Masters's Thesis Computing Science

Developing a Comprehensive Ontology for Representing Defensive Capabilities in IT Infrastructures

Luuk Maas

September 20, 2024

External supervisor
dr. Hidde-Jan Jongsma, TNO

Internal supervisor/assessor
prof. dr. ir. Harald Vranken, Radboud University

Second assessor
dr. ir. Erik Poll, Radboud University

Radboud University

Acknowledgements

I would like to express my deepest gratitude to my supervisors dr. Hidde-Jan Jongsma and prof. dr. ir. Harald Vranken, for their continued support, guidance, and invaluable feedback throughout the process of writing this thesis. Their insights and expertise were instrumental in shaping my research.

Furthermore, I would like to extend heartfelt thanks to my family and friends, whose unwavering personal support has been a constant source of encouragement, not only during the writing of this thesis but throughout my entire studies.

Abstract

As cyber attacks become increasingly sophisticated and automated, there is a growing necessity for IT infrastructures to implement automated security reasoning and decision-making processes. This thesis presents the development of the Security Actuator Ontology (SAO), which has been designed to model the defensive capabilities of IT infrastructures at a tactical level. The SAO builds upon the MITRE D3FEND framework by extending existing ontologies to incorporate security actuators and their relationships with infrastructure components. This enables more precise and actionable security reasoning. The ontology was validated using a case study involving a typical corporate infrastructure, demonstrating its ability to support automated reasoning by mapping infrastructure elements to security actuators and inferring relationships between them. The results show that the SAO is effective in enhancing cybersecurity operations by bridging the gap between abstract defensive techniques and their real-world applications. This research highlights the potential of ontology-based models for automated security reasoning and paves the way for future extensions, such as dynamic threat intelligence integration and temporal reasoning.

Contents

Chapter 1

Introduction

Cyber attacks are becoming increasingly frequent and complex, employing sophisticated techniques that often outpace traditional defense mechanisms. Among the most significant causes is the reliance on human operators for cyber defense, while attacks are becoming more automated [1]. This escalating threat has profound implications for the security of IT infrastructures, as organizations struggle to keep up with the rapidly evolving tactics of malicious actors [2, 3]. To address this issue, TNO proposes an automated approach for cyber security [1].

TNO considers that significant advancements towards automated security can be made by addressing two directions: automated security reasoning and automated response. Automated security reasoning technology aims to provide a contextual understanding of threats and events and subsequently make informed decisions about response strategies. The foundation of such reasoning is a sophisticated model of the IT infrastructure to be defended [1]. However, there are few solutions available for modeling infrastructures, particular those that are capable of representing defensive capabilities at a tactical level. It is challenging to reason about potential cyber countermeasures without having a holistic model of the infrastructure, which includes servers, network devices, applications, databases, user endpoints, and their configurations.

Relevant previous research covering infrastructure modeling includes the thesis by Oosterhof in 2019, which presents an automatic method for modeling an IT infrastructure from different sources – such as network scanners. However, this research did not result in a reusable standard that can be used to reason about the defensive capabilities of such an infrastructure. This is because the proposed method is limited to represent only very basic information about the infrastructure [4].

A more recent (2023) study by Ganesh explored the feasibility of using the SBOM (Software Bill of Material) concept to describe IaC (Infrastructure as Code) deployments. While this was found possible, it precluded the application to security analysis due to the static and verbose nature [5].

Because of these limitations, we attempt to develop an extendable ontology that can be used to express the defensive capabilities of an IT infrastructure at a tactical level. By doing so we aim to make progress towards automated decision-making in cyber security.

1.1 Research Questions

The following research questions describe the aim of this research:

RQ1. What does an ontology of security actuators look like? The development of the ontology will be carried out in accordance with the guidelines set forth in [6]. The precise methodology is outlined in Chapter 3, while the resulting ontology is detailed in Chapter 4.

RQ2. How can such an ontology be mapped to an IT infrastructure, such that the defensive capabilities of such an infrastructure can be expressed? In order to validate the ontology, it will be applied to a case study involving an example IT infrastructure, as presented in Chapter 5. This practical application will demonstrate how the ontology models and represents the defensive capabilities of the infrastructure, thus allowing for an assessment of its effectiveness and completeness. The insights gained from this case study will contribute to the refinement of the ontology and ensure that it meets the intended goals of enabling automated reasoning and decision-making for cyber security at a tactical level.

Finally, this thesis is outlined as follows: Chapter 2 provides some background information on the important topics covered in this thesis; Chapter 3 explains *how* the Security Actuator Ontology was developed; Chapter 4 describes *what* the resulting ontology looks like; Chapter 5 illustrates how the ontology can be applied through a case study. Finally, we discuss related work and present our conclusions in Chapter 6 and 7, respectively.

Chapter 2

Background

In order to gain a comprehensive understanding of this thesis, it is essential to understand the key concepts that form its foundation. This chapter serves to explain those concepts.

2.1 Security Actuators

We conduct this research at TNO[1], which uses some internal terminology, including *security actuator*. Since no formal definition has been established, we will define it as follows for the scope of this study:

Definition 2.1.1 (Security Actuator)**.** A security actuator is a component within an IT infrastructure that executes specific defensive actions or responses with the objective of mitigating, countering or neutralizing detected cyber threats. These actions can include automated tasks such as blocking malicious traffic, updating firewall rules, isolating compromised systems or deploying security patches, thereby enhancing the overall security posture of the infrastructure.

Although some literature employs the terms "sensors" (which take input) and "actuators" (which produce output), a security actuator may perform both of these functions.

[1]TNO is the Netherlands Organization for Applied Scientific Research. It is the largest fully independent research, development and consultancy organisation in the Netherlands.

2.2 Ontologies

As the concept of ontologies forms part of the field of information sciences, we will provide a brief explanation of the concept in the next subsection. If the reader is already familiar with the concept of ontologies, they may proceed to Section 2.2.2.

Noy and McGuinness state that an ontology defines a common vocabulary for researchers who need to share information in a certain domain. Developing such an ontology is rarely an objective by itself: "it is akin to defining a set of data and their structure for other programs to use" [6].

Ontologies provide a standardized framework for representing knowledge, which is particularly valuable in complex domains such as cybersecurity. By structuring information in a formal and consistent way, ontologies allow concepts and relationships to be clearly defined, reducing ambiguity and improving communication between stakeholders [7].

Ontologies facilitate automated reasoning by providing a formal, structured representation of knowledge that can be processed by machines. In the context of cybersecurity, this is particularly advantageous for the generation and analysis of attack and defense graphs. Attack graphs model the potential paths an adversary could take to compromise a system, while defense graphs map out the corresponding defensive measures [8]. By employing ontologies, these graphs can be dynamically generated and updated based on new threat information, enabling systems to automatically identify vulnerabilities and suggest appropriate countermeasures. The ontology developed in this research pertains only to defense graphs, as its focus is on defensive capabilities.

2.2.1 Developing an Ontology

In essence, the process of developing an ontology entails the following [6]:

- defining **classes** in the ontology;
- arranging these classes in a **taxonomic** hierarchy;
- defining **properties** and describing the allowed values for these properties.

In order to illustrate these concepts, we will present an example from the MITRE D3FEND Ontology, as will be further discussed in Section 2.2.2. A small extract of the ontology is provided in Figure 2.1. The boxes represent

classes, and the arrows indicate the hierarchical structure. The top-level class, in this case `Digital Artifact`, is the most abstract. As we proceed in a downward direction through the hierarchy of classes, the classes become increasingly concrete. In other words, the arrows display a *subclass-of* relationship between the classes.

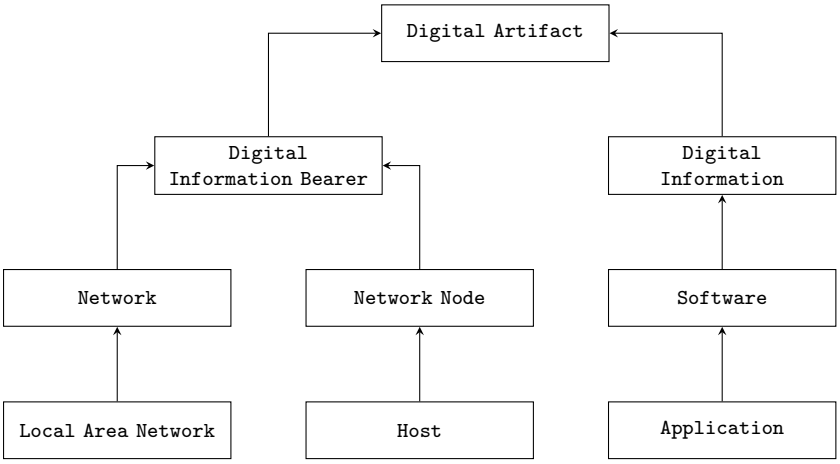

Figure 2.1: Extract from the D3FEND Ontology class hierarchy

From this, we can infer that a `Host` is a `Network Node` which in turn is a `Digital Information Bearer`. In other words, `Host` is a subclass of `Network Node` which is a subclass of `Digital Information Bearer`. Finally, some example properties from this ontology include (not shown in Figure 2.1):

- `Local Area Network` *may-contain* `Host`
- `Host` *contains* `Application`

Properties are inherited by its subclasses. For example, `Network Node` is involved in the following property: `IP Address` *identifies* `Network Node`. By the class hierarchy from Figure 2.1, we derive that a `Host` is also identified by an `IP Address`.

