Efficient Semantic Representation of Network Access Control Configuration for Ontology-based Security Analysis

Florian Patzer[a] and Jürgen Beyerer[b]
Fraunhofer IOSB, Institute of Optronics, System Technologies and Image Exploitation, Fraunhoferstr. 1, 76131 Karlsruhe, Germany

Keywords: Network Access Control, Security Analysis, Ontology-based Security Analysis, Security Ontology.

Abstract: Assessing countermeasures and the sufficiency of security-relevant configurations within networked system architectures is a very complex task. Even the configuration of single network access control (NAC) instances can be too complex to analyse manually. Therefore, model-based approaches have manifested themselves as a solution for computer-aided configuration analysis. Unfortunately, current approaches suffer from various issues like coping with configuration-language heterogeneity or the analysis of multiple NAC instances as one overall system configuration, which is the case for the maturity of analysis goals. In this paper, we show how deriving and modelling NAC configurations' effects solves the majority of these issues by allowing generic and simplified security analysis and model extension. The paper further presents the underlying modelling strategy to create such configuration effect representations (hereafter referred to as effective configuration) and explains how analyses based on previous approaches can still be performed. Moreover, the linking between rule representations and effective configuration is demonstrated, which enables the tracing of issues, found in the effective configuration, back to specific rules.

1 INTRODUCTION

IT and OT Systems consist of various networks and components, often with thousands of *network access control* (*NAC*) configurations. This makes them too complex to manually assess their configuration, for example to validate security policy and countermeasure deployment. Model-based computer-aided analysis addresses this issue by modelling security-relevant information in a machine-interpretable manner and using algorithms to find issues. The last two decades, ontologies, which can be defined as explicit specification of a conceptualization (Gruber, 1993), have proven themselves to provide adequate models for this matter (cf. Section 3). Most noticeable, the *Web Ontology Language* (*OWL*[1]) has been successfully applied for various such tasks. OWL and the *Semantic Web Rule Language* (*SWRL*[2]) have been applied widely to model security-relevant knowledge about a system. These models can then be extended by various existing reasoners (e.g. *Pellet*[3]) or rule engines (e.g. *Drools*[4]), and can further be queried by semantic query languages like *SPARQL*[5] or *SQWRL* (*Semantic Query-Enhanced Web Rule Language*) (O'Connor and Das, 2009).

Ontology-based security analysis, is usually applied on manually created ontological representations of abstract information about systems. Only view research focusses detailed configurations in order to identify issues with impact to the security of the respective system. With the increasing availability of machine-interpretable information about the systems under investigation, provided by configuration management databases, software-defined networking, engineering tool exports, asset management interfaces, etc., detailed configuration gets more accessible as time goes by. Therefore, security analysis on detailed configuration is becoming increasingly feasible and relevant.

[a] https://orcid.org/0000-0001-6620-9753
[b] https://orcid.org/0000-0003-3556-7181
[1] https://www.w3.org/TR/owl2-overview/, last accessed: 15.12.2020
[2] https://www.w3.org/Submission/SWRL/, last accessed: 15.12.2020
[3] https://github.com/stardog-union/pellet, last accessed: 15.12.2020
[4] https://www.drools.org/, last accessed: 15.12.2020
[5] https://www.w3.org/TR/sparql11-overview/, last accessed: 15.12.2020

Patzer, F. and Beyerer, J.
Efficient Semantic Representation of Network Access Control Configuration for Ontology-based Security Analysis.
DOI: 10.5220/0010285305500557
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 550-557
ISBN: 978-989-758-491-6

The already mentioned network access control is of particular importance for security analysis. This type of countermeasure, which includes host- and network-based firewalls and proxies, is particularly prone to configuration errors, like missing, obsolete, conflicting or shadowed policies. At the same time NAC is critical for the security of a system. Thus, a significant part of ontology-based configuration security analysis focusses on NAC and its configuration issues (Khelf and Ghoualmi-Zine, 2018). Nevertheless, only few research has concentrated on analysing the compliance of NAC configurations to security policies. Moreover, existing approaches addressing this kind of challenge either suffer from crucial scalability issues or from non-applicability for more sophisticated NAC configurations (cf. Section 3).

