Using Normalized Systems to Explore the Possibility of Creating an Evolvable Firewall Rule Base

Geert Haerens

Faculty of Business and Economics
University of Antwerp, Belgium
and Engie IT — Dir. Architecture
Email: geert.haerens@engie.be

Peter De Bruyn

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

***Abstract*—A firewall is an essential network security component. The firewall rule base, the list of filters to be applied on network traffic, can have significant evolvability issues in a context where companies consider their firewall as complex. Whereas sufficient literature exists on how to analyze a rule base which is running out of control, little research is available on how to properly construct a rule base upfront, preventing the evolvability issues to occur. Normalized Systems (NS) theory provides proven guidance on how to create evolvable systems. In this paper, NS is used to study the combinatorics involved when creating a firewall rule base. Based on those combinatorics, an artifact (method) is proposed to create a firewall rule base which has evolvability in its design.**

Keywords–Normalized Systems; Firewall; Rule base

I. INTRODUCTION

Firewalls are an essential component of network security. They have been protecting network connected resources for over 25 years and will continue to do so for the next decades [14] [15]. Initially, firewalls were used to protect a company against threats coming from the outside (i.e., the "evil Internet"). This is referred to as filtering North-South traffic [20]. But security breaches are not only caused by access through the Internet. A significant portion of security breaches are caused from within the company network [18] where hacks have become more sophisticated. Getting a foothold on one resource on the internal network and from there on hopping between resources, is a known hacking strategy against which filtering North-South traffic offers no protection. For this reason, protecting the network connected resources from internal traffic, referred to as East-West traffic [20], is gaining ground.

Networks are becoming more and more complex: they often contain multiple firewalls, which protect multiple network segments. The rule base of those firewalls (i.e., the definitions of which traffic is allowed or not) is becoming equally complex, up to the point where it becomes almost unmanageable. In a survey organized by Firemon [13], 73 % of survey participants stated that their firewall ranges from "somewhat complex" to "out of control". Further, complexity is the challenge which was ranked the highest for firewall management [14] [15].

The firewall rule base is a classic example of a system that needs to evolve over time. It starts with one firewall, and two network segments and filtering rules between them. As the network grows, the number of resources connected to the network grows, the number of services offered on the network grows and the number of security threats grows. The resulting firewall rule base will enlarge dramatically. This evolution will at some point result in a rule base where normal changes (i.e., the addition of a rule or the removal of a rule) result in unforeseen side effects. Those effects are proportional to the size of the rule base: the bigger the system (rule base), the worse it gets [14].

Normalized Systems (NS) theory [23]–[27] studies combinatorics in modular systems and provides a set of theorems to design modular systems exhibiting ex-ante proven evolvability. Here, the goal is to avoid so-called combinatorial effects (CE). CE's are impacts which are proportional to the type of change as well as the size of the system to which the change is applied. When all modules of a system respect the NS theorems, the system will be free of such CE's. At that point, the system can be considered stable under change with respect to a set of anticipated changes (such as adding and removing components from the system).

There are multiple vendors which sell tools to analyze a firewall rule base and can even be used to simplify it (e.g., Firemon, Tufin, Algosec). Some academic research on such analyses is available as well. Both industry and academics seem to focus on improving existing rule bases. However, a more ambitious objective would be to avoid this type of problems upfront through the deliberate design of the rule base and incorporate evolvability by design.

This paper will study the combinatorics involved in the firewall rule base. We will propose an artifact (a method), which translates the general NS theorems into a set of firewall rule base principles. When applied, this will result in an ex-ante proven evolvable (free of CE) rule base with respect to the addition and removal of rules to the firewall rule base.

The paper uses a Design Science approach [28] [29]. Therefore, Section 2 starts with explaining some firewall basics and explains the evolvability issues of a firewall rule base. In Section 3, the artifact goals and design are described. Different changes will be applied on a rule base created with the artifact to demonstrate the evolvability in Section 4. In Section 5, the artifact is evaluated based on the demonstration. Section 6 includes a literature review to link the newly created artifact with existing work, and points out the weaknesses of the artifact which also includes some suggestions for future research. Finally, our conclusions are offered in Section 7.

This paper elaborates on earlier research [27], where the

 ISBN: 978-1-61208-708-5

applicability of NS for IT infrastructure systems was being explored. The current paper focuses on a practical case where NS and domain specific knowledge on firewalls are combined resulting in a design strategy for an evolvable firewall rule base.

II. General background and problem description

In this section, some fundamental concepts about firewalls will be explained, followed by a summary of the issues regarding the evolvability of a firewall rule base. The section continues with explaining the notion of firewall group objects, their value and related issues, and finishes with a brief explanation of the "Zero Trust" concept, which is one of the design objectives of the envisioned artifact.

A. Firewall concepts

An IPV4 TCP/IP based firewall can filter traffic between TCP/IP network connected resources based on Layer 3 (IP addresses) and 4 (TCP/UDP ports) information of those resources [21] [22]. The firewall must be in the network path between the resources. Filtering happens by making use of rules. A rule is a tuple containing the following elements: <Source IP, Destination IP, Destination Port, Protocol, Action>. IP stands for IP address and is a 32 bit number which uniquely identifies a networked resource on a TCP/IP based network. The rule is evaluated by the firewall, meaning that when it 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>, then the firewall will perform an action = <Action>. The action can be "Allow" or "Deny".

A firewall rule base is a collection of order-sensitive rules. The firewall will evaluate all inbound traffic against the ordered rule base, starting at the top of the rule base untill it encounters the first rule that matches the criteria (Source, Destination, Destination Port, Protocol) of the traffic, and perform the action as specified in the rule it hits. In a firewall rule, <Source IP>, <Destination IP>, <Destination Port> and <Protocol> can be one value or a range of values. Protocol can be TCP or UDP. In the remainder of this document, the notion of protocol is omitted as it can be included in the Port variable (for example TCP port 58 or UDP port 58).

