Investigating the Creation of an Evolvable Firewall Rule Base and Guidance for Network Firewall Architecture, using the Normalized Systems Theory

Geert Haerens

Department of Management Information Systems
Faculty of Business and Economics
University of Antwerp, Belgium
and Engie IT — Dir. Architecture
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*—A firewall is an essential network security component. It protects network connected company resources from potential malicious traffic. The firewall rule base, the list of filters to be applied to network traffic, can quickly become complex up to the point where companies consider the rule base as unmanageable. The complexity leads to unforeseen and painful side effects when the firewall rule base is changed (add/remove filtering rules). Sufficient literature exists on the root cause of rule base evolvability issues. However, little research is available on how to properly construct a rule base such that the evolvability issues do not occur. Normalized Systems (NS) theory provides proven guidance on how to create evolvable modular 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, that has evolvability in its design. As a network rarely contains only one firewall, the impact of different filtering strategies and changes on multiple firewalls, is studied as well.**

Keywords–Normalized Systems; Firewall; Rule Base; Filtering Strategies.

I. INTRODUCTION

This paper is an extended version of "Using Normalized Systems to Explore the Possibility of Creating an Evolvable Firewall Rule Base" [1] 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 [2] [3]. Initially, firewalls were used to protect a company against threats coming from the outside (i.e., the "evil Internet"). Such kind of filtering is called North-South traffic filtering [4]. 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 [5] 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 [4], is gaining ground.

Networks are becoming more and more complex: they often contain multiple firewalls, which protect numerous 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 [6], 73 % of survey participants stated that their firewall ranges from "somewhat complex" to "out of control". Further, complexity is the highest-ranked challenge for firewall management [2] [3].

The firewall rule base is a classic example of a system that needs to evolve. 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 regular 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 [2].

A network rarely contains only one firewall. Large companies have networks containing many firewalls. Valuable IT assets, located in data centers, are protected by multiple layers of firewalls. A single firewall can quickly become a non-evolvable system. Multiple firewalls only make the problem worse. Besides the question on how to create the correct rule and implement it on the rule base, one also has to decide on which firewall(s) this rule should be applied.

Normalized Systems (NS) theory [7]–[11] studies combinatorics in modular systems and provides a set of theorems to design modular systems exhibiting ex-ante proven evolvability. The goal is to avoid so-called combinatorial effects (CE). CE's are impacts that 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 for a set of anticipated changes (such as adding and removing components from the system).

Multiple vendors 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 problem 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), that 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.

We will start with a literature review and relate work. The remainder of the paper is structured according to the Design Science approach [12] [13]. Therefore, Section III starts by explaining some firewall basics and explains the evolvability issues of a firewall rule base. Section IV describes the artifact goals and design. The artifact is demonstrated (apply changes to a rule base) and evaluated in Section V. Section VI elaborates on different filtering strategies and Section VII will address the problems and possible solutions related to multiple firewalls. In Section VIII, automation and scaling of the propose solution is discussed and a link is made with the concept of Software Defined Network. In Section IX a part of the literature review is revised and weaknesses of the artifact are pointed out. Finally, Section X wraps up the paper and proposes future research.

This article builds on earlier research [11], where the 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 and network firewall architecture.

II. Literature review and related work

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 [14]–[25]. To the best of our knowledge, we did not find papers which specifically address and try to solve the evolvability issues of the firewall rule base. Next to academic papers, reports from Forrester and white papers from industry leaders were used as well [2]–[6], [26]–[28]. 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 [2]–[6], [26]–[28]. 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 [14]–[25] [26].
- ***Size of the rule base issues***: a large rule base is considered complex [6] [16] [20] [22] [23].
- ***Error and anomalies issues***: A rule base is error-prone due to complexity and manual interventions [2]–[6], [15], [16], [23], [26]–[28] and can contain firewall rule conflicts or anomalies [6], [14]–[16], [19], [21]–[23], [25], [27].

The ***"Translation-issue"*** is tackled by proposing tools, which could translate high level security concepts into low level firewall rules. FANG [19], FIRMATO [16], LUMETA [18] 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 [16]. 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"*** receives a lot of attention. Effort is put in reducing the rule base to a minimum list of rules, that still answer to the filtering requirements. The motivation for this "reduction of the rule base" is performance, although in [16] it was suggested that the actual size of the rule base is not related to the way the hardware actually applies the rules. This suggests a decorrelation between the size of the rule base and the firewall performance. This point will be revisited in Section IX.

The ***"Error issue"*** due to complexity and manual intervention is recognized and confirmed in recent surveys [2]–[6], [26]–[28]. 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 [15] [21] [22] [23] [24] [25]. Artifacts have been put forward [15] [16], [20]–[22], [25], 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 or worry about unforeseen side effects of the change.

III. General background and problem description

This section explains some fundamental concepts about firewalls, followed by a summary of the issues regarding the evolvability of a firewall rule base. The section continues by explaining the notion of firewall group objects, their value, and related issues. The section continues with a brief explanation of the Zero Trust (ZT) filtering strategy, which is one of the design objectives of the envisioned artifact, and terminates with an introduction to the Normalized Systems Theory.

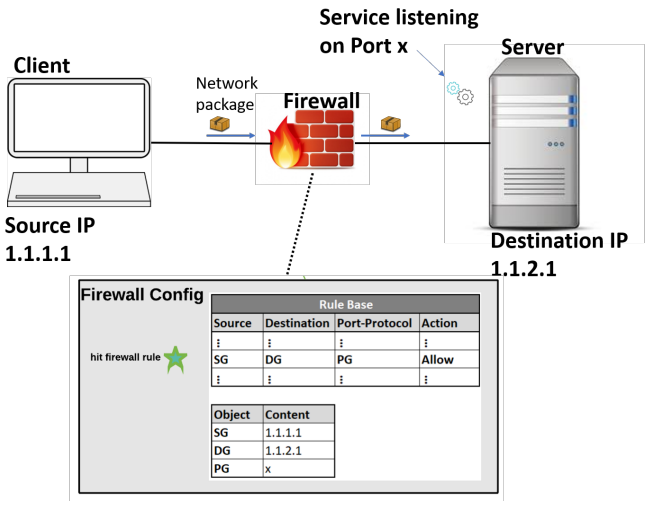

Figure 1. Firewall concepts

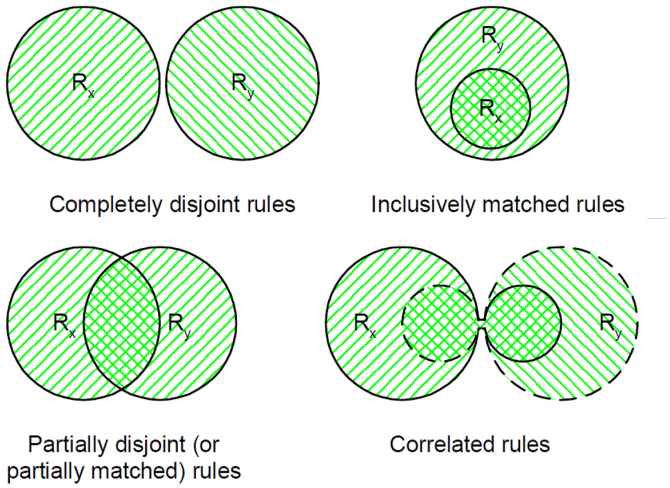

Figure 2. Possible relationships between rules (from [21])

A. *Firewall concepts*

An 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 [29] [30]. 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 that 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". See Figure 1 for a graphical representation of the explained concepts.