2.2.2 Existing Cyber Security Ontologies

MITRE D3FEND

While ontologies in the field of Computing Science are not uncommon, there are few relevant cyber security ontologies. With respect to defensive cyber

security, we consider the MITRE D3FEND Ontology to be state-of-the-art: "a framework in which we encode a countermeasure knowledge base, but more specifically, a knowledge graph. The graph contains semantically rigorous types and relations that define both the key concepts in the cybersecurity countermeasure domain and the relations necessary to link those concepts to each other" [9].

The core of the D3FEND Ontology is the D3FEND Matrix[2], which describes what needs to be done in response to some adversary action in the form of *tactics*. These tactics are further specified by so-called *base-techniques*, which in turn are further specified by *techniques* [9]. As an example, see Table 2.1, which displays an extract from the matrix. We see the tactic `Detect`, base-techniques `Network Traffic Analysis` and `File Analysis` and techniques `DNS Traffic Analysis` and `Dynamic Analysis`. The concepts progress from the most abstract to the most specific, from top to bottom. The relation between the concepts (classes) are the same as explained in Figure 2.1. This example also indicates the level of abstraction from D3FEND: it outlines the countermeasures to be performed, but it does not explain *how* these techniques should be performed. Furthermore, it does not specify what security actuators are capable of performing these techniques.

tactic →	**Detect**	
base technique →	**Network Traffic Analysis**	**File Analysis**
technique →	DNS Traffic Analysis	Dynamic Analysis
technique →	...	...

Table 2.1: Hierarchy of the D3FEND Matrix

MITRE ATT&CK

In contrast to MITRE D3FEND, which is focused on defensive cybersecurity measures, the MITRE ATT&CK framework is oriented towards understanding the offensive tactics, techniques, and procedures (TTPs) used by adversaries. The acronym ATT&CK, which stands for Adversarial Tactics, Techniques, and Common Knowledge, denotes a matrix that categories the methods employed by attackers to infiltrate and compromise IT systems. This framework provides a structured approach for organizations to analyze and document potential threats. By mapping out the stages of an attack, from initial access to data exfiltration, the ATT&CK framework serves as a valuable tool for security professionals to gain a deeper understanding of adversarial behavior and inform their defensive planning. When used in con-

[2] `https://d3fend.mitre.org/`

junction with D3FEND, these frameworks offer a complementary perspective on cybersecurity, covering both offensive and defensive considerations [10].

2.3 Infrastructure as Code

Infrastructure as Code (IaC) is a methodology whereby the management and provisioning of infrastructure, including servers, networks, and storage, is conducted through the utilization of machine-readable configuration files, as opposed to manual hardware configuration or the use of interactive configuration tools. This approach permits the automation of consistent and repeatable deployments of IT environments, thereby reducing the risk of human error and enabling more rapid and reliable infrastructure management.

Widely-used tools that facilitate the implementation of the IaC concept include Ansible and Puppet.

- **Ansible** is renowned for its straightforwardness and user-friendliness, utilizing YAML for configuration management [11]. However, its configurations are often too specific to the tasks at hand, which limits their adaptability for broader automated security reasoning. To illustrate this, consider an example configuration, depicted in Listing 1. The reasoning process is constrained by its narrow focus on specific tasks, such as enabling a firewall, rather than on more encompassing countermeasures that could be applied across diverse infrastructure contexts.

```
- name: Ensure firewall is enabled on all hosts
  hosts: all
  tasks:
    - name: Check if firewall is running
      shell: ufw status | grep "Status: active"
      register: firewall_status

    - name: Enable firewall if not active
      shell: ufw enable
      when: firewall_status.rc != 0
```

Listing 1: Example Ansible configuration

- In contrast, **Puppet** employs a declarative language for configuration management, making it more powerful in managing complex environments [12]. Nevertheless, Puppet's proprietary formats and comprehensive configurations frequently result in verbosity, which complicates

```
class firewall {
  package { 'ufw':
    ensure => installed,
  }

  service { 'ufw':
    ensure => running,
    enable => true,
  }

  exec { 'Enable firewall':
    command => '/usr/sbin/ufw enable',
    unless  => '/usr/sbin/ufw status | grep "Status: active"',
  }
}
```

Listing 2: Example Puppet configuration

the extraction of high-level security insights essential for automated reasoning. Consider the example configuration in Listing 2. Puppet's comprehensive configuration settings, which require the explicit definition of each resource, present a significant challenge to the extraction of high-level security insights.

These constraints underscore the necessity for a more semantically transparent approach to automated security reasoning that extends beyond the scope of these particular IaC tools.

2.4 RDF/OWL

RDF (Resource Description Framework) represents a standard model for the interchange of data on the web. It permits the structured representation of data as triples (subject, predicate, object), thereby facilitating the sharing and reuse of information across different systems in a machine-readable manner. RDF serves as the foundation for semantic web technologies, providing a flexible, graph-based data structure that is suitable for a variety of applications [13]. Consider Listing 3, which depicts an example of RDF indicating a simple relationship between `ServerA` and `NetworkA`.

The Web Ontology Language (OWL) builds on the Resource Description Framework (RDF) to offer an expressive framework for representing complex knowledge. OWL enables the definition of classes, properties, and relationships between concepts in a formalized logical structure. It supports automated reasoning, enabling systems to infer new information from existing

```
<rdf:Description rdf:about="http://example.com/ServerA">
    <ex:isPartOf rdf:resource="http://example.com/NetworkA"/>
</rdf:Description>
```