B. Firewall evolvability issues

A typical firewall rule base might become complex over time and consist of many different rules that can have different types of relations with regard to each other. In [2], the following relations are defined between rules:

- **Disjoint:** Two rules **R1** and **R2** are disjoint, if they have at least one criterion (source, destination, port, action) for which they have completely disjoint values (= no overlap or match).
- **Exactly Matching:** Two rules **R1** and **R2** are exactly matched, if each criterion (source, destination, port, action) of the rules match exactly.
- **Inclusively Matching:** A rule **R1** is a subset, or inclusively matched to another rule **R2**, if there exists at least one criterion (source, destination, port, action) for which **R1**'s value is a subset of **R2**'s value and for the remaining attributes, **R1**'s value is equal to **R2**'s value
- Correlated: Two rules **R1** and **R2** are correlated, if **R1** and **R2** are not disjoint, but neither is a subset of the other.

Exactly matching, inclusively matching and correlated rules can result in the following firewall anomalies [2]:

- *Shadowing Anomaly*: A rule **R1** is shadowed by another rule **R2** if **R2** precedes **R1** in the policy, and **R2** can match all the packets matched by **R1**. The result is that **R1** is never activated.
- *Correlation Anomaly*: Two rules **R1** and **R2** are correlated if they have different filtering actions and **R1** matches some packets that match **R2** and **R2** matches some packets that **R1** matches.
- *Redundancy Anomaly*: A redundant rule **R1** performs the same action on the same packets as another rule **R2** so that if **R1** is removed the security policy will not be affected.

A fully consistent rule base should only contain disjoint rules, as in that case, the order of the rules in the rule base is of no importance and the anomalies described above will not occur [2] [8]–[12]). However, due to a number of reasons such as unclear requirements, a faulty change process, lack of organization, manual interventions and system complexity [13], the rule base will include correlated, exactly matching and inclusively matching rules. Combined with the order-sensitivity of the rule base, changes to the rule base (the addition or removal of a rule) can result in unforeseen side effects. To be confident that a change will not introduce unforeseen side effects, the whole rule base needs to be analyzed. Therefore, the effect of the change is proportional to the change and the size of the system, being the complete rule base. According to NS, this is a CE. As a result, a firewall rule base containing rules other than disjoint rules, is unstable under change.

C. Firewall group objects

A rule base which is made up of IP's as Source/Destination and port numbers is difficult to interpret by humans. It is just a bunch of numbers. 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 then populated with IP addresses or ports. Groups can be nested.

Although it should improve the manageability of the firewall, using groups can easily result in the introduction of exactly matching, inclusively matching or correlated rules.

Example:

"Group_Windows_APP" and "Group_Windows_APPS" could be two groups with each contain the IP addresses of all Windows Application Servers. The second group may have been created without knowledge of the existence of the other [13], introducing exactly matching rules. The group memberships may start to deviate from each other, introducing correlated or inclusively matching rules, which could lead to anomalies in the rule base. The group structure must be well designed to avoid this.

D. Zero Trust

In [18] [19] [20] Forrester advocates the usage of a Zero Trust model:

- Ensure all resources are accessed securely, regardless of location and hosting model,

- Adapt a “least privilege” strategy and strictly enforce access control,
- Inspect and log all traffic for suspicious activity.

The working assumption in the case of protecting network connected resources, is that all traffic towards those resources is considered a threat and must be inspected and secured. A network connected resource should only expose those services via the network which are minimally required and each network connected resource should only be allowed access to what it really needs.

III. Creating an artifact for an evolvable rule base

Based on the analysis of the problem space in the previous section, the objective is:

- To create a rule base compliant with the “Zero Trust” concept.
- To create a rule base which only contains disjoint rules.
- To create a rule base making use of firewall group objects to improve readability and manageability.
- To create a rule base which is evolvable with respect to the following anticipated changes: the addition and removal of rules.

NS will be used to structure this evolvable rule base.

In order to obtain this goal, this section starts with the investigation of the modular structure of a firewall rule base and is followed by a discussion of the issues which surface when the modular structure is instantiated. The section continues with a set of formal definitions of the firewall rule base components, from which the combinatorics are derived when creating a firewall rule base. Based on these combinatorics, the design rules for the evolvable rule base are distilled and translated into the actual artifact.

A. Modular structure of the rule base

A rule base is the aggregation of rules. A rule is the aggregation of: Source, Destination, Service and Action. Source is the aggregation of Clients requiring services. Destination is the aggregation of Hosts offering services. Service is the aggregation of Ports (combination of port number and protocol) which compose a service. Figure 1 represents the implicit modular data structure of a rule base in a firewall. Implicit because firewall vendors do not publish the internal data structure they use, but based on the input one needs to enter a rule in the rule base. Therefore, we assume that the model is a sufficient representation of a firewall rule base. In NS terms, the modular structure would be considered as evolvable when “Separation of Concern” is respected, as the theorems “Separation of State” and “Data and Action Version Transparency” are not relevant. As each of the mentioned modules seems to be focused on one concern, one tends to conclude that the design of a rule base can be considered as stable under change.

B. Module instantiation

If the modular structure of the rule base seems to be stable under change, then where does the problem of non-evolvable rule bases come from? In this respect, it is important to be aware of the fact that a firewall rule base is an order-sensitive system. More specifically, each instantiation of a rule must be given the correct place in the rule base or the rule will have an impact on existing rules (see Section II). Due to this, there