A firewall rule base is a collection of order-sensitive rules. The firewall will evaluate all inbound traffic against the ordered rule base. The firewall starts at the top of the rule base until it encounters the first rule that matches the criteria (Source, Destination, Destination Port, Protocol) of the traffic. The firewall then performs the action as specified in the rule. In a firewall rule, <Source IP>, <Destination IP>, <Destination Port> and <Protocol> can be one value or a range of values. The 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*

As a rule base changes over time, different rules start interfering with each other, resulting in complexity. In [15], the following relations are defined between rules:

- **Disjoint:** Two rules **R1** and **R2** are disjoint (completly or partially), if they have at least one criterion (source, destination, port) that has completely disjoint values (= no overlap or match).
- **Exactly Matching:** Two rules **R1** and **R2** are exactly matched, if each criterion (source, destination, port) 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) 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 a subset of the other.

Figure 2 represents the differnet relation in a graphical manner. Exactly matching, inclusively matching and correlated rules can result in the following firewall anomalies [15]:

- *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. Disjoint rule are completely disjoint or partially disjoint. In that case, the order of the rules in the rule base is of no importance and the anomalies described above will not occur [15] [21]–[25]). 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. 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 impact 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 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 populated with IP addresses or ports. Groups can be nested.

Using groups should improve the manageability of the firewall. But, using groups can easily result in the introduction of exactly matching, inclusively matching or correlated rules as well.

Example:

"Group_Windows_APP" and "Group_Windows_APPS" could be two groups with each contain the IP addresses of all Windows Application Servers. The latter may have been created without knowledge of the former [6], 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 (ZT) 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. Also, each network connected resource should only be allowed access to what it needs.

E. Introduction to Normalized Systems

The Normalized Systems Theory [7]–[10] originates from the field of software development. There is a widespread belief in the software engineering community that using software modules decrease complexity and increases evolvability. It is also well known that one should strive towards "low coupling and high cohesion". The problem is that the community does not seem to agree on how exactly "low coupling and high cohesion" needs to be achieved and what the size of a module should be, to achieve low complexity and high evolvability.

The Normalized Systems 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 equals 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 states that the system exhibits a Combinatorial Effect and is considered unstable under change. Normalized Systems Theory proves that, in order to eliminate Combinatorial Effects, the software system must have a certain modular structure, where 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)

Only by respecting those rules, the system can infinity grow and still be able to incorporate new requirements.

Although NS originates in software design, the applicability of the NS principles in other disciplines such as process design, organizational design, accounting, document management, and physical artifacts. The theory can be used to study evolvability in any system that can be seen as a modular system and derive design criteria for the evolvability of such a system. In this paper, NS will be used to study the evolvability of the firewall rule base.

IV. Creating an artifact for an evolvable rule base

This section starts with investigating the modular structure of a firewall rule base, followed by a discussion of the issues that 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. The combinatorics are used to distill the design rules for the evolvable rule base. The design rules are translated into the actual artifact.

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

- To create a rule base compliant with the ZT concept.
- To create a rule base that contains only disjoint rules.
- To create a rule base, making use of firewall group objects to improve readability and manageability.
- To create a rule base that is evolvable for the following anticipated changes: the addition and removal of rules.

NS will be used to structure this evolvable rule base.

A. Modular structure of the rule base

A rule base is the aggregation of rules. A rule is an 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 3 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. The model corresponds with the type of information one needs to enter to create a rule in a firewall. 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 (the theorems "Separation of State" and "Data and

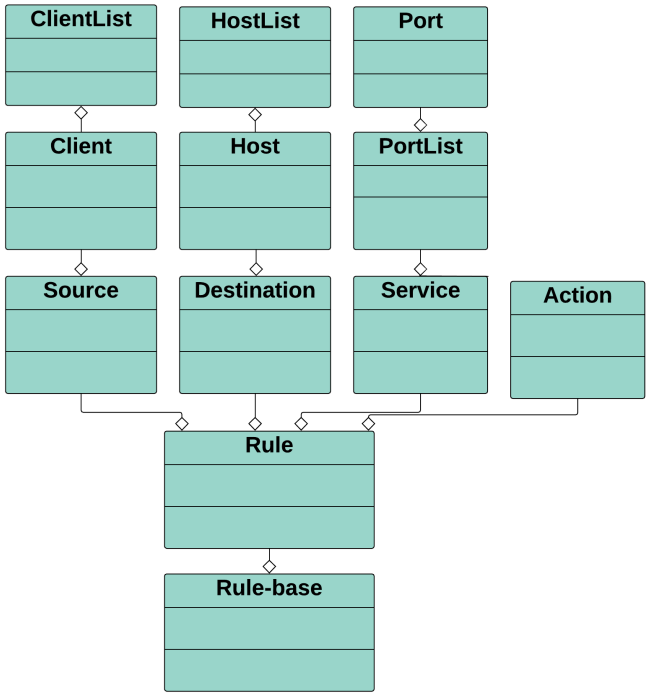

Figure 3. Modular Structure of a rule base

Action Version Transparency" are not relevant for the analysis of the rule base structure). As each of the mentioned modules focusses 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 comes from? In this respect, it is important to be aware 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 III). The order sensitivity is the root cause of the evolvability issues when the modular structure is instantiated. Indeed, it seems that —in some specific situations— certain evolvability issues of a modular structure only show up at instantiation time. Therefore, it is interesting to look at the application of the NS theorems at the 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.

Eliminating the order-sensitivity of the rule base is the key to solving the problem. A firewall rule base should 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 ZT principle should be applied, meaning that clients have only access to those services on hosts they have been given explicit access to.

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}.