This paper suggests to model the effects of NAC configurations (called effective configuration), to allow compliance-related queries or reasoning in a configuration-language-independent manner. The approach exploits the fact that compliance and inter-NAC configuration analysis is focussed on the traffic a NAC-instance is configured to permit or deny. Therefore, it follows the strategy to model patterns for the allowed or prohibited traffic, resp. the classes of data/information flows a NAC instance allow or block. It is important to note that effective configuration is not targeting analysis on rule level and thus cannot contribute to that type of analysis. For example, the policy ‘‘*Every rule needs to define a source and destination port range*” has to be analysed at rule level and not on the modelled effective configuration. However, this paper shows that, compared to currently available approaches, providing such a model of effective configuration significantly improves the scalability and efficiency of security analysis and model extension (cf. Section 4).

The next sections are structured as follows: Section 2 describes an example system architecture to support the understanding of later sections. Subsequently, Section 3 describes and compares related approaches and their issues. Afterwards, the effective configuration approach is presented in Section 4 and used to solve an example analysis question. Finally, the presented approach is further discussed in Section 5, before the paper is summarized in Section 6

2 WORKING EXAMPLE

This section describes a small but realistic system architecture to support the understanding of the next sections. Figure 1 shows the example system architecture consisting of three different internal subnets and one subnet mimicking the internet. Following the IEC 62443[6] security standards the networks and their connected assets are distributed into different security zones, namely Internet, DMZ, Office and Field.

Listing 1 shows a rule chain excerpt of the *pfSense*[7] firewall `Pfsense 1`. In the pfSense configuration language, `inet` signals the IPv4 context of a rule. The rule processing in pfSense is rather complex. However, for didactic reasons, the processing strategy can be simplified as follows:

1. All rules without a `quick` instruction follow the “last match wins” strategy.
2. Rules with a `quick` instruction follow the “first match wins” strategy.

Listing 1: Reduced rules of the firewall Pfsense 1.

```
 1  block in inet all label
 2  block out inet all label
 3  block in on ! em0 inet from
        172.16.110.0/24 to any
 4  block in inet from 172.16.110.118
        to any
 5  block in on ! em1 inet from
        172.16.121.0/24 to any
 6  block in inet from 172.16.121.1 to
        any
 7  block in on ! em2 inet from
        172.16.122.0/24 to any
 8  block in inet from 172.16.122.1 to
        any
 9  pass in quick on em0 inet proto
        tcp from 172.16.110.0/24 to
        172.16.122.0/24 port = http
10  pass in quick on em1 inet proto
        udp from 172.16.121.0/24 to
        172.16.122.103 port = domain
11  pass in quick on em2 inet proto
        udp from 172.16.122.103 to any
```

The rules state the following:

- Rules 1 and 2 are the IPv4 default blocking rules.
- Rules 3, 5 and 7 block incoming packages on each Ethernet interface which is not internally configured for the IPv4 source networks of the packages.
- Rules 4, 6 and 8 are again default blocking rules but for the firewall interface’s IPv4 addresses.
- Rule 9 allows HTTP traffic to pass into the DMZ.
- Rule 10 allows DNS traffic from the Office zone to the DNS server.
- Rule 11 allows any UDP traffic from the DNS server to pass through.