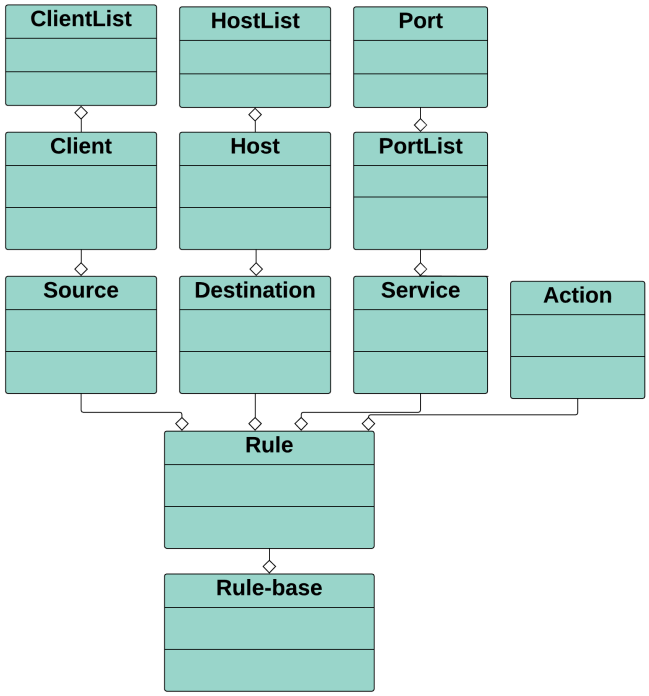

Figure 1. Modular Structure of a rule base

are evolvability issues at the level of the instantiations of the modular structure. Indeed, it seems that —in some specific situations— certain evolvability issues of a modular structure only show up at runtime. Therefore, it is interesting to look at the application of the NS theorems at this instantiation level as well. In the context of this research, this would mean that we need to look whether the addition or removal of instantiations (of rules) can result in CE’s and thus evolvability issues making an operational system unmanageable. Here, eliminating the order-sensitivity of the rule base is the key to the problem. In order to do this, the rule base can only contain disjoint rules. Disjoint rules have no coupling with other rules, and are thus compliant with the “Separation of Concern” theorem of NS.

C. Formal definitions of rule base components

Let **N** represent a Layer 4 TCP/IP based network, in which 2 groups of network connected resources can be defined:

- The hosts, providing network services via TCP/IP ports.
- The clients, requiring access to the services offered by the host.

The network contains a firewall with configuration **F**, which is configured in a way that only certain clients have access to certain services on certain hosts. The “Zero Trust” principle should be applied, meaning that clients have only access to those services on hosts to which they have been given explicit access.

Let **Port** represent a Layer 4 TCP/IP defined port.

- Port.name = the name of the port.
- Port.protocol = the layer 4 TCP/IP protocol, being one of the following two values: TCP or UDP.
- Port.number = the number of the port, represented as an integer ranging from 1 to 2^{16}.

 ISBN: 978-1-61208-708-5

Let **P** represent the list of **Ports**, of length = pj .

- **P**[1] ... **P**[pj].
- **P**[j] contains a **Port**.
- $1 \leq j \leq pj$.

Let **Service** represent a network service accessible via a list of layer 4 TCP/IP ports.

- **Service**.name = name of the service.
- **Service**.ports = list of ports = **P**.

Let **S** represent a list of **Services**, of length = sj.

- **S**[1] ... **S**[sj].
- **S**[i] contains a **Service**.
- $1 \leq i \leq sj$.

Let **Host** represent a network host which provides services.

- **Host**.name = the Fully Qualified Domain Name (FQDN) of the network host.
- **Host**.IP = the IP address of the network host.

Let **H** represent a list of **Hosts**, of length = hj. The length of **H** is a function of the network **N**.

- **H**[1] ... **H**[hj].
- **H**[k] contains a **Host**.
- $1 \leq k \leq hj$.
- hj = f_h(**N**)

Let **Client** represent a network client that requires access to hosted services.

- **Client**.name = the FQDN of the network client.
- **Client**.IP = the IP address of the network client.

Let **C** represent a list of **Clients**, of length = cj. The length of **C** is a function of the network **N**.

- **C**[1] ... **C**[cj].
- **C**[l] contains a **Client**.
- $1 \leq l \leq cj$.
- cj = f_c(**N**)

Let **R** represent a firewall rule.

- **R**.Source = a list of Clients **Cs** of length = csj, where
  - $1 \leq csj \leq cj$
  - **Cs** $\subset$ **C**
- **R**.Destination = a list of Hosts **Hd** of length = hdj, where
  - $1 \leq hdj \leq hj$.
  - **Hd** $\subset$ **H**.
- **R**.Ports = a list of Ports = a Service **Sp**
  - where **Sp** $\in$ **S**$[sj]$.
- **R**.Action = either "Allow" of "Deny".

Let **F**, representing a list of rules **R** of length = fj, be the ordered firewall rule base **F**

- **F**[1] ... **F**[fj]
- **F**[m] contains a firewall rule **R**
- $1 \leq m \leq fj$

F is order-sensitive. If **R**x is a firewall rule at location y in **F**, then the behavior of the firewall can be different if **Rx** is located at position z instead of y, where z:1$\rightarrow$ fj and z $\neq$ y. Whether or not the behavior is different depends on the relation **R**x has with the other rules of **F**.

D. Combinatorics

1) Ports: Port numbers are represented by 16 bit binary number and thus go from 1 to 2^{16}. Assuming that only TCP and UDP protocols are considered for OSI Layer 4 filtering, the possible number of values for Ports is equal to $2.2^{16} = 2^{17}$.

2) Services: **S** is the list of all possible services delivered via all ports exposed on the network **N**.
$\mathbf{S_{max}}$ is the largest possible list of services, with length = sj_{max}, in which all possible combinations of possible **Ports** are being used, where

$$sj_{\max} = \sum_{k=1}^{2^{17}} \binom{2^{17}}{k} \tag{1}$$

3) Hosts: The size of the list **H**, hj, is function of the network **N** and expressed as hj = f_h(**N**).
$\mathbf{H_{max}}$ is the list of all possible lists of hosts which are part of **H**. The length of this list is hj_{max}, where

$$hj_{\max} = \sum_{a=1}^{hj} \binom{hj}{a} \tag{2}$$

and where hj = f_h(**N**).

4) Services on Host: The maximum number of Hosts/Services combinations = $hj_{max}.sj_{max}$ =

$$hj_{\max}.sj_{\max} = \left(\sum_{a=1}^{hj} \binom{hj}{a}\right) \cdot \left(\sum_{k=1}^{2^{17}} \binom{2^{17}}{k}\right) \tag{3}$$

where hj = f_h(**N**).

5) Clients: The size of the list **C**, cj, is a function of the network **N**. and expressed as cj = f_c(**N**).
$\mathbf{C_{max}}$ is the list of all possible lists of clients which are part of **C**. The length of this list is cj_{max} where

$$cj_{\max} = \sum_{a=1}^{cj} \binom{cj}{a} \tag{4}$$

where cj = f_c(**N**).

