended version of [1], and applies the researches discussed in [1] to operational firewall exports provided by Engie (an utilities multinational at which one of the author works). This paper also provides additional insights in the implications of using the research in an operational environment, as discussed in the PhD dissertation of one of the authors [2].

The TCP/IP based firewall has been and will continue to be an essential network security component in protecting network-connected resources from unwanted traffic. The increasing size of corporate networks and connectivity needs has resulted in firewall rule bases increasing considerably. Large rule bases have a nasty side effect. It becomes increasingly difficult to add the right rule at the correct location in the firewall. Anomalies start appearing in the rule base, resulting in the erosion of the firewall's security policy or incorrect functioning. Making changes to the firewall rule base becomes more complex as the size of the system grows. An observation shared by Forrester [3] and the firewall security industry [4] [5]. A more detailed literature review on the topic can be found in [6].

Normalized Systems (NS) theory [7] defines a Combinatorial Effect (CE) as the effect that occurs when the impact of a change is proportional to the nature of the change and the system's size. According to NS theory, a system that suffers from CE is considered unstable under change or non evolvable. A firewall suffers from CE. The evolvability issues are the root cause of the growing complexity of the firewall as time goes by.

The order sensitivity plays a vital role in the evolvability issues of the rule base. The necessary condition to remove the order sensitivity is known, being non-overlapping or disjoint rules. However, firewall rule bases do not enforce that condition, leaving the door open for misconfiguration. While previous work investigates the causes of anomalies [8] [9], detecting anomalies [10] [11] [12] and correcting anomalies at the time of entering new rules in the rule base [10], to the best of our knowledge and efforts, no work was found that tries to construct a rule base with ex-ante proven evolvability (= free of CE). While previous methods are reactive, this work proposes a proactive approach.

Issues with evolvability of the firewall rule base induce business risks. The first is the risk of technical communication paths not being available to execute business activities properly. The second is that flaws in the rule base may result in security issues, making the business vulnerable for malicious hacks resulting in business activities' impediment.

In this paper, we propose an artifact, an algorithm, that aims at converting an existing non-evolvable rule base into an evolvable rule base. Design Science [13] [14] is suited for research that wants to improve things through artifacts (tools, methods, algorithms, etc.). The Design Science Framework (see Figure 1) defines a relevance cycle (solve a real and relevant problem) and rigor cycle (grounded approach, usage of existing knowledge and methodologies).

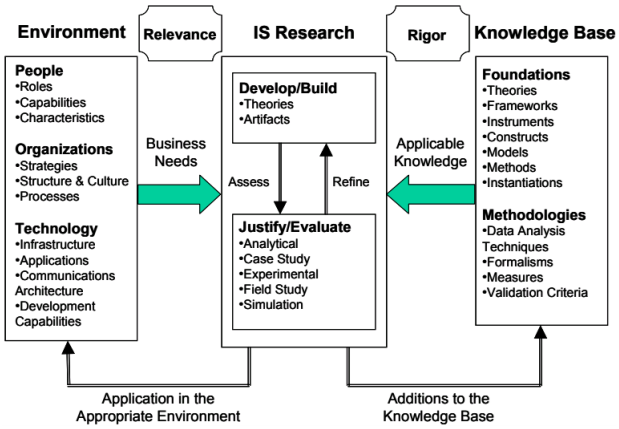

Fig. 1. The Design Science Framework - from [13] .

The Design Science Process (see Figure 2) guides the artifact creation process according to the relevance and rigor cycle. What follows is structured according to the Design Science process.

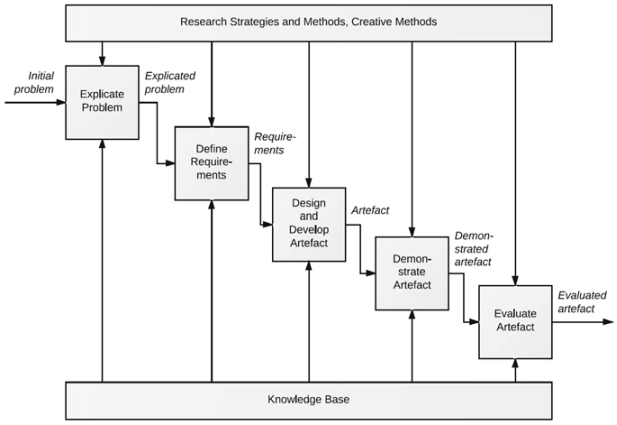

Fig. 2. The Design Science Process - from [14].

Section II introduces the basic concepts of firewalls, firewall rule relationships, Normalized Systems, and the evolvability issues of the firewall. In Section III, we will discuss the requirements for an algorithm that will transform a non-evolvable rule base, into an evolvable rule base. Section IV will build the different components of the proposed algorithm using the Iterated Local Search metaheuristic. In Section V, the algorithm will be demonstrated with real operational data. In Section VI, we evaluate and discuss our findings and we wrap-up with a conclusion in Section VII.

II. Problem Description

The first part of this section will explain how a firewall works and the concept of firewall group objects. The second part will discuss the relationships between firewall rules and introduces the Normalized Systems theory.

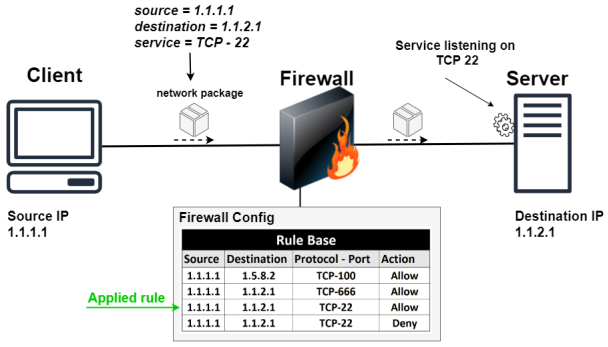

Fig. 3. Firewall concepts.

A. Firewall basics

An Internet Protocol Version 4 (IP4) TCP/IP based firewall, located in the network path between resources, can filter traffic between the resources, based on the Layer 3 (IP address) and Layer 4 (TCP/UDP ports) properties of those resources [15]. UDP stands for User Datagram Protocol and is, next to TCP, a post based communication protocol at the 4th level of the Open Systems Interconnection Model (OSI Model) [16]. Filtering happens by making use of rules. A rule is a tuple containing the following elements: <Source IP, Destination IP, Protocol, Destination Port, Action>. IP stands for IP address and is a 32-bit number that uniquely identifies a networked resource on a TCP/IP based network. The protocol can be TCP or UDP. Port is a 16-bit number (0 - 65.535) representing the TCP or UDP port on which a service is listening on the 4th layer of the OSI-stack.

When a firewall sees traffic coming from a resource with IP address =<Source IP>, going to resource =<Destination IP>, addressing a service listening on Port = <Destination port>, using Protocol = <Protocol>, the firewall will look for the first rule in the rule base that matches Source IP, Destination IP, Protocol and Destination Port, and will perform an action = <Action>, as described in the matched rule. The action can be "Allow" or "Deny". See Figure 3 for a graphical representation of the explained concepts. A firewall rule base is a collection of order-sensitive rules. The firewall starts at the top of the rule base until it encounters the first rule that matches the traffic. In a firewall rule, <Source IP>, <Destination IP>, <Destination Port> and <Protocol> can be one value or a range of values. In the remainder of this paper, protocol and port are grouped together in service (for example, TCP port 58 or UDP port 58 are 2 different services).

The firewalls discussed in this work are stateful, meaning that filtering happens on inbound traffic (towards the destination), but that the same firewall does not require rules to allow the response from the destination to the source. The firewall keeps track of the allowed inbound traffic and by default allows the response toward the source. In a stateless firewall this is not the case. A more elaborate discussion about the impact of inbound and outbound traffic on the evolvability of the firewall can be found in [2].

B. Firewall group objects

Rules containing IP addresses for source/destination and port numbers, are difficult to interpret by humans. Modern firewalls allow the usage of firewall objects, called groups, to give a logical name to a source, a destination, or a port, which is more human-friendly. Groups are populated with IP addresses or ports and can be nested. The groups are used in the definition of the rules. Using groups should improve the manageability of the firewall. See Figure 4 for an example.

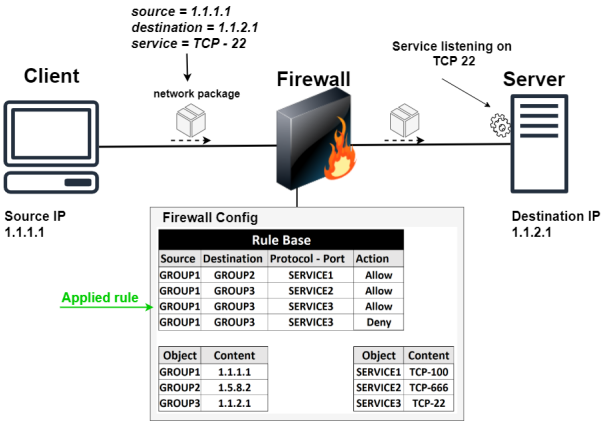

Fig. 4. Firewall concepts, including groups.

C. Firewall rule relationships

Based on [11], the relationships between rules and rule components are defined as follows:

- **Field**: A field in a rule is defined as a source, destination or service. A field is a set of values, with a minimum of size one.
  Example: *The source field of a rule contains 3 IP addresses/values - (10.10.10.1, 10.10.10.2, 10.10.10.3)*
- **Equal Fields**: Two corresponding fields of two rules are equal if the set of values of the fields are the same.
  Example: *The source field of a rule **R1** and source field in rule **R2** contain the same 3 IP addresses - (10.10.10.1, 10.10.10.2, 10.10.10.3)*
- **Inclusive Fields:** Two corresponding fields of two rules are inclusive if the set of values of the field of the first rule are a subset of, but not equal to, the second rule field's set of values.
  Example: *The source field of **R1** contains (10.10.10.1, 10.10.10.2) and the source field of **R2** contains (10.10.10.1, 10.10.10.2, 10.10.10.3). The IPs (10.10.10.1, 10.10.10.2) are a subset of (10.10.10.1, 10.10.10.2, 10.10.10.3). The source field of **R1** is inclusive with regards to the source field of **R2**.*
- **Correlated Fields**: Two corresponding fields of two rules are correlated if there are some values, but not all, of the field of the first rules that are equal to some values, but not all, of the field of the second rule. The intersection between the sets of values of the fields is not empty, but the fields are not equal or inclusive either.
  Example: *The source field of **R1** contains (10.10.10.1, 10.10.10.2, 10.10.10.3) and the source field of **R2** contains (10.10.10.2, 20.20.20.20, 30.30.30.30). The two source fields are correlated as they intersect with the IP 10.10.10.2.*
- **Distinct Fields:** Two corresponding fields in two rule are distinct if they are not equal, not inclusive or not correlated. The intersection between the sets of values of the fields is empty.
  Example: *Source field (10.10.10.10) of rule **R1** and source field (10.10.10.100) of rule **R2** are distinct.*
- **Matching Fields:** Two corresponding fields in two rules match if they are equal or inclusive.
  Example: *Source field of **R1** = (10.10.10.1, 10.10.10.10) and the source field of **R2** = (10.10.10.1, 10.10.10.10, 10.10.10.30), are matching.*
- **Exactly Matching Rules:** Rules **R1** and **R2** are exactly matched if every field in **R1** is equal to the corresponding field in **R2**.
  Example: *Rule **R1**: (source = (10.10.10.10); destination = (20.20.20.20); service = (TPC 100); action = allow) and **R2**: (source = (10.10.10.10); destination = (20.20.20.20); service = (TPC 100); action = deny), are exactly matching rules.*
- **Completely Disjoint Rules:** Rules **R1** and **R2** are completely disjoint if every field in **R1** and **R2** is distinct.
  Example: *Consider rule **R1**: (source = (10.10.10.10); destination = (20.20.20.20); service = (TPC 100); action = allow) and rule **R2**: (source = (30.30.30.30, 30.30.30.21); destination = (40.40.40.40, 40.40.40.41); service = (TPC 200,201); action = deny). Both rules are completely disjoint.*
- **Partially Disjoint Rules or Partially Matching Rules:** Rules **R1** and **R2** are partially disjoint (or partially matched) if there is at least one field in **R1** and **R2** that is distinct. The other fields can be equal, inclusive or correlated.
  Example: *Consider rule **R1**: (source = (10.10.10.10); destination = (20.20.20.20); service = (TPC 100); action = allow) and rule **R2**: (source = (10.10.10.10); destination = (40.40.40.40, 40.40.40.41); service = (TPC 100,201); action = deny). **R1** and **R2** are partially disjoint, as destination is a distinct field.*
- **Inclusively Matching Rules:** Rules **R1** and **R2** are inclusively matched if there is at least one field that is inclusive, and the remaining fields are either inclusive or equal.
  Example: *Consider Rule **R1**: (source = (10.10.10.10); destination = (20.20.20.20); service = (TPC 100); action = allow) and **R2**: (source = (10.10.10.10, 10.10.10.11); destination = (20.20.20.20, 20.20.20.21); service = (TPC 100); action = deny). Then rule **R1** inclusively matches rule **R2**.*
- **Correlated Rules**: Rules **R1** and **R2** are correlated there is at least one field that is correlated, while the remaining fields are either equal or inclusive.

