# Ontological Analysis of the Evolvability of the Network Firewall Rule Base

Geert Haerens[1,2]

[1] Antwerp University, Antwerp 2000, Belgium
[2] Engie , Simon Bolivarlaan 34, 1000 Brussels, Belgium
geert.haerens@engie.com

**Abstract.** The TCP/IP based firewall is a notorious non-evolvable system. Changes to the firewall often result in unforeseen side effects, resulting in the unavailability of network resources. The root cause of these issues lies in the order sensitivity of the rule base. It is not only essential to define the correct rule. The rule must be placed at the right location in the rule base. As the rule base becomes more extensive, the problem increases. According to Normalized Systems, this is a Combinatorial Effect. This paper studies the ontology of a rule base and its implementation in an actual firewall. Based on this study, we explain why existing firewalls do not prevent evolvability issues. A new ontological model and implementation are proposed, using Normalized Systems, which drastically increases the firewall rule base's evolvability.

**Keywords:** Ontology · Evolvability · Normalized Systems · Firewall Rule Base.



## 1 Introduction

The TCP/IP based firewall has been and will continue to be an essential network security component in protecting network-connected resources from unwanted traffic. The increasing size of corporate networks and the need for connectivity have resulted in firewall rule bases increasing considerably. Large rule bases have a nasty side effect. It becomes increasingly difficult to add the right rule at the correct location in the firewall. Anomalies start appearing in the rule base, resulting in the erosion of the firewall's security policy or incorrect functioning. Making changes to the firewall rule base becomes more complex as the size of the system grows. An observation shared by Forrester [1] and the firewall security industry [2] [3].

Normalized Systems theory (NS) [4] defines a Combinatorial Effect (CE) as the effect that occurs when the impact of a change is proportional to the nature of the change and the system's size. According to NS, a system that suffers from CE is considered unstable under change. A firewall suffers from CE. The evolvability issues are the root cause of the growing complexity of the firewall as time goes by.

The order sensitivity plays a vital role in the evolvability issues of the rule base. The necessary condition to remove the order sensitivity is known, being non-overlapping or disjoint rules. However, firewall rule bases don't enforce that condition, leaving the door open for misconfiguration.

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

The topic presented in this paper is technical. Using domain-specific knowledge and instruments of Enterprise Engineering, such as DEMO [5] for the ontological analyses and NS [4] to study stability under change, we propose a scientifically sound solution to the problem of firewall rule base evolvability.

Design Science [6] has been used to come up with a solution for the problem at hand, and the paper is structured according to the Design Science process [7]. Section 2 introduces the basic concepts of firewalls and Normalized Systems. In Section 3 we will explain why the current ontological model and implementation model of the firewall rule base gives rise to evolvability issues. In Section 4 we will list the requirements for a solution to the problem and introduce an artifact that results in the creation of an evolvable rule base. In Section 5, we will see that the usage of this artifact puts forward a new ontological model and implementation model for the firewall rule base. In Section 6, we discuss our findings and wrap-up with a conclusion in Section 7.

## 2 Introduction to firewalls and Normalized Systems

This section starts by explaining the basic concepts of the firewall and continues with the introduction of NS.

### 2.1 Firewall concepts

**Firewall basics:** 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 [8]. Filtering happens by making use of rules. A rule is a tuple containing the following elements: <Source IP, Destination IP, Protocol, Destination Port, Action>. IP stands for IP address and is a 32-bit number that uniquely identifies a networked resource on a TCP/IP based network. The protocol can be TCP or UDP. Port is a 16-bit number (0 - 65.535) representing the TCP or UDP port on which a service is listening on the 4th layer of the OSI-stack [9]. When a firewall sees traffic coming from a resource with IP address =<Source IP>, going to resource =<Destination IP>, addressing a service listening on Port = <Destination port>, using Protocol = <Protocol>, the firewall will look for the first rule in the rule base that matches Source IP, Destination IP, Protocol and Destination Port, and will perform an action = <Action>, as described in the

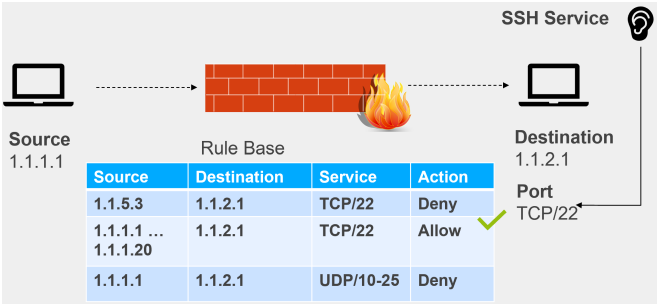


**Fig. 1.** Firewall concepts

matched rule. 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 starts at the top of the rule base until it encounters the first rule that matches the traffic. In a firewall rule, <Source IP>, <Destination IP>, <Destination Port> and <Protocol> can be one value or a range of values. In the remainder of this paper, protocol and port are grouped together in service (for example, TCP port 58 or UDP port 58 are 2 different services).