6) Rules and rule base: In a rule **R**,

- **R**.Source can contain any element of $\mathbf{C_{max}}$.
- **R**.Destination can contain any element of $\mathbf{H_{max}}$.
- **R**.Ports can contain any element of $\mathbf{S_{max}}$.
- **R**.Action is the maximum number of action combinations, which is 2 ("Allow" or "Deny")

The firewall rule base $\mathbf{F_{max}}$ contains all possible rules which can be made with $\mathbf{C_{max}}$, $\mathbf{H_{max}}$ and $\mathbf{S_{max}}$

$$fj_{\max} = cj_{\max}.hj_{\max}.sj_{\max}.2 \tag{5}$$

$$fj_{\max} = 2.\left(\sum_{a=1}^{cj} \binom{cj}{a}\right) \cdot \left(\sum_{a=1}^{hj} \binom{hj}{a}\right) \cdot \left(\sum_{k=1}^{2^{17}} \binom{2^{17}}{k}\right) \tag{6}$$

where cj = f_c(**N**) and hj = f_h(**N**)

The possible design space for a rule base is phenomenal. Multiple rules can deliver one particular required functionality. Choosing the right rule is a real challenge. As the network grows and f_c(**N**) and f_h(**N**) grow, choosing the right firewall rule from the design space becomes even more challenging. To gain control over the design space, it needs to be consciously reduced.

 ISBN: 978-1-61208-708-5

E. Designing an evolvable rule base

A rule will be made up of:

- **Cs** representing the Source, where **Cs**$\subset$ **C**.
- **Hd** representing the Destination, where **Hd**$\subset$ **H**.
- **Sp** representing the Ports, where **Sp** $\in$ **S**.
- Action is to be "Allow" as each rule in the rule base explicitly provides access to allowed services on allowed hosts.
- **R** = (**Cs**, **Hd**, **Sp**, "Allow')

Note that the last rule in the rule base **F**, **F**[fj] has to be the default deny rule ($\mathbf{R}_{\text{default\_deny}}$) as, when no rule explicitly provides access to a service on a host, the traffic needs to be explicitly blocked.

- $\mathbf{R}_{\text{default\_deny}}$.Source = ANY,
- $\mathbf{R}_{\text{default\_deny}}$.Destination=ANY,
- $\mathbf{R}_{\text{default\_deny}}$.Port= ANY,
- $\mathbf{R}_{\text{default\_deny}}$.Action = "Deny".

From Section II-B, it is known that:

- A Firewall rule base is order-sensitive.
- Different types of relations/coupling can exist between rules.
- If all rules are disjoint from each other, there is no coupling between the rules.
- If all rules are disjoint, the rule base is no longer order-sensitive.
- If a new rule which is added to the rule base is disjoint with all existing rules, the location of the rule in the rule base is not important.

If the whole firewall rule base needs to be checked to see if the a rule is disjoint to all existing rules, a CE is being introduced. Introducing a new rule to, or removing a rule from the system should result in work which is proportional to the newly required functionality and not into work which has no logical link to the required functionality and which requires searching throughout the whole system (being the entire rule base). Or as NS formulates it: the impact of the change should be proportional to the nature of the change itself, and not proportional to the system to which the change is applied.
Disjoint rules have no overlap in source or destination or ports. This gives the following combinations:

- No overlap in sources - don't care about destination and port overlaps.
- No overlap in destinations - don't care about source and port overlaps.
- No overlap in ports - don't care about source and destination overlap.
- No overlap in source-destination combination, don't care about ports.
- No overlap in source-ports combinations, don't care about destinations.
- No overlap in destination-ports combinations, don't care about sources.
- No overlap in source-destination-port combination.

Cs is $f_c(\mathbf{N})$ and **Hd** is $f_h(\mathbf{N})$. The network is an uncontrollable variable. Trying to find a way to structure **Cs** and **Hd** to allow for disjoint rules starting from this variable, will not yield to anything useful. On the other hand, **S** represents the ports and is bound: the nature of TCP/IP limits the amount of possible ports and thus all port combinations. It thus makes sense to look for a way to guarantee that there is no overlap at port/service level.

Let us consciously restrict **S** to **Su**, so that **Su** only contains unique values.
$\exists!$**Su**[m] in **Su**.
Su[u] $\cap$ **Su**[v] = $\emptyset$, where u, v:1$\rightarrow$suj, and u $\neq$ v

If each service is represented by 1 port, **Su** will contain 2^{17} elements, which is the max size of **Su** in this restricted case.
The service **Su**[m] can be delivered by many hosts.
Let $\mathbf{Hd}_{\mathbf{Su}[m]}$ represent the list of hosts which offer service **Su**[m].
$\mathbf{Hd}_{\mathbf{Su}[m]} \subset$ **Hd** and $\mathbf{Hd}_{\mathbf{Su}[m]}$[x] contains a single host.
$\mathbf{Hd}_{\mathbf{S}[m]}$ contains unique and disjoint elements.

Combining hosts and services gives, ($\mathbf{Hd}_{\mathbf{Su}[m]}$[x],**Su**[m]) where x:1$\rightarrow$hdmj, a list of tuples which are disjoint. This hold for all m:1$\rightarrow$hdmj. At his point, all services and hosts who deliver the services, form tuples which are disjoint and can thus be used as a basis for creating an order independent firewall rule base. $\mathbf{Cs}_{\mathbf{Hd}_{\mathbf{Su}[m]}[x]}$ is the list of clients which have access to service **Su**[m], defined on host $\mathbf{Hd}_{\mathbf{Su}[m]}$[x].
By using :

- **Su**[m] where m:1$\rightarrow$suj, with suj=number of disjoint services offered on the network, for defining **R**.Port
- $\mathbf{Hd}_{\mathbf{Su}[m]}$[x], x:$\rightarrow$hdmj, with hdmj=number of hosts offering **Su**[m], for defining **R**.Destination
- $\mathbf{Cs}_{\mathbf{Hd}_{\mathbf{Su}[m]}[x]}$ being the list of clients requiring access to service **Su**[m] on host $\mathbf{Hd}_{\mathbf{Su}[m]}$[x], of length = cjs, for defining **R**.Source
- "Allow", for **R**.action

disjoint rules are being created, usable for an evolvable firewall rule base.

F. The artifact

What has been discussed in the previous section needs to be transformed into a solution usable in a real firewall. As discussed in Section II-C, firewalls work with groups. Groups can be used to represent the concepts discussed in the previous sections.