Listing 3: Example RDF

data. OWL is commonly used in domains such as knowledge representation and ontology development, and will also be used for the development of the ontology in this research [14]. To demonstrate the contribution of OWL to RDF, consider the example illustrated in Listing 4, which depicts the definition of classes in OWL and the addition of restrictions on the properties of these classes.

```
<owl:Class rdf:about="http://example.com/Network"/>

<owl:Class rdf:about="http://example.com/Server">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://example.com/isPartOf"/>
            <owl:allValuesFrom rdf:resource="http://example.com/Network"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Listing 4: Example OWL

Chapter 3

Methodology

3.1 Security Actuator Ontology

This section outlines the process by which the Security Actuator Ontology was developed. This process broadly aligns with the methodology outlined in [6].

At the beginning of the ontology development process, it is recommended that the domain and scope be defined. This can be achieved by answering four key questions.

1. **What is the domain that the ontology will cover?** The domain of the ontology is that of security actuators, as defined in Section 2.1. Furthermore, in order to express the capabilities of these security actuators with respect to the infrastructure in which they are situated, it is necessary for the domain to also include IT infrastructure components. It is anticipated that this ontology will not contain *all* security actuators, given the wide range of such actuators and the duration of this research. However, we will design this ontology to be easily extendable in the future. For the scope of this research, we will define the security actuators such that we cover a wide range of the countermeasures from the D3FEND Matrix. Furthermore, it should be possible to map the ontology to a basic infrastructure, as presented in Chapter 5.

2. **What are we going to use the ontology for?** The ontology will be employed to effectively model and represent the defensive capabilities of IT infrastructures. In particular, the ontology will facilitate the identification of various security actuators, the mapping of their

relationships with the components they protect, and the automation of reasoning about the infrastructure's defensive posture.

3. **For what kind of questions should the information in the ontology provide answers?** In order to provide a more concrete answer to the preceding question, we present the following questions, which should be answered using an implementation of this ontology. Note that this list is not exhaustive; it just represents the type of questions to be answered by the ontology.

    - What are the defensive capabilities of security actuator X – in terms of D3FEND techniques?
    - What hosts/network segments are affected by security actuator X?
    - Given host X, what are the defensive capabilities for that host?

    It should be noted that it should also be feasible to make these queries in the inverse order. For instance for the first question, it should be possible to query what security actuators are capable of executing defensive measure X, too.

4. **Who/what will use the ontology?** The ontology will be used primarily internally at TNO as a basis for automated security reasoning solutions.

Having defined the domain and scope of the ontology, the subsequent step was to consider the possibility of reusing existing ontologies [6]. As mentioned earlier, many ontologies about general computing science concepts exist. Following an initial literature review of a number of existing ontologies, two were chosen for further investigation to determine whether they could be extended in a manner that would align with the aforementioned aims.

The ontologies that were included in the initial review are listed in Table 3.1. Firstly, the Communications Network Modeling Ontology (CNMO) places an emphasis on structural network components and does not include any security mechanisms, countermeasures or other security concepts [15]. This renders CNMO unsuitable as a basis for our ontology. The Open Cybersecurity Schema Framework [16] lacks any infrastructure components, which makes it challenging to model the impact of security actuators on the underlying infrastructure. INDL, NML and MITRE D3FEND are discussed in the following sections.

Ontology	Review	Reference
Communications Network Modeling Ontology (CNMO)	Too much focus on network design, no security mechanisms	[15]
Open Cybersecurity Schema Framework (OCSF)	Lacking core concepts and technically not an ontology	[16]
NML/INDL	See Section 3.1.1	[17, 18]
MITRE D3FEND	See Section 3.1.2	[9]

Table 3.1: Reviewed Ontologies

3.1.1 Infrastructure and Network Description Language

Initially, the Infrastructure and Network Description Language (INDL) ontology was reviewed. The objective of INDL is to offer technology independent descriptions of computing infrastructures, encompassing both physical resources and network infrastructure [18]. INDL is an extension of the Network Markup Language (NML), presented in [17].

The NML Base Schema provides a model for the description of network topologies, which is primarily utilized in research and educational contexts. It allows for the representation of network components and their relationships, including nodes, ports, links, and various network services, such as switching and adaptation services. By utilizing XML and RDF/XML syntaxes, the language is both machine-readable and human-interpretable, which allows for interoperability with other systems. Its extensibility allows for updates to accommodate technological advancements and evolving network requirements [17].

The INDL ontology is designed to provide a technology-agnostic framework for the description of computing infrastructures, encompassing both physical resources and network elements. By employing the Network Markup Language, INDL aims to facilitate the generation of reusable and extensible models that accommodate resource virtualization and service description. The INDL ontology's design permits the modeling of complex infrastructures, thereby facilitating tasks such as resource management, energy monitoring, and workflow planning. Furthermore, the incorporation of INDL into existing models allows for a comprehensive representation of the infrastructure, thereby facilitating more effective infrastructure management and optimization [18].

The following sections will present an analysis of the strengths and limi-

tations of the NML/INDL model in relation to its potential extension to accommodate the inclusion of security actuators.

Strengths and Limitations

In order to successfully model the defensive capabilities of an IT infrastructure, it is necessary to work at an abstraction level that enables the identification of such capabilities while avoiding the trap of becoming mired in extraneous detail. However, while both INDL and NML provide frameworks to describe physical and network resources in detail, they do so at a granularity that is ill-suited to express higher-level defensive mechanisms.

The INDL ontology was presented with a particular emphasis on the provision of technology-independent descriptions of computing infrastructures, encompassing the full range of details pertaining to physical resources, virtualization, and service offerings [18]. It is closely integrated with NML for the realization of network topologies, encompassing the specifics of nodes, ports, links, and the diverse range of network services. While these models are useful for resource management and workflow planning at a fine level of detail, they also significantly complicate the characterization of general defensive capabilities across such an infrastructure.

The NML Base Schema is an extensive model of network topologies, with a particular focus on the detailed relationships and characteristics of network elements. The necessity for low-level granularity in the model, which is productive when describing complex network configurations and services, is ultimately inappropriate for expressing the more general defensive capabilities of an IT infrastructure in a few words. The description of nodes, ports and their interactions inevitably diverts attention from the higher-level security features and their interaction within the infrastructure.

Therefore, despite the strengths of the INDL and NML frameworks, their granular approach makes it challenging to effectively model defensive capabilities. What is required is a higher-level ontology that abstracts from the fine detail and focuses on the essentials of defensive mechanisms in IT infrastructures.

3.1.2 MITRE D3FEND Ontology

In contrast to the granular models of INDL and NML, the MITRE D3FEND ontology offers a more abstract and higher-level ontology. More specifically, this ontology provides a comprehensive framework that abstracts complex security measures, thereby rendering it more suitable for the expression and

management of the defensive strategies of modern IT systems. For more information about the D3FEND ontology, refer to Section 2.2.2.

Strengths and Limitations

The D3FEND framework currently serves as a useful high-level overview of the taxonomy of defensive measures, encompassing a set of defensive tactics and techniques. Such a structure allows for the organisation of defensive strategies in a structured way, however the ontology does not provide details on the specific security devices or software – the security actuators – that are responsible for executing the aforementioned tactics and techniques. Furthermore, the ontology does not provide any information regarding the hosts or network segments that are protected by such security measures.

As an example, consider the `Firewall` class. It is already part of the ontology, and typically a firewall is capable of performing `Network Traffic Filtering` – also part of the ontology. However, this relationship is not defined in the ontology right now.

An extension of the D3FEND framework would facilitate the linking of tactics and techniques to specific security actuators. Such an extension would facilitate the modeling of not only the abstract countermeasures themselves, but also the concrete implementations of these countermeasures within the IT infrastructure. The relationships between the security actuators and the hosts they protect could be defined, thus enabling a more complete visualization of the defensive posture of the infrastructure.

Following a comprehensive evaluation of the strengths and weaknesses, the D3FEND ontology has been selected as the foundation for our own ontology. This decision was based on the ontology's substantial number of relevant concepts and its level of abstraction, which renders it a robust starting point.

3.1.3 Defining Classes

In accordance with the guidelines set forth in [6], we employed a top-down approach to define the classes for our ontology of security actuators. This approach enabled us to commence with a comprehensive classification and subsequently refine it into increasingly specific categories. At the most abstract level, we present the `Security Actuator` class, which encompasses all components responsible for performing some defensive action within an IT infrastructure. This general category was subsequently subdivided into increasingly specific subcategories, which were then further subdivided as

needed. A detailed explanation of the resulting classes is provided in Chapter 4.

The hierarchy within classes has certain implications. When we re-examine the example illustrated in Figure 2.1, we can derive the following:

1. If a class A is a parent/super-class of class B, then every instance of B is also an instance of A. For example, all instances of `Host` are also an instance of `Network Node`, which in turn are also instances of `Digital Information Bearer` and `Digital Artifact`.

2. If a class has certain properties, then all its sub-classes also have those properties. For example, if the `Network Node` class has the property *runs* `OperatingSystem`, then its subclass `Host` also has that property.

3.1.4 Defining Properties

A property is a fundamental element that describes the relationships between classes and the attributes of those classes. Properties can be classified into two principal categories: object properties and data properties. Object properties define the relationships between different classes, indicating the manner in which instances of one class are related to instances of another class. To illustrate, a property indicating that one entity, designated as a `Firewall` protects another entity, designated as a `Network Segment`, may be established. In contrast, data properties describe the attributes or characteristics of a class by linking it to data values. For example, a `hasRuleset` property might link a `Firewall` class to a specific set of rules it enforces. These properties are essential for defining the structure and semantics of an ontology, enabling a detailed and precise representation of the domain being modeled.

In a similar manner to the classes, we also develop the properties top-down. This methodology is logical, given that properties propagate in a downward direction towards more specific classes.

A more thorough account of the resulting properties is provided in Chapter 4.

Chapter 4

Security Actuator Ontology

This chapter introduces the Security Actuator Ontology (SAO). This ontology was developed using the RDF/OWL format. The Resource Description Framework (RDF) and the Web Ontology Language (OWL) are widely-used formats for ontology development due to their flexibility, standardization, and support for semantic reasoning [19]. RDF provides a robust framework for representing information about resources in a graph structure, thereby facilitating the linking and integration of data across different systems. OWL builds on RDF by enabling more expressive representations of complex relationships and constraints, thus making it ideal for capturing the detailed semantics required in an ontology [20]. Additionally, both RDF and OWL are W3C standards, ensuring broad compatibility and long-term viability, which is crucial for maintaining and extending the ontology over time.

The objective of this chapter is to present an explanation of the resulting ontology and to discuss some of the design decisions that were made. We will discuss the class structure (Section 4.1) and the properties and relationships of these classes (Section 4.2). These sections present excerpts from the ontology, which are included here as illustrative examples. The complete ontology is available for reference in Appendix A.

Note: the prefixes `d3fend:` *and* `sao:` *are employed in the excerpts below, in order to distinguish between components from the D3FEND ontology and the security actuator ontology as delineated in this study.*

4.1 Class Structure

The entire ontology derives from the base class `Security Actuator` – see Figure 4.1 for an overview of the class structure. It is important to note that

this base class should not be instantiated directly, nor should the subsequent five subclasses (all other subclasses can be instantiated):

- `Access Control Actuator`
- `Data Security Actuator`
- `Endpoint Security Actuator`
- `Management and Monitoring Actuator`
- `Network Security Actuator`

If a certain security actuator cannot be instantiated without violating the aforementioned, a new class will need to be introduced.

Listing 5 shows how classes are defined using the RDF/OWL format. Furthermore, Listing 6 illustrates how subclasses are defined, exemplified by the `Access Control Actuator` class.