***Example**: Consider rule **R1**: source = (10.10.10.10, 10.10.10.11); destination = (20.20.20.20, 40.40.40.41); service = (TPC 100); action = allow) and rule **R2**: (source = (10.10.10.10); destination = (40.40.40.40, 40.40.40.41); service = (TPC 100,201); action = deny). Rules **R1** and **R2** are correlated.*

Figure 5 represents the different relations in a graphical manner. Exactly matching, inclusively matching and correlated rules can result in the following firewall anomalies [10]:

- *Shadowing Anomaly*: A rule **Rx** is shadowed by another rule **Ry** if **Ry** precedes **Rx** in the policy, and **Ry** can match all the packets matched by **Rx**. The result is that **Rx** is never activated.
- *Correlation Anomaly*: Two rules **Rx** and **Ry** can cause a correlation anomaly if, the rules **Rx** and **Ry** are correlated and if **Rx** and **Ry** have different filtering actions.
- *Redundancy Anomaly*: A redundant rule **Rx** performs the same action on the same packets as another rule **Ry** so that if **Rx** is removed the security policy will not be affected.

A fully consistent rule base should only contain disjoint (completely or partial) rules. In that case, the order of the rules in the rule base is of no importance, and the anomalies described above will not occur [8] [9] [10] . However, due to several reasons such as unclear requirements, a faulty change management process, lack of organization, manual interventions, and system complexity [13], the rule base will include correlated, exactly matching, and inclusively matching rules, and thus resulting in evolvability issues.

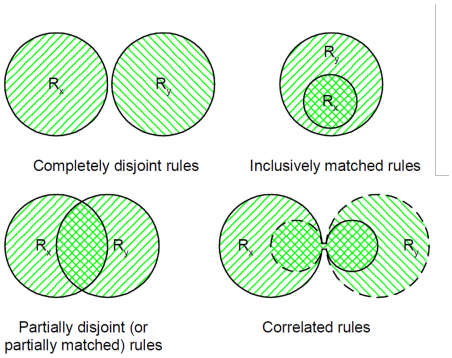

Fig. 5. Possible relationships between rules (from [11]).

D. Normalized Systems concepts

Normalized Systems (NS) theory [7] [17] originates from the field of software development. NS theory takes the concept of system theoretic stability from the domain of classic engineering to determine the necessary conditions a modular structure of a system must adhere to in order for the system to exhibit stability under change. Stability is defined as Bounded Input results in Bounded Output (BIBO). Transferring this concept to software design, one can consider bounded input as a certain amount of functional changes to the software and the bounded output as the number of effective software changes. If the amount of effective software changes is not only proportional to the amount of functional changes but also the size of the existing software system, then NS theory states that the system exhibits a CE and is considered unstable under change.

NS theory proves that, in order to eliminate CE, the software system must have a certain modular structure, were each module respects four design rules. Those rules are:

- Separation of Concern (SoC): a module should only address one concern or change driver.
- Separation of State (SoS): a state should separate the use of a module by another module during its operation.
- Action Version Transparency (AVT): a module, performing an action should be changeable without impacting modules calling this action.
- Data Version Transparency (DVT): a module performing a certain action on a data structure, should be able to continue doing this action, even is the data structures has undergone change (add/remove attributes).

NS theory can be used to study evolvability in any system, which can be seen as a modular system and derive design criteria for the evolvability of such a system [18] [19].

III. Requirements for the Solution

In [1] [2] [6] the necessary conditions for an evolvable firewall rule base are discussed. All the rules in the rule base must be disjoint or partially disjoint from each other. In [6] an artifact, a method, is proposed to create disjoint rules. Following the method will result in a firewall rule base that is free from CE for ADD and REMOVE changes.

For a given network **N**, containing $\mathbf{C_j}$ sources and $\mathbf{H_j}$ destinations, offering 2^{17} services (protocol/port) (= the max amount of possible UDP and TCP ports according to the TCP/IP V4 standard), and having a firewall **F** between the sources and the destinations, it can be shown (see [6]) that $\mathbf{f_{max}}$ is the number of possible rules (including both "allow" and "deny" rules) that can be defined on the firewall **F** (the solution space):

$$\mathbf{f_{max}} = 2.\left(\sum_{a=1}^{Hj}\binom{Cj}{a}\right).\left(\sum_{a=1}^{Hj}\binom{Hj}{a}\right).\left(\sum_{k=1}^{2^{17}}\binom{2^{17}}{k}\right) \quad (1)$$

where $\mathbf{C_j}$ and $\mathbf{H_j}$ are function of **N**: $\mathbf{C_j} = f_c(\mathbf{N})$ and $\mathbf{H_j} = f_h(\mathbf{N})$

Out of this design space, the amount of firewall rules that will exhibit ex-ante proven evolvability are the explicit "allow" rules that are disjoint, and it equal to:

$$\mathbf{f_{disjoint}} = \mathbf{H_j}.2^{17} \quad (2)$$

where $\mathbf{H_j}$ is the number of hosts connected to the network. $\boldsymbol{H_j} = \boldsymbol{fh(N)}$ and 2^{17} the max amount of services available on a host.

Applying the artifact drastically reduced the solution space. The artifact describes the green-field situation; building a rule

base from scratch. The luxury of a green-field is often not present. We require a solution that can convert an existing rule base, into a rule rule base that only contains disjoint rules. Of course, the original filtering strategy expressed in the rule base must stay the same. From [1] [2] [6] we know that we require disjoint service definitions. If we can disentangle the service definitions, and adjust the rules accordingly, we have our basic building block for a disjoint rule base. For each disjoint service definition, we need to create as many destination groups as there are host offering that service (lookup in rule base), and for each host-service combination, we require one source group definition. All components are then present to expand a non-evolvable rule base into a normalized evolvable rule base. Figure 6 visualizes what we want the solution to do.

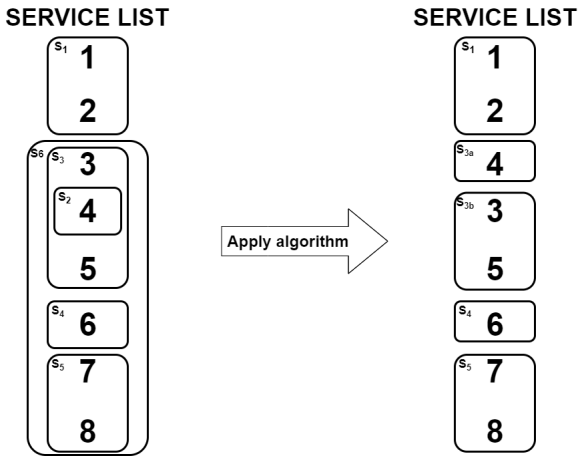

Fig. 6. Algorithm objective.

IV. Artifact Design

In this section we will discuss a different artifact, the brown-field artifact, that will convert a non evolvable rule base into an evolvable rule base, by disentangling the service definitions. The different components that comprise the algorithm will be discussed in dept. We begin by rationalizing the choice for Iterated Local Search (ILS) as metaheuristic [20] [21] [22]. We will discuss the nature of the initial solution, the set of feasible solutions, and the objective function associated with a solution. We continue by defining the move type, move strategy, perturbation and stop condition of the Iterated Local Search. The final part of this section provides a high level algorithm that represents the brown-field artifact.

A. Metaheuristic Selection

The objective is to disentangle/reshuffle the service definitions into a set of new service definitions that are disjoint but maximally large. The simplest solution is to create one service definition per port. However, some ports belong together to deliver a service. This filtering logic is embedded in the rule base and service definitions. It must be preserved.

Service definitions containing ports that appear in multiple service definitions must be split into non-overlapping service definitions. The result should be that the degree of overlap (or disjointness) of all service definitions decreases as more service definitions are split.

Let us say that a user measures the degree of disjointness of the entirety of the service definitions (pre-change and post-change) and then observes a post-change improvement in the degree of disjointness. It would be correct to conclude that the change represents an improvement to the previous version.

A Local Search (LS) heuristic is a suitable method for organizing such gradual improvement processes. To avoid getting stuck in a local optimum (see further), the Local Search will be upgraded to an Iterated Local Search. The Iteration component should result in avoiding becoming stuck in a local optimum where we can no longer perform splits and improve the disjointness. The Iteration component should perform a special kind of split called a ”perturbation” that will allow the continuation of the search for improvement.

B. Initial Solution and Neighborhood

The initial solution is the rule base containing all of the service definitions. It is the rule base with all the service definitions. The set of all service definitions is our neighborhood. We will have to pick a service definition, confirm whether or not it is disjoint and, if not, split it and see how this affects the solution - that is whether or not disjointness has improved. The solution space (SP) for the service definitions consists of all possible combinations of ports. If the number of distinct ports in the service groups equals **P**, then the **SP** is:

$$\mathbf{SP} = \sum_{k=1}^{\mathbf{P}} \binom{\mathbf{P}}{k} \tag{3}$$

P can be max 2^{17}. We are looking to find a new solution that is part of the solution space, in which all service definitions are disjoint yet grouped within groups of maximum size.

C. Objective Function

To know whether or not the splitting of a service definition results in improving the solution, we need a mechanism to express the degree of disjointness of a service definition and of the total rule base.

Let $\boldsymbol{p}$ represent a service port.

Let **S** be a set of ports, representing a service definition.
$\mathbf{S} = \{ \boldsymbol{p}_1 ... \boldsymbol{p}_{nS} \}$
where$|\ \mathbf{S}\ | = nS$ = number of ports in the service definition.

Let σ be the set of all service definitions $\mathbf{S}_i$ used in the firewall rule base.
$\sigma = \{\mathbf{S}_1 ... \mathbf{S}_{n\sigma}\}$
where$|\ \sigma\ | = n\sigma$= number of service definitions

Let $\mathbf{P}\mathrm{F}(\boldsymbol{p}_x)_\sigma$ be the port frequency of port $\boldsymbol{p}_x$ in σ, as

the number of times $\boldsymbol{p}_x$ is used in services of σ.

$$\mathbf{PF}(\boldsymbol{p}_x)\sigma = \sum_{i=1}^{n\sigma} \mid \mathbf{S}_\mathrm{i} \cap \boldsymbol{p}_x \mid \tag{4}$$

We define the Disjointness Index $\mathbf{DI}(\mathbf{S}_\mathrm{x})_\sigma$, of a service definition $\mathbf{S}_\mathrm{x}$, in σ as the sum of the port frequencies $\mathbf{PF}(\boldsymbol{p}_x)_\sigma$ of the ports $\boldsymbol{p}x$ of $\mathbf{S}_\mathrm{x}$, divided by the number of ports in $\mathbf{S}_\mathrm{x}$.

$$\mathbf{DI}(S_\mathrm{x})\sigma = \frac{\sum_{i=1}^{nx} PF(p_x)_\sigma}{nx} \tag{5}$$

where nx = $\mid S_\mathrm{x} \mid$= number of ports in $\mathbf{S_x}$.

A disjoint service is a service whereby each port **p** appears in only one service definition. The **DI** of a disjoint service will have a value of **1** and a value greater then 1 if the service is not disjoint.

We define the Objective Function $\mathbf{OF}_\sigma$, in σ, as the sum of all $\mathbf{DI}(\mathbf{S}_\mathrm{x})_\sigma$ and of all service definitions in σ.

$$\mathbf{OF}\sigma = \sum_{i=1}^{n\sigma} \mathbf{DI}(\mathbf{S}_i)\sigma \tag{6}$$

with $n\sigma$ the number of service definitions in the solution.

We define an Optimal Solution as a solution where $\mathbf{OF}\sigma$ equals the number of service definitions, as this means that all **DI** of all service definitions are equal to 1.

$$\mathbf{OF}\sigma = \mid \sigma \mid \tag{7}$$

An Optimal Solution is not necessarily a Global Optimum as making service definitions of one port would also yield an objective function value that is equal to the total number of service definitions.

D. Feasible Solutions

Whatever kind of splits we will be performing, the original filtering logic of the rule base must be maintained. When a service is split, all rules that contain this service must to be modified. The original service must be replaced by the result of the split. As we want a rule to contain only one service definition, it may be required to split the rules containing the split result.

Example: R1 contains service Sx. Sx is split into Sx1 and Sx2. To reflect this, we replace Sx with Sx1 and Sx2 in rule R1. However, a rule must only contain one service. R1 needs to be split into R1.1 and R1.2, where R1.1 is a copy of R1 but with Sx being replaced by Sx1, and R1.2 is a copy of R1 but with Sx being replaced by Sx2. Both rules are put in consecutive locations in the rule base.