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

$$\begin{cases} \mathbf{P}[1] \text{ ... } \mathbf{P}[\text{pj}]. \\ \mathbf{P}[\text{j}] \text{ contains a } \mathbf{Port}. \\ 1 \leq \text{j} \leq \text{pj}. \end{cases}$$

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.

$$\begin{cases} \mathbf{S}[1] \text{ ... } \mathbf{S}[\text{sj}]. \\ \mathbf{S}[\text{i}] \text{ contains a } \mathbf{Service}. \\ 1 \leq \text{i} \leq \text{sj}. \end{cases}$$

Let **Host** represent a network host that 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**.

$$\begin{cases} \mathbf{H}[1] \text{ ... } \mathbf{H}[\text{hj}]. \\ \mathbf{H}[\text{k}] \text{ contains a } \mathbf{Host}. \\ 1 \leq \text{k} \leq \text{hj}. \\ \text{hj} = f_h(\mathbf{N}) \end{cases}$$

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**.

$$\begin{cases} \mathbf{C}[1] \text{ ... } \mathbf{C}[\text{cj}]. \\ \mathbf{C}[\text{l}] \text{ contains a } \mathbf{Client}. \\ 1 \leq \text{l} \leq \text{cj}. \\ \text{cj} = f_c(\mathbf{N}) \end{cases}$$

Let **R** represent a firewall rule.

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

- **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 **R**x is located at position z instead of y, where z:1→ fj and z ≠ 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 that 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 that 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, being 2 ("Allow" or "Deny")

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

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

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

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 difficult. To gain control over the design space, it needs to be consciously reduced.

E. Designing an evolvable rule base

A rule will be made up of:

- **Cs** representing the Source, where $\mathbf{Cs} \subset \mathbf{C_{max}}$.
- **Hd** representing the Destination, where $\mathbf{Hd} \subset \mathbf{H_{max}}$.
- **Sp** representing the Ports, where $\mathbf{Sp} \in \mathbf{S_{max}}$.
- 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 ($\mathbf{R}_{default\_deny}$) as, when no rule explicitly provides access to a service on a host, the traffic needs to be explicitly blocked.

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

From Section III-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 is added to the rule base and it's disjoint with all existing rules, then 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 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 that is proportional to the newly required functionality and not into work, that has no logical link to the required functionality and that 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. The following combinations are possible:

- No overlap in sources - do not care about destination and port overlaps.

- No overlap in destinations - do not care about source and port overlaps.
- No overlap in ports - do not care about source and destination overlap.
- No overlap in source-destination combination, do not care about ports.
- No overlap in source-ports combinations, do not care about destinations.
- No overlap in destination-ports combinations, do not care about sources.
- No overlap in source-destination-port combination.

Cs is f_c(**N**) and **Hd** is f_h(**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, **Sp** represents the ports and is bound: the nature of TCP/IP limits the number 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 **Sp** to **Su**, so that **Su** only contains unique values.

$$\begin{cases} \exists! \mathbf{Su}[\mathrm{m}] \text{ in } \mathbf{Su} \text{ for m:1}\rightarrow\text{suj.} \\ \mathbf{Su}[\mathrm{u}] \cap \mathbf{Su}[\mathrm{v}] = \emptyset, \text{ where u, v:1}\rightarrow\text{suj, and u} \neq \text{v} \end{cases}$$

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 $\mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}$ represent the list of hosts that offer service **Su**[m].

$$\begin{cases} \mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]} \subset \mathbf{Hmax} \text{ and } \mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}[\mathrm{x}] \text{ contains a single host.} \\ \mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]} \text{ contains unique and disjoint elements.} \\ \exists! \mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}[\mathrm{x}] \text{ in } \mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]} \text{for x:1}\rightarrow\text{hdm} \\ \mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}[\mathrm{u}] \cap \mathbf{Hd}_{\mathbf{Su}}[\mathrm{v}] = \emptyset, \text{ where u, v:1}\rightarrow\text{hdmj, and u} \neq \text{v} \end{cases}$$

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

- **Su**[m] where m:1→suj, with suj=number of disjoint services offered on the network, for defining **R**.Port
- $\mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}[\mathrm{x}]$, x:→hdmj, with hdmj=number of hosts offering **Su**[m], for defining **R**.Destination
- $\mathbf{Cs}_{\mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}[\mathrm{x}]}$ being the list of clients requiring access to service **Su**[m] on host $\mathbf{Hd}_{\mathbf{Su}[\mathrm{m}]}[\mathrm{x}]$, 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 III-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 $\mathbf{F}[1] = \mathbf{R}_{\text{default\_deny}}$ with

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

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:

$$\begin{cases} \mathbf{R}.\text{Source} = \mathbf{C}\_\mathbf{H}\_\textit{host.name}\_\mathbf{S}\_\textit{service.name} \\ \mathbf{R}.\text{Destination} = \mathbf{H}\_\textit{host.name}\_\mathbf{S}\_\textit{service.name} \\ \mathbf{R}.\text{Port} = \mathbf{S}\_\textit{service.name} \\ \mathbf{R}.\text{Action} = \text{"Allow"} \end{cases}$$

   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 that form the building blocks for an evolvable rule base. Each building block 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

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

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

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

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

V. Demonstrate and evaluate artifact

In this section, we will demonstrate the artifact. We will apply different changes on a rule base (add/remove rule) and on the components that make up rule (add/remove a service, add/remove a host, add/remove a client). We also show what happens if rules are aggregated. The section terminate with an evaluation of the proposed artifact.

A. Add and remove a rule

Creating rules according to the artifact's design principles, leads to rules that 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 that 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). The source groups are populated with all clients requiring access to the service one 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 V-A).

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

A new host is a host that 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 hosts. The source 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 SectionV-A).

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

Combining Sections V-C and V-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. The same 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 V-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 needs to be added to those source groups that give access to the required services/hosts combinations. No CE's are being introduced during these operations.

F. Removing a service from the network

Let **sr** be the service that 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 that require removing from the rule base are now available. For each host offering **sr**, the corresponding rule

$$\begin{cases} \mathbf{R}_{\text{default\_deny}}.\text{Source} = \mathbf{C\_H}\_\textit{host.name}\_\mathbf{S}\_\text{sremove} \\ \mathbf{R}_{\text{default\_deny}}.\text{Destination}=\mathbf{H}\_\textit{host.name}\_\mathbf{S}\_\text{sremove} \\ \mathbf{R}_{\text{default\_deny}}.\text{Port}= \mathbf{S}\_\text{sremove} \\ \mathbf{R}_{\text{default\_deny}}.\text{Action} = \text{“Allow”} \end{cases}$$

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 V-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 that require removal from the rule base are available. For each service offered by **hr**, the corresponding rule

$$\begin{cases} \mathbf{R}_{\text{default\_deny}}.\text{Source} = \mathbf{C\_H}\_\text{hremove}\_\mathbf{S}\_\textit{service.name} \\ \mathbf{R}_{\text{default\_deny}}.\text{Destination}=\mathbf{H}\_\text{hremove}\_\mathbf{S}\_\textit{service\_name} \\ \mathbf{R}_{\text{default\_deny}}.\text{Port}= \mathbf{S}\_\textit{service.name} \\ \mathbf{R}_{\text{default\_deny}}.\text{Action} = \text{“Allow”} \end{cases}$$

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 V-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

$$\begin{cases} \mathbf{R}_{\text{default\_deny}}.\text{Source} = \mathbf{C\_H}\_\text{hremove}\_\mathbf{S}\_\text{sremove} \\ \mathbf{R}_{\text{default\_deny}}.\text{Destination}=\mathbf{H}\_\text{hremove}\_\mathbf{S}\_\text{sremove} \\ \mathbf{R}_{\text{default\_deny}}.\text{Port}= \mathbf{S}\_\text{sremove} \\ \mathbf{R}_{\text{default\_deny}}.\text{Action} = \text{“Allow”} \end{cases}$$

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 V-A). The service **sr** does not need to be removed from **Su** and neither does the corresponding group as the service is still 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 that 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.