```
<owl:Class rdf:about="sao:SecurityActuator">
    <rdfs:label>Security Actuator</rdfs:label>
</owl:Class>
```

Listing 5: Implementation of the base Class – `Security Actuator`

```
<owl:Class rdf:about="sao:AccessControlActuator">
    <rdfs:subClassOf rdf:resource="sao:SecurityActuator"/>
    ...
</owl:Class>
```

Listing 6: Example of subclasses – `Access Control Actuator` is a subclass of `Security Actuator`

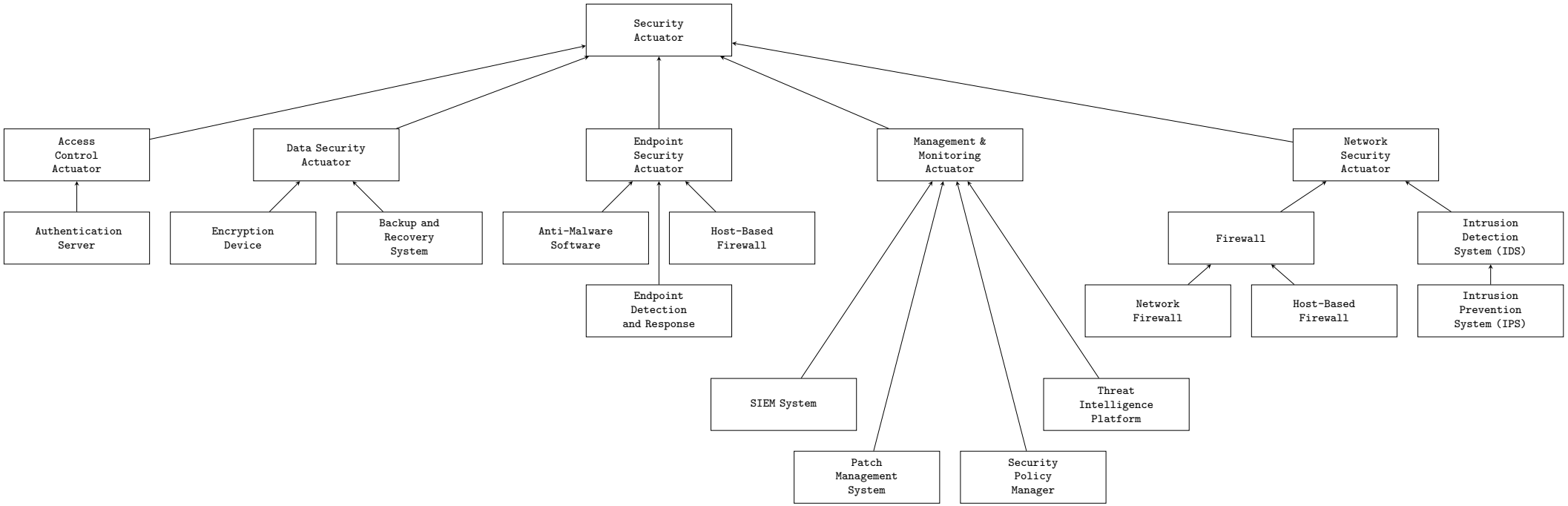

Figure 4.1: Class Hierarchy of the Security Actuator Ontology

4.2 Properties and Relationships

As outlined in Chapter 3, Section 3.1, one of the objectives is to determine which hosts or network segments are affected by a specific security actuator. For this purpose, we define the `protects` property as depicted in Listing 7. The use of this property enables the linkage of security actuators to the affected components of the underlying infrastructure. As an example, an `Endpoint Security Actuator` protects a certain `Host`, which we define as illustrated in Listing 8.

```
<owl:ObjectProperty rdf:about="sao:protects">
    <definition>x protects y: Security actuator x protects Host/Network
    ↪  (segment) y</definition>
    <rdfs:label>protects</rdfs:label>
</owl:ObjectProperty>
```

Listing 7: Definition of the `protects` property

```
<owl:Class rdf:about="sao:EndpointSecurityActuator">
    ...
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom rdf:resource="d3fend:Host"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    ...
</owl:Class>
```

Listing 8: Example usage of the `protects` property

4.2.1 Relation to D3FEND

An additional criterion of the ontology was to be able to express the defensive countermeasures that a security actuator was capable of executing, in the form of D3FEND techniques. For this purpose, we define the `performs` property, as depicted in Listing 9.

Using this property, we are able to link D3FEND techniques to security actuators. For instance, an `Authentication Server` is capable of executing `Credential Hardening`, `Credential Eviction` and `User Behavior Analysis`. We restrict the defensive techniques a security actuator can perform, as exemplified in Listing 10.

```
<owl:ObjectProperty rdf:about="sao:performs">
    <definition>x performs y: Security actuator x performs technique
    ↪  y</definition>
    <rdfs:label>performs</rdfs:label>
</owl:ObjectProperty>
```

Listing 9: Definition of the `performs` property

```
<owl:Class rdf:about="sao:AuthenticationServer">
    ...
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:CredentialHardening"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:CredentialEviction"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:UserBehaviorAnalysis"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    ...
</owl:Class>
```

Listing 10: Example usage of the `performs` property

```
<owl:ObjectProperty rdf:about="sao:is-protected-by">
    <owl:inverseOf rdf:resource="sao:protects"/>
    <definition>x is protected by y: Host/Network x is protected by agent
    ↪  x</definition>
    <rdfs:label>is-protected-by</rdfs:label>
</owl:ObjectProperty>
```

Listing 11: Definition of the `is-protected-by` property

When a specific D3FEND technique is linked to a security actuator, it is assumed that all subclasses of that technique can also be executed by that security actuator, and vice versa. Thus, if a security actuator is linked to a certain technique, it can be inferred that all subclasses of that security actuator can also execute that technique.

Since defensive techniques from the D3FEND ontology also contain information about their function, we can also deduce this information thanks to this relationship between security actuators and defensive techniques. For example, D3FEND states that `Credential Eviction` *`deletes`* `Credential` and *`disables`* `User Account`.

4.2.2 Inverse Properties

As previously stated in Chapter 3, the objective is to facilitate queries in two distinct directions. For example, given a specific security actuator, the aim is to learn which components of the infrastructure are influenced by it. Conversely, when a particular component of the infrastructure, such as a specific host, is given, the goal is to determine which security actuators have an influence on that host.

To facilitate that, we define inverse properties. For example, we define `is-protected-by` as the inverse property of `protects`, as depicted in Listing 11. Thus, while a certain security actuator `protects` – for example – a certain host, a host `is-protected-by` a certain security actuator.

Although inverse properties can be derived implicitly, explicitly defining them enhances the ontology's quality. It optimizes the model's usability and clarity, enabling users to query in both directions with greater intuitiveness. Furthermore, explicitly defined inverse properties can improve reasoning performance by reducing the computational load on reasoners, as they eliminate the necessity to infer these relationships dynamically.

By defining such a property as the inverse of an existing property, an on-

tology reasoner[1] can automatically infer that property. For example, if we define a `protects` relationship between two instances, the `is-protected-by` relationship can be inferred, and vice versa.

4.2.3 Chained Properties

Another way of automatically inferring properties, is through a concept called *property chains*. Property chains facilitate the definition of complex relationships by enabling the combination of multiple object properties into a specified sequence. This mechanism enables the automatic inference of new relationships between individuals based on the presence of existing linked properties.

In order to illustrate this concept, we will examine an example from our ontology. For the purposes of this example, we will instantiate our ontology by defining the following three individuals:

- Individual `NF`, instance of class `Network Firewall`
- Individual `LAN`, instance of class `Local Area Network`
- Individuals `PC1` and `PC2`, instances of class `Personal Computer` (subclass of `Host`)

Subsequently, some object properties are defined, stating that `PC1` and `PC2` are part of `LAN` and that `NF` protects `LAN`:

- `LAN contains PC1`
- `LAN contains PC2`
- `NF protects LAN`

Currently, there is no explicit relationship between the network firewall and the PCs. However, it can be deduced that these PCs are (indirectly) protected by the network firewall. This relationship can be inferred by chaining the properties as follows:

$$\texttt{protects} \circ \texttt{contains} \rightarrow \texttt{protects}$$

[1] An ontology reasoner is a software tool that automatically infers new knowledge and relationships from the explicitly defined concepts and rules in an ontology. This ensures logical consistency and enables advanced querying and analysis.