[6]isa99.isa.org/ISA99\%20Wiki/Home.aspx, last accessed: 15.12.2020

[7]https://www.pfsense.org/, last accessed: 15.12.2020

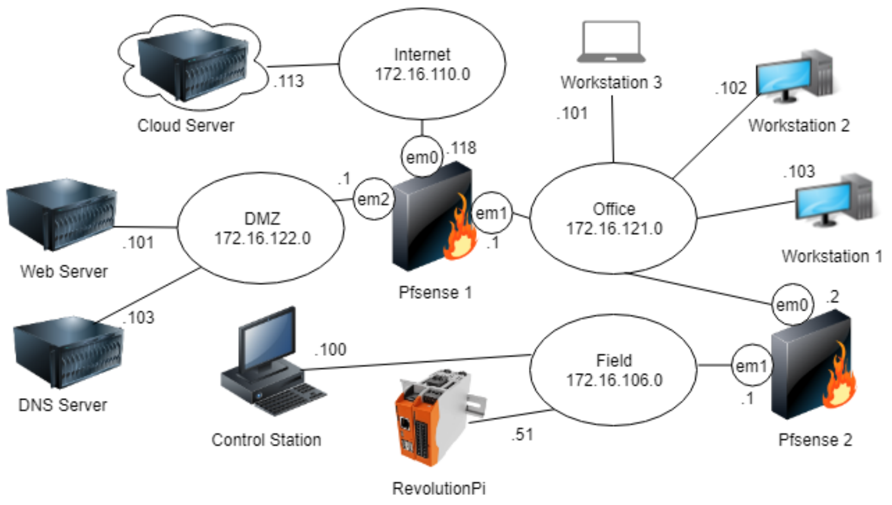

Figure 1: Example system architecture.

Due to space limitations, the remaining rules and the configuration of the second firewall are not listed here.

Ontological Representation. The ontological model of the example architecture was created by command parsing and modelling of the information with the Web Ontology Language (OWL) Version 2 using the *description logic* (*DL*) subset (cf. (Patzer et al., 2019)). Individual component models were created, merged and extended by reasoning and SWRL rules to ensure the following assumptions about the model hold:

1. IPv4 interfaces are linked to IPv4 networks.
2. Instead of modelling every possible IPv4 address as an individual, only the networks are modelled as individuals. Moreover, interfaces and networks contain data properties with numeric representations of the respective addresses.
3. The order in which the firewall rules are configured is modelled via linked list (cf. Figure 2).
4. To simplify queries regarding the order of rules, each rule has a link to all successors.
5. IPv4 subnets are inferred to be within the same zone as their parent networks.
6. Each rule addressing an IPv4 source or destination contains respective IPv4 interfaces.
7. Firewall types are modelled by different classes.
8. The rules are modelled as distinct individuals.

This system ontology provides a suitable model to embed the effective configuration. However, it is not part of the approach and therefore not the subject of evaluation here. Rather, it serves as an “environmental example” to help the reader understand the approach.

3 RELATED APPROACHES

Previous approaches of ontology-based security analysis covering NACs can be summarized as follows:

- *Approach 1* - Modelling policies in an abstract configuration language which can be used to evaluate the rules and to automatically generate NAC-specific rules (Martínez Pérez et al., 2007), (Davy et al., 2013).
- *Approach 2* - Automatically translate NAC-specific rules into an abstract configuration language which can be used to evaluate the rules (Hu et al., 2011), (Tong et al., 2015).
- *Approach 3* - Modelling each NAC configuration with language-dependent representations and searching for issues on rule chain level (Fitzgerald et al., 2007), (Foley and Fitzgerald, 2008), (Cordova et al., 2018).
- *Approach 4* - Modelling inter-policy relations manually and searching for issues on the so created abstract relationship (Fitzgerald et al., 2008), (Fitzgerald et al., 2009).

In the following paragraphs, some of this research is explained in more detail.

Research following Approach 1 concentrates on generic policies, which are automatically transformable into concrete configurations. Besides work on general policy languages (Davy et al., 2013), the

Table 1: Comparison of the four main classes of NAC modelling.

	A 1	A 2	A 3	A 4	EC
Finding configuration issues in rule chains, like shadowed, redundant, generalized rules	1	3	4	0	(4)
Finding inter-NAC issues like inconsistencies regarding certain policies	3	2	0	1	4
Finding network-centred issues not focussing NAC-rules (cf. example question of Section 4)	2	2	0	3	4
Providing a vendor-agnostic NAC configuration representation	4	3	0	0	2
Providing a representation covering all NAC configuration languages	2	2	0	0	3
Providing a fully automated model generation strategy	2	2	2	0	3

term most related to this approach is *policy-based network management* (*PBNM*). Pérez et al. (Martínez Pérez et al., 2007) presented a framework and policy representation for PBNM - more precisely the distributed firewall strategy, where firewall rules are defined in a centralized manner and are then enforced at each network endpoint. Their policy representation is based on the – meanwhile outdated – OWL mapping[8] of the *common information model* (*CIM*) (Textor et al., 2010), which the authors extended with concepts for the formalization of NAC policies that are supposed to be translated to the respective NAC-specific rules.