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

### 2.2 Normalized Systems concepts

Normalized Systems [4] [10] originates from the field of software development.

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 CE and is considered unstable under change.

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

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

NS can be used to study evolvability in any system, which can be seen as a modular system and derive design criteria for the evolvability of such a system [11] [12].

## 3 Problem Description

This section investigates the root cause of evolvability issues in firewall rule bases, being relationships between rules. We will create an ontological model of a firewall rule base by analyzing the DEMO FACT model that is associated with the "create firewall rule" transaction. We continue by reverse-engineering the relational model used in a firewall rule base, based on a rule base's export. We conclude this section by investigating the rule relationships, firewall ontology model, and firewall implementation model with the NS principles.

### 3.1 Relationships between firewall rules

In [13], 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 relations in a graphical manner. Exactly matching, inclusively matching and correlated rules can result in the following firewall anomalies [13]:

- *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.

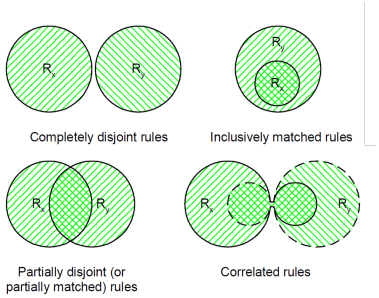


**Fig. 2.** Possible relationships between rules (from [14])

- *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. In that case, the order of the rules in the rule base is of no importance, and the anomalies described above will not occur [13] [15] [16]. However, due to several reasons such as unclear requirements, a faulty change management process, lack of organization, manual interventions, and system complexity [13], the rule base will include correlated, exactly matching, and inclusively matching rules, and thus resulting in evolvability issues. With this paper, we add to the list of reasons that the implicit ontology and implementation model are not helping to reduce these issues.

### 3.2 Ontological model

The analysis of a firewall rule base's ontology will be done with a DEMO FACT model [5]. It is a simple, straightforward ontological modeling method that sufficiently reveals the issues we want to surface.

In the scope of a security department that manages the firewalls, the "create firewall rule" transaction is considered an ontological transaction. During the request phase, information must be provided to the executor, corresponding with the filtering objective: source, destination, service, and action. The ontology of a rule base can be expressed in a DEMO FACT model, as shown in Figure 3. For simplicity reasons, details about value types and the transactions related to the coming about the facts and products are omitted. Note that a host has the dual notion of a source and destination. There are multiple ways to identify a source or a destination, depending on whether the network resource(s) are identified by their IP address(es) (IPRESOURCE, IPRESOURCEGROUP)

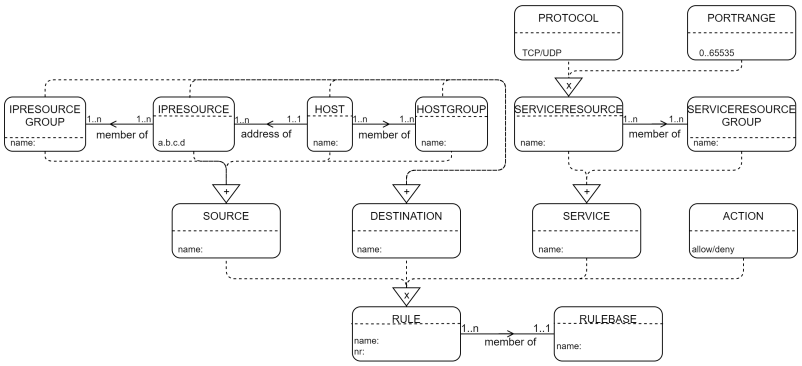


**Fig. 3.** Ontological FACT model of a firewall rule base

or by their logical name(s) (HOST, HOSTGROUP). SOURCE and DESTINATION are a generalization of those different resource identifications. A similar reasoning holds for SERVICERESOURCE and SERVICERESOURCEGROUP. A SERVICERESOURCE is the aggregation/cartesian product of PROTOCOL and PORTRANGE. A RULE is the aggregation/cartesian product of the different entity types making up a rule.