J. The impact of aggregations

When following the prescriptions of the artifact, many groups and rules will be created (see Section V-K 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 complicated 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 ZT 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 ZT model.

We provide two 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 particular host. A client will have access to the service on all hosts offering the service, desired or not. In such a configuration, ZT 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 ZT and non-ZT rules, non-disjoint rule will pop-up. 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. The aggregation method excludes 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, ZT can no longer be guaranteed. As long as the destination groups are unique, disjoint rules can still be created. The moment that ZT and non-ZT rules are combined, non-disjoint rule will pop-up. The rule base can no 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.

K. Evaluation

The previous 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). The size of the rule base might be consider this as a drawback. Large size is often regarded as complex. A large size rule base may also impact firewall performance, as surching for a matching rule in a large rule base, has a direct impact on firewall performance. In Section VIII, the impact of rule base size on performance is further investigated. Some operations on rules may indeed result in CE's at group level, such as adding and removing a client from the network. Aggregations will violate the ZT constraint. Combining aggregation and non-aggregation based rules results in non-disjoint rules and CE's at rule base level.

VI. Filtering strategies

The artifact discussed in the previous section was created to be compliant with the ZT filtering strategy. In this section, we will discuss other filtering strategies: Interconnect strategy and Outbound filtering and see what kind of impact they have on the artifact.

A. Interconnect filtering strategy

The ZT filtering strategy can be considered as an inbound filtering strategy. Only traffic corresponding with exposed services is allowed. The filtering strategy used to interconnect different network segments and control the traffic between those segments is an Interconnect (IC) filtering strategy. The focus is on traffic between network segments, like VLANs or groups of VLANs, and not on the resources connected to those network segments. The rules are different compared to ZT rules. The level of granularity is a network subnet, not the resource. Filtering does not happen at port/service level. This means that there is one less parameter to enforce disjointness between the rules.

The proposed artifact can still be used to create an IC strategy based rule base. The group objects used in an IC strategy rule base would represent the following:

- Destination group: a group containing the IP addresses, expressed in subnets (VLAN's), that make up a logical part of the network.

- Source group: a group of IP addresses expressed in subnets (VLAN's), that make up a logical part of the network.

The VLANs can be organized in different ways. They can be organized according to a physical location or organizational department. In the former case, there is a VLAN per building floor, and the sum of all VLANs represents the building. In the latter case, there are VLANs per organizational unit, grouped in different parts of the building. The sum of all VLANs based organizational units in the building represents the full building.

In ZT based filtering, the most fine-grained component filtering is performed at, is the port. In IC based filtering, the most fine-grained component filtering is performed at, is the VLAN. The design of the rule base will be structured around the VLAN.

Using the artifact previously designed artifact:

- Start from an empty firewall rule base **F**. Add as the first rule; the default deny rule.
- For each VLAN requiring access control, create a destination group. Populate the group with the relevant IP address ranges representing the VLAN. The intersection between all groups must be empty! A VLAN cannot be present in 2 different logical parts of the network and thus in 2 groups. The naming convention of those groups: **D**_VLAN-LogicalName-VLANnr
- For each part of the network, which requires potential shielding from other parts of the network, create a source group. Populate the source group with the VLAN's that require access. The naming convention of those groups: **S_D**_VLAN-LogicalName-VLANnr.
- For each VLAN that requires protection, create a rule:
  - Source: **S_D**_VLAN-LogicalName-VLANnr
  - Destination: **D**_VLAN-LogicalName-VLANnr
  - Protocol: ANY

The **D**_VLAN-LogicalName-VLANnr groups will enforce the disjointness of the rules in the rule base. Add, remove, change operation on a rule base created according to the artifact are compliant with the evolvability conditions. It should be clear that this kind of filtering cannot be combined with ZT based filtering. The disjointness of rule cannot be guaranteed if ZT and IC based rules are used in the same firewall rule base:

- Protocol: violates disjointness
- Destination: ZT rules will be a subnet of IC rules and thus violate disjointness.
- Source: is not used to enforce disjointness

An example of an IC strategy use case is the merger between two companies. Each has their network. As long as the security policies are not aligned between both companies, there is a good reason not to interconnect the two networks directly. The interconnection is best done via a firewall. The firewall will filter between IP ranges, for instance, allowing traffic between the two headquarters, but not yet between remote sites (simplified example, not considering potential IP range overlap, NATing etc.).

As change is the only constant in companies, IC based filtering is complicated. Moves between buildings, reorganization in buildings, add and removal of sites, organizational changes, all make upfront, and stable segmentation of a network difficult. Segmentation rules change, segmentation principles are mixed, and logical network segments no longer become disjoint. The result will be evolvability issues in the rule base(s) and unforeseen side effects due to changes. Till now, the IC problem has been addressed in a network-centric approach. As network segmentation and company organization can result in implementation conflicts, solutions such as identity-based firewalls emerged. In those solutions, IC' based filtering happens based on the identity of the user. When a user tries to connect to certain parts of the network and hits an identity-based firewall enforcing the IC, the firewall will check the identity of the user and will filter based on this identity. This only works if:

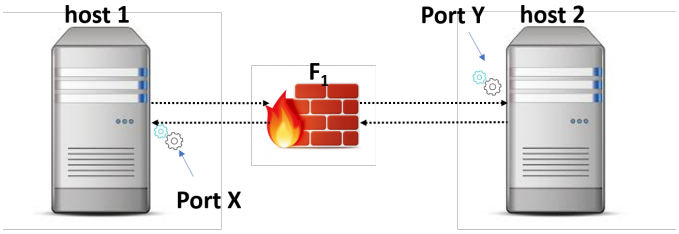

Figure 4. Inbound and outbound on a single firewall

- The firewall can establish the identity of the user associated with the source (who's working on PC with IP = x.y.z.u).
- The firewall has access to a DB containing the identities and has mechanisms to validate the identity.
- The firewall has a set of rules stating which identity has access to which destinations.