1) Starting from an empty firewall rule base **F**. Add as first rule the default deny rule **F**[1]= $\mathbf{R}_{\text{default\_deny}}$ with
   - $\mathbf{R}_{\text{default\_deny}}$.Source = ANY,
   - $\mathbf{R}_{\text{default\_deny}}$.Destination=ANY,
   - $\mathbf{R}_{\text{default\_deny}}$.Port= ANY,
   - $\mathbf{R}_{\text{default\_deny}}$.Action = "Deny".
2) For each service offered on the network, create a group. All service groups need to be completely disjoint from each other: the intersection between groups must be empty.
   Naming convention to follow:
   - **S**_*service.name*,
   - with *service.name* as the name of the service.
3) For each host offering the service defined in the previous step, a group must be created containing only one item (being the host offering that specific service).
   Naming convention to follow:
   - **H**_*host.name*_**S**_*service.name*,
   - with *host.name* as the name of the host offering the service

4) For each host offering the service from the first step, a client group must be created. That group will contain all clients requiring access to the specific service on the specific host.
Naming convention to follow:
- **C_H**_*host.name*_**S**_*service.name*

5) For each **S**_*service.name*,**H**_*host.name*_**S**_*service.name* combination, create a rule **R** with:
- **R**.Source =**C_H**_*host.name*_**S**_*service.name*
- **R**.Destination = **H**_*host.name*_**S**_*service.name*
- **R**.Port= **S**_*service.name*
- **R**.Action = "Allow"

Add those rules to the firewall rule base **F**.
The default rule $\mathbf{R}_{\text{default}}$ should always be at the end of the rule base.

By using the artifact's design principles, group objects are created which form the building blocks for an evolvable rule base. Each building brocks addresses one concern.
If each service of **Su** is made up of only one Port, then the **Su** will contain maximum 2^{17} elements, resulting in maximum 2^{17} service groups **S**_*service.name* being created. For each host, maximum 2^{17} services can be defined, expressed in **H**_*host.name*_**S**_*service.name* destination groups. According to the artifact, one rule per host and per service, must be created. This reduced the rule base solution space from

$$2.\left(\sum_{a=1}^{cj}\binom{cj}{a}\right)\cdot\left(\sum_{a=1}^{hj}\binom{hj}{a}\right)\cdot\left(\sum_{k=1}^{2^{17}}\binom{2^{17}}{k}\right) \quad (7)$$

where cj = *fc*(**N**) and hj = *fh*(**N**)
to:

$$fj = hdj.suj = hdj.2^{17} + 1 \quad (8)$$

with hdj = number of hosts connected to the network.
hdj = *fh*(**N**). The "+1" is the default deny rule $\mathbf{R}_{\text{default\_deny}}$

IV. DEMONSTRATE ARTIFACT

To demonstrate the artifact, we will investigate the effect of different changes on the rule base (add/remove rule) and the components making up the rule base (add/remove a service, add/remove a host, add/remove a client). We also show what happens if rules would be aggregated.

A. Add and remove a rule

Creating rules according to the artifact's design principles, leads to rules which are disjoint from each other. Disjoint rules can be added and removed from the firewall rule base without introducing CE's.

B. Adding a new service to the network

A new service is a service which is not already defined in **Su**. The new services results in a new definition of a service being added to **Su**. The artifact prescribes that a new group **S**_*service.name* must be created for the new service. The group will contain the ports required for the service. For each new host offering the service, the artifact prescribes to create a new group destination **H**_*host.name*_**S**_*service.name*, and an associated source group **C_H**_*host.name*_**S**_*service.name*. The destination groups are populated with only one host (the host offering the service), and the source groups are populated with all clients requiring access to the service on specific host. All building blocks to create the disjoint rules are now available. For each host offering the new service, a rule must be created using the created groups. No CE's are being introduced during these operations. Adding the new rules to the rule base does not introduce CE's (see Section IV-A).

C. Adding a new host offering existing services, to the network

A new host is a host which is not already defined in **Hd**. The new host results in a new host definition being added to **Hd**. The artifact prescribes that a new group **H**_*host.name*_**S**_*service.name* must be created for each service delivered by the host and a corresponding source group **C_H**_*hostname*_**S**_*service.name* must be created as well. The destination groups are populated by their corresponding host and the sources groups are populated with all clients requiring access to the service on that host. All building blocks to create the disjoint rules are now available. For each service offered by the new host, a rule must be created using the created groups. No CE's are being introduced during these operations. Adding the new rules to the rule base does not introduce CE's (see sectionIV-A).

D. Adding a new host offering new services, to the network

Combining Sections IV-C and IV-B delivers what is required to complete this type of change. The artifact prescribes that new service groups must be created for new services. An equal amount of destination groups needs to be created and each populated by the new host. An equal amount of source groups needs to be created and populated by the clients requiring access to one of the new services on the new host. All building blocks to create the disjoint rules are now available. For each combination (new host, new service) a rule must be created using the created groups. No CE's are being introduced during these operations. Adding the new rules to the rule base does not introduce CE's (see Section IV-A).

E. Adding a new client to the network

Adding a new client to the network does not require the creation of new rule building blocks or the addition of new rules. The new client only requires to be added to those source groups which give access to the services it requires on the hosts it needs access to. No CE's are being introduced during these operations.

F. Removing a service from the network

Let **sr** be the service which needs to be removed from the network. The name of the service is **sr**.name=sremove. The service is part of **Su**. The group corresponding with **sr** is **S**_sremove. The hosts offering the service correspond with the groups **H**_*host.name*_**S**_sremove. The clients consuming the service are defined in **C_H**_*host.name*_**S**_sremove. All building blocks to identify the rules which require removing from the rule base are now available. For each host offering **sr**, the corresponding rule

- $\mathbf{R}_{\text{default\_deny}}$.Source = **C_H**_*host.name*_**S**_sremove
- $\mathbf{R}_{\text{default\_deny}}$.Destination=**H**_*host.name*_**S**_sremove,
- $\mathbf{R}_{\text{default\_deny}}$.Port= **S**_sremove,
- $\mathbf{R}_{\text{default\_deny}}$.Action = "Allow".

must be removed from the rule base. No CE's are being introduced during these operations. Removing rules from the rule base does not introduce CE's (see Section IV-A). The service **sr** needs to be removed from **Su** as well as the corresponding group **S**_remove in the firewall.