### 3.3 Reverse-engineerd implementation model from a firewall

Firewall vendors do not share or publish the firewall's internal data model explicitly. We will reverse-engineer the implemented data model based on a firewall configuration export.

**Firewall export** Two exports were made from firewalls inside the Engie network. One firewall is used to protect data center resources in Belgium, the other firewall is used to interconnect an Azian branch of the Tractebel Engie business entity to the Tractebel network. Both firewalls are from PaloAlto. The firewall configuration contains the following information: the Service objects, ServiceGroup objects, Address objects, AddressGroup objects, and Rule objects.

**Service objects** The Services objects export contains a data definition header. The relevant fields of this header are: Name (reference name), Protocol (TCP or UDP), and Destination Port (one or more ports or port ranges). Analysing the different objects, the following observations are made:

– The naming of the Services: No formal naming convention. The name explains the usage and meaning of the service or repeats the service content.

**ServiceGroup objects** The ServicesGroup objects export contains a data definition header. The relevant fields of this header are: Name (reference name)

and Services (one or more Service objects) Analysing the different objects, the following observations are made:

- The naming of the ServiceGroups: No formal naming convention. The name includes the usage of a grouping of Service objects.
- Uniqueness/overlap: The ServiceGroups overlap
- As ServiceGroups are an aggregation of services, ServiceGroup and Service objects are by design overlapping.

**Address objects** The Address objects export contains a data definition header. The relevant fields of this header are: Name (reference name), Type (a Fully Qualified Domain Name (= DNS resolvable name) or IP Netmask (/32 for a single IP or a /x for a range of IP addresses) or an IP range (from a.b.c.d to a.b.c.f)) and Address (the address according to one of the three type). Analysing the different objects, the following observations are made:

- Naming: No formal naming convention. The name includes the IP Mask, the FQDN, the hostname (different from FQDN), or a logical name for an IP range.
- Uniqueness/overlap: Multiple Address objects were found containing the same information of existing Address objects. The Address objects are not unique.

**AddressGroup objects** The AddressGroup objects export contains a data definition header. The relevant fields of this header are: Name (reference name), Address (an Address Object). Analysing the different objects, the following observations are made:

- Naming: No formal naming convention. The name includes the function of the AdressGroup.
- Uniqueness/overlap: AddressGroups overlap.
- As AddressGroups are an aggregation of addresses, AddressGroup and Address objects are by design overlapping.

**Rule objects** The Rule objects export contains a data definition header. The relevant fields of this header are: Nr (location in the rule base (1 = top)), Name (reference name), SourceAddress (an Address object and/or a AddressGroup object referenced via its Name), DestinationAddress (An Address object and/or a AddressGroup object referenced via its Name), Services (a Services object and/or ServiceGroup object referenced via its Name), Action (Allow or Deny). Analysing the different objects, the following observations are made:

- Source- and DestinationAddress: Source- and Destination Address contains AddressGroup objects and/or Address objects.
- Source- and DestinationAddress: AddressGroup objects or Address objects can be used for both SourceAddress and DestinationAddress.

– The rule base contains both allow and deny rules, mixing a whitelist (rule base only contains "allow" rules and a default "deny" rule at the end) and blacklist (rule base only contains "deny" rules and a default allow rule at the end) approach.

### 3.4 Reverse-engineered implementation model

An ontology is an implementation independent model and needs to be translated towards a technology implementable solution. Certain design decisions need to be made during this conversion. The FACT model shown in Figure 3 contains generalization relations or (inverse) inheritance relations. The way inheritance relations are translated towards an implementation can profoundly impact the evolvability of the implementation model [17].