Such a setup is more user-centric. Access to the network is linked to the identity of the user and not the building or organizational layout. Elegant as the solution may seem, it just shifts the problem from the network space to the identity space. This research will not further investigate this. However, it is worth pointing out that, user identities, identify verification (authentication), identity authorization, identity definition, identity implementation, identity and HR policies, identity synchronization solutions, are among the most complex IS systems of an IT landscape. Researching the associated evolvability issues and proposing solutions is worthy of a separate Ph.D. research.

B. Inbound and outbound filtering strategy

An inbound filtering strategy, as ZT, will filter traffic close to the destination. The outbound filtering strategy will filter close to the source. From a security point of view, it makes sense to stop the traffic as early as possible on the network. On a single firewall, the notion of inbound and outbound is relative. A firewall rule base is not aware of inbound or outbound. It only knows source and destination and both can be located on the two sides of the firewall.

The artifact we propose started from a scenario where all sources are located on the left and all destination to the right of the firewall, effectively implementing an inbound filtering strategy. The same artifact can be used in a single firewall setup where sources and destination are located at both sides of the firewall. As long as the artifact is strictly followed, all rules will stay disjoint. There are some dangers involved. Take the case described in Figure 4 where a host1, located on the left side of the firewall, needs to access a host2 on the right side. Host2 also requires access to a service offered by host1. According to the artifact, the 2 following rules would

be created.

- **C_H**_host2_**S**_Y, **H**_host2_**S**_Y, **S**_Y, Allow
  - traffic from left to right
  - **H**_host2_**S**_Y contains host2
  - **C_H**_host2_**S**_Y contains host1
- **C_H**_host1_**S**_X, **H**_host1_**S**_X, **S**_X, Allow
  - traffic from right to left
  - **H**_host1_**S**_X contains host1
  - **C_H**_host1_**S**_X contains host2

What the firewall will do internally is look at the content of the groups, not the group names itself, and the rules are internally translated as

- host1, host2, Y, Allow
- hotst2, host1, X, Allow

Both host1 and host2 are member of different group. Interchaning those groups will result in rules which do not follow the logic of the artifact but that do represent the same rules inside the firewall

- **H**_host1_**S**_Y, **C_H**_host1_**S**_Y, **S**_X, Allow
  - host2, host1, X, Allow
- **H**_host2_**S**_Y, **C_H**_host1_**S**_X, **S**_Y, Allow
  - host1, host2, Y, Allow

Group objects are used to increase the manageability of rule bases. The above makes it clear that, if not used correctly, manageability will decrease. Groups created to represent destinations cannot be used to represent souces in rules, and vise versa. This is a manifestation of Separation of Concern. Representing sources and destination are different concerns. They should not be mixed.

Inbound and outbound filtering are also two different concerns. In the above scenario, both are mixed on one firewall yet, no immediate impact seems to surface. The impact will become visible when there are multiple firewalls in the network. This will be discussed in the next section.

VII. MULTIPLE FIREWALLS

In the previous sections, the assumption was taken that the network only contains one firewall. In this section, we will investigate the impact of multiple firewalls between the source and the destination.

A. The serial firewall filtering function

Let **Pa** be a package traveling over the network.

- **Pa**.source = the IP adress of the source sending package **Pa**.
- **Pa**.destination = the IP address of the destination for package **Pa**.
- **Pa**.port = the **Port** targetted on destination **Pa**.destination.

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

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

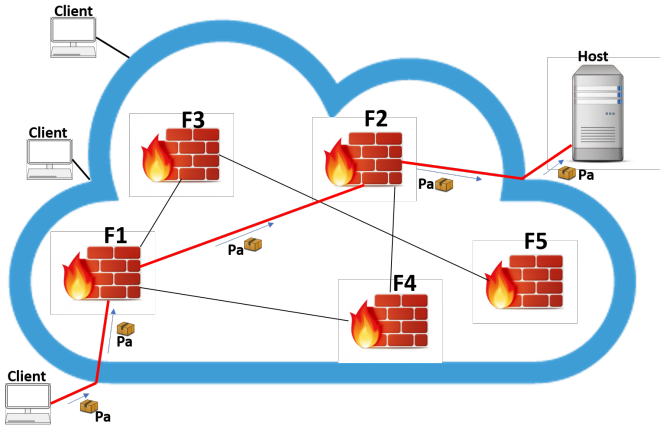

Figure 5. Multiple firewalls in a network

Let f_total be the total amount to firewalls in a given network. Let $\Phi^{\mathbf{s}}_{\mathbf{fw}}$ be the serial firewall filtering function for a network path containing fw firewalls in serie. Then

$$\Phi^{\mathbf{s}}_{\mathbf{fw}}(\mathbf{Pa}) = \prod_{f=1}^{f=fw} \varphi_{\mathbf{f}}(\mathbf{F_f}, \mathbf{Pa}) \quad (9)$$

$$\Phi^{\mathbf{s}}_{\mathbf{fw}}(\mathbf{Pa}) = \varphi_{\mathbf{1}}(\mathbf{F_1}, \mathbf{Pa}).\varphi_{\mathbf{2}}(\mathbf{F_2}, \mathbf{Pa})...\varphi_{\mathbf{fw}}(\mathbf{F_{fw}}, \mathbf{Pa}) \quad (10)$$

Where:

$fw : 1 \rightarrow f\_total$
$\Phi^{s}_{fw}(\mathbf{Pa}) = 0$ if **Pa** is blocked by at least one of the fw firewalls

$\Phi^{s}_{fw}(\mathbf{Pa}) = 1$ if **Pa** is allowed by all fw firewalls

See Figure 5 for a graphical representation of these concepts.

B. Applying the rules on some firewalls

In a given network, fw and thus $\Phi^{\mathbf{s}}_{\mathbf{fw}}$, will differ from the location of the source, destination, and the internal routing of the network. Let us assume that in such a network, all firewalls have an evolvable rule base according to the proposed artifact. The addition of a new resource, host_new offering service S_new, requires the addition of new rules $\mathbf{R_{new}}$, such that host_new is protected according to the ZT filtering strategy. Let us assume that $\mathbf{R_{new}}$ is only implemented on the firewalls in the path between the initially identified sources (members of **C_H**_host_new_**S**_new), and destination host_new. As time moves on, the initially identified sources require modification: a new client needs to access the host, or a client is removed from the network.