G. Removing a host from the network

Let **hr** be the host that needs to be removed from the network. The name of the host is **hr**.name=hremove. The host is part of **Hd**. There will be as much destination groups for **hr** as there are services offered by **hr**. They are defined by **H**_hremove_**S**_*service_name*. The same holds form the source groups, defined by **C_H**_hremove_**S**_*service.name*. All building blocks to identify the rules which require removal from the rule base are available. For each service offered by **hr**, the corresponding rule

- $\mathbf{R}_{\text{default\_deny}}$.Source = **C_H**_hremove_**S**_*service.name*
- $\mathbf{R}_{\text{default\_deny}}$.Destination=**H**_hremove_**S**_*service_name*
- $\mathbf{R}_{\text{default\_deny}}$.Port= **S**_*service.name*,
- $\mathbf{R}_{\text{default\_deny}}$.Action = "Allow".

must be removed from the rule base. No CE's are being introduced during these operations. Removing rules from the rule base does not introduce CE's (see Section IV-A). The host **hr** needs to be removed from **Hd** and the corresponding groups **H**_remove_**S**_*service.name* in the firewall, must be removed as well.

H. Removing a service from a host

Let **sr** be the services with **sr**.name=sremove, which needs removing from host **hr** with **hr**.name = hremove. The service is part of **Su**. The group corresponding with **sr** is **S**_sremove. The destination group for service **sr** on host **hr**, is **H**_hremove_**S**_sremove. The corresponding source group is **C_H**_hremove_**S**_sremove. All building blocks to identify the rule

- $\mathbf{R}_{\text{default\_deny}}$.Source = **C_H**_hremove_**S**_sremove
- $\mathbf{R}_{\text{default\_deny}}$.Destination=**H**_hremove_**S**_sremove
- $\mathbf{R}_{\text{default\_deny}}$.Port= **S**_sremove,
- $\mathbf{R}_{\text{default\_deny}}$.Action = "Allow".

which require removing from the rule base are available. No CE's are being introduced during these operations. Removing rules from the rule base does not introduce CE's (see Section IV-A). The service **sr** does not need to be removed from **Su** and neither does the corresponding group as the service is till offered on other hosts.

I. Removing a client from the network

Let **cr** be a client that needs to be removed from the network. The client is part of **Cs**. Removing a client from the network does not require removing rules from the rule base. The client needs to be removed from the different source groups which provide the client access to specific services on specific hosts. If the services and hosts to which the client has access are known, then the source group from which the client needs to be removed, are known as well. If the services and/or hosts are not known, then an investigation of all the source groups is required to see if the client is part of the group or not. If part of the group, the client needs to be removed. The client also needs to be removed from **Cs**. Determining if a client is part of a source group can be considered as a CE as all source groups require inspection. This will further be elaborated upon in Section VI.

J. The impact of aggregations

When following the prescriptions of the artifact, many groups and rules will be created (see Section V for more details). The urge to aggregate and consolidate rules into more general rules, will be a natural inclination of firewall administrators as a smaller rule base will be (wrongfully) considered as a less complex rule base. However, any form of aggregation will result in loss of information. It is because the artifact consciously enforces fine-grained information in the group naming and usages, that disjoint rules can be created and the "Zero Trust" model can be enforced. If due to aggregations it can no longer be guaranteed that rules are disjoint, then a CE-free rule base can no longer be guaranteed either. Aggregation will also lead to violations of the "Zero Trust" model.

We provide 2 examples of aggregations.

Aggregation at service level: all hosts offering the same service are aggregated into one destination group. Such an aggregation excludes the possibility of specifying that a client needs access to a specific service on a specific host. A client will have access to the service on all hosts offering the service, desired or not. In such a configuration, "Zero Trust" can no longer be guaranteed. As long as the services on the network are unique, so will be the port groups. Rules will stay disjoint and the rule base CE-free. The moment that one starts combining "Zero Trust" and non "Zero Trust" rules, non-disjoint rule will pop-up and the rule base can no longer be guaranteed to be CE-free.
Example: if for some reason, it cannot be allowed that a client has access to the service on all hosts and a special service group is being created (no longer disjoint with the existing service group) with a special associated destination group (no longer disjoint with existing destination groups), the rule created with those groups is not disjoint with existing rules in the rule base and the effect of adding this rule to the rule base is no longer guaranteed CE-free.

Aggregation at host level: all services offered on a host, are aggregated into one host-bound port/service group. Such an aggregation excludes the possibility of specifying that a client needs access to some of the services on the host. A client will have access to all services defined on the host, desired or not. In such a configuration, "Zero Trust" can no longer be guaranteed. As long as the destination groups are unique, disjoint rules can still be created. The moment that one starts combining "Zero Trust" and non "Zero Trust" rules, non-disjoint rule will pop-up and the rule base can not longer be guaranteed CE-free.
Example: if for some reason, it cannot be allowed that a client has access to all services on the host and a special service group is being created (no longer disjoint with existing service groups) with a special associated destination group (no longer disjoint with existing destination group), the rule created with those groups is not disjoint with existing rules in the rule base and the effect of adding this rule to the rule base is no longer guaranteed CE-free.

V. EVALUATION

The previous section demonstrates that, when applying the artifact, the rules are guaranteed to be disjoint and adding and removing such rules has no unwanted side effects on the existing rule base. Such a rule base will be fine-grained (i.e., having many rules). One might consider this as an important drawback from using such an approach as the idea of a fine-grained structure is often considered as a complex one (but this

 ISBN: 978-1-61208-708-5

does not necessarily has to be the case). Some operations on rules may indeed result in CE's at group level, such as adding and removing a client from the network. In Section VI, this will further be elaborated upon.
Aggregations will violate the "Zero Trust" constraint. Combining aggregation and non-aggregation based rules results in non-disjoint rules and CE's at rule base level.
"Zero Trust" has been defined at host level but one can imagine a setup where "Zero Trust" is defined at another level such as VLAN's. A VLAN is a section of the network with a continuous IP range (for example 10.10.10.1 till 10.10.10.254). In such a setup, instead of "specific service on a specific host", the "Zero Trust" could become a "specific service on a specific VLAN". If in Section III, **H** would be replaced by **V**, the list of VLAN's, and the content of **V** could be used to create destination groups **V**_vlan.name_**S**_service.name with source groups **C**_**V**_vlan.name_**S**_service.name, the building blocks would be created for disjoint rules respecting a "Zero Trust" at VLAN level model. The artifact would still be applicable and the resulting rule base would be smaller (and less fine grained). An evolvable rule base is possible as long as one sticks to a defined level of "Zero Trust" and one does not start mixing up different levels of "Zero Trust". From the moment one starts mixing different levels, evolvability can no longer guaranteed.