Approach 2 is comparable to Approach 1, but concentrates on the generation of the generic policies from specific configuration. Hu et al. (Hu et al., 2011) presented a framework to build generic policies from arbitrary access control policies, using an ontological approach. The authors evaluated it using a simple firewall configuration and *XACML*[9]. Furthermore, as an application of the approach, the authors showed how redundant and conflicting configurations can be discovered on that abstract layer. However, due to the simplicity of the firewall configuration utilized, which does not contain complex control, state or substitution commands, the approach is not applicable for modern firewalls, as it is. Tong et al. (Tong et al., 2015) presented another example of Approach 2 which also presents a generic algorithm to create the generalization which, according to the authors, covers all NAC configuration languages.

Approach 1 and Approach 2 share the disadvantage of unpredictable expressiveness (henceforth called *Expressiveness Issue*) of the generic language. This means that it is unclear to what extend the various configuration languages are covered by the generic language. This is a natural issue of any language which is supposed to formalize configurations on a generic level. Moreover, the approaches suffer from *Scalability Issues* - which are discussed in more detail later in this section - and the dependency on the transformation algorithms. The latter dictates what really can be analysed within the respective generic representation of rules and is henceforth called *Transformation Issue*.

Approach 3 enables NAC-configuration-language-specific analysis. As an example, Foley and Fitzgerald (Fitzgerald et al., 2007), (Foley and Fitzgerald, 2008) proposed an OWL representation of the Netfilter firewall and explained how this representation is used to find configuration issues using SWRL and reasoning. Later, the authors extended their approach by adding risks and thresholds in order to analyse the sufficiency of NAC policies, which the authors suggest to use for inter-NAC analysis (Approach 4) (Fitzgerald et al., 2008). Moreover, the authors suggested to perform inter-NAC identification of configuration issues, like shadowing, using the intra-NAC approach. However, (Fitzgerald et al., 2008) does not show how this should be scalable, since with their method each rule would have to be written for every possible NAC combination separately (no generic NAC model is utilized). This aligns to the fact that the authors later presented the risk-based method as the inter-NAC analysis solution of their choice in (Fitzgerald et al., 2009).

The previously mentioned Scalability Issues manifest through information which needs to be evaluated by every query or rule (Approach 1, Approach 2 and Approach 3), vendor-specific modelling which makes it's application for inter-NAC analysis infeasible (Approach 3) and the necessity to manually add the relevant knowledge to the model (Approach 4), instead of generating or inferring it.

During our research, we identified six aspects a modelling strategy of NAC configuration should cover, in order to be adequate for security analysis. Table 1 displays the estimated degree in which each aspect is covered by an approach class (abbreviated as A 1 - A 4), using a scale from 0 to 4, where 0 is no and 4 is full coverage. Furthermore, the approach of modelling the effective configuration (abbreviated as

[8]https://wwwvs.cs.hs-rm.de/oss/cimowl/index.html, last accessed: 15.12.2020

[9]https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, last accessed: 16.12.2020

EC) is analogously assessed and thus presents a quantification of the improvement compared to related approaches.

The table shows that even approaches targeting a certain aspect are not being estimated to cover the aspect to a full extend. The main reasons for this estimation are their expressiveness and scalability issues which are significantly mitigated by the approach described in the following section. A more in depth discussion of the effective configuration approaches' assessment is provided in Section 5.

4 EFFECTIVE CONFIGURATION

This section presents the effective configuration approach which provides a solution for the Expressiveness Issue, Transformation Issue and Scalability Issues.

Listing 2 describes the strategy of generating the effective configuration model, which can be applied for common NAC-configurations, as pseudo code. This strategy is configuration language agnostic, but contains configuration-language-specific functions an implementation has to provide.