According to our artifact, adding or removing a client is just a question of adding and removing the client from the group **C_H**_host_new_**S**_new. In our current scenario, this is no longer the case. If a new client has a different network path towards the host_new compared to the path in which the rule $\mathbf{R_{new}}$ was initially implemented, then the rule $\mathbf{R_{new}}$ must now be implemented on the all firewalls in the path between the new client and host_new as well. In addition, the source group must be updated on all firewalls in all paths between all current clients and host_new. As the possible network paths are a function of the network, and the network can grow infinitely, a CE is being introduced. This is the worst kind of CE, as we will not know upfront where adjustments are required, and the

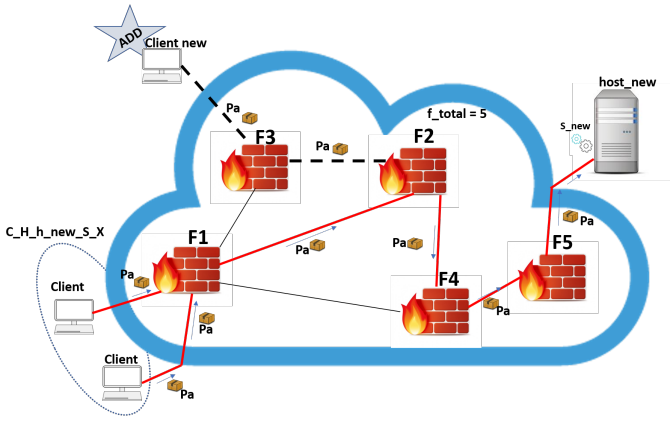

Figure 6. Apply the rules on some firewalls

full investigation of the network is required. An example of the described scenario can be found in Figure 6.

C. Applying the rules on all firewalls

The only way to avoid the problem described in the previous section, is to have all firewalls contain the same rule base. All manipulations of rules must be done on all firewalls simultaneously. As the network grows, so will the number of firewalls, and again, a CE is being introduced. This CE is less aggressive as it is know now the manipulations are required on all firewalls. We have already discussed the impact of the size of the rule base on the firewall. Having to duplicate all rules all over the network will make the rule base even larger and less coherent. Rules are added to firewalls, which will never be activated, and groups contain objects that are not relevant to the context of that specific firewall. The manageability of the firewalls will decrease. All firewalls are addressing the same concern. Normalized Systems learns that this will have a negative impact on evolvability, as can be concluded from the above.

D. Restricting Inbound traffic filtering

The paper "Minimizing the Maximum Firewall Rule Set in a Network with Multiple Firewalls" [31] is closely related to the problem we are trying to solve. According to [31], placing firewalls in a network such that the rule base is minimal, is an NP-complete problem, which requires a heuristics-based approach. Although applying the heuristic-base algorithm described in [31] may minimize the rule base over all firewalls, the evolvability of those rule bases is not discussed.

In Section VI-B, we mentioned that a network with one firewall is combining both inbound and outbound filtering rules. If we have a network with two firewalls that are connected in a back-to-back configuration - meaning the firewalls are directly interconnected and no resources are located in this interconnection - inbound and outbound traffic filtering can be separated. This can be done by adding a new default rule, which states that all outbound traffic is allowed. Figure 7 illustrates the setup, while Algorithm 1 and Algorithm 2 show the construction of the rule bases of F_1 and F_2.

The rules **R**1 on both firewalls are disjoint with respect to the rule base in which they are located as:

- on F_1: **C_H**_F_1Any_**S**_Any - represents all hosts protected by inbound traffic by F_1

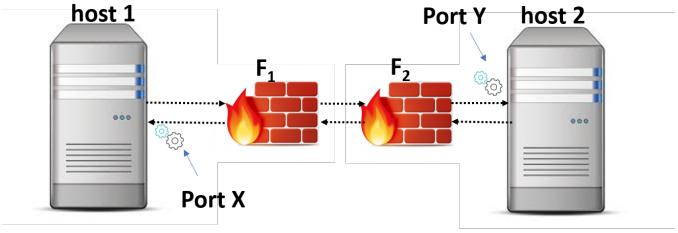

Figure 7. back-to-back firewalls

Rules *Firewall* F_1
- **R**1: **C_H**_F_1Any_**S**_Any, **H**_F_1Any_**S**_Any, **S**_Any, Allow
- **R**2: **C_H**_host1_**S**_X, **H**_host1_**S**_X, **S**_X, Allow
- **R**3: Any, Any, Any, Deny
- **with** *group contents*
  - in **R**1: **H**_F_1Any_**S**_F_1Any: any
  - in **R**1: **S**_F_1Any: any
  - in **R**2: **C_H**_host1_**S**_X: all hosts needing access to host1, host2 in this case
  - in **R**2: **H**_host1_**S**_X: host1
  - in **R**2: **S**_X: port X

Algorithm 1: Rule base of F_1

- on F_2: **C_H**_F_2Any_**S**_Any - represents all hosts protected by inbound traffic by F_2
- **C_H**_F_1Any_**S**_Any ∩ **C_H**_F_2Any_**S**_Any = ∅

and

- All source groups on F1 are subsets of **C_H**_F_2Any_**S**_Any - represents all hosts protected by inbound traffic by F_2
- All source groups on F2 are subsets of **C_H**_F_1Any_**S**_Any.

Thus, on both F1 and F2, the default outbound rule is disjoint with all other groups.

We see here appearing Separation of Concern. The concern of protecting a resource is only assigned to one firewall. If given to multiple firewalls, evolvability issues will occur. The leads to the following design criteria:

- A firewall should be clearly assigned to protect a set of resources. Those resources are protected by the firewall via the inbound ZT traffic filtering strategy.
- The firewall allows all outbound traffic from the set of resources it protects, to the rest of the network.

Rules *Firewall* F_2
- **R**1: **C_H**_F_2Any_**S**_Any, **H**_F_2Any_**S**_Any, **S**_Any, Allow
- **R**2: **C_H**_host2_**S**_Y, **H**_host2_**S**_Y, **S**_Y, Allow
- **R**3: Any, Any, Any, Deny
- **with** *group contents*
  - in **R**1: **H**_F_2Any_**S**_F_2Any: any
  - in **R**1: **S**_F_2Any: any
  - in **R**2: **C_H**_host2_**S**_Y: all hosts needing access to host2, host1 in this case
  - in **R**2: **H**_host2_**S**_X: host2
  - in **R**2: **S**_Y: port Y

Algorithm 2: Rule base of F_2

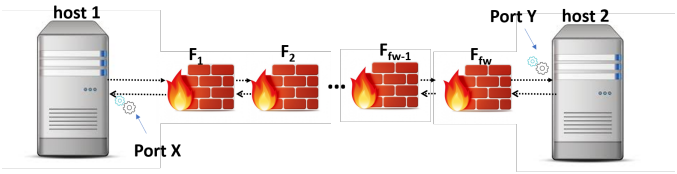

Figure 8. Path with multiple firewalls

- If all firewalls are protecting their resources, there is no need for outbound filtering.

As illustrated, our artifact can be made compliant with such as setup, simply by adding the "default allow" rule and the creation of some extra groups.