VI. Discussion

By means of discussion, we will cover three items. First, we will provide a short literature review to position our research with respect to earlier contributions found in literature. Second, we will reflect on the nature of the CE's which are still present when applying our proposed artifact. And third, we will highlight some opportunities for future research.

A. Literature review

The academic literature about firewalls can be divided into 3 groups. The first group (published roughly before the year 2000)focuses on the performance of the firewall and the hardware used to perform the actual package filtering. The second group (published roughly between 2000 and 2006) focuses on the complexity and issues with the rule base of the firewall. The third group (published roughly after 2006) focuses on the firewall in a Software Define Network (SDN) context, where distributed firewalls and software defined firewalls are used. As this paper focuses on the complexity and issues related to the firewall rule base, the following literature review will only focus on the second group of papers [1]–[12]. Next to academic papers, reports from Forrester and white papers from industry leaders were used as well [13]–[20]. Those reports include surveys which give information on the current state-of-affairs. One might think that, because academic publication about rule base issues have diminished after 2006, the problem is solved. However, the surveys provide a different view. Companies are still struggling with their firewall [13]–[20]. This can be due to the "knowing-doing" gap or because the issue is not fully resolved.

Most papers start by stating that there is a problem with the firewall rule base because of:

- ***Translation issues***: how to convert a high level security policy into a low-level language of firewall rules [1]–[12] [17].
- ***Size of the rule base issues***: a large rule base is considered complex [3] [7] [9] [10] [13].
- ***Error and anomalies issues***: A rule base is error-prone due to complexity and manual interventions [2] [3] [10], [13]–[20] and can contain firewall rule conflicts or anomalies [1] [2] , [3] [6] [8]–[10] [12] [13] [16].

The ***"Translation-issue"*** is tackled by proposing tools which could translate high level security concepts into low level firewall rules. FANG [6], FIRMATO [3], LUMETA [5] are artifacts proposed and described, which help translating high level security requirements into a low level firewall rule base. There are however no guarantees that these tools deliver a small and simple firewall rule base free of anamolies [3]. Companies such as TUFIN, ALGOSEC, FIREMON, VMWare also deliver commercial tools which claim to help managing the complexity of network security. The tools do not prescribe, neither enforce how a rule base should be created in order to be free of anomalies and exhibit evolvability.

The ***"Size of the rule base issue"*** is not treated as an issue related to the stability of a system under change. To the best of our knowledge, most contributions do not focus on this point, whereas it is a corner stone of NS. The different artifacts all start with ideas similar to "For each rule in the firewall, do the following ...". One might consider such an approach as a CE in itself. There is attention to reducing the rule base to a minimum list of rules, which still answer to the filtering requirements. The motivation for this "reduction of the rule base" is performance, although in [3] it was suggested that the actual size of the rule base is not related to the way in which the hardware actually applies the rules. This suggests a decorrelation between the size of the rule base and the firewall performance. If this would be the case, why bother about the reduction of the size of the rule base? Further research of the literature and real-life measurements are required to clarify this point.
Looking at the combinatorics of Section III-D, the design space is enormous. By applying the artifact, there is a conscious reduction of the design space. But the size of the rule base is still large as for each combination (host, service) a rule must be created in the rule base.

$$2^{17}.hdj \tag{9}$$

The maximum number of services is 2^{17} = 131.075. However, in reality this number will never be reached. A sample in Engie (a multinational and world leader in energy services) on 100 servers revealed that on average 39 services are exposed. The standard deviation in the sample is 14. It can be stated with a statistical probability of 98% that a host exposes less then 67 services. The sample was taken from a population of 1000 servers. Those 1000 serves are currently protected by about 890 firewall rules. If the artifact would be applied, it would mean implementing 67.000 rules. However, at Engie, a "Zero Trust" model at host level is not applied. Instead, a "Zero Trust" at VLAN level (see Section V) is present. If the realistic assumption is a made that the 1000 server are spread over 20 VLAN's, it would mean that 20 x 67 = 1340 rules are required for an evolvability rule base. This would mean 50% more rules to gain full evolvability.
Applying the artifact will probably imply a larger rule base and

there may be worries about performance. It should be noted that a firewall vendor such as CheckPoint, suggests to put the rules which are most frequently hit (and applied) at the top of the firewall table. In a rule base which is order-sensitive, this may be a real issue. In a rule base which is not order-sensitive, one could monitor the firewall and see which rules are hit most and move those rules around without having to worry of the potential impact on other rules. Doing this dynamically would even be more powerful as the firewall would be able to reorganize his rules according to the traffic of the day.

The ***"Error issue"*** due to complexity and manual intervention is recognized and confirmed in recent surveys [13]–[20]. The academic papers focus more on the technical root causes of the errors, being the anomalies in the rule base. Over time, the definitions of the types of anomalies, their formal definition and proof, have evolved and resulted in a definition of how a firewall rule base should look like in order to remain stable under change: a firewall rule base should only include disjoint rules ([2], [8], [9], [10], [11], [12]). Artifacts have been put forward [2] [3], [7]–[9], [12] which allow to scan the rule base for non-disjoint rules and make them disjoint if required. The same artifacts allow to assess the impact of adding a new rule and adjusting the rules in such way that the rule base only contains disjoint rules. However, each time a rule is entered, the whole rule base needs to be scanned to detect potential anomalies between the existing rule base and the new rule. The effort of making a change to the system is thus proportional to the size of the system.

The literature review shows that the problems related to the firewall rule base are well known and the necessary condition to keep the rule base under control (i.e., having disjoint rules) is also known. However, clear architectural guidance on how to create a disjoint rule base as of the moment of conception, is lacking. It is exactly this architectural guidance, making use of NS, which is the main contribution of this paper. By structuring the rules in such a way that they are always disjoint, one can add and remove rules without having to analyze the rule base (source of CE) or worry about unforeseen side effects of the change.