E. Move Type

Before we decide on the move type, we must first investigate the impact that splitting of service definitions has on the objective function. Based on this analysis, a selection of type of split (move type) will be made.

*1) The Impact of Splitting Service Definitions on the **OF**:*

A service definition can:

- be a subset of existing service definitions.
- be the superset of existing service definitions.
- be partially overlapped with other service definitions.
- be a combination of the above.

Let $\mathbf{S}_\mathrm{ca}$ be the candidate service we will split.

$$\begin{cases} \mathbf{S}_\mathrm{ca} = \{\boldsymbol{p}_1 ... \boldsymbol{p}_{nca}\} \\ nca = \mid \mathbf{S}_\mathrm{ca} \mid = \text{number of ports in the } \mathbf{S}_\mathrm{ca} \end{cases}$$

Let $\mathbf{S}_\mathrm{co}$ be an arbitrary set of ports that are part of $\mathbf{S}_\mathrm{ca}$, making up the new service $\mathbf{S}_\mathrm{co}$, that is to be extracted from $\mathbf{S}_\mathrm{ca}$.

$$\begin{cases} \mathbf{S}_\mathrm{co} = \{\boldsymbol{p}_j ... \boldsymbol{p}_{j+nco}\} \\ nco = \mid \mathbf{S}_\mathrm{co} \mid = \text{number of ports in the } \mathbf{S}_\mathrm{co}. \\ \mathbf{S}_\mathrm{ca} \cap \mathbf{S}_\mathrm{co} = \{\boldsymbol{p}_j ... \boldsymbol{p}_{j+nco}\} \\ \mid \mathbf{S}_\mathrm{ca} \cap \mathbf{S}_\mathrm{co} \mid = nco \end{cases}$$

Let $\mathbf{S'}_\mathrm{ca}$ be the new service comprised of ports that are part of $\mathbf{S}_\mathrm{ca}$ but not of $\mathbf{S}_\mathrm{co}$. $\mathbf{S'}_\mathrm{ca}$ is what is left of $\mathbf{S}_\mathrm{ca}$, after splitting-up or carving-out $\mathbf{S}_\mathrm{co}$

$$\begin{cases} \mathbf{S'}_\mathrm{ca} = \mathbf{S}_\mathrm{ca} \backslash \mathbf{S}_\mathrm{co} = \{\boldsymbol{p}_1 ... \boldsymbol{p}_{j-1}, \boldsymbol{p}_{j+nco+1}, ... \boldsymbol{p}_{nca}\} \\ \mid \mathbf{S'}_\mathrm{ca} \mid = nca - nco \end{cases}$$

Let $\sigma_{\mathbf{S}_{ca}}$ be the set of services that contains ports that are also part of service $\mathbf{S}_\mathrm{ca}$.

$$\begin{cases} \sigma_{\mathbf{S}_{ca}} = \{\mathbf{S}_{\mathrm{V}1} ... \mathbf{S}_{\mathrm{V}n}\} \\ \forall \mathbf{S}_{\mathrm{V}x} \ \mathrm{x}{=}1 \rightarrow \mathrm{n} \mid \\ * \ \mid \mathbf{S}_\mathrm{ca} \cap \mathbf{S}_{\mathrm{V}x} \mid \neq \emptyset \\ * \ \mid \mathbf{S}_{\mathrm{V}x} \mid = Vnx \\ * \ \mid \mathbf{S}_\mathrm{ca} \cap \mathbf{S}_{\mathrm{V}x} \mid = \mathrm{q_x} = \text{the amount of port overlap between } \mathbf{S}_\mathrm{ca} \text{ and } \mathbf{S}_{\mathrm{V}x} \end{cases}$$

See Figure 7 for a visual representation of these definitions.

When the split or carve-out of **S**co from **S**ca is performed, the port frequencies, the **DI** and the **OF** change, depending on the effect of the split. We shall now investigate under which conditions the split will improve the objective function.

Let σ_B be the set of services before the split and σ_A be the set of services after the split. We want to know which conditions will improve the Objective Function, or

$$\begin{cases} \Delta\mathbf{OF} = \mathbf{OF}_{\sigma_B} - \mathbf{OF}_{\sigma_A} > 0 \\ \Delta\mathbf{OF} > 0 \text{ means } \mathbf{OF} \text{ improved (=lowered).} \\ \Delta\mathbf{OF} < 0 \text{ means } \mathbf{OF} \text{ deteriorated (=increased).} \end{cases}$$

$\mathbf{S}_\mathrm{co}$ is a random subgroup of $\mathbf{S}_\mathrm{ca}$, meaning not necessarily part of σ_B. Subsequent to a split $\mathbf{S}_\mathrm{ca}$ becomes $\mathbf{S'}_\mathrm{ca}$. Both $\mathbf{S'}_\mathrm{ca}$ and $\mathbf{S}_\mathrm{co}$ are part of σ_A.

There are three possible cases:

- $\mathbf{S'}_\mathrm{ca}$ and $\mathbf{S}_\mathrm{co}$ also exist in σ_B. The split results in two existing services. We merge them into the existing services — the split results in a reduction of the total number of services with 1.
  $\mid \sigma_A \mid$ - $\mid \sigma_B \mid$ = -1

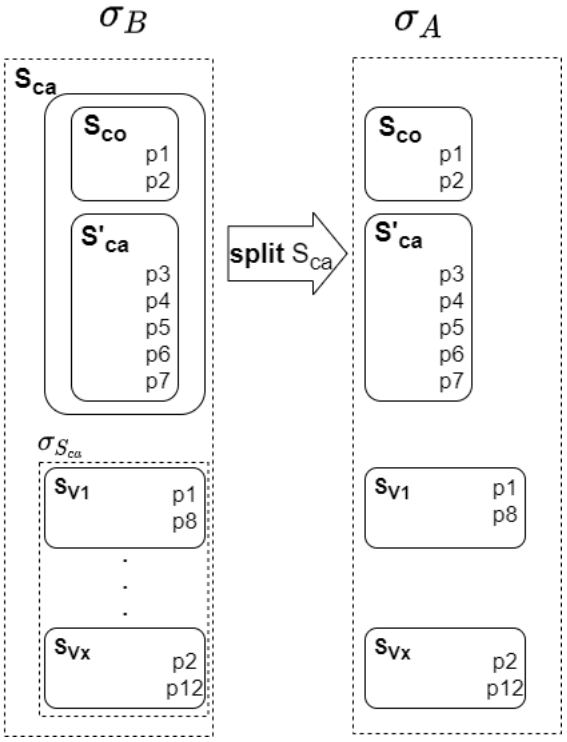

Fig. 7. Split example

- **S'**$_{ca}$ or $\mathbf{S}_{co}$ exists in σ_B. The split results in a new service and an existing service. The existing service merges and the split results in an equal number of services.
  $| \sigma_A | - | \sigma_B | = 0$
- **S'**$_{ca}$ and $\mathbf{S}_{co}$ do not exist in σ_B. The split results in two new services and the split results in an increase of the total number of services with 1.
  $| \sigma_A | - | \sigma_B | = +1$

We shall now investigate what kind of change in Objective Function value we can expect, based on the three following cases.

Case 1: $| \sigma_B | - | \sigma_A | = -1$
S'$_{ca}$ and $\mathbf{S}_{co}$ are elements of σ_A and σ_B.
$\mathbf{S}_{ca}$ only exists in σ_B.
$\mathbf{S}_{ca} = \{p_1 ... p_{nca}\}$
As $\mathbf{S}_{ca}$ is not part of σ_A, the port frequencies of all ports of S_{ca} decreased by 1 in σ_A.

$$\forall \mathbf{p} \in \mathbf{S}_{ca} \mid (\mathbf{PF}(\mathbf{p})_{\sigma_A} = \mathbf{PF}(\mathbf{p})_{\sigma_B} - 1 \tag{8}$$

See Figure 8 for a graphical representation.

As the port frequencies of all ports that are part of $\mathbf{S}_{ca}$ decrease, the **DI** of all groups that contain one or more port of $\mathbf{S}_{ca}$ are also impacted. These are all $\mathbf{S}_{Vi}$ service groups.

When calculating ΔOF, only the impacted service groups must be taken into account.

$$\Delta\mathbf{OF} = \mathbf{OF}_{\sigma_B} - \mathbf{OF}_{\sigma_A}$$

$$\implies \Delta\mathbf{OF} = [\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} + \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} + \mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} + \sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_B}] - [\mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} + \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} + \sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A}]$$

$$\implies \Delta\mathbf{OF} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} + [\mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A}] + [\mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A}] + [\sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_B} - \sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A}]$$

Taking (5) and (8) into account:

(a) $\mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} = \frac{\sum_{i=1}^{nca-nco} \mathbf{PF}(\mathbf{p}_i)_{\sigma_A}}{nca-nco}$

$$\implies \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} = \frac{\sum_{i=1}^{nca-nco} \mathbf{PF}(\mathbf{p}_i)_{\sigma_B} - (nca-nco)}{nca-nco}$$

$$\implies \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} = \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} - 1$$

$$\implies \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} = 1$$

(b) $\mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} = \frac{\sum_{i=1}^{nco} \mathbf{PF}(\mathbf{p}_i)_{\sigma_A}}{nco}$

$$\implies \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} = \frac{\sum_{i=1}^{nco} \mathbf{PF}(\mathbf{p}_i)_{\sigma_B} - (nco)}{nco} = \mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} - 1$$

$$\implies \mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} = 1$$

(c) $\sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_B} - \sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A} =$

$$\sum_{i=1}^{n} \frac{\sum_{j=1}^{nvi} \mathbf{PF}(\mathbf{p}_j)_{\sigma_B}}{nvi} - \sum_{i=1}^{n} \frac{\sum_{j=1}^{nvi} \mathbf{PF}(\mathbf{p}_j)_{\sigma_A}}{nvi} =$$

$$\sum_{i=1}^{n} \frac{\sum_{j=1}^{nvi} \mathbf{PF}(\mathbf{p}_j)_{\sigma_B}}{nvi} - \sum_{i=1}^{n} \frac{\sum_{j=1}^{nvi} \mathbf{PF}(\mathbf{p}_j)_{\sigma_B} - qvi}{nvi}$$

$$\implies \sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_B} - \sum_{i=1}^{n} \mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A} = \sum_{i=1}^{n} \frac{qvi}{nvi}$$

Putting (a), (b) and (c) into ΔOF

$$\Delta\mathbf{OF} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} + 2 + \sum_{i=1}^{n} \frac{qvi}{nvi}$$

Conclusion: If $| \sigma_B | - | \sigma_A |$ ***= -1, then*** Δ***OF is always*** > 0 ***(all terms are positive), and the Objective Function always improves.***

Case 2: $| \sigma_B | - | \sigma_A | = 0$
S'$_{ca}$ or $\mathbf{S}_{co}$ are part of σ_A or σ_B (exclusive OR).
Assume $\mathbf{S}_{co}$ already exists in σ_B (carve-out of an existing group)
$\mathbf{S}_{ca}$ does not exist in σ_A, but **S'**$_{ca}$ does exist in σ_A. The port frequencies of all ports of **S'**$_{ca}$ do not change.

$$\forall \mathbf{p} \in \mathbf{S}_{ca} \setminus \mathbf{S}_{co} \mid \mathbf{PF}(\mathbf{p})_{\sigma_A} = \mathbf{PF}(\mathbf{p})_{\sigma_B} \tag{9}$$

$\mathbf{S}_{co}$ already exists in σ_B. The group cancels out in σ_B and the port frequencies of all ports in $\mathbf{S}_{co}$ decrease.

$$\forall \mathbf{p} \in \mathbf{S}_{co} \mid \mathbf{PF}(\mathbf{p})_{\sigma_A} = \mathbf{PF}(\mathbf{p})_{\sigma_B} - 1 \tag{10}$$