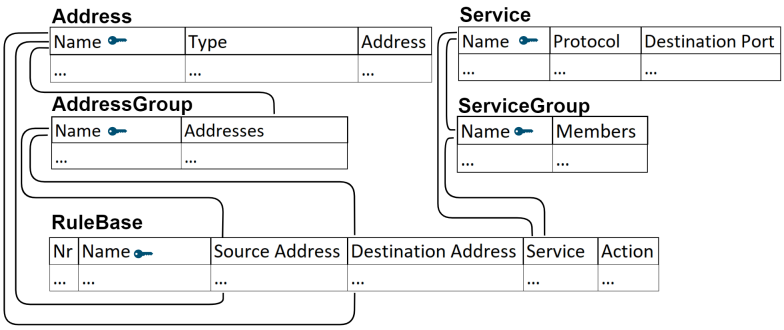


**Fig. 4.** Implementation model of a firewall rule base

Based on the firewall export, we can reverse-engineer the implementation model. We can see that Source, SourceGroup, Destination, and DestinationGroup are not present in the implementation model. Only the notion of Address and AddressGroup exists. The Address object represents both a Host, IPResource, and IPResourceGroup (in the form of an IP Mask or an IP Range). The AddressGroup object represents a HosGroup. IPResources/IPResourceGroups cannot be added directly to the Rule SourceAddress or Rule DestinationAddress. Only an Address or an AddressGroup can be added. IPResources/IPResourceGroups cannot be added directly to a rule. Similar principles hold for the services. Rule Services can only be populated by either Service objects or ServiceGroup object, not by ServcieResource/ServiceResourceGroups directly. Figure 4 shows the derived implementation model as a set of tables and relationships between the tables. The different objects found in the firewall export map to rows in these tables.

### 3.5 Looking at the evolvability issues with NS

According to NS, a modular structure will be free of CE regarding a set of anticipated changes if each module respects the 4 NS theorems – SoC, SoS, AvT, and DvT.

From Section 3.2 we can deduce that a source, a destination, and a service can be specified in multiple ways as they are a generalization of other types. At instantiation time, the same logical concept of a network resource can be represented by different objects, thus resulting in overlap. These overlaps could lead to non-disjoint rules. The ontology does not contain restrictions that would favor the creation of non-disjoint rules. If the implementation follows this ontology, it is expected that no restrictions will be present to avoid the creation of non-disjoint rules.

From Section 3.3 we can deduce that SoC between the different rule base objects is not respected. Service and ServiceGroup objects manage the same concern, being a protocol/port(s) pair combination. The same reasoning hold for Address and AddressGroup. The same network resource can be represented in three ways: IP Mask, IP Range, or FQDN. The three representations are generalized in the Address and AddressGroup objects.

From the observation made in Section 3.3 we conclude that SoC is violated for Address and AddressGroup. The same concern, being either a source or a destination, are combined into Address and AddressGroup. A considerable amount of changes would need to be made to the rule base's design if a source would require new attributes and actions compared to a destination. An example could be integrating identity-based filtering (the user using the source) into the filtering actions. It would require a redesign of the rule base model, effectively representing a CE. The fact that identity-based filtering is not a simple "add-on" to an existing IP-based filtering firewall demonstrates the point.

We are more concerned about different relations that can exist between firewall rules. Those relations represent a form of coupling inside the rule base that only becomes visible when the design model is being instantiated. These couplings result in evolvability issues at runtime.

The namings of the Service, ServiceGroup, Address, AddressGroup objects are independent of the objects' content. They are considered as two concerns that can evolve independently from each other. Like the naming of variables in programming should be anthropomorphic with respect to what they represent, so should the naming of the rule-based objects be anthropomorphic to their content. If name and content deviate from each other, incorrect objects can be chosen to make rules, thus creating incorrect rules. The separation of naming and content result in objects with different names, representing the same thing. Changes to the resource, like a new IP, need to be propagated to all objects representing that resource. As you cannot deduce this from the design of the objects, it means that at runtime, you will need to search through the rule base objects to determine the impact. The change will ripple through the system and is a function of the size of the system; thus, it is a CE.

The implemented rule base's design violates SoC in 2 ways - combine different concerns - introducing additional coupling - and decompose concerns - violating cohesion. As changes to names, IP Masks, IP ranges, ports, protocols will happen over time, violation of SoC will lead to ripple effects in the configuration of the firewall rule base. They can result in non-disjoint rules and become CE with respect to the addition and deletion of rules in the rule base.

Given the above, we can conclude at this point that neither the ontological model nor the implementation model facilitates the creation of disjoint rules.