```
<owl:ObjectProperty rdf:about="sao:protects">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <rdf:Description rdf:about="sao:protects"/>
        <rdf:Description rdf:about="d3fend:contains"/>
    </owl:propertyChainAxiom>
    <definition>x protects y: Security actuator x protects Host/Network
    ↪  (segment) y</definition>
    <rdfs:label>protects</rdfs:label>
</owl:ObjectProperty>
```

Listing 12: Definition of the `protects` ∘ `contains` → `protects` property chain

Figure 4.2 provides a visual representation of the explicit properties (solid arrows) and the resulting inferred properties (dashed arrows). This approach allows for the convenient definition of a network firewall's protective function for an entire local network while still enabling reasoning about a single host.

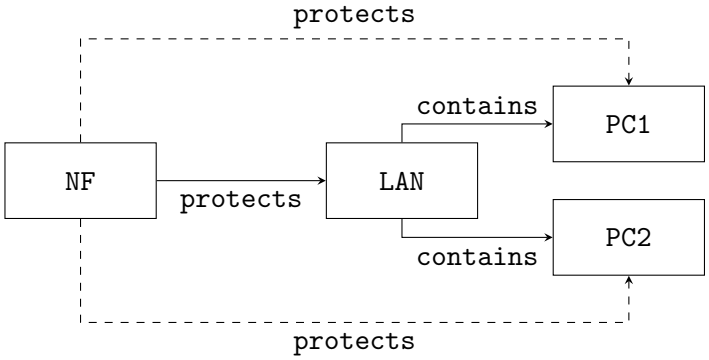

Figure 4.2: Inferred properties through property chaining

Finally, Listing 12 illustrates how a chain of properties is defined in the ontology in RDF/OWL – in this case within the `protects` property.

4.2.4 Domains and Ranges

Finally, we define domains and ranges for the `protects` and `performs` properties. The domain defines the type of subject pertaining to the property, whereas the range defines the type of object associated with the property. The domains and ranges of the aforementioned properties are relatively straightforward, as illustrated in Listing 13 and 14.

```
<owl:ObjectProperty rdf:about="sao:protects">
    ...
    <rdfs:domain rdf:resource="sao:SecurityActuator"/>
    <rdfs:range rdf:resource="d3fend:DigitalArtifact"/>
    ...
</owl:ObjectProperty>
```

Listing 13: Definition of the `protects` domain and range

```
<owl:ObjectProperty rdf:about="sao:performs">
    ...
    <rdfs:domain rdf:resource="sao:SecurityActuator"/>
    <rdfs:range rdf:resource="d3fend:DefensiveTechnique"/>
    ...
</owl:ObjectProperty>
```

Listing 14: Definition of the `performs` domain and range

Chapter 5

Ontology Validation

This chapter presents a case study designed to validate the Security Actuator Ontology, which was previously presented in Chapter 4. A representative IT infrastructure, comprising common components such as routers, company computers (workstations), network firewalls, and so forth, will be mapped to the ontology. This process will establish a correspondence between the elements of the infrastructure and their representations in the ontology. To further validate the ontology, queries will be posed to test its reasoning capabilities, ensuring that it supports the desired inferences related to defensive capabilities and security management. This process will demonstrate the effectiveness of the ontology in enabling (automated) reasoning and identifying security relationships within IT infrastructures.

5.1 The Case

The case study presents a straightforward corporate infrastructure, comprising a number of components that are commonly encountered. Figure 5.1 presents a visual overview of this infrastructure. In addition to the information in Figure 5.1, we assume the following:

- All laptops are equipped with anti-malware software and an Endpoint Detection and Response (EDR) tool.
- The file server is equipped with anti-malware software, an encryption tool, and a backup system.
- The application server incorporates patch management software, which facilitates the updating of applications in response to the discovery of a vulnerability.

- All client computers (as depicted on the right in Figure 5.1) are reliant on the servers (as depicted on the left in Figure 5.1) for providing services such as launching applications and file storage.
- The routers and switches are solely responsible for routing, and therefore do not perform any filtering of network traffic.

5.2 Mapping to the Ontology

Having delineated the infrastructure in this case study, the subsequent step is to construct a mapping of the components of the infrastructure to the ontology. This mapping is demonstrated in Figure 5.2. The class of each component is presented in **bold**, and below that the name of the corresponding instance is presented in `typewriter` style.

Although the individual components are self-explanatory, we also define three networks that serve as logical boundaries to facilitate the grouping of components. This approach allows us to reason about a group of components while also defining properties on a number of components simultaneously. For example, we define a *`protects`* relationship between `NetworkFirewallB` and `LocalAreaNetworkA`. By doing so, we can infer that all components contained in `LocalAreaNetworkA` – such as `FileServerA` and so forth – are also protected by `NetworkFirewallB`.

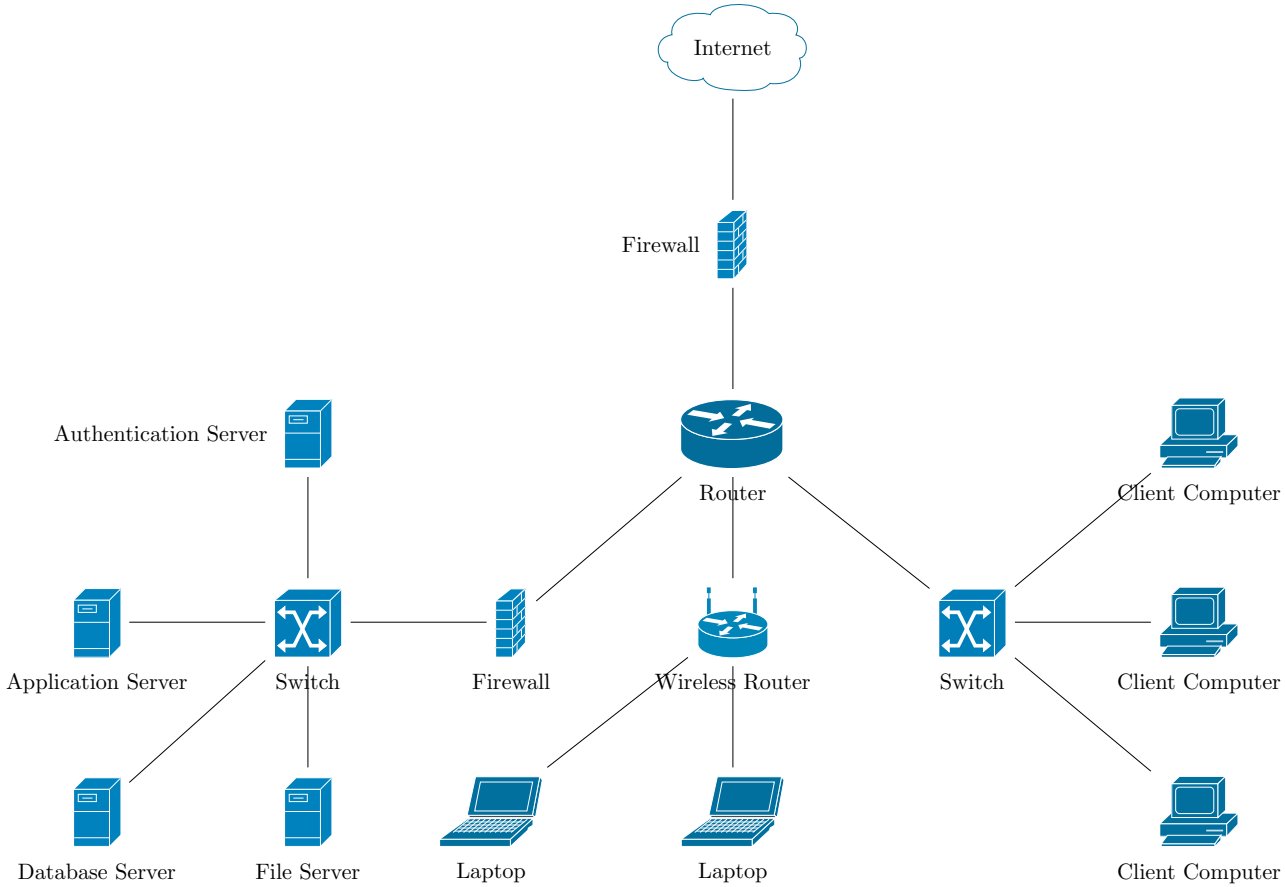

Figure 5.1: Example Infrastructure for Case Study

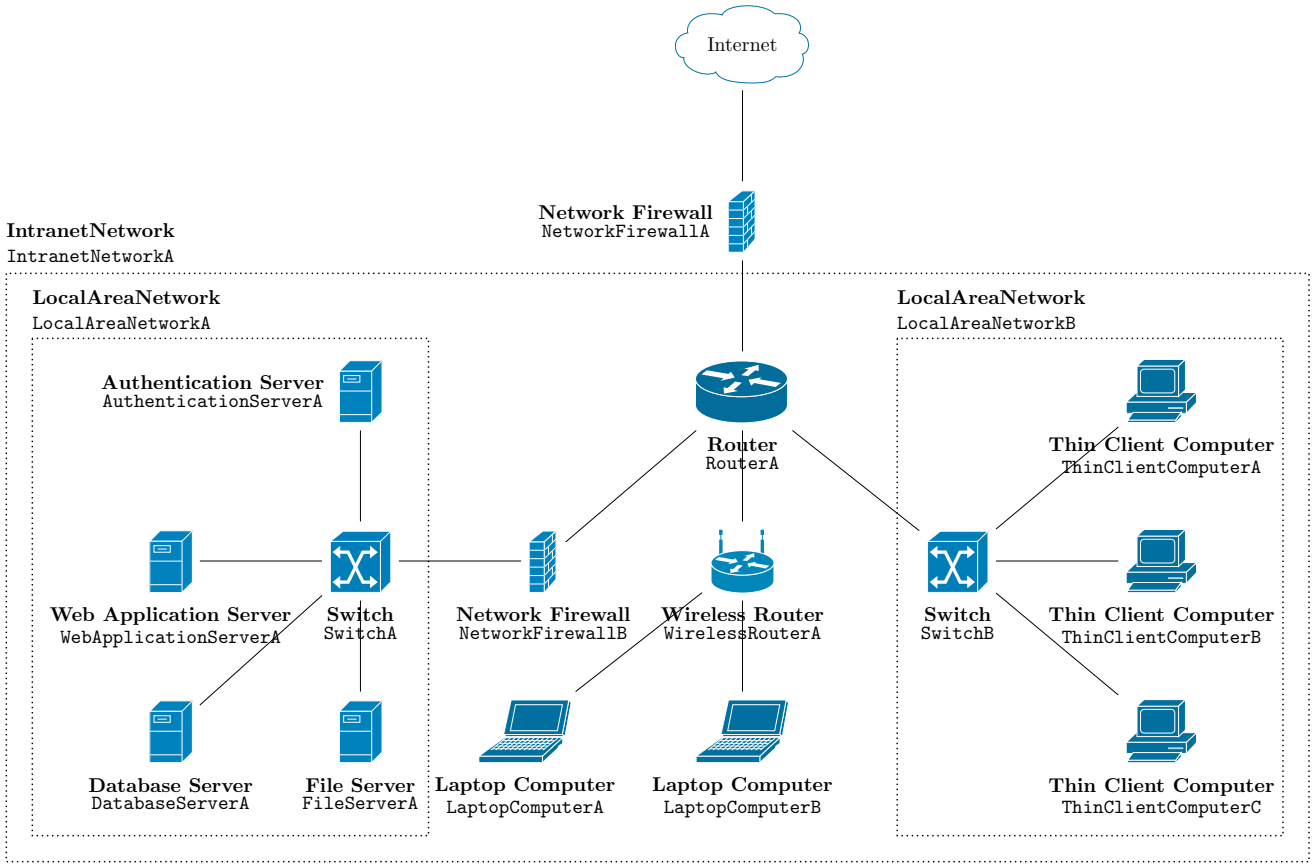

Figure 5.2: Mapping of the Case Study to the Ontology

5.2.1 Defining Individuals (Instantiating Classes)

For all the instances displayed or described above, we define a so-called NamedIndividual, as depicted in Listing 15.