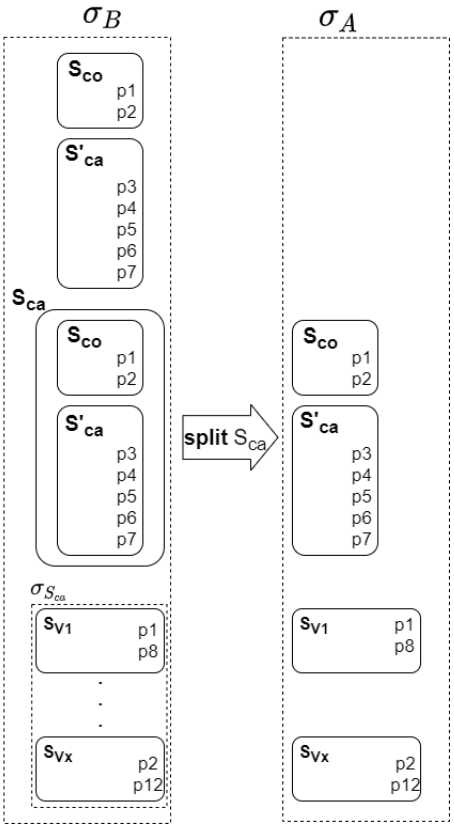

Fig. 8. Split case 1

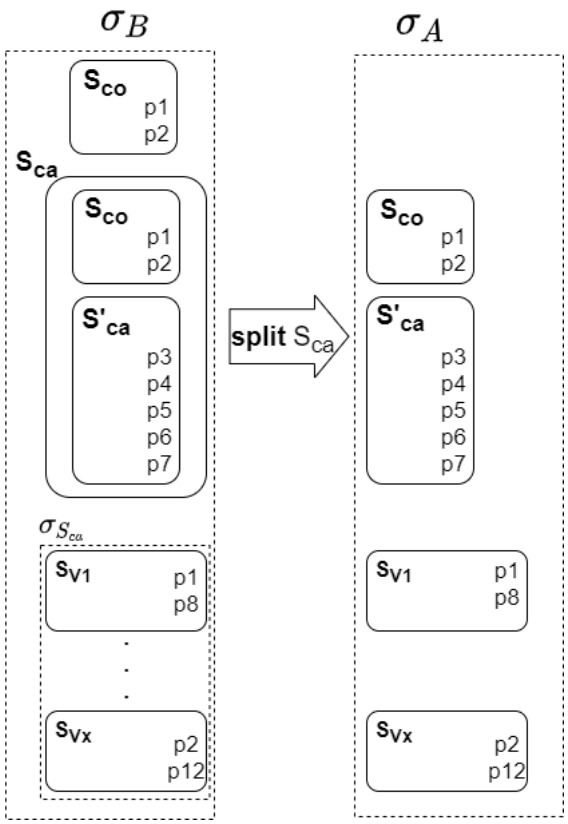

Fig. 9. Split case 2

See Figure 9 for a graphical representation.

As the port frequencies decrease, the **DI** of all groups that contain one or more port of $\mathbf{S}_{\text{co}}$ are also impacted. These are all $\mathbf{S}_{\text{Vi}}$ service groups.

When calculating Δ**OF**, only impacted service groups must be taken into account.

$$\Delta\mathbf{OF} = [\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} + \mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} + \sum_{i=1}^{n}\mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_B}] - [\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_A} + \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} + \sum_{i=1}^{n}\mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A}]$$

$$\Longrightarrow \Delta\mathbf{OF} = [\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A}] + [\mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A}] + [\sum_{i=1}^{n}\mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_B} - \sum_{i=1}^{n}\mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A}]$$

Taking (5) and (10) into account and using the same type of calculations as in Case 1:

(d) $\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B}$

(e) $\mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} = 1$

(f) $\mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} + \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A} + \sum_{i=1}^{n}\mathbf{DI}(\mathbf{S}_{Vi})_{\sigma_A} = \sum_{i=1}^{n}\frac{qvi}{nvi}$ (see case 1)

Putting (d), (e) and (f) into ΔOF

$$\Delta\mathbf{OF} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} + 1 + \sum_{i=1}^{n}\frac{qvi}{nvi} > 0$$

The same result is obtained when the assumption is made that $\mathbf{S'}_{\text{ca}}$ already exits in σ_B

Conclusion: If $|\ \sigma_B\ |$ **-** $|\ \sigma_A\ |$ **= 0, then** Δ***OF (due to*** $-\mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B}$***) can be*** < 0***, and the Objective Function can thus deteriorate***

Case 3: $|\ \sigma_B\ |$ **-** $|\ \sigma_A\ |$ **= 1**

$\mathbf{S}_{\text{ca}}$ splits into 2 mutually-exclusive new services ($\mathbf{S'}_{\text{ca}}$ and $\mathbf{S}_{\text{co}}$). Neither $\mathbf{S'}_{\text{ca}}$ nor $\mathbf{S}_{\text{co}}$ are part of σ_B.
$\mathbf{S'}_{\text{ca}}$ and $\mathbf{S}_{\text{co}}$ are both part of σ_A.
The port frequencies **PF** of any **p** do not change. No other services are impacted.

$$\forall \mathbf{p} \in \mathbf{S}_{ca} \mid \mathbf{PF}(\mathbf{p})_{\sigma_A} = \mathbf{PF}(\mathbf{p})_{\sigma_B} \tag{11}$$

The only factors playing a role in the calculation of Δ**OF** are $\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B}, \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A}$, and $\mathbf{DI}(\mathbf{S}_{co})_{\sigma_A}$

$$\Delta\mathbf{OF} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_A} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_A}$$

$$\Delta\mathbf{OF} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_B}$$

Conclusion: If $|\sigma_B|$ **-** $|\sigma_A|$ **= 1, then** Δ**OF can be** < 0 ***(due to*** $-DI(S'_{ca})_{\sigma_B} - DI(S_{co})_{\sigma_B}$***), and the Objective Function can thus deteriorate***).

See Figure 7 for a visual representation of this case.

Only case 1, $|\sigma_B|$ - $|\sigma_A|$ = -1, provides full certainty of how **OF** will evolve. To gain more certainly, we shall also investigate the relationship between the **DI** of service $\mathbf{S}_{ca}$ and the **DI**s of sub services $\mathbf{S'}_{ca}$ and $\mathbf{S}_{co}$.

Relationship between DIs

(1) $\mathbf{DI}(\mathbf{S}_{ca})_\sigma = \frac{\sum_{i=1}^{nca} \mathbf{PF}(\mathbf{p}i)_\sigma}{nca}$ =

$\frac{\sum_{i=1}^{1} \mathbf{PF}(\mathbf{p}i)_\sigma + \sum_{i=j+1}^{j+nco} \mathbf{PF}(\mathbf{p}i)_\sigma + \sum_{i=j+nco+1}^{nca} \mathbf{PF}(\mathbf{p}i)_\sigma}{nca}$

$\implies nca.\mathbf{DI}(\mathbf{S}_{ca})_\sigma - \sum_{i=j+1}^{j+nco} \mathbf{PF}(\mathbf{p}i)_\sigma =$

$\sum_{i=1}^{j} \mathbf{PF}(\mathbf{p}i)_\sigma + \sum_{i=j+nco+1}^{nca} \mathbf{PF}(\mathbf{p}i)_\sigma$

(2) $\mathbf{DI}(\mathbf{S'}_{ca})_\sigma = \frac{\sum_{i=1}^{nca} \mathbf{PF}(\mathbf{p}i)_\sigma + \sum_{i=j+nco+1}^{nca} \mathbf{PF}(\mathbf{p}i)_\sigma}{nca-nco}$

(3) $nco.\mathbf{DI}(\mathbf{S}_{co})_\sigma = \sum_{i=j+1}^{j+nco} \mathbf{PF}(\mathbf{p}i)_\sigma$

Putting (1), (2) and (3) together

$\mathbf{DI}(\mathbf{S'}_{ca})_\sigma = \frac{nca.\mathbf{DI}(\mathbf{S}_{ca}) - \sum_{i=j+1}^{j+nco} \mathbf{PF}(\mathbf{p}i)_\sigma}{nca-nco}$

$\implies \mathbf{DI}(\mathbf{S'}_{ca})_\sigma = \frac{nca}{nca-nco}.\mathbf{DI}(\mathbf{S}_{ca})_\sigma - \frac{nco}{nca-nco}.\mathbf{DI}(\mathbf{S}_{co})_\sigma$

$\implies \frac{nca}{nca-nco}.\mathbf{DI}(\mathbf{S}_{ca})_\sigma = \mathbf{DI}(\mathbf{S'}_{ca})_\sigma + \frac{nco}{nca-nco}.\mathbf{DI}(\mathbf{S}_{co})_\sigma$

$\implies \mathbf{DI}(\mathbf{S}_{ca})_\sigma = \frac{nca-nco}{nca}.\mathbf{DI}(\mathbf{S'}_{ca})_\sigma + \frac{nco}{nca}.\mathbf{DI}(\mathbf{S}_{co})_\sigma$

Let $\alpha = \frac{nco}{nca}$ be the split-factor, where $0 \leq \alpha \leq 1$

Then

$$\mathbf{DI}(\mathbf{S}_{ca})_\sigma = (1-\alpha).\mathbf{DI}(\mathbf{S'}_{ca})_\sigma + \alpha.\mathbf{DI}(\mathbf{S}_{co})_\sigma \qquad (12)$$

This formula expresses $\mathbf{DI}(\mathbf{S}_{ca})_\sigma$ as the linear interpolation between $\mathbf{DI}(\mathbf{S'}_{ca})_\sigma$ and $\mathbf{DI}(\mathbf{S}_{co})_\sigma$, with α as the interpolation factor. See Figure 10 for a visualization of this linear interpolation function.

Two cases are possible:

- Case a: $\mathbf{DI}(\mathbf{S'}_{ca})_\sigma > \mathbf{DI}(\mathbf{S}_{co})_\sigma$
- Case b: $\mathbf{DI}(\mathbf{S}_{co})_\sigma > \mathbf{DI}(\mathbf{S}_{ca})_\sigma$

Based on the relationship between DIs, we may conclude that:

If $|\sigma_B|$ - $|\sigma_A|$ = 1
then Δ**OF=** $\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S}_{co})_{\sigma_B} < 0$
as according to (12) either $\mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B}$ or $\mathbf{DI}(\mathbf{S}_{co})_{\sigma_B}$ is

$DI(S_{ca})_\sigma = (1-\alpha).DI(S'_{ca})_\sigma + \alpha.DI(S'_{co})_\sigma$

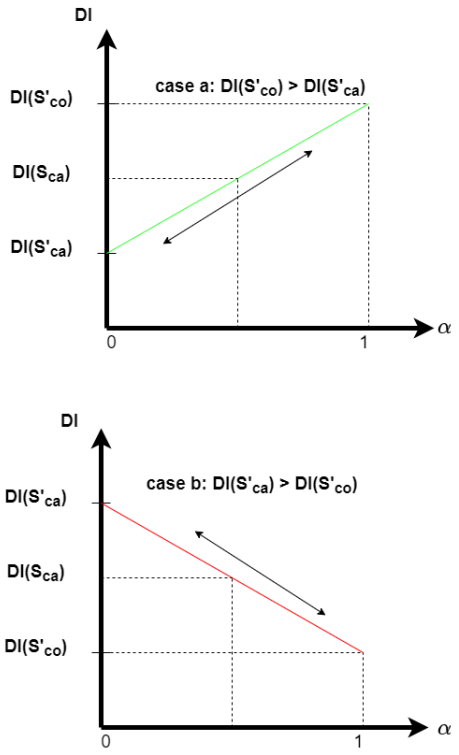

Fig. 10. Linear interpolation

$> \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B}$.
The Objective Function thus deteriorates when $|\sigma_B|$ - $|\sigma_A|$ = 1.

If $|\sigma_B|$ - $|\sigma_A|$ = 0
then $\Delta\mathbf{OF} = \mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} - \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B} + 1 + \sum_{i=1}^{n} \frac{qvi}{nvi}$ can be < 0
if $\Delta\mathbf{OF} < 0$ then $\mathbf{DI}(\mathbf{S}_{ca})_{\sigma_B} < \mathbf{DI}(\mathbf{S'}_{ca})_{\sigma_B}$ must be ¡ 0
Taking (12) into account, we can rewrite Δ**OF** as:

$(1-\alpha).\mathbf{DI}(\mathbf{S'}_{ca}) + \alpha.\mathbf{DI}(\mathbf{S}_{co}) - \mathbf{DI}(\mathbf{S'}_{ca}) + 1 + \sum_{i=1}^{n} \frac{qvi}{nvi}$

$\implies -\alpha.(\mathbf{DI}(\mathbf{S'}_{ca}) - \mathbf{DI}(\mathbf{S}_{co}) + 1 + \sum_{i=1}^{n} \frac{qvi}{nvi} > 0$

$\implies \Delta\mathbf{OF} > 0$ if $\alpha < \frac{1+\sum_{i=1}^{n} \frac{qvi}{nvi}}{\mathbf{DI}(\mathbf{S'}_{ca}) - \mathbf{DI}(\mathbf{S}_{co})}$

Thus, a smaller α gives a higher probability of an **OF** improvement.

2) Split Selection: From the preceding, we conclude that carving-out subgroups has a high likelihood to result in an improvement of the Objective Function. We will even go a step further and define our move type as the carving-out of all subgroups of a service definition. We call our split operator the full-carve-out move. Example: A service definition $\mathbf{S}$={1,2,3,4,5,6,7}. There also exists service definitions $\mathbf{S_1}$={1,2} and $\mathbf{S_2}$={5,7}. Carving out $\mathbf{S_1}$ and $\mathbf{S_2}$ from $\mathbf{S}$ gives,

$\mathbf{S_1}$= {1,2}, $\mathbf{S_2}$={5,7} and **S'**={3,4,6}.