Listing 2: Pseudo code describing the strategy to generate the effective configuration representation.

```
$integrate\_additional\_config(\Lambda)$
$transform\_action\_types(\Lambda)$
for each $s \in \Sigma${
    $\Lambda'_s$ = $transform\_to\_last\_match\_wins(\Lambda_s)$
    integrate $\Lambda'_s$ into $\Lambda$
}
for each rule $r \in \Lambda$ {
    $p_r$ = $get\_pattern(r)$
    $P$ = $get\_overlapping\_patterns(p_r)$
    if $P \neq \emptyset$ {
        $P'$ = $split\_expand\_reduce(P, p_r)$
        remove $P$ from $\Pi$
        add $P'$ to $\Pi$
    } else {
        add $p_r$ to $\Pi$
    }
}
$\Upsilon$ = $create\_flow\_concepts(\Pi)$
for each $p \in \Pi$ {
    for each $t \in \Upsilon$ {
        if $p$ contains $t$-specific
            information $i_t$ {
            $add\_to\_ontology(i_t, p)$
        }
    }
}
```

The following list provides definitions and clarifications for Listing 2:

- Λ contains the rule chain, i.e. the main configuration of the NAC instance. It is assumed that any substitution or similar additional configuration has already been resolved, or is transformed into the rule chain as part of the *integrate_additional_config* routine, so that Λ after this function only consists of the rule chain and any necessary configurations which cannot be integrated. Listing 1 contains an example for a necessary substitution in the form of `domain` and `http` which are placeholders for the actual ports and need to be replaced.
- ***transform_action_types*** performs the transformation of the rule's actions to "allow" and "block". This reduction is useful because later on the exact behaviour of the NAC instance is no longer relevant. Instead, it is only of interest whether certain network traffic is allowed to pass the instance or not. As an example, two actions that block ICMP, where one sends a response on receiving the ICMP package and the other does not, would both be substituted with the "block" action.
- Σ is the set of evaluation strategies of the NAC configuration.
- $\Lambda_s \subset \Lambda$ is the partial rule chain of an evaluation strategy $s \in \Sigma$.
- "**integrate Λ'_s into** Λ" means that the modified partial rule chain Λ'_s replaces the original partial rule chain Λ_s in Λ such that it integrates into the correct overall order. This reordering is important because the evaluation strategy of Λ'_s is different to the strategy ofΛ_s.
- Π contains both, the patterns for allowed and for blocked traffic, in order to avoid a significant expansion of the algorithm with various redundancy due to the additional case distinctions. Instead, all pattern processing functions are defined as action-type-aware.
- ***get_overlapping_patterns*** returns the patterns which overlap with the passed pattern.
- ***split_expand_reduce*** creates as many sub-patterns as necessary to represent the corresponding passed patterns without containing contradictions regarding the associated actions. This is explained in more detail later in this section.
- ***create_flow_concepts*** creates the flow concepts that do not yet exist in the model. These are classes and roles that can be used per combination of action, identifier type (e.g. IPv4 address range or ports), ISO/OSI layer and protocol to represent the respective subpattern, including its relationships to other subpatterns and the respective

rules (if also part of the model). An example of this is shown in Figure 2.

- Υ is the ordered list of flow concept types (cf. `AllowedIpV4Flow` and `AllowedTcpFlow` in Figure 2). The order of the list is ascending with reference to the concerned ISO/OSI layer.
- ***add_to_ontology*** adds the passed information of the pattern to the model using the flow concepts of the corresponding flow concept type. In this step, the already added information of the same pattern is also linked to the newly added information (cf. *usesFlow* from Figure 2). For the case of allow actions, this reflects that each child concept of `AllowedFlow` (e.g. `AllowedEthernetFlow`, `AllowedIpV4Flow` and `AllowedTcpFlow`) is either a self-contained pattern or a refinement of another `AllowedFlow`.

The following paragraphs provide additional insights to the strategy.

The first part of the strategy (lines 1-6) generalizes the NAC configuration, by dissolving substitutions - like names for IP networks or hosts, service names for port numbers or NAT tags for hosts - and by transforming individual processing control strategies into a "last-match-wins" strategy. For pfSense firewalls, the latter can be achieved by separating the normal from the quick rules, reversing the order of the quick rules and concatenating the list of normal rules to the reversed list of quick rules.