B. Remaining CE's

The artifact proposed in the paper is not completely free of CE. The evaluation has shown that there is are CE's at the level of groups. However, these CE's are not related to the technical coupling within the rule base but due to the size and topology of the network. The bigger the network, the more objects and rules. Such CE's are considered acceptable given that:

- The actions leading to the CE can be automated (search for, or through, groups)
- The CE is predictable and is the logical effect of the change which needs to be applied (remove a client = look in all groups where the client is present)

CE's which cannot be automated because their impact is not predictable are not acceptable as there is no logical link between the change and the extra work one needs to do to implement the change. For example, the addition of a rule to activate a service on the network that would require the inspection of the whole rule base to find conflicting rules (not related to the newly activated service) would be considered as an unacceptable CE. Remark that the proposed artifact facilitates the removal of such unacceptable CE's.

C. Future research

Applying the artifact to create the rule base in a manual way, is highly inadvisable. The risk of introducing errors is too large. NS at the software level advocates the use of expanders, which will automate the creation of a skeleton code structure which has ex-ante proven evolvability. A similar expander should be built for the proposed artifact. The expander would enforce the application of the naming conventions and the disjoint rules, thus automatically enforcing the architectural principles. Future efforts could be invested in building a tool which would enforce the artifact on a firewall rule base. Like many of the tools discussed in literature, creating a system (with the desired configuration) which is positioned in parallel to the existing firewalls (which can later on be pushed towards the actual production firewall) is the most likely approach to apply.

The artifact currently focuses on a single firewall implementation. The same concept could now be investigated in a context where the network has multiple firewalls. How would this affect the rule base and which part of the rule base goes to which firewall? One could even consider a distributed firewall in a software defined setup. For this purpose, the concepts behind the proposed artifact should be extended towards distributed firewalls and software defined firewalls.

Because the consulted literature did not contain conclusive information on the impact of the size of the rule base on firewall performance, further literature review on this topic is required.

VII. Conclusion

Firewall rule bases are typically non-evolvable systems. Tools and literature exist on how to show and potentially reduce the complexity and conflicts in firewall rule bases, but practical guidance on how to make a rule base which has proven evolvability by design, is lacking. Using the NS paradigm and domain specific knowledge, we have proposed an artifact which has the desired evolvability for a firewall rule base. The most important drawback of the resulting rule base could be the size due to its fine-grained structure, although this should be further analyzed in future research efforts.

Acknowledgment

The authors would like to thank Prof. Dr. Herwig Mannaert for his guidance, Stefan Thys, Frederik Leemans, Stefan Biesbroeck for their extra input, feedback and editing, Sam Gozin and Bruno De Becker for the Engie operational data.

References

[1] P. Eronen and J. Zitting, "An expert system for analysing firewall rules", In Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec 2001),pp. 100–107, November 2001.

[2] M. Abedin et al., "Detection and Resolution of Anomalies in Firewall Policy Rules", In Proceedings of the IFIP Annual Conference Data and Applications Security and Privacy, 2006, LNCS 4127, pp. 15–29

[3] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, "Firmato: A novel firewall management toolkit", Proceedings of the 1999 IEEE Symposium on Security and Privacy,pp. 17-31,; Oakland, California, May 1999

[4] A. Wool, "Architecting the Lumeta firewall analyser", In Proceedings of the 10the USENIX Security Symposium, Washington DC, August 2001

[5] S. Hinrichs, "Policy-based management: Bridging the gap", In Proceedings of the 15th Annual Computer Security Applications Conference, Phoenix, Arizona, December 1999, IEEE Computer Society Press.

[6] A. Mayer, A. Wool, and E. Ziskind. "Fang: A firewall analysis engine", In Proceedings, IEEE Symposium on Security and Privacy,pp. 177-187, IEEE CS Press, May 2000

 ISBN: 978-1-61208-708-5

[7] S. Hazelhurst, “Algorithms for analysing firewall and router access lists”. Technical Report TR-WitsCS-1999-5, Department of Computer Science, University of the Witwatersrand, South Africa, July 1999

[8] 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

[9] E. Al-Shaer and H. Hamed, “Discovery of policy anomalies in distributed firewalls”, In Proceedings of the 23rd Conf. IEEEE Communications Soc. (INFOCOM 2004), Vol 23, No.1,pp. 2605-2616, March 2004

[10] E. Al-Shaer and H. Hamed, “Taxonomy of conflicts in network security policies”, IEEE Communications Magazine, 44(3), March 2006

[11] 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), October 2005

[12] A. Hari, S. Suri, and G.M. Parulkar., “Detecting and resolving packet filter conflicts”, In INFOCOM (3),pp. 1203-1212, March 2000.

[13] Firemon whitepaper, Firewall cleanup recommendations, URL https://www.firemon.com/resources/, [retrieved: April, 2019]

[14] Firemon whitepaper, 2017 State of the firewall , URL https://www.firemon.com/resources/, [retrieved: April, 2019]

[15] Firemon whitepaper, 2018 State of the firewall , URL https://www.firemon.com/resources/, [retrieved: April, 2019]

[16] Algosec whitepaper, Firewall Management: 5 challenges every company must address, URL https://www.algosec.com/resources/ [retrieved: April, 2019]

[17] D. Monahan EMA, Research Summary: “Network Security Policy Management tools – Tying Policies to Process, Visibility, Connectivity and Migration”, https://web.tufin.com/network-security-policy-management-tools-ema-research, [retrieved: April, 2019]

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

[19] C. Cunningham and J.Pollard, “The Eight Business and Security Benefits of Zero Trust”, Forrester Reseach November 2017

[20] M. Bennet, “Zero Trust Security: A CIO’s Guide to Defending Their Business From Cyberattacks”, Forrester Research June 2017

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

[22] H. Zimmermann, and J.D. Day, “The OSI reference model - Proceedings of the IEEE”, Volume: 71, Issue: 12, Dec 1983

[23] 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

[24] 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 to 1222, 2011

[25] H. Mannaert, J. Verelst, and K. Ven, “Towards evolvable software architectures based on systems theoretic stability”, Software Practice and Experience : Volume 42, Issue 1, 2012

[26] 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

[27] 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

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

[29] 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

 ISBN: 978-1-61208-708-5