This move type, however, is unable to handle partial overlapping service definitions. It is expected, therefore, that when all carve-outs are done, there will be a number of overlaps remaining that require a different type of operation.

F. Move Strategy

All services with a **DI** greater than one are candidates for splitting. It seems logical to begin by splitting the service with the largest **DI**. If that service cannot be split (no subgroups), then the second-largest **DI** is taken, etc. If a group can be split, the impact of the split is calculated. When **OF** improves (=descends), the move is accepted and executed. If not, the next service in the sorted service **DI** list is chosen. The move-strategy is a variant of the First Improvement strategy of the ILS meta heuristic; a variant as we first order the service **DI** list and take the top element from the list.

G. Perturbation

The carve-out of subgroups cannot remove all forms for non-disjointness. Correlated (partially overlapping) service definitions cannot be split this way. This creates a requirement for a new split operator when no additional carve-outs are possible. The operator will determine if a service definition overlaps with another service definition. If it does, the intersection is split-off. By splitting off this intersection, a new services definition will be created. Splitting off an intersection will result in $|\sigma_B| - |\sigma_A| = 1$. In the previous section we observed that the Objective Function will deteriorate. This is a transitory situation, due to the fact that the newly-formed service definitions may be subgroups of the existing service definitions. We consciously allow the **OF** temporary deterioration so that a better optimum may be found in the next Local Search iteration. We consider this kind of split as the perturbation.

H. Stop Conditions

Once all possible carve-outs and perturbations are complete, then there are no more inclusively matching and correlated rules. All port frequencies are equal to one, all service group **DI**'s are equal to one, and **OF** will equal the number of service definitions. The solution cannot be additionally improved.

Figure 11 shows how we expect the Objective Function to evolve over time, via consecutive local searches (doing full-carve-out moves) and perturbations (doing intersection-carve-out-moves), until the end condition is reached (i.e., full services are disjoint).

I. Algorithm Overview

The algorithm has been implemented in JAVA. The different components of the solution are implemented as JAVA classes. We attempted to stay as true as possible to NS principles by defining data classes, which only contain data and convenience methods to get and set the data, and task classes used to perform actions and calculations on the data objects. A high level overview of algorithm can be found in Algorithm 1. More

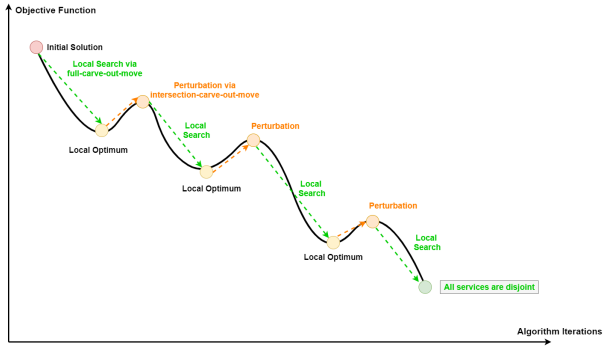

Fig. 11. Expected evolution of the Objective Function

algorithm details and implementation details can be found in [2].

V. Artifact Demonstration

The artifact outlined in the previous section will be applied to operational firewall rule bases provided by Engie. In [1] a manually created test rule base (containing a lot of exceptions to properly test the algorithm encoding) was used to validate the concept and the implementation. We now apply the artifact to operational data to see the real impact it has on a rule base. Before an export from a firewall can be used as input for the algorithm, some pre-processing is required. We start this section by explaining these operations. We continue by discussing the components we added to the algorithm that allow the adjustments to the rule base and thus measure the impact of service disjointness on the size of the rule base. The different demonstration sets will be elucidated before they become subject to the algorithm. We conclude with a summary of the algorithm's results and a description of some in-depth behavioral characteristics of the algorithm.

A. Firewall Export Pre-Processing

Engie provided firewall rule base exports that are implemented on Palo Alto firewalls (a leading manufacturer and provider of firewalls). Those exports required some pre-processing before the brown-field artifact can be used. The pre-processing steps include:

- Remove non-relevant information from the exported CSV files. This is the only manual step.
- Prepare data structures that allow historization (tracking of all changes to the rule base and services during algorithm execution) of the rule base and services.
- Replace firewall group objects that aggregate other group objects and adjust the rules accordingly.
- Adjust the rule base such that each rule only contains one service group.
- Version the rules and services (for historization).
- Remove non-unique services.

B. Adjusting the Rules

Each time a sub-service gets carved-out or an intersection between two services gets split off, adjustments to the rules

base are required. All the rules containing the original service must be adjusted to reflect the result of the split. The rules must also be split on the basis that rules must adhere to our SoC design criteria. Adjustments to the rule base occur at two instances of the algorithm: when a successful sub services carve-out is performed and when a successful overlapping services carve-out is performed.

C. Demonstration Data Sets

Engie provided exports from 15 Palo Alto firewalls in use within Belgium- and Paris-based data centers. The data centers contain multiple firewalls with different filtering strategies. We requested firewall exports that would represent the different types of filtering strategies. Additional contextual information for each firewall can be found below.

- AIMv2: Firewall used to filter in- and outbound traffic of Internet-exposed resources.
- AdminBE: Firewall used to filter traffic between data center client hosting zones and a shared management zone containing services as backup, and monitoring and system management tools. The firewall is located in the Belgium-based data center.
- AdminFR: Idem as AdminBE but for a firewall located in the Paris-based data center.
- AWSDCN: Firewall that acts as a filter between the Engie backbone network and the AWS Direct Connect (dedicated connection to AWS cloud data center in Dublin).
- HOSTING-BE-EBL: Firewall protecting the client hosting zone for Electrabel (a business unit of the Engie Group) in the Belgium-based data center.
- HOSTING-BE-ORES: Firewall protecting the client hosting zone for ORES (a former part of Electrabel, no longer part of the Engie Group), in the Belgium-based data center.
- HOSTING-BE-RAS: Firewall protecting Remote Access Resources, in the Belgium-based data center.
- HOSTING-BE-SHARED: Firewall protecting resources that are shared between various business units of the Engie Group, in the Belgium-based data center.
- HOSTING-BE-TRACTEBEL: Firewall protecting the client hosting zone for TRACTEBEL (a business unit of the Engie Group), in the Belgium-based data center.
- HOSTING-FR-COFELY: Firewall protecting the client hosting zone for COFELY (a business unit of the Engie Group), in the Paris-based data center.
- HOSTING-FR-GRDF: Firewall protecting the client hosting zone for GRFD (a former business unit of the GDF, no longer part of the Engie Group), in the Paris-based data center.
- HOSTING-FR-RAS: Firewall protecting Remote Access Resources, in the Paris-based data center.
- HOSTING-FR-SHARED: Firewall protecting resources that are shared between various business units of the Engie Group, in the Paris-based data center.

Algorithm 1: ILS for service list normalization

```
load_initial_solution (= rule base);
create neighborhood list (= list of services and their DI);
fully_disjoint = FALSE;
end_of_neighborhood = FALSE;
objective_function_improvement = FALSE;
calculate current_objective_function (sum of all service DI of
  neighborhood);
while NOT full_disjoint AND NOT end_of_neighborhood do
    sort neighborhood list (highest DI at top of list);
    neighborhood_pointer = 1 (top of list);
    objective_function_improvement = FALSE;
    while NOT improvement_objective_function AND NOT
      fully_disjoint AND NOT end_of_neighborhood do
        service_to_split = service to which
          neighborhood_pointing is referring;
        perform full-carve-out move on service_to_split;
        calculate new_objective_function;
        objective_function_improvement =
          (new_objective_function <
          current_objective_function?);
        if objective_function_improvement = TRUE then
            reset neighborhood based on full-carve-out
              move;
            reflect full-carve-out move in rule base;
            current_objective_function =
              new_objective_function
            fully_disjoint = (are all service DI of the
              neighborhood = 1);
        else
            neighborhood_pointer ++
        end
        end_of_neighborhood = (neighborhood_pointer
          pointing to last element in neighborhood list?);
    end
    if end_of_neighborhood then
        look for overlapping services in the neighborhood
        if overlapping_services_exists then
            perform intersection-carve-out move;
            reset neighborhood based on
              intersection-carve-out move;
            reflect intersection-carve-out move in rule base;
            calculate new_objective_function;
            current_objective_function =
              new_objective_function
            fully_disjoint = (are all service DI of the
              neighborhood = 1);
            end_of_neighborhood = FALSE;
        else
            end_of_eighborhood = TRUE;
        end
    end
end
if fully_disjoint then
    PRINT "Probably the Global Optimum has been found";
else
    PRINT "Local Optimum found";
end
PRINT RESULT = neighborhood;
```

- IAF: Firewall protecting access between the resources of the user network and data center, via Identify Aware filtering rules.
- IoT-BE: Firewall protecting IoT related resources in the Belgium-based data center.

The demonstration data set also contains an artificially-created rule base entitled Demoset which was used to test the algorithm. Demoset contains as many anomalies as possible to test special conditions that could occur but that are difficult to filter out of the given exports.

D. Demonstration Results

In the following subsections, we review the demonstration environment, the summary table of the demonstrations, and the relationship between the Objective Function and the number of rules in the rule base. We continue with a discussion of the impact of the algorithm on the number of service definitions, and have a closer look at the evolution of the Objective Function during the algorithm's execution. We conclude with an example of how rule and service definition changes are tracked during algorithm execution.

1) Demonstration Environment: The algorithm is written in JAVA using JAVA SDK 1.8.181, developed in the NetBeans IDE V8.2. The demonstration ran on an MS Surface Pro (5th Gen) Model 1796 i5 - Quad Core @ 2.6 GHz with 8 GB of memory, running Windows 10.

2) Demonstration Overview: The algorithm results for the different rule bases can be found in Table I which contains the following information:

- Initial Number of Rules (NoR): number of rules as read from the firewall export files.
- Initial Number of Services (NoS): number of services as read from the firewall export files.
- Initial Number of Service Groups (NoSR): number of service groups as read from the firewall export files.
- Pre-Processing Number of Rules (NoR): number of rules after pre-processing.
- Pre-Processing Number of Unique Services (NoUS): number of unique services after pre-processing.
- Pre-Processing **OF**: the value of the Objective Function, after pre-processing and thus at start of the algorithm.
- Final Number of Rules (NoR): the number of rules after applying the algorithm.
- Final Number of Services (NoS): the number of service definitions after applying the algorithm.
- Final **OF**: the value of the Objective Function after applying the algorithm.

The algorithm performance indicators can be found in Table II.

- Algorithm execution time: time required to disentangle the services and adjust the rules.
- Total execution time: time required to perform the data loading, pre-processing, disentanglement, to print the end result and log, and for the result to be displayed on the screen.
- Level 1 Iterator: number of times the outer loop of the algorithm has run.
- Level 2 Iterator: number of times the inner loop of the algorithm has run.

TABLE I
OVERVIEW DEMONSTRATION RESULTS

Rule Base	Initial			After Pre-Processing			Final		
	NoR	***NoS***	***NoSG***	***NoR***	***NoUS***	**OF**	***NoR***	***NoS***	**OF**
AIMV2	207	250	6	498	226	577.9	1,263	288	288
AdminBe	461	597	41	1,443	547	3,214.1	8,994	547	547
AdminFR	655	717	46	2,584	699	4,469.3	29,377	668	668
AWSDCN	13	13	1	13	13	14.9	22	13	13
Demoset	21	24	8	104	21	44.9	249	34	34
HOSTING-BE-EBL	350	304	10	759	259	877.6	3,841	256	256
HOSTING-BE-ORES	462	336	13	1,306	274	1,205.6	4,936	267	267
HOSTING-BE-RAS	20	16	0	28	16	17.5	29	16	16
HOSTING-BE-SHARED	107	120	7	223	10 7	188.7	360	106	106
HOSTING-BE-TRACTEBEL	10	5	1	10	5	5	10	5	5
HOSTING-FR-COFELY	10	9	1	16	9	9	16	9	9
HOSTING-FR-GRDF	118	46	4	213	42	50	223	40	40
HOSTING-FR-RAS	21	16	1	29	16	17,5	30	16	16
HOSTING-FR-SHARED	198	139	6	359	126	250.2	509	127	127
IAF	32	10	0	34	10	10	34	10	10
IOT-BE	23	28	0	38	25	36.5	47	24	24

TABLE II
PERFORMANCE OF THE ALGORITHM

Rule Base	Execution Information			
	ILS (ms)	Total (ms)	L1	L2
AIMV2	187,227	396,000	25	1,507
AdminBe	520,944	824,000	135	13,609
AdminFR	820,847	1,242,000	154	23,165
AWSDCN	811	18,000	2	3
Demoset	1,358	120,000	22	430
HOSTING-BE-EBL	76,039	265,000	34	2,016
HOSTING-BE-ORES	193,436	632,000	59	2,587
HOSTING-BE-RAS	99	1,000	2	3
HOSTING-BE-SHARED	35,139	202,000	12	427
HOSTING-BE-TRACTEBEL	63	1,000	2	2
HOSTING-FR-COFELY	54	1,000	2	2
HOSTING-FR-GRDF	122	1,000	3	6
HOSTING-FR-RAS	96	1,000	2	36
HOSTING-FR-SHARED	78,503	210,000	19	929
IAF	68	1,000	2	2
IOT-BE	28,731	130.000	3	19

TABLE III
%**OF** IMPROVEMENT VS INITIAL NUMBER OF RULES (NOR)

Rule Base	**Initial NoR**	%**OF** Improvement
HOSTING-BE-TRACTEBEL	10	0%
HOSTING-FR-COFELY	10	0%
AWSDCN	13	10%
HOSTING-BE-RAS	20	9%
Demoset	21	24%
HOSTING-FR-RAS	21	9%
IOT-BE	23	34%
IAF	32	0%
HOSTING-BE-SHARED	107	44%
HOSTING-FR-GRDF	118	20%
HOSTING-BE-SHARED	198	49%
AIMv2	207	61%
HOSTING-BE-EBL	350	71%
AdminBE	461	83%
HOSTING-BE-ORES	462	78%
AdminFR	655	85%

3) Objective Function and the Number of Rules: In Table III and Figure 12, we represent the relationship between the % of **OF** improvement and the number of initial rules in the rule base (NoR).