## 4 Solution

### 4.1 Requirements

An evolvable rule base should only contain disjoint rules. From the previous section, we know that the current ontological and implementation model of a rule base contains insufficient inherent restrictions to create disjoint rule components and thus disjoint rules. We require a systematic approach that tells us how to create the rule components and how to combine them into rules. Applying such an approach would mean introducing a more restrictive ontology for a rule base, one that supports disjoint rules. In the next subsection we present an artifact, a method, to create an evolvable rule base by applying a strict design for rule components and rules.

### 4.2 Artifact for an evolvalbe rule basel

In previous work [19], the combinatorics involved when creating a rule base are discussed. For a given network $\mathbf{N}$, containing $\mathbf{C_j}$ sources and $\mathbf{H_j}$ destinations, offering $2^{17}$ services (protocol/port), and having a firewall $\mathbf{F}$ between the sources and the destinations, it can be shown that $\mathbf{f_j}$ is the number of possible rules that can be defined on the firewall $\mathbf{F}$:

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

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

A subset of those rules will respresent the intended security policy and only a subset of that subset will be the set of rules that are disjoint. The maximum size of the disjoint set of "allow" rules (aka a while list) is:

$$\mathbf{f_{disjoint}} = \mathbf{H_j}.2^{17} \tag{2}$$

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

The probability that a firewall administrator will always pick rules from the disjoint set is low if there is no conscious design behind the selection of rules.

In previous work [19] an artifact is being proposed to create a rule base free of CE for a set of anticipated changes. The artifact takes the "Zero Trust" [21] [20] design criteria into account as well, meaning that access is given to the strict minimum: in this case, the combination of host and service.

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
   - $\mathbf{R}_{\text{default\_deny}}$.Source = ANY,
   - $\mathbf{R}_{\text{default\_deny}}$.Destination=ANY,
   - $\mathbf{R}_{\text{default\_deny}}$.Service= ANY,
   - $\mathbf{R}_{\text{default\_deny}}$.Action = "Deny".
2. For each service offered on the network, create a group. All service groups need to be completely disjoint from each other: the intersection between groups must be empty.
   **Naming convention to follow:**
   - **S**_*service.name*,
   - with *service.name* as the name of the service.
3. For each host offering the service defined in the previous step, a group must be created containing only one item (being the host offering that specific service).
   **Naming convention to follow:**
   - **H**_*host.name*_**S**_*service.name*,
   - with *host.name* as the name of the host offering the service
4. For each host offering a service, a client group must be created. That group will contain all clients requiring access to the specific service on the specific host.
   **Naming convention to follow:**
   - **C**_**H**_*host.name*_**S**_*service.name*
5. For each **S**_*service.name*,**H**_*host.name*_**S**_*service.name* combination, create a rule **R** with:
   - **R**.Source =**C**_**H**_*host.name*_**S**_*service.name*
   - **R**.Destination = **H**_*host.name*_**S**_*service.name*
   - **R**.Service= **S**_*service.name*
   - **R**.Action = "Allow"

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

## 5 Evaluation

The artifact's application leads to a CE-Free rule base with respect to a large set of anticipated changes. Adding rules to the rule base due to the activation of a new host, the activation of a new service on a host, the addition of a new client requiring access to a service on a host becomes free of CE when the artifact is

used. Removal of rules due to removing a host, or a service, is also free of CE. Formal proof can be found in [19].

The section shows that applying such an artifact is enforcing a new ontology for the firewall rule base. Applying a more restrictive ontology will also mean that the implementation model will be more restrictive.

### 5.1 Impact on the ontological

A DEMO FACT model taking those restrictions into account can be found in Figure 5. As the model is the basis for the implementation mode, the restrictions will be present there as well and will favor the usage of disjoint components and rules.