The approach described above might be turned around: by default allow all inboud traffic and filter on outbound traffic. Separation of Concerns would be respected. The artifact would need to be revised as disjointness would need to be enforced based on the combination of Service and Source instead of Service and Destination. The same reasoning applies for a the inbound default allow rule. Although technically possible, this filtering strategy would be confusing. Compare with the following scenario: A city needs to close an entry road due to construction works. Traffic will be blocked as close to the yard as possible (inbound filtering). It is impossible to block all roads, which could potentially lead to the city (outbound filtering).

E. Multiple firewalls revised

What happens when there are more than 2 firewalls between 2 resources? Figure 8 illustrates the setup. If we apply the design criteria from the previous section, we have to conclude that F_2 to F_{fw-1} are not allowed to filter inbound traffic. Those concerns are already assigned to F_1 and F_{fw}. Firewall F_2 to F_{fw-1} must handle other concerns such as:

- **chokepoint**: Use a firewall as a kind of valve: allow all or deny all. This comes in handy in case of network intrusions, and traffic needs to be blocked asap in a simple way, without impacting existing routing.
- **Interconnect filtering strategy**: use those firewalls to control connectivity between network segments (see Section VI-A).

Note that for the Interconnect filtering strategy, Separation of Concern needs to be respected as well. A "IC" firewall should be assigned to handle the interconnect of assigned ranges, and no other "IC" firewall should filter on the same ranges. This can again become quickly complex and evolve into an NP-complete problem. The best advice is to refrain from the usage of "IC" and chokepoint firewalls, limiting the number of firewalls in any network path as much as possible.

VIII. Additional Aspects of Firewall Rules Bases

Applying the Normalized Systems Principles results in a fine-grained modular structure. The creation of an evolvable firewall rule base is no exception; it leads to a fine-grained rule base. Creating and managing a large rule base requires automation, and a large rule base may lead to performance issues. In this section, the scalability of an evolvable rule base will be disussed, together with a possible approach to automate the creation and management of an evolvable rule base. The section ends with a reflection on Software Defined Networks (SDN) and Software Defined Firewalls (SDF) and why SDF has interesting evolvabilty features.

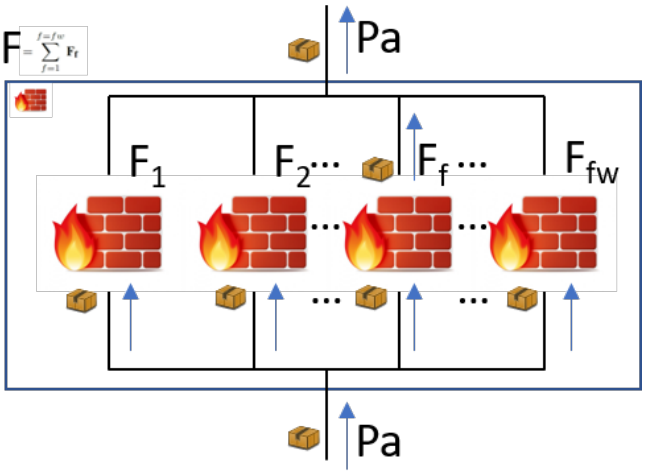

Figure 9. Scaling of Firewalls with normalized rule base

A. Scaling

In an evolvable rule base, all the rules are disjoint from each other and every 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 can be split in multiple pieces and distributed parallelly over different firewalls. Let **F** be a firewall rulebase containing only disjoint rules, created according to the artefact described in Section IV-F. As visualized in Figure 9, **F** can be split in fw sub rule bases, which are spread over fw parallel firewalls. Each of the fw rule bases contains the "Default Deny" rule at the end.

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

$$\mathbf{F} = \sum_{f=1}^{f=fw} \mathbf{F_f} \tag{11}$$

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

- $\varphi_{\mathbf{f}}(\mathbf{F_f},\mathbf{Pa}) = 0$ if the package is blocked - there is no rule **R** in $\mathbf{F_f}$ such that the package is allowed
- $\varphi_{\mathbf{f}}(\mathbf{F_f},\mathbf{Pa}) = 1$ 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 for fw firewalls in parallel. Then

$$\Phi^{\mathbf{p}}_{\mathbf{fw}}(\mathbf{Pa}) = \sum_{f=1}^{f=fw} \varphi_{\mathbf{f}}(\mathbf{F_f}, \mathbf{Pa}) \tag{12}$$

$$\Phi^{\mathbf{p}}_{\mathbf{fw}}(\mathbf{Pa}) = \varphi_{\mathbf{1}}(\mathbf{F_1}, \mathbf{Pa}) + \varphi_{\mathbf{2}}(\mathbf{F_2}, \mathbf{Pa}) + \ldots + \varphi_{\mathbf{fw}}(\mathbf{F_{fw}}, \mathbf{Pa}) \tag{13}$$

Where:

$$\begin{cases} \boldsymbol{\Phi}^{\boldsymbol{p}}_{\boldsymbol{fw}}(\mathbf{Pa}) = 0 \textbf{ if } \mathbf{Pa} \text{ is blocked by all of the fw firewalls} \\ \boldsymbol{\Phi}^{\boldsymbol{p}}_{\boldsymbol{fw}}(\mathbf{Pa}) = 1 \textbf{ if } \mathbf{Pa} \text{ is allowed by one rule of one of the fw firewalls, as:} \\ \exists! \boldsymbol{F_f} \in \boldsymbol{F} \text{ for} f = 1 \rightarrow fw \implies \boldsymbol{R} \in \boldsymbol{F_j} \end{cases}$$

This mechanism shows that the size of the evolvable rule base does not matter, as the solution scales. Firewalls with a

non-evolvable rule base cannot scale the same way. Scaling comes with a cost. Modern firewalls allow virtualization, but each virtual instance comes at a cost as well.

In addition to the horizontal scaling posibilities of an evolvable rule base, the performance of an evolvale rule base can be boosted by moving the most frequently used rules at the top. A firewall vendor such as CheckPoint, suggests to put 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 may be a real issue. In a rule base that 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 about 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.

B. Automation

The creation of the fine-grained rule base by humans can be an issue. The procedure regarding definitions of the groups needs to be followed strictly, and the creation of a catalog of all possible services is a must. For standard services and tools, lists of assigned ports/protocols and international standardization organizations related to the Internet (like iana.org) exist and can be reused. The management of the groups, their content, and the rules, should be done in a tool outside of the firewall (see Figure 10). This tool could expand the firewall rules in the fine-grained format, according to the naming conventions, performing checks against the group definitions and content via a user-friendly interface. The tool could then push the rules towards the firewall, effectively separating the management of rules and implementation of rules. Such tools exist on the market. Examples are Algosec, Tuffin, Firemon. However, none of those tools consciously restrict the design space and will thus enforce the creation of an evolvable rule base.