Since each rule can be interpreted as a pattern and an action, in lines 7-17 the algorithm iterates over the patterns, derives which patterns overlap with already processed ones and adds the others to the pattern list. However, if a pattern p_r overlaps with an already processed one, the following cases can occur:

1. Patterns overlap with p_r partially and the respective rules have a different action than r.
2. Patterns overlap with p_r partially and the respective rules have the same action as r.

In the first case, each already processed pattern p will be split into the parts of the pattern which are not covered by p_r and the parts covered by p_r. The first parts are reintroduced to the stored pattern as substitution for p, while the second parts get substituted by p_r. In the second case, the already processed patterns are merged with p_r. Patterns overlapping with other patterns occur in every NAC configuration, since each rule overlaps with the default rules (cf. rules 1 and 2 of Listing 1). During the processing of the algorithm, Π remains consistent and free of contradictions.
Lines 18-25 address the actual modelling of patterns for allowed or blocked traffic, i.e. the effective configuration. The already referenced Figure 2 displays an excerpt of a model for allow patterns. As the diagram visualizes, each allowed flow is linked to the network knowledge of the model (cf. Section 2), which significantly simplifies queries and processing tasks that include, but are not limited to, the respective NAC instance. Furthermore, the blue concepts and properties are part of the effective configuration whereas the black concepts and properties only exist if the NAC configuration is modelled as well. This is optional, but has the advantage of supporting the intra-NAC misconfiguration detection Approach 3 allows (cf. Section 3). Moreover, if issues are discovered when analysing the effective configuration, conclusions on the causing rule can obviously only be drawn if a representation of the rule is present and linked. For this, an identifier of the rule would already be a sufficient degree of knowledge, although existing misconfiguration detection approaches would not be applicable.

Example Analysis. An ontology, built using the method described in the preceding paragraphs, enables configuration-language-agnostic rules and queries for connection- or traffic-based model extension and analysis (cf. Section 3). To substantiate this claim, the following paragraphs present an exemplary analysis based on the following question:
Is there a TCP connection possible from the Field Zone (containing the connection initiator) to the DMZ?

To shorten the here described example, let the question only be answered positively if a possible TCP/IP link can be derived, able to pass both firewalls, which implies that no bypasses or pivoting opportunities are evaluated. Moreover, the example analysis assumes that only one firewall is connected to the Field Zone and only one path exists from the Field Zone to the DMZ, with respect to firewalls to pass.

In a first step, the SWRL rule in Listing 3 links the IPv4 flows directly to the known subnets of their source networks. An analogue rule is applied for destination networks as well.

Listing 3: SWRL rule adding source networks to allowed IPv4 flows to simplify later rules and queries.

```
AllowedIpV4Flow(?aif)
 ^ srcNetwork(?aif, ?srcNet)
 ^ parentNetwork(?subNet, ?srcNet)
-> srcNetwork(?aif, ?subNet)
```

In consequence, the query from Listing 4 directly results in a selection of all firewalls, supporting the requested connection.