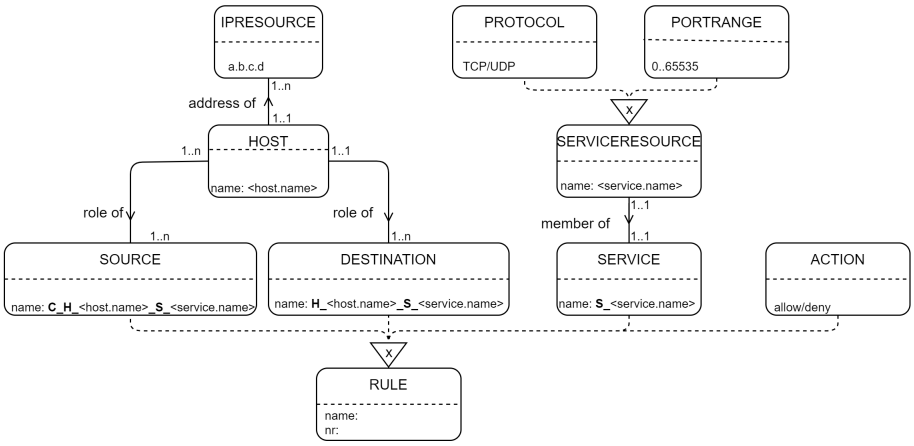


**Fig. 5.** Ontology of an evolvable rule base

### 5.2 Impact on the implemenation model

The artifact enforces an implementation model that is shown in Figure 6. The necessary restriction to ensure disjoint rules are put in place. If a firewall were to use this implementation mode, it would favor the usage of disjoint rules and be evolvable under change.

## 6 Discussion and further research

This paper refines previous work [19] done around the evolvability of the firewall rule base. We demonstrated that the data model used in the firewall opens the door for configuring a non-evolvable rule base. When the proposed artifact is used to create a new rule base, the evolvability issues can be drastically reduced.

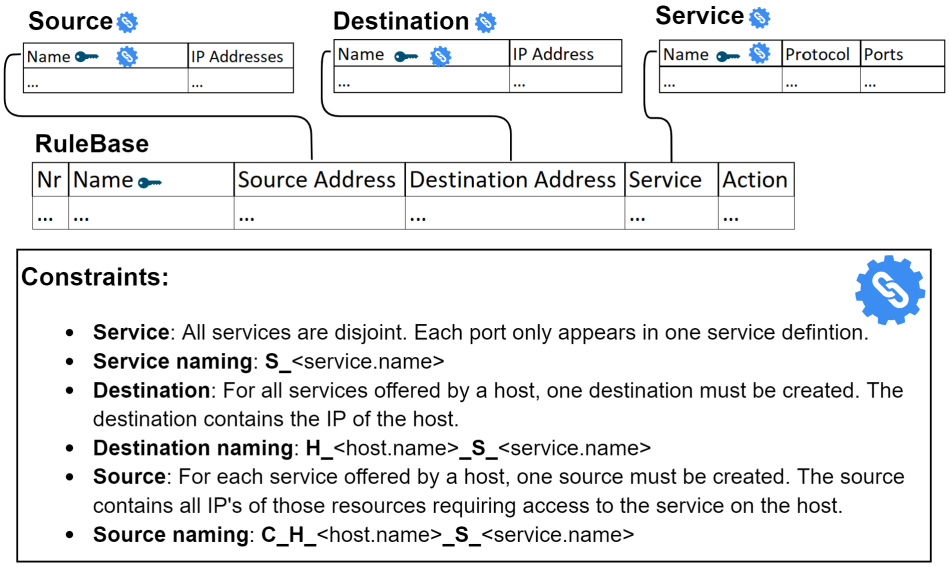


**Fig. 6.** Implementation model of an evolvable rule base

The result of applying the artifact is a fine-grained rule base. For each host/service pair, there is a rule. As those pairs are disjoint, the rule base is evolvable. A fine-grain rule base results in a large rule base. This could result in a performance issue as a firewall is sized for a certain max number of rules. If a disjoint rule base is used, horizontal scaling of the rule base becomes possible and even dynamic adjustments of the rule base, putting more frequently used rules at the top. Formal proof and information on the mechanism to apply for horizontal scaling, can be found in previous work [19].

Making the rules manually is not a good idea. Errors or the urge to aggregate could result in loss of evolvability. For this reason, the rule base should be managed and expanded in a tool sitting next to the firewall. The tools should expand the rules according to the artifact and push them to the firewall. Adding, removing, changing rules may no longer happen directly on the firewall, as the internal data model opens the door for non-evolvable design. Creating a tool enforcing the artifact on the firewall configuration is the subject of further research and work. Introducing such a tool in an enterprise would require changes in the current operational procedures. The firewall would no longer be managed directly, but indirectly.

The initial ontological model and implementation model are reverse-engineered. The model mimics the behavior of firewalls. This study was performed with data coming from a PaloAlto firewall. Additional verification should happen with other firewalls such as CheckPoint, Fortinet, and Cisco. Firewall administrators who helped with this study expect similar implementation for firewalls of other firewall vendors. Explicit validation is required.