Defining a rule for each service may be considered cumbersome. Roles could be created, like ”monitoring and management”, which are a grouping of smaller, disjoint services. The firewall administrator can 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 will expand this role into the individual rules for each disjoint service. Example:

- ”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

C. Software Defined Network and Software Defined Firewall

Pushing the inbound filtering strategy discussed in previous section to the limit equals providing each resource with its firewall. This is what is happening in a Software Defined Network (SDN) combined with a Software Defined Firewall (SDF). In an SDN, the network layer is virtualized inside a virtualization

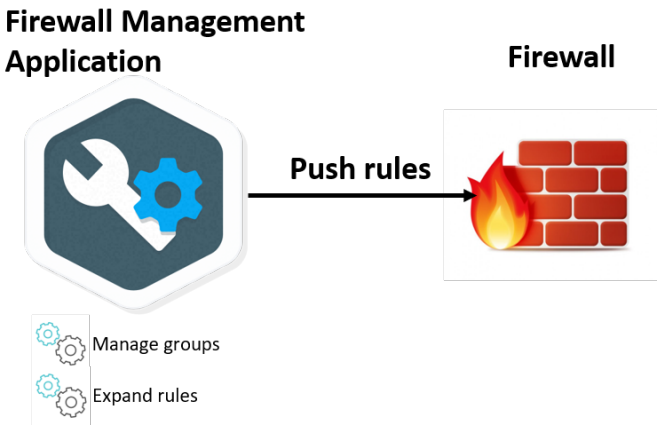

Figure 10. Firewall Management Tool

layer called the hypervisor. The SDN is decoupled from the actual underlying physical network. In the hypervisor layer, network components such as routers, switches, VLANs, load balancers, firewalls are all defined entirely in software. To each virtual host defined in/on the hypervisor, a virtual firewall can be attached. A package does not enter the network layer of the virtual hosts unless it successfully passes the firewall. SDF is better compared to an Operating System (OS)-based firewall (like IP tables or Windows Group Policies). OS-based firewalls can only perform their filtering function if the package is already ”inside” the host.

For an SDF, the rule base is configured by my means of policies. A policy defines the protocol and port that can pass though the firewall. The policies are attached to the firewall. As the firewall is attached to only one host, by default, disjointness for the destination is guaranteed. But, multiple policies can be attached to one host, and in those policies, overlaps and conflicts of protocols/ports and actions can be defined. Again, the conscious restriction of design space is required.

The previously proposed artifact can be adjusted for an SDN context by creating policies for Software Defined Firewalls. The policies are the equivalent of the Service Groups. They must be as fine-grained as possible. For each service exposed on a host, a policy must be created. Policies cannot overlap. Instead of creating a destination group, the polices are being attached to the host. As many policies are attached to the host as there are services offered by the host. Access to the host is provided by giving explicit access of a client to the host. This corresponds to creating a client group as defined in the artifact. Belonging to the group means you can access the host, and the policy attached to the host will check authorized protocols and ports.

A Software Defined Firewall in a Software Defined Context is the best way to guarantee the ZT filtering strategy. SDF also offers the most evolvable setup. Add/remove of hosts to the hypervisor automatically adds/removes the associated host firewalls. Add/remove of rules means add/remove of policies and/or attach/detach of policies. If the policies are created according to the proposed artifact, evolvability is guaranteed.

IX. Discussion

By means of discussion, we will cover two items. First, we will revisit the literature related to the size of the rule base, followed by a reflection on the nature of the CE’s that are still present when applying our proposed artifact.

A. Size of the rule base: revisiting the literature

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, motivated to the impact of the size of the rule base on performance. However, in [16] it is suggested that the actual size of the rule base is not related to the way the hardware actually processing 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? In Section VIII we pointed out that a rule base that has built-in evolvability, can scale and thus circument the potential performance issues due to its size. Non-evolvable rule bases cannot scale this way. Scaling does come at a cost. Either in terms of the purchase of more physical firewalls or adding resources to firewalls which allow virtualization. The higher cost will result in a firewall setup which will behave as is exptected. Security always comes at a cost. Further research of the literature and real-life measurements are required to clarify this point.

Looking at the combinatorics of Section IV-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.

$$2^{17}.hdj + 1 \tag{14}$$

The maximum number of services is 2^{17} = 131,072. 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 1,000 servers. Those 1,000 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 ZT model at host level is not applied. Instead, ZT at VLAN level is present (still filter at port level, but instad of at host level, filtering happens at VLAN level = a collection of hosts). If the realistic assumption is a made that the 1,000 server are spread over 20 VLAN's, it would mean that 20 x 67 = 1,340 rules are required for an evolvability rule base. This would mean 50% more rules to gain full evolvability.

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. Note that the proposed artifact facilitates the removal of such unacceptable CE's.

X. 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. 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. In addition to the proposed artifact, the evolvability implications of filtering strategies and firewall placement, has been investigated, showing that the Software Defined Firewall, promisses evolvability in a multi firewall network.

What is currently lacking is an acutal tool that could create, push and manage firewall rule bases according to the outlined principles of this paper. Having such a tool is one thing, implementing it and proving that it inhances security and operational efficity related to security, is something completly different. The creation of a tool in combination with the organizational impact, are subject for future research.

Another topic for future reseach is the size of the rule base. More real-life use cases are required to see to what extend existing rule bases can be transformed into evolvalbe rule bases, what the size of those rule bases will be and what the cost if implementing such rule bases would be.

Acknowledgment

The authors would like to thank Stefan Thys, Frederik Leemans and Stefan Biesbroeck for their help in writing and editting the article, and Sam Gozin and Bruno De Becker for providing the Engie operational data.

References

[1] G. Haerens and P. De Bruyn, "Using normalized systems to explore the possibility of creating an evolvable firewall rule-base", The 11th International Conferences on Pervasive Patterns and Applications (PATTERNS), pp. 7-16, May 2019

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

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

[4] M. Bennet, "Zero Trust Security: A CIO's Guide to Defending Their Business From Cyberattacks", Forrester Research June 2017

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

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

[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] 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

[9] 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

[10] 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

[11] G. Haerens, “Investigating the Applicability of the Normalized Systems Theory on IT Infrastructure Systems”, Enterprise and Organizational Modeling and Simulation”, 14th International workshop (EOMAS), pp. 123-137, June 2018

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

[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. 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.

[15] 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

[16] 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

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

[18] 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.

[19] 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

[20] 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

[21] 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

[22] 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

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

[24] 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

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

[26] 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]

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

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

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

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

[31] S.Chen, M. Yoon and Z. Zhang, ”Minimizing the Maximum Firewall Rule Set in a Network with Multiple Firewalls”, IEEE Transactions on Computers, Volume 59, No.2, 2010