Three out of the sixteen firewalls contain fully disjoint service definitions: HOSTING-BE-TRACTEBEL, HOSTING-FR-COFELY and IAF. Those are also the firewalls with the fewest rules and service definitions. The algorithm detects the full disjointness and leaves the service definitions and rule base as is.

Six out of the sixteen firewalls are fairly close to having disjoint service definitions: AWSDCN, Demoset, HOSTING-BE-RAS, HOSTING-FR-GRDF and IOT-BE. Those firewalls have a number of rules and services definitions that are below 100 (HOSTING-FR-GRDF having a number of rules a bit above 100). The total improvement of the **OF** is limited to about 25 %.

The remaining eight firewalls contain many more rules and service definitions and the value of the difference between the initial and final value of the **OF** is at least 50 %, with a maximum of 85 %. These numbers confirm that, without proper rule design criteria, the probability of getting a non-evolvable rule base drastically increases with the size of the rule base. The trend between the size of the rule base and the percentage of **OF** improvement (a good indicator for the status of the initial evolvability), should be an asymptotic function trending toward 100 %. A logarithmic regression provides a good fit.

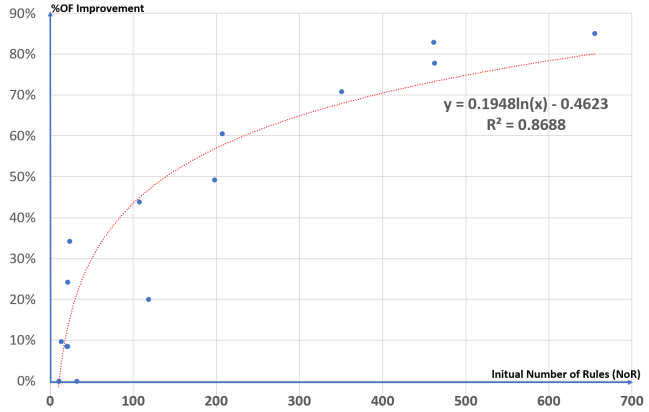

Fig. 12. %**OF** Improvement vs number of rules in the rule base

In Table IV and Figure 13, we represent the relationship between the % **OF** improvement and the % of extra rules (growth rule base) due to the service disentanglement algorithm.

4) Impact of the Algorithm on the Number of Service Definitions: Upon examination of the number of service definitions at the end of the algorithm, we note that splitting the service definitions does not have a large impact on the total number of services. See Figure 14 for an overview. There is even a tendency toward the total number of service definitions decreasing slightly. It seems to be that the algorithm rearranges the ports into more suitable groups, without having the number of service definitions proliferate.

5) Evolution of the Objective Function During Algorithm Execution: To visualize what occurs during the algorithm execution, three indicators are tracked: the **OF**, the outer loop iterations, and the inner loop iterations. The "Level 1 Indicator" is the number of times that the outer **DO** loop of the algorithm has run. The indicator measures the number of times a perturbation or successful carve-out is done. The "Level 2 Indicator" (L1I) is the number of times the inner **DO** loop of the algorithm runs within a given number of level 1 iterations. Each time the "Level 1 Iterator" increments, the "Level 2 Iterator" (L2I) is reset. We plot the evolution of these three indicators against the cumulative number of level 2 iterations for two of the firewalls with a number of rules below 100, in Figure 15 and Figure 16. In Figure 17 and Figure 18 we show the evolution of the three indicators for two firewalls containing in excess of 100 rules.

TABLE IV
% **OF** IMPROVEMENT VS %GROWTH RULE BASE

Rule Base	%OF Improvement	%Growth Rule Base
HOSTING-BE-RAS	9%	4%
HOSTING-FR-RAS	9%	3%
AWSDCN	10%	69%
HOSTING-FR-GRDF	20%	5%
Demoset	24%	139%
IOT-BE	34%	24%
HOSTING-BE-SHARED	44%	61%
HOSTING-FR-SHARED	49%	42%
AIMv2	61%	154%
HOSTING-BE-EBL	71%	406%
HOSTING-BE-ORES	78%	278%
AdminBE	83%	523%
AdminFR	85%	1037%

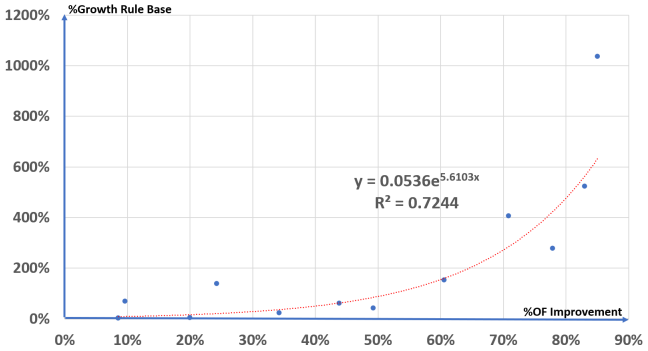

Fig. 13. % extra rules vs %Δ**OF**

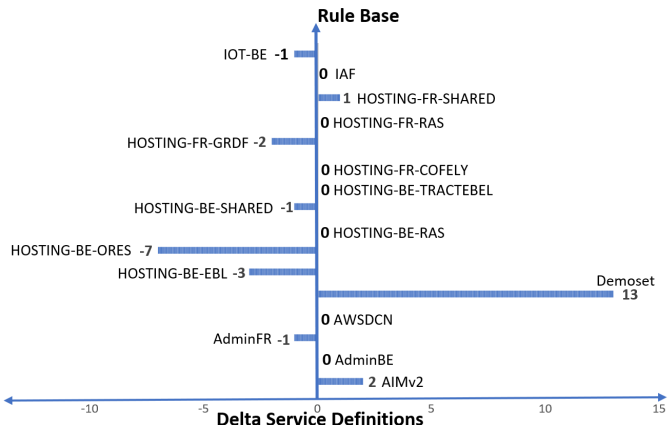

Fig. 14. Impact of the algorithm on number of services.

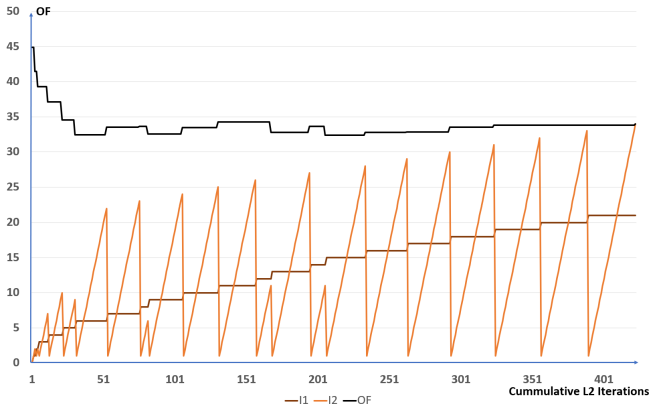

Fig. 15. **OF**, L1 and L2 for the Demoset firewall.

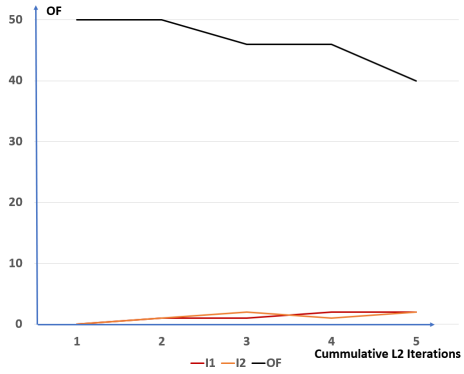

Fig. 16. **OF**, L1 and L2 for the HOSTING-FR-GRDF firewall.

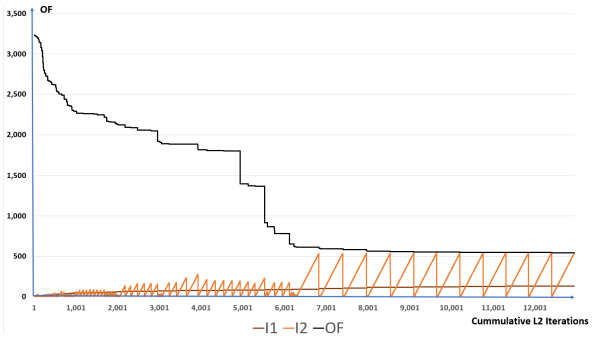

Fig. 17. **OF**, L1 and L2 for the AdminBE firewall.

6) Tracking of Rule and Service Definition Changes: The algorithm tracks all changes that are made to the rules. As an example, the log excerpt below shows the evolution of rule 6 from the Demoset, as provided by the algorithm at end of execution.

- During pre-processing, the Service Group ”SERVICE25”, is replaced by its members ”SERVICE17” and ”SERVICE19”. The rule now has 6.1 as identifier.
- During pre-processing, the rule is split into two rules, 6.1.1 and 6.1.2 since rule 6.1 was contained in two service definitions.
- Duinrg pre-processing, rules 6.1.1 and 6.1.1.2 get the versioned service definitions. At this point, rule 6 is replaced by 6.1.1.1 and 6.1.2.1.
- During the ILS, the service ”SERVICE17 V0” gets split into ”Service 17 V0.2” and the existing service ”SERVICE19 V0”, and the rule 6.1.1.1 splits into 6.1.1.1.1 and 6.1.1.1.2.

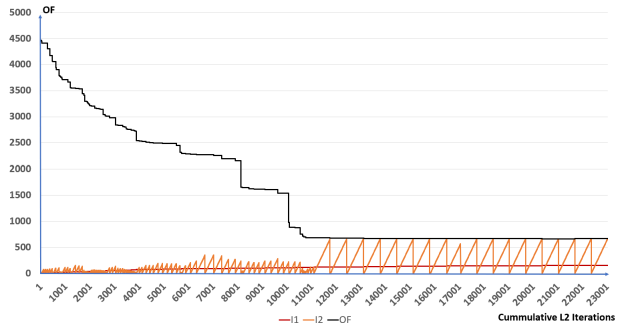

Fig. 18. **OF**, L1 and L2 for AdminFR firewall.

```
6;R6;SERVICE25,
*6.1;R6.1;SERVICE17,SERVICE19,
**6.1.1;R6.1.1;SERVICE17,
***6.1.1.1;R6.1.1.1;SERVICE17 V0,
****6.1.1.1.1;R6.1.1.1.1;SERVICE19 V0,
****6.1.1.1.2;R6.1.1.1.2;SERVICE17 V0.2,
**6.1.2;R6.1.2;SERVICE19,
***6.1.2.1;R6.1.2.1;SERVICE19 V0,
```

In summary, rule 6 was replaced by rules 6.1.1.1.1, 6.1.1.1.2 and 6.1.2.1.

The evolution of the services is tracked in a similar manner. In the log excerpt below, the evolution of ”SERVICE17” and ”SERVICES19” is shown (versioning, splitting).

```
SERVICE17;UDP;40-41
*SERVICE17 V0;UDP;40-41
**SERVICE19 V0;UDP;40
**SERVICE17 V0.2;UDP;41

SERVICE19;UDP;40
*SERVICE19 V0;UDP;40
```

VI. Evaluation and Discussion

This section starts with evaluating the functioning of the artifact followed by the analysis of the worst case number of operations required to complete the algorithm and the comparison with the artifact performance. We continue with pondering on the question whether the artifact has found the most optimal solution, and by analysing the impact of the algorithm on size of the rule base and firewall scaling. We continue with discussing the entropy present in a rule base and how the algorithm measures this entropy. We finish by positioning the artifact in a firewall management tool.