The artifact assumes that we start from an empty rule base. A method must be put in place to convert an existing rule base into a rule base that only contains disjoint rules. At that point, the firewall can be managed according to the artifact. Tooling for this transformation is the subject of future research and work. A prototype based on an iterated local search heuristic algorithm is being built and provides promising results. Being able to "normalize" an existing rule base will also bring more clarity on the size of an evolvalbe rule base. It is expected that the size will increase but unknown how much.

This paper does not include an explicit literature study as this has been done extensively in previous work [19].

## 7 Conclusion

We started by identifying the evolvability issues of the firewall rule base. We looked at the problem from the point of view of relationships between rules, from the point of view of an ontological DEMO FACT model and a reverse-engineered firewall rule-based implementation model. We learned that the current ontology and implementation of a firewall rule base have insufficient restrictions and guarantees that only disjoint rules will be created. NS has been used to create an artifact that has been discussed in previous research. The artifact will enforce a set of restrictions on how to create rules. It effectively implements a more restrictive ontology of the firewall rule base.

## Acknowledgment

The author would like to thank Philippe Hendrickx and Jean-François Brison of Engie for providing the firewall export.

## References

1. Shel, H., Spiliotes, A.: The State of Network Security: 2017 to 2018'. Forrester Research November 2017
2. Firemon whitepaper: 2018 State of the firewall , URL https://www.firemon.com/resources/, [retrieved: October, 2020]
3. Algosec whitepaper: Firewall Management - 5 challenges every company must address, URL https://www.algosec.com/resources/ [retrieved: October, 2020]
4. Mannaert, H., Verelst, J., De Bruyn,P.: Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design, ISBN 978-90-77160-09-1, 2016
5. Dietz, J., Mulder, H.: Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation, ISBN 978-3-030-38854-6, Springer, 2020
6. Hevner, A. R., March, S. T., Park, J.,Ram, S.: Design Science in Information Systems Research. MIS Quarterly, pp. 75..105, Volume 38, Issue 1, 2004
7. P. Johannesson and E. Perjons, An Introduction to Design Science, ISBN 9783319106311, 2014

8. Stevens, W.R.: TCP/IP Illustrated, Volume 1, the Protocols, Addison-Wesley Publishing Company, ISBN 0-201-63346-9, 1994
9. Zimmermann, H., Day, J.D.: The OSI reference model. Proceedings of the IEEE, Volume 71, Issue 12, Dec 1983
10. Mannaert, H., Verelst, J., Ven, K.: 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
11. Huysmans, P., Oorts, G., De Bruyn,P., Mannaert, H., Verelst, J.: Positioning the normalized systems theory in a design theory framework. Lecture notes in business information processing, ISSN 1865-1348 - 142, pp. 43-63, 2013
12. Haerens, G.: Investigating the Applicability of the Normalized Systems Theory on IT Infrastructure Systems, Enterprise and Organizational Modeling and Simulation. In: 14th International workshop (EOMAS) 2018, pp. 23-137, June 2018
13. Abedin, M. 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
14. Al-Shaer, E., Hamed, H.: Design and Implementation of firewall policy advisor tools. Technical Report CTI - techrep0801, School of Computer Science Telecommunications and Information Systems, DePaul University, August 2002
15. Al-Shaer, E., Hamed, H.: Taxonomy of conflicts in network security policies. IEEE Communications Magazine, 44(3), March 2006
16. Al-Shaer, E., Hamed, H.: Boutaba, R., Hasan, M.: Conflict classification and analysis of distributed firewall policies. IEEE Journal on Selected Areas in Communications (JSAC), 23(10), October 2005
17. Suchanek, M. and Pergl, R.: Evolvability Evaluation of Conceptual-Level Inheritance Implemtation Patterns. In: The 11th International Conferences on Pervasive Patterns and Applications (EMPAT), pp. 1-6, May 2019
18. Hinrichs, S.: 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. Haerens, G., Mannaert, H.: Investigating the Creation of an Evolvable Firewall Rule Base and Guidance for Network Firewall Architecture, using the Normalized Systems Theory. International Journal on Advances in Security, pp. 1-16, 2020 vol 13 nr. 1&2
20. Bennet,M.: Zero Trust Security - A CIO's Guide to Defending Their Business From Cyberattacks. Forrester Research June 2017
21. Cunningham, C., Pollard, J.: The Eight Business and Security Benefits of Zero Trust. Forrester Reseach November 2017