```
<owl:NamedIndividual rdf:about="sao:NetworkFirewallA">
    <rdf:type rdf:resource="sao:NetworkFirewall"/>
</owl:NamedIndividual>
```

Listing 15: Example of a NamedIndividual

5.2.2 Instantiating Properties

Once all individuals have been defined, object properties are instantiated on those individuals, as illustrated by the example in Listing 16. This object property defines that `AntiMalwareSoftware-FileServerA` *`protects`* `FileServerA`.

```
<owl:NamedIndividual rdf:about="sao:AntiMalwareSoftware-FileServerA">
    <rdf:type rdf:resource="sao:AntiMalwareSoftware"/>
    <sao:protects rdf:resource="sao:FileServerA"/>
</owl:NamedIndividual>
```

Listing 16: Example instantiation of a property

5.2.3 Application of a Technique

In order to define the scope of defensive techniques that a security actuator is capable of performing, a series of restrictions have been established, as outlined in Section 4.2. However, the *`performs`* property, as defined in Section 4.2.1, is a relationship between individuals, not between classes. Therefore, we must instantiate the defensive techniques in order to map an instance of a security actuator to a defensive technique. To achieve this, we utilize an RDF concept called *anonymous nodes*. These are individuals that indicate the existence of something (in this case, a defensive technique) without naming it. The anonymous node represents the reification of the execution of a technique by a security actuator on an asset. To demonstrate this, we will examine the instance of **Encryption Tool** (from the case study) performing file encryption, as outlined in Figure 5.3.

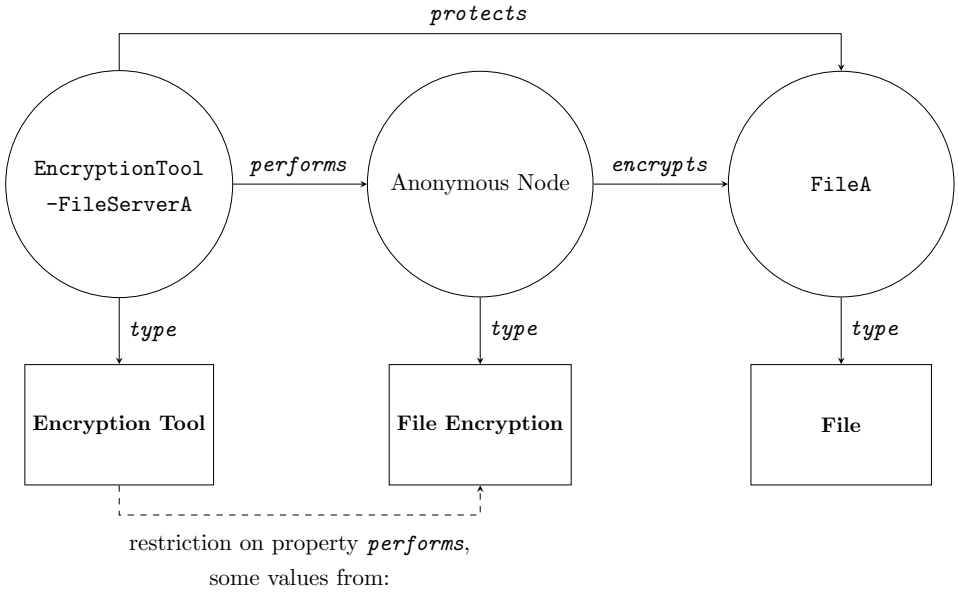

Figure 5.3: Example of *performs* property using anonymous node

5.2.4 Inferred Knowledge

Once all object properties have been defined, an ontology reasoner is able to acquire new knowledge using inverse properties (as defined in Section 4.2.2) and chained properties (as defined in Section 4.2.3). Table 5.1 provides illustrative examples of these inferred properties.

5.3 Querying the Ontology

This section illustrates the capacity of the developed ontology to respond to key questions pertaining to the IT infrastructure and its security. By posing a series of example queries, the objective is to test the ontology's reasoning capabilities, thereby ensuring that it provides accurate and meaningful inferences based on the defined classes and properties.

In Section 3.1 some example questions were posed as part of defining the goal of the ontology. We will now reiterate these questions and provide responses in the context of the case study.

1. What are the defensive capabilities of security actuator X – in terms of D3FEND techniques?

In order to gain insight into the defensive measures that can be executed by

Rule	Defined properties	Inferred property
inverse	NetworkFirewallA *protects* IntranetNetworkA	IntranetNetworkA *is-protected-by* NetworkFirewallA
chain	NetworkFirewallB *protects* LocalAreaNetworkA and LocalAreaNetworkA *contains* AuthenticationServerA	NetworkFirewallB *protects* AuthenticationServerA
chain	NetworkFirewallA *protects* IntranetNetworkA and IntranetNetworkA *contains* LocalAreaNetworkA and LocalAreaNetworkA *contains* DatabaseServerA	NetworkFirewallA *protects* DatabaseServerA

Table 5.1: Examples of inferred properties

an arbitrary security actuator, consider the EncryptionTool-FileServerA actuator. The *performs* property has been established precisely to infer this type of information. In this particular case, we learn that this actuator is capable of performing FileEncryption. Similarly, we can deduce this information for any security actuator.

2. What hosts/network segments are affected by security actuator X?

The *protects* property enables the identification of the network components that are subject to protection by a specific security actuator. For instance, considering NetworkFirewallB, we find that it has an effect on

the following infrastructure components:

- `LocalAreaNetworkA`
- `FileServerA`
- `FileA`
- `SwitchA`
- `DatabaseServerA`
- `AuthenticationServerA`
- `WebApplicationServerA`

3. Given host X, what are the defensive capabilities for that host?

In order to respond to this query, it is necessary to integrate the insights derived from two properties. Initially, it must be established which security actuators affect the host in question. This is achieved by utilizing the property *`protects`*, as demonstrated in query 2. Subsequently, the defensive countermeasures can be inferred from the security actuators that are identified.

Chapter 6

Related Works

This section presents a review of relevant literature pertaining to the development of security ontologies and automated reasoning in IT infrastructures. Through an analysis of existing works, we identify limitations that justify the necessity of the proposed Security Actuator Ontology.

Firstly, we discuss prior studies conducted at TNO. We then proceed to discuss both widely recognized models, such as MITRE D3FEND, and lesser-known models, such as Network Markup Language (NML).

6.1 Prior Research at TNO

Previous research in this field includes two studies, both of which were conducted at TNO. Firstly, Oosterhof conducted a study which facilitated the automated modeling of networks, primarily utilizing scanners. This information was stored in a database, but the semantics of the data model were not clearly defined. In addition, this method only resulted in limited information about the infrastructure. Consequently, the resulting model did not allow for automatic security reasoning [4].

Similarly, Ganesh conducted a study that explored the potential of utilizing the Software Bill of Material (SBOM) concept to model an infrastructure. While this approach was deemed feasible, its verbose nature posed a challenge in conducting a comprehensive security analysis [5].

The aforementioned issues have been addressed in the Security Actuator Ontology in the following ways. Firstly, the ontology provides sufficient information for reasoning about the security posture of an infrastructure. In the event that this is not the case, the ontology has been designed in such

a way that it can be easily extended. Secondly, the ontology's clear semantics will ensure a concise representation, facilitating automated reasoning.

6.2 D3FEND

As demonstrated in Chapter 2, the MITRE D3FEND Ontology [9] is effective at cataloging defensive techniques but lacks the ability to represent the practical defensive capabilities of an infrastructure. Specifically, the D3FEND Ontology does not specify which security actuators, such as firewalls or intrusion detection systems, are responsible for performing the defensive techniques it describes. This limitation constrains its utility in providing actionable insights into the practical application of defensive techniques within real-world systems.

To address this gap, the Security Actuator Ontology extends the D3FEND framework by incorporating security actuators and defining explicit relationships between actuators and defensive techniques. This integration enables a more precise understanding of which security actuators are responsible for protecting specific infrastructure components, making it easier to make tactical decisions based on practical defensive capabilities. As a result, D3FEND evolves from being a catalog of techniques into a comprehensive ontology that supports actionable infrastructure defense strategies.

6.3 NML/INDL

In Chapter 3, we argued that models like NML [17] and INDL [18] model infrastructures at a granularity that is ill-suited to express higher-level defensive mechanisms. These models tend to prioritize low-level details, which makes them effective for infrastructure description but limits their utility in addressing cybersecurity at a tactical level. This granularity presents a challenge in representing more abstract security concepts, such as the relationships between security actuators and the components they protect.