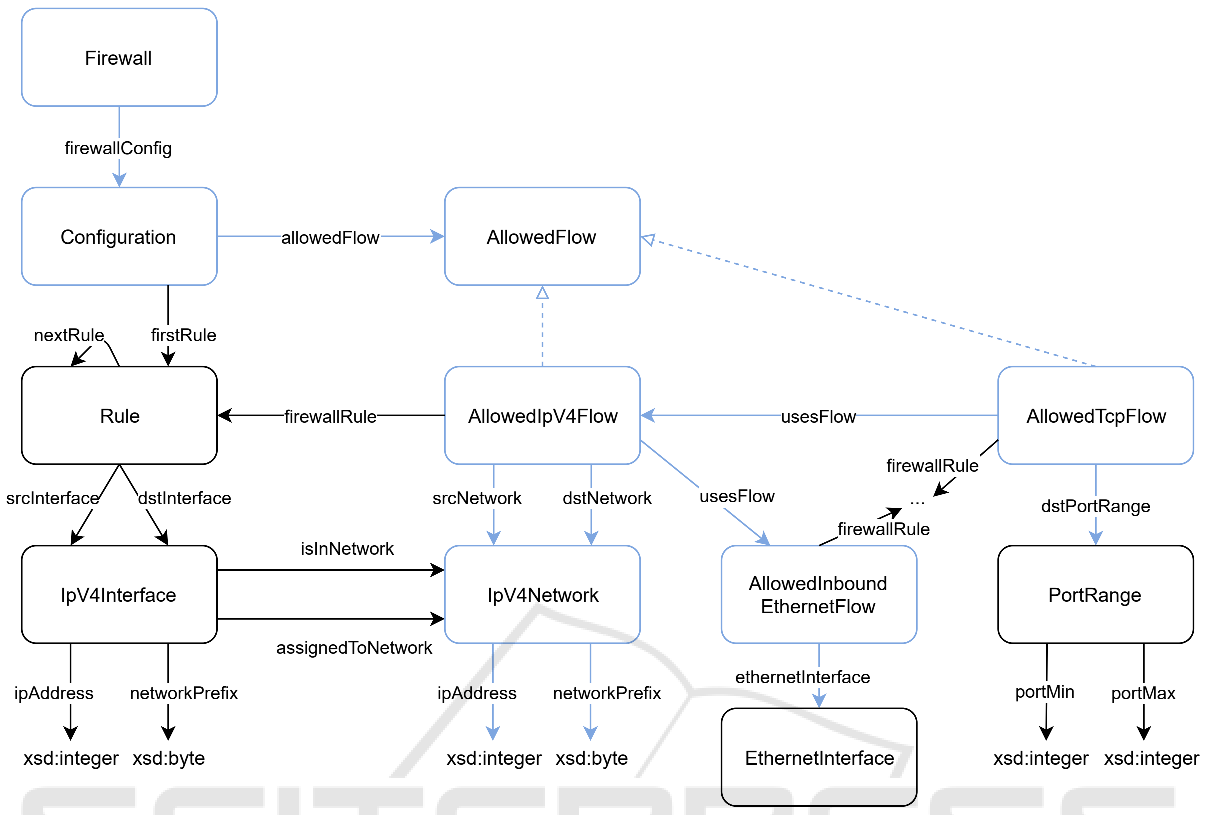

Figure 2: Excerpt of the effective configuration ontology.

Listing 4: SQWRL statement selecting all firewalls, which have a matching allowed flow.

```
containsNetwork(FieldZone, ?sn)
^ containsNetwork(DMZ, ?dn)
^ Firewall(?f)
^ firewallConfig(?f, ?fc)
^ AllowedTcpFlow(?atf)
^ allowedFlow(?fc, ?atf)
^ usesFlow(?atf, ?aif)
^ AllowedIpV4Flow(?aif)
^ srcNetwork(?aif, ?sn)
^ dstNetwork(?aif, ?dn)
-> sqwrl:select(?f)
```

Consequently, if the result is empty, the link is not allowed and vice versa.

Let's assume that the analysis should further determine whether a path from the Field Zone to the DMZ exists which only passes firewalls from the queried set. Because each IPv4 interface (including the ones of firewall components) is directly linked to an IPv4 network, a transitive "connected" relation between firewalls can be inferred, if they share a network. If Listing 4 is extended by the condition `connected(?f, ?f)`, the additional requirement is fulfilled.

5 DISCUSSION AND FUTURE WORK

As this section shows, the effective configuration approach does not suffer from Scalability Issues, since the model is created once and generic model extension and analysis can be applied to it. Instead of formalizing configuration on a generic or abstract level, the effective configuration strategy models all protocol-specific patterns that can trigger NAC actions. This is the reason for the allocation of only 2 points for aspect 4 in Table 1.

The strategy expresses all network-level effects of the configuration by design and thus does not suffer from the Expressiveness Issue. Furthermore, if used accompanied by a model of the rule chain, the approach can even provide inference of a set of rules responsible for an identified problem. Therefore, although the approach does not improve rule-level analysis, to the best of our knowledge, it is the first approach to provide traceability to corresponding NAC rules as part of an identification of a cross-NAC problem. The ability to support configuration-language-specific rule models implies that the approach fully

supports Approach 3 which means that it indirectly earns the high score of aspect 1 in Table 1. In addition, the fixed strategy of deriving the disjoint patterns accompanied by their affiliation to the block or allow actions mitigates the Transformation Issue. Nevertheless, it should be noted that a reverse transformation has not been analysed. This and the fact that still a configuration-language specific transformation is needed once for every language is the reason why the approach does not get allocated the high score regarding aspects 5 and 6 in Table 1.
Stateful control and filtering configurations are currently ignored by our implementations. In future work we will analyse to what extend they should be part of the effective configuration, resp. linked to it.