A. Algorithm functioning

Based on the demonstration, we can conclude that an algorithm based on an ILS meta-heuristic disentangles service definitions and is able to adjust the rule base accordingly. The algorithm is an essential building block in a solution that can

convert an existing firewall rule base into a rule base that is fully compliant with the green-field artifact.

It is an essential building block but not the sufficient building black. It is possible that fully-overlapping rules emerge during the algorithm execution.

Example:

Take a rule R1 that has C1 as source, H1 as destination, and S1 as service.

Now take a rule R2 that has C1 as source H1 as destination, and S2 as source.

Let's consider that S1 and S2 overlap. The overlapping service is S3

Applying the algorithm would give:

– R1: C1 H1 S'1
– R2: C1 H1 S3
– R3: C1 H1 S'2
– R4: C1 H1 S3

As can be seen, R2 and R3 become identical rules which still need to be filtered out.

The demonstration has provided insight into how the Objective Function evolves during algorithm execution, as well as into the relationships between the number of initial rules in the rule base and the corresponding value of the objective function, and the number of rules at the end of the algorithm execution and the change in Objective Function.

B. Big O of the Artifact

The **Big O** of an algorithm expresses the algorithm's complexity, calculated based on the worst-case scenario in terms of the number of operations required in function of the size of the problem to be solved. This formula reflects the worst-case effort required to complete the algorithm execution. The algorithm contains two nested loops that both can iterate over the full neighborhood, meaning the algorithm will be quadratic with respect to the size of the neighborhood. The number of operations performed in the innermost loop, such as Service_DI_list_Creator, Service_split_Evaluator are also proportional to the size of the neighborhood.

We may thus conclude that the **Big O** of the complete algorithm is cubic - $O = n^3$, where **n** is the size of the neighborhood (= size of the solution = the number of service definitions).

C. Performance of the Artifact

Algorithm execution time is measured as the time it takes to disentangle the services after pre-processing. Figure 19 shows the relationship between the initial size of the neighborhood (number of unique services) and algorithm execution time. The exponent of the power function is a bit above two. This is consistent with the **Big O**, where we expected a worst-case exponent of three.

Measures could be taken to ensure better algorithmic performance. The innermost loop iterates over all services until it locates one that contains subgroups. All services that already have a **DI** of 1 should not be further investigated. As the neighborhood is sorted from high to low **DI** at the start of the inner loop, the inner loop could stop as of the first service where a **DI** of 1 is encountered. According to meta-heuristics, this value represents a form of algorithmic memory, indicating parts of the neighborhood that can no longer improve and should thus not be investigated.

TABLE V
ARTIFACT PERFORMANCE

Rule Base	Number of Unique Services (NoUS)	Algorithm Execution Time (ms)
HOSTING-BE-TRACTEBEL	5	63
HOSTING-FR-COFELY	9	54
IAF	10	68
AWSDCN	13	811
HOSTING-BE-RAS	16	99
HOSTING-FR-RAS	16	96
Demoset	21	1,358
IOT-BE	25	28,731
HOSTING-FR-GRDF	42	122
HOSTING-BE-SHARED	107	35,139
HOSTING-FR-SHARED	126	78,503
AIMv2	226	187,227
HOSTING-BE-EBL	259	76,039
HOSTING-BE-ORES	274	193,436
AdminBE	547	520,944
AdminFR	699	820,847

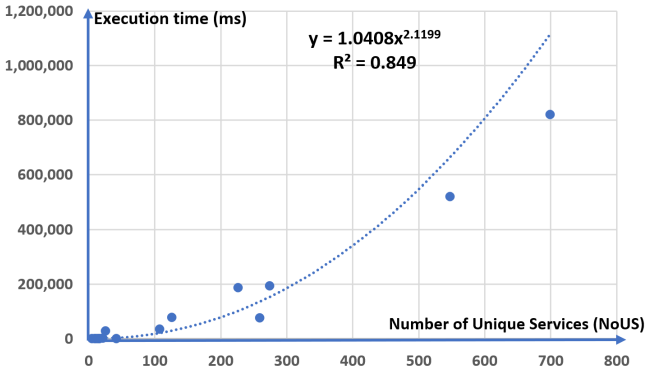

Fig. 19. Artifact performance

D. Global Optimum

Does the heuristics-based algorithm establish the Global Optimum? It is quite difficult to formally prove that heuristic algorithms always provide the most optimal solution. After all, the full solution space of all possible groups combining all possible ports is exponential (see combinatorics) and quickly becomes impossible to fully search.

We do think that, given the initial solution, we have succeeded in converging on the most optimal solution. Sub-optimal solutions always will have either subgroup and/or overlapping groups. The algorithm filters out all subgroups in the inner loop and, if no additional subgroups are found, it searches for overlaps, after which it again scans for subgroups. As both inner and outer loop iterations search the entire neighborhood, all possible subgroups and overlaps are located and eliminated. While we are not presently able to provide formal proof, we nonetheless believe that, from a given initial solution, the set of services that are disjoint and maximum in size is found.

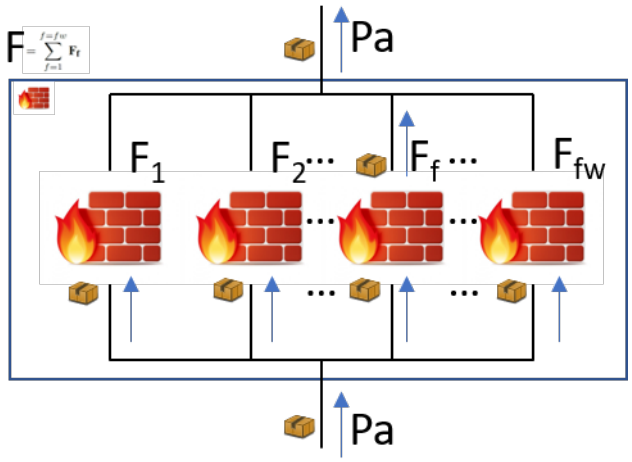

Fig. 20. Scaling of firewalls with an evolvable rule base

E. Implication of the Artifact on Firewall Rule Base Size and Scaling

In all application of NS on design of systems, the result is a more fine grained structure [7] [17] [18]. Applying NS on a firewall, and thus making it more evolvable, is no exception. From the previous section we can clearly see that the amount of extra rules can increase considerably. This brings forward the question of firewall performance - how quickly will the firewall find the rule to apply to the traffic.

In an evolvable rule base, all the rules are disjoint from one another and each network package can only hit one rule. This rule can be located in the beginning or near the end of the rule base. As there is only one rule that can be hit, the rule base may be split into multiple parts and distributed in parallel across different firewalls.

Let **F** be a firewall rule base containing only disjoint rules created according to the green-field artifact. As visualized in Figure 20, **F** can be split into **f**w sub rule bases, which are spread over **f**w parallel firewalls. Each of the **f**w rule bases conclude with the “Default Deny” rule.

A network package will attempt to traverse each firewall, but only one of the firewalls has a rule it can hit.

$$\mathbf{F} = \sum_{f=1}^{f=fw} \mathbf{F}_{\mathrm{f}}$$

Let $\phi_{\mathbf{f}}(\mathbf{F_f},\mathbf{Pa})$ be the firewall filtering function that takes rule base $\mathbf{F_f}$ and package **Pa** as input.

- $\phi_{\mathbf{f}}(\mathbf{F_f},\mathbf{Pa})$ = 0 if the package is blocked - there is no rule **R** in $\mathbf{F_f}$ such that the package is allowed
- $\phi_{\mathbf{f}}(\mathbf{F_f},\mathbf{Pa})$ = 1 if the package is allowed - there is a rule **R** in $\mathbf{F_f}$ such that the package is allowed

Let $\Phi^{\mathbf{P}}{}_{\mathbf{fw}}$ be the parallel firewall filtering function. Then:

$$\Phi^{\mathbf{P}}{}_{\mathbf{fw}}(\mathbf{PA}) = \sum_{f=1}^{f=fw} \phi_{\mathbf{f}}(\mathbf{F_f},\mathbf{Pa})$$

Where:

- $\Phi^{p}_{fw}(\mathbf{Pa})$ = 0, if **Pa** is blocked by all of the fw firewalls.
- $\Phi^{p}_{fw}(\mathbf{Pa})$ = 1 if **Pa** is allowed by one of the fw firewalls.
- $\exists! F_f \in F$ for $f = 1 \rightarrow fw \implies | \, R \in F_j$

A rule base that exclusively contains disjoint rules can scale horizontally (i.e., employ parallel firewalls). Firewalls with a non-evolvable rule base can only scale vertically (i.e., employ a larger firewall). Scaling, however, does not come without significant cost. Modern firewalls allow virtualization, but each virtual instance comes at a cost as well.

In addition to the horizontal scaling possibilities of an evolvable rule base, the performance of an evolvable rule base can be boosted by moving the most frequently used rules to the top. Check Point, a firewall vendor, suggests locating the rules that are most frequently hit (and applied) at the top of the firewall table. In a rule base that is order-sensitive, this is a real issue. In a rule base that is not order-sensitive, one could monitor the firewall to determine which rules are hit most and then prioritize those rules without having to worry about the potential impact to other rules. Doing this dynamically would be even more powerful as the firewall would be able to reorganize its rules according to variable daily traffic.

F. Measuring the entropy of a filewall rulebase

We will now examine the question ”What is the impact of the service group disjointness level of a rule base on the size of the aforementioned rule base after application of the brown-field artifact?”. We shall define the Services Disjointness Index (**SDI**) as the ratio between the value of the objective function **OF** and the number of services **S**.

$$\mathbf{SDI} = \frac{\mathbf{OF}}{\mathbf{S}}$$

SDI is 1 in a rule base exclusively containing disjoint services and greater then 1 if the rule base contains non-disjoint services. We would like to know whether or not we may determine the increase in number of rules as a result of the application of the brown-field artifact, based on the initial value of **SDI**.

The **SDI** is a fairly accurate measure for the statistical entropy of a rule base. The macro-state is the number of services in a rule base, the micro-states being the number of possible services within a rule base. An evolvable and perfectly stable rule base would have a ratio of micro-states to macro-state equalling 1. There are multiple configurations of services that deliver a statistical entropy of 1. We are aware of at least two: one port per service, and the one we discovered with the brown-field algorithm by disentangling the services. The **SDI** is, however, an imperfect representation of the statistical entropy of the rule base. Indeed, we have demonstrated that the brown-field algorithm may result in shadowing rules. An additional operationalization to measure this would be required in order to fully express the statistical entropy of a rule base.

TABLE VI
RNIR VS **SDI**

Rule Base	SDI	RNIR
HOSTING-BE-TRACTEBEL	1	0
HOSTING-FR-COFELy	1	0
IAF	1	0
HOSTING-BE-RAS	1.0938	0.0357
HOSTING-FR-RAS	1.0938	0.035
AWSDCN	1.1069	0.6923
HOSTING-FR-GRDF	1.1905	0.0469
IOT-BE	1.4600	0.2368
HOSTING-BE-SHARED	1.7632	0.6143
HOSTING-FR-SHARED	1.9853	0.4178
Demoset	2.1377	1.3942
AIMv2	2.5573	1.5361
HOSTING-BE-EBL	3.3882	4.0606
HOSTING-BE-ORES	4.3999	2.7795
AdminBE	5.8759	5.2328
AdminFR	6.3938	10.3688

In our experiment, the independent variable is **SDI** and the dependent variable is the relative increase in the number of rules due to application of the treatment (i.e., application of the brown-field artifact). The relative increase in the number of rules (**RINR**) is calculated as the difference between the number of rules (**NR**) after and before application of the artifact, divided by the number of rules before application of the artifact.

$$\textbf{RINR} = \frac{NR_{after} - NR_{before}}{NR_{before}}$$

The result can be found in Table VI and Figure 21. The correlation between the independent and dependent variable is 0.9257.