In contrast, the Security Actuator Ontology was developed with the objective of representing infrastructure information at a higher level of abstraction, capturing the defensive capabilities without becoming constrained by irrelevant technical details that are not pertinent to tactical decision-making. By abstracting from low-level configuration details, the ontology permits reasoning about security countermeasures, such as firewalls or encryption mechanisms, and how they can be applied across different parts of the infrastructure.

Chapter 7

Conclusions

The objective of this thesis was to develop a comprehensive ontology for the representation of defensive capabilities of IT infrastructures, with a particular focus on tactical-level considerations. The primary objective was to address the limitations of existing models, such as the MITRE D3FEND Ontology and NML/INDL, which lacked the capacity to represent practical security mechanisms or were excessively detailed, making it difficult to capture higher-level defensive strategies. The objective of developing the Security Actuator Ontology (SAO) was to create a model of infrastructure components and their corresponding security actuators, thereby facilitating more effective automated reasoning about defensive capabilities.

The findings of this study illustrate the effectiveness of the Security Actuator Ontology (SAO) in modeling a range of security actuators and their interconnections with infrastructure components. The ontology was validated by applying it to an example infrastructure in the case study, demonstrating its capability to support automated queries and inferences between security actuators and infrastructure elements. The ontology effectively bridged the gap between abstract defensive techniques and practical implementations, thereby enabling a higher-level view of security mechanisms across an IT infrastructure.

This work contributes to the field by addressing the lack of higher-level abstraction in current cybersecurity ontologies. The SAO provides a more flexible and extendable model for reasoning about defensive tactics and their real-world implementation, thereby facilitating better tactical decision-making in cybersecurity operations.

7.1 Future Work

Although the Security Actuator Ontology (SAO) has made notable advancements in modeling the defensive capabilities of IT infrastructures, there are still several avenues for future research. One promising avenue for future research is the integration of dynamic threat intelligence data into the ontology, which would allow it to adapt in real-time to emerging vulnerabilities and attack patterns. Another valuable extension would be the incorporation of a temporal component into the ontology. The current model is predicated on the assumption of a static infrastructure. However, in practice, the structure of an IT environment is subject to change over time, and the defenses in place at one point may differ from those in place at another. It is of great importance to be able to ascertain how an infrastructure was configured and defended at a specific point in time, as this would be crucial for post-incident analysis and historical auditing of security measures. This, however, is beyond the scope of the research presented in this thesis.

The ontology would be improved by the inclusion of a broader range of security actuators, particularly those applicable to emerging technologies such as cloud computing. Another potential avenue for extension would be to enhance the ontology's interoperability with existing cybersecurity frameworks, such as MITRE ATT&CK, in order to provide a comprehensive view of both offensive and defensive strategies. Furthermore, the application of machine learning techniques for automated reasoning within the ontology could facilitate the discovery of new insights into infrastructure protection and assist in the optimization of vulnerability management.

By pursuing these future research directions, the SAO could evolve into a more powerful tool, capable of providing time-sensitive insights into the evolution of IT infrastructures and more effectively supporting cybersecurity professionals in safeguarding modern infrastructures.

Bibliography

[1] F. Fransen, B. Gijsen, R. Kerkdijk, R. Wolthuis, and R. Montalto, "Innovative directions for automation in cyber security operations," TNO, White paper, Apr. 2023. [Online]. Available: https://publications.tno.nl/publication/34640813/suEUzW/TNO-2023-P11346.pdf

[2] ENISA, "ENISA Threat Landscape Report 2017," Jan. 2018. [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017

[3] NCTV and NCSC, "Cyber Security Assessment Netherlands 2023," NCTV, NCSC, publicatie, Jul. 2023. [Online]. Available: https://english.nctv.nl/topics/cyber-security-assessment-netherlands/news/2023/07/03/cyber-security-assessment-2023-expect-the-unexpected

[4] J. Oosterhof, "Automated ICT Infrastructure Modeling as a first step of Automated Cyber Security Analysis," Master's thesis, University of Groningen, Sep. 2019. [Online]. Available: https://fse.studenttheses.ub.rug.nl/20954/

[5] A. Ganesh, "Securing Infrastructure as Code deployments with Bills of Materials Approaches," Master's thesis, University of Groningen, 2023. [Online]. Available: https://fse.studenttheses.ub.rug.nl/31358/

[6] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," 2001.

[7] D. Preuveneers and W. Joosen, "An Ontology-Based Cybersecurity Framework for AI-Enabled Systems and Applications," Future Internet, vol. 16, no. 3, p. 69, Mar. 2024, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1999-5903/16/3/69

[8] V. Shandilya, C. Simmons, and S. Shiva, "Use of Attack Graphs in Security Systems," Journal of Computer Networks and Communications, vol. 2014, Oct. 2014.

[9] P. E. Kaloroumakis and M. J. Smith, "Toward a Knowledge Graph of Cybersecurity Countermeasures," The MITRE Corporation, Tech. Rep., Apr. 2020.

[10] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and Philosophy," MITRE, Technical Report 01ADM105-PI, Jul. 2018. [Online]. Available: https://apps.dtic.mil/sti/citations/AD1108016

[11] L. Hochstein and R. Moser, Ansible: Up and Running: Automating Configuration Management "O'Reilly Media, Inc.", Jul. 2017.

[12] J. Turnbull and J. McCune, Pro Puppet. Apress, Aug. 2011, google-Books-ID: p2h7FRdRjxIC.

[13] "RDF 1.1 XML Syntax." [Online]. Available: https://www.w3.org/TR/rdf-syntax-grammar/

[14] "OWL 2 Web Ontology Language Document Overview (Second Edition)." [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[15] M. Rahman, A. Pakstas, and F. Wang, "Towards Communications Network Modelling Ontology for Designers and Researchers," in International Conference on Intelligent Engineering Systems, 2006. INES '06, Jan. 2006, pp. 258–263.

[16] "Open Cybersecurity Schema Framework," Aug. 2023. [Online]. Available: https://schema.ocsf.io/

[17] J. Van Der Ham, F. Dijkstra, R. Lapacz, and J. Zurawski, "Network Markup Language Base Schema version 1," Open Grid Forum, Tech. Rep., 2013.

[18] M. Ghijsen, J. Van Der Ham, P. Grosso, C. Dumitru, H. Zhu, Z. Zhao, and C. De Laat, "A semantic-web approach for modeling computing infrastructures," Computers & Electrical Engineering, vol. 39, no. 8, pp. 2553–2565, Nov. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S004579061300222X

[19] "OWL 2 Web Ontology Language Profiles (Second Edition)," Dec. 2012. [Online]. Available: https://www.w3.org/TR/owl2-profiles/#OWL_2_RL

[20] G. Antoniou and F. v. Harmelen, "Web Ontology Language: OWL," in Handbook on Ontologies, S. Staab and R. Studer, Eds. Berlin, Heidelberg: Springer, 2009, pp. 91–110. [Online]. Available: https://doi.org/10.1007/978-3-540-92673-3_4

Appendix A

Security Actuator Ontology

A.1 Classes