6 CONCLUSION

This paper discusses approaches for modelling network access control configuration for security analysis, identifies issues and proposes an alternative approach solving these issues. For this, the effective configuration of rule chains is derived and modelled in a configuration-language-agnostic way. The paper shows that the approach does not suffer from scalability problems, since the model is created once and generic model extensions and analyses can be applied to it. The effective configuration represents all protocol-specific patterns that can trigger actions in a rule chain. Therefore, it expresses all network-level effects of the configuration and, if used with a model of the rule chain, can even provide inference of a set of rules responsible for an identified problem. Therefore, the effective configuration approach is an improvement over existing approaches and, moreover, can be used in combination with them.

ACKNOWLEDGEMENTS

This work has been funded by the Federal Ministry of Education and Research (BMBF, Germany) in the project AICAS (project number 16KIS1063K).

REFERENCES

Cordova, R. F., Marcovich, A. L., and Santivanez, C. A. (2018). An Efficient Method for Ontology-Based Multi-Vendor Firewall Misconfiguration Detection: A Real-Case Study. In Cely Callejas, J. D., editor, *2018 IEEE ANDESCON*, pages 1–3, [Piscataway, New Jersey]. IEEE.

Davy, S., Barron, J., Shi, L., Butler, B., Jennings, B., Griffin, K., and Collins, K. (2013). A language driven approach to multi-system access control. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1004–1008.

Fitzgerald, W. M., Foley, S. N., and Foghlú, M. Ó. (2008). Network access control interoperation using semantic web techniques. In *WOSIS*, pages 26–37.

Fitzgerald, W. M., Foley, S. N., Foghlú, M. Ó., et al. (2009). Network access control configuration management using semantic web techniques. *Journal of Research and Practice in Information Technology*, 41(2):99.

Fitzgerald, W. M., Foley, S. N., and O'Foghlu, M. (2007). Confident firewall policy configuration management using description logic. In *12th Nordic Workshop on Secure IT Systems*, pages 11–12.

Foley, S. N. and Fitzgerald, W. M. (2008). Semantic web and firewall alignment. In *2008 IEEE 24th International Conference on Data Engineering Workshop*, pages 447–453.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Hu, H., Ahn, G.-J., and Kulkarni, K. (15.10.2011 - 18.10.2011). Ontology-based policy anomaly management for autonomic computing. In *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE.

Khelf, R. and Ghoualmi-Zine, N. (26.11.2018 - 27.11.2018). Ipsec/firewall security policy analysis: A survey. In *2018 International Conference on Signal, Image, Vision and their Applications (SIVA)*, pages 1–7. IEEE.

Martínez Pérez, G., García Clemente, F. J., and Gómez Skarmeta, A. F. (2007). Managing semantic–aware policies in a distributed firewall scenario. *Internet Research*, 17(4):362–377.

O'Connor, M. and Das, A. (2009). SQWRL: A Query Language for OWL. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529*, OWLED'09, pages 208–215, Aachen, DEU. CEUR-WS.org.

Patzer, F., Volz, F., Usländer, T., Blocher, I., and Beyerer, J. (2019). The industrie 4.0 asset administration shell as information source for security analysis. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 420–427. IEEE.

Textor, A., Stynes, J., and Kroeger, R. (2010). Transformation of the common information model to owl. In *Current trends in web engineering*, Lecture Notes in Computer Science SL 3, Information Systems and Application, incl. Internet/Web and HCI, pages 163–174, Berlin. SpringerLink.

Tong, W., Liang, X., Li, X., Zhao, J., and Liang, X. (2015). An analysis method of nac configuration conflict based on ontology. In *Proc. of the 3rd International Conference on Digital Enterprise and Information Systems (DEIS2015)*, page 46.