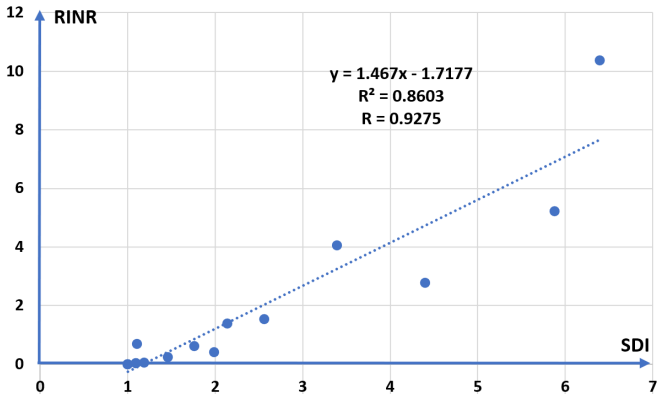

Fig. 21. **RINR** vs **SDI**

G. The Firewall Rule Base Analyser and Normalizer System

As the firewall provides considerable design freedom which could potentially lead to evolvability issues, firewall management should be undertaken outside of the firewall, ideally using a specialized tool that incorporates the artifacts discussed in this dissertation. We characterize this tool as a Firewall Rule Analyser and Normalizer System or FRANS (see Figure 22). Such a tool would ideally have the following features:

- Enforces the usage of the green-field artifact.
- Analyzes an existing rule base — measures disjointedness levels — with the brown-field artifact.
- Converts an existing rule base into an evolvable rule base using the brown-field artifact.
- Will centrally manage all definitions: services, sources, destinations, rules.
- Provides full traceability on all changes performed on the definitions.
- Makes firewalls scale horizontally.
- Changes the rule order dynamically to increase performance.

All firewall rule management activities are done in the tool as opposed to via firewall management consoles. As modern firewalls publish their management functionalities via APIs, the tool can use these APIs to change rules and objects.

The creation of the fine-grained rule base by humans is an issue. The green-field artifact defines criteria for groups and rules that need to be followed strictly. The creation of a catalog of all possible services is required. For standard services and tools, lists of assigned ports/protocols and international standardization organizations related to the Internet (e.g., iana.org) exist and may be reused.

FRANS should expand the firewall rules in the fine-grained format, in accordance with the naming conventions. Checks must also be performed against the group definitions and content in accordance with the green-field artifact and via a user-friendly interface. With this configuration, the tool could then push the rules towards the firewall, which would effectively separate the management from the implementation of rules. Such tools exist on the market. Examples include Algosec, Tufin, Firemon. However, none of those tools consciously restrict the design space for the purpose of enforcing the creation of an evolvable rule base.

While defining a rule for each service may be considered cumbersome, it is possible to create roles such as ”monitoring and management” (i.e., establishing which is a grouping of smaller, disjoint services) in order to mitigate this. In this example, the firewall administrator could create a rule specifying this ”monitoring and management” role to express that the server needs to allow access to all monitoring and management services. The tool would ideally expand these roles into the individual rules for each disjoint service. Examples:

- ”Monitoring and Management” = SSH + SFTP + FTP + SMTP + TELNET
- Host = x
- Rule : C_Hx_SMaM; Hx_S_MaM; S_MaM; allow
- Will be expanded to :
  - C_Hx_S_SSH; Hx_S_SSH; S_SSH; allow
  - C_Hx_S_SFTP; Hx_S_SFTP; S_SFTP; allow
  - C_Hx_S_FTP; Hx_S_FTP; S_FTP; allow

– C_Hx_S_SMTP; Hx_S_SMTP; S_SMTP; allow
– C_Hx_S_TELNET; Hx_S_TELNET; S_TELNET; allow

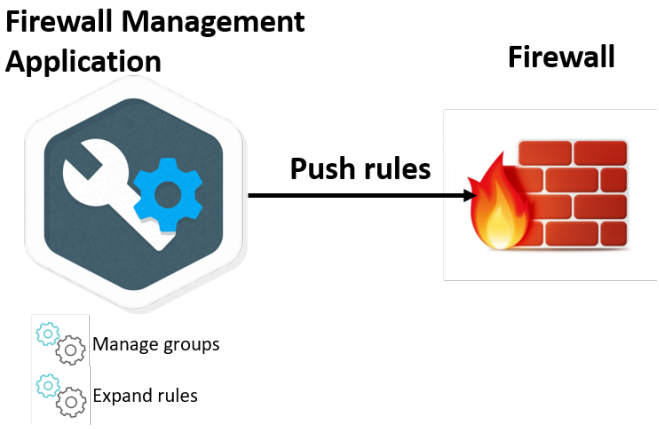

Fig. 22. Firewall management tool

The brown-field artifact should be included in the tool to read and analyze an existing rule base. The Disjoint Index of all groups and the total value of the Objective Function can be calculated. These are important indicators for the level of evolvability and the impact the firewall normalization process will have on the size of the rules base. Highly non-evolvable rule bases may require additional firewall infrastructure to allow horizontal scaling. The tool could create new firewall instances on a virtual infrastructure or spin up new cloud based firewalls on a cloud platform.

FRANS should convert existing rule bases into evolvable rule bases and deploy those on the firewall infrastructure. As FRANS should be a central firewall management platform, it could compare at all times the defined policy in the tool to the active policy on the firewall. This would allow detection of rule adjustments made directly on the firewall and even make firewall rule bases immutable. In FRANS, additional information reflecting why rules are deployed and links with application management tools could be made in order to allow centralized and easily understandable security documentation.

VII. Conclusion

In this section, we will summarize important conclusions. We start with pointing out that the artifact allows the measuring of the evolvability of the firewall. We continue by stating that the artifact leads to a fine-grained rule base and by stating that such fine-grained rule bases allow true horizontal scaling. We finish be pointing out some limitations, lack of comparison with related literature and proposing future work.

A. Measuring the Evolvability of a Firewall

We were able to operationalize one aspect of the evolvability of a firewall, namely the need for disjoint services. Independent from the meaning and functions of the various service ports within the rule base, the **SDI** is an important indicator for the evolvability of the firewall. If **SDI** is greater than 1, the door is left open to the creation of non-evolvable rules. The **SDI** represents the statistical entropy of the service configurations in a rule base and is a good proxy for the statistical entropy of the rule base.

To measure all aspects of evolvability and statistical entropy in accordance with the green-field artifact, a second index concerning the destinations would need to be developed.

B. Impact of the Brown-Field Artifact on the Size of the Rule Base

As SoC has been meticulously applied, the choice of a fine-grained rule base is unsurprising. The relationship between the level of service disjointness and extra rules has been investigated. Additional runs of the algorithm with firewalls from different companies would provide further insight into the complexity of this relationship (and establish whether it is linear, polynomial or exponential).

C. Firewall Scaling

The artifacts produce a fine-grained rule base. A large number of rules in a rule base will have a detrimental impact on performance. But creating an evolvable rule base also provides the answer to this problem, given that only an evolvable rule base will scale truly horizontally.

D. Artifact Limitations

The artifact tracks all changes in the service definitions by means of continuously changing the name of the services via a versioning mechanism. Although the end result is disjoint services according to the green-field artifact, the naming of those services is not compliant with the naming convention put forward in the green-field artifact. A mechanism to generate meaningful names is currently lacking. According to the green-field artifact, we should add a rules in the rule base for each host-service combination. The current version of the brown-field artifact only disentangles the services. Although this leads to disjoint rules, it can still lead to CE [2]. It is quite straight forward to add this step as it just a matter of splitting rules to make sure they only contain one destination and not an aggregation of destinations. The brown-field artifact can result in identical rules and so can the above mentioned splits in destinations. The algorithm does not filter those out. An extra pass through the rule base is required to eliminate those.

E. Related Literature

A more detailed literature study related to this work can be found in [2]. We would however like to stress that, to the best of our knowledge, no other work has been found that addresses the evolvability issues of the firewall. Over the past 40 years, sufficient research has been done boosting performance of the firewall and on problems with firewall management. We have not found work that does an analysis based on a grounded theory of evolvability such as NS. The concept of true horizontal scaling of firewall has also not been observed in literature. The closest we found is work that put an amount of firewalls in parallel but all firewalls contain the

same rule base [23]. This will indeed boost performance as each firewall has less traffic to handle but each firewall still has the same rule base size. In our solution, the size of the rule base can be reduced by spreading it over multiple firewalls.

F. Future Work

This work is an important yet incomplete step toward the evolvable TCP/IP firewall. The green-field artifact needs to be converted into software that will take a high-level security requirement as input, "expand" it into the required fine-grained rules, and push the rules to a firewall. The artifact discussed in the paper (the brown-field artifact) needs to be extended to re-organize the destinations and sources in accordance with the green-field artifact, and requires a solution to naming services such that they are in line with the green-field artifact.

While the requisite groundwork has been established, the remainder needs to be built. We thus regard this not as future research, but rather as future work.

This work has limited itself to the TCP/IP based firewall, which provides a basic security layer like decent locks on the doors and windows of a house. Other security devices, such as application level firewalls, operate at other layers of the OSI stack, above TCP/IP. Filtering rules are installed there as well and if again the filtering rules overlap or contradict, evolvability issues may appear. It is also native to only rely on application level firewalls and no longer on TCP/IP based firewalls, as it is like removing locks from doors and windows and only react on camera surveillance - by the some something shows up on camera, the damage can already be done. Multi level security applies in both the physical and virtual world.

References

[1] G. Haerens, "Usage of iterated local search to improve firewall evolvability", PATTERNS-2021, 2021.

[2] G. Haerens, "On the Evolvability of the TCP-IP Based Network Firewall Rule Base", 2021, PhD Thesis, ISBN: 978-90-5728-716-9, University of Antwerp, 2021

[3] H. Shel and A. Spiliotes, "The State of Network Security: 2017 to 2018", Forrester Research, November 2017

[4] "2018 State of the firewall", Firemon whitepaper, URL https://www.firemon.com/resources/, [retrieved: April, 2021]

[5] "Firewall Management - 5 challenges every company must address", Algosec whitepaper, URL https://www.algosec.com/resources/, [retrieved: April, 2021]

[6] G. Haerens and H. Mannaert, "Investigating the Creation of an Evolvable Firewall Rule Base and Guidance for Network Firewall Architecture, using the Normalized Systems Theory", International Journal on Advances in Security, Volume 13 nr. 1&2, pp. 1-16, 2020

[7] H. Mannaert, J. Verelst and P. De Bruyn, "Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design", ISBN 978-90-77160-09-1, 2016

[8] E. Al-Shaer and H. Hamed, "Taxonomy of conflicts in network security policies", IEEE Communications Magazine, 44(3), pp. 134-141, March 2006

[9] E. Al-Shaer, H. Hamed, R. Boutaba and M. Hasan, "Conflict classification and analysis of distributed firewall policies", IEEE Journal on Selected Areas in Communications (JSAC), 23(10), pp. 2069-2084, October 2005

[10] M. Abedin, S.Nessa, L. Khan and B. Thuraisingham, "Detection and Resolution of Anomalies in Firewall Policy Rules", Proceedings of the IFIP Annual Conference Data and Applications Security and Privacy, pp. 15–29, 2006

[11] E. Al-Shaer and H. Hamed, "Design and Implementation of firewall policy advisor tools", Technical Report CTI - techrep0801, School of Computer Science Telecommunications and Information Systems, DePaul University, August 2002

[12] S. Hinrichs, "Policy-based management: Bridging the gap", Proceedings of the 15th Annual Computer Security Applications Conference, pp. 209-218, December 1999

[13] A. R. Hevner, S. T. March, J. Park and S. Ram, "Design Science in Information Systems Research", MIS Quarterly, Volume 38, Issue 1, pp. 75-105, 2004

[14] P. Johannesson and E. Perjons, "An Introduction to Design Science", ISBN 9783319106311, 2014

[15] W. R. Stevens, "TCP/IP Illustrated - Volume 1 - the Protocols", Addison-Wesley Publishing Company, ISBN 0-201-63346-9, 1994

[16] H. Zimmermann and J. D. Day, "The OSI reference model", Proceedings of the IEEE, Volume 71, Issue 12, pp. 1334-1340, Dec 1983

[17] H. Mannaert, J. Verelst and K. Ven, "The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability", Science of Computer Programming, Volume 76, Issue 12, pp. 1210-1222, 2011

[18] P. Huysmans, G. Oorts, P. De Bruyn, H. Mannaert and J. Verelst, "Positioning the normalized systems theory in a design theory framework", Lecture notes in business information processing, ISSN 1865-1348-142, pp. 43-63, 2013

[19] G. Haerens, "Investigating the Applicability of the Normalized Systems Theory on IT Infrastructure Systems, Enterprise and Organizational Modeling and Simulation", 14th International workshop (EOMAS) 2018, pp. 23-137, June 2018

[20] M. Rafael, P. M. Pardalos and M. G. C. Resende, "Handbook of Heuristics", ISBN 978-3-319-07123-7 IS, 2018

[21] Z. Michalewicz and D. B. Fogel, "How to Solve It: Modern Heuristics", ISBN 978-3-642-06134-9, 2004

[22] E. Talbi, "Metaheuristics - From Design to Implementation", ISBN 978-0-470-27858-1, 2009

[23] K. Salah, P. Calyam, R. Boutaba, "Analytical model for elastic scaling of cloud-based firewalls", IEEE Transactions on Network and Service Management, 2016, 14.1: 136-146.