```
<owl:Class rdf:about="sao:SecurityActuator">
    <rdfs:label>Security Actuator</rdfs:label>
</owl:Class>

<owl:Class rdf:about="sao:AccessControlActuator">
    <rdfs:subClassOf rdf:resource="sao:SecurityActuator"/>
    <rdfs:label>Access Control Actuator</rdfs:label>
</owl:Class>

<owl:Class rdf:about="sao:DataSecurityActuator">
    <rdfs:subClassOf rdf:resource="sao:SecurityActuator"/>
    <rdfs:label>Data Security Actuator</rdfs:label>
</owl:Class>

<owl:Class rdf:about="sao:EndpointSecurityActuator">
    <rdfs:subClassOf rdf:resource="sao:SecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom rdf:resource="d3fend:Host"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Endpoint Security Actuator</rdfs:label>
</owl:Class>

<owl:Class rdf:about="sao:ManagementMonitoringActuator">
    <rdfs:subClassOf rdf:resource="sao:SecurityActuator"/>
    <rdfs:label>Management and Monitoring Actuator</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="sao:NetworkSecurityActuator">
    <rdfs:subClassOf rdf:resource="sao:SecurityActuator"/>
    <rdfs:label>Network Security Actuator</rdfs:label>
</owl:Class>


<owl:Class rdf:about="d3fend:AuthenticationServer">
    <rdfs:subClassOf rdf:resource="d3fend:Server"/>
    <rdfs:subClassOf rdf:resource="sao:AccessControlActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:CredentialEviction"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:CredentialHardening"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:RestoreUserAccountAccess"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:UserBehaviorAnalysis"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:Host"/>
                        <rdf:Description rdf:about="d3fend:Network"/>
                        <rdf:Description rdf:about="d3fend:UserAccount"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Authentication Server</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="sao:EncryptionTool">
    <rdfs:subClassOf rdf:resource="sao:DataSecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom rdf:resource="d3fend:FileEncryption"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:File"/>
                        <rdf:Description rdf:about="d3fend:Host"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Encryption Tool</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="sao:BackupRecoverySystem">
    <rdfs:subClassOf rdf:resource="sao:DataSecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom rdf:resource="d3fend:RestoreObject"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:ConfigurationResource"/>
                        <rdf:Description rdf:about="d3fend:Credential"/>
                        <rdf:Description rdf:about="d3fend:Database"/>
                        <rdf:Description rdf:about="d3fend:File"/>
                        <rdf:Description rdf:about="d3fend:Software"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Backup and Recovery System</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="sao:AntiMalwareSoftware">
    <rdfs:subClassOf rdf:resource="sao:EndpointSecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom rdf:resource="d3fend:FileAnalysis"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:File"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Anti-Malware Software</rdfs:label>
</owl:Class>


<owl:Class rdf:about="sao:EndpointDetectionResponse">
    <rdfs:subClassOf rdf:resource="sao:EndpointSecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:FileAnalysis"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:NetworkTrafficAnalysis"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:PlatformMonitoring"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:ProcessAnalysis"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:UserBehaviorAnalysis"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Endpoint Detection and Response</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="sao:SIEMSystem">
    <rdfs:subClassOf rdf:resource="sao:ManagementMonitoringActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:OperatingSystemMonitoring"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:NetworkTrafficAnalysis"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:Host"/>
                        <rdf:Description rdf:about="d3fend:Network"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Security Information and Event Management (SIEM)
    ↪  System</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="d3fend:Firewall">
    <rdfs:subClassOf rdf:resource="d3fend:NetworkNode"/>
    <rdfs:subClassOf rdf:resource="sao:NetworkSecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom
            ↪  rdf:resource="d3fend:NetworkTrafficFiltering"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Firewall</rdfs:label>
    <skos:altLabel>Network Firewall</skos:altLabel>
</owl:Class>
```

```
<owl:Class rdf:about="sao:PatchManagementSystem">
    <rdfs:subClassOf rdf:resource="sao:ManagementMonitoringActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom rdf:resource="d3fend:SoftwareUpdate"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:Host"/>
                        <rdf:Description rdf:about="d3fend:Software"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Patch Management System</rdfs:label>
</owl:Class>

<owl:Class rdf:about="sao:NetworkFirewall">
    <rdfs:subClassOf rdf:resource="d3fend:Firewall"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom rdf:resource="d3fend:Network"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Network Firewall</rdfs:label>
</owl:Class>


<owl:Class rdf:about="d3fend:Host-basedFirewall">
    <rdfs:subClassOf rdf:resource="d3fend:SystemSoftware"/>
    <rdfs:subClassOf rdf:resource="d3fend:Firewall"/>
    <rdfs:label>Host-Based Firewall</rdfs:label>
</owl:Class>


<owl:Class rdf:about="d3fend:ApplicationLayerFirewall">
    <rdfs:subClassOf rdf:resource="d3fend:Firewall"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom rdf:resource="d3fend:Application"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Application Layer Firewall</rdfs:label>
    <skos:altLabel>Application Firewall</skos:altLabel>
</owl:Class>
```

```
<owl:Class rdf:about="sao:SecurityPolicyManager">
    <rdfs:subClassOf rdf:resource="sao:ManagementMonitoringActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:NetworkTrafficPolicyMapping"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:DomainTrustPolicy"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:StrongPasswordPolicy"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:Host"/>
                        <rdf:Description rdf:about="d3fend:Network"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Security Policy Manager</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="sao:ThreatIntelligencePlatform">
    <rdfs:subClassOf rdf:resource="sao:ManagementMonitoringActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:IdentifierActivityAnalysis"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:NetworkTrafficSignatureAnalysis"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="d3fend:Host"/>
                        <rdf:Description rdf:about="d3fend:Network"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Threat Intelligence Platform</rdfs:label>
</owl:Class>
```

```
<owl:Class rdf:about="d3fend:IntrusionDetectionSystem">
    <rdfs:subClassOf rdf:resource="d3fend:DigitalInformationBearer"/>
    <rdfs:subClassOf rdf:resource="sao:NetworkSecurityActuator"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:protects"/>
            <owl:someValuesFrom rdf:resource="d3fend:Network"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="sao:performs"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description
                        ↪  rdf:about="d3fend:OperatingSystemMonitoring"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:NetworkTrafficAnalysis"/>
                        <rdf:Description
                        ↪  rdf:about="d3fend:UserBehaviorAnalysis"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label>Intrusion Detection System</rdfs:label>
    <skos:altLabel>IDS</skos:altLabel>
</owl:Class>


<owl:Class rdf:about="d3fend:IntrusionPreventionSystem">
    <rdfs:subClassOf rdf:resource="d3fend:IntrusionDetectionSystem"/>
    <rdfs:label>Intrusion Prevention System</rdfs:label>
    <skos:altLabel>IDPS</skos:altLabel>
    <skos:altLabel>IPS</skos:altLabel>
    <skos:altLabel>Intrusion Detection and Prevention System</skos:altLabel>
</owl:Class>
```

A.2 Properties

```
<owl:ObjectProperty rdf:about="sao:performs">
    <rdfs:domain rdf:resource="sao:SecurityActuator"/>
    <rdfs:range rdf:resource="d3fend:DefensiveTechnique"/>
    <definition>x performs y: Security Actuator x performs Technique
    ↪  y</definition>
    <rdfs:label>performs</rdfs:label>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="sao:is-performed-by">
    <definition>x is performed y: Technique x is performed Security Actuator
    ↪  y</definition>
    <rdfs:label>is-performed-by</rdfs:label>
    <owl:inverseOf rdf:resource="sao:performs"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="sao:protects">
    <rdfs:domain rdf:resource="sao:SecurityActuator"/>
    <rdfs:range rdf:resource="d3fend:DigitalArtifact"/>
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <rdf:Description rdf:about="sao:protects"/>
        <rdf:Description rdf:about="d3fend:contains"/>
    </owl:propertyChainAxiom>
    <definition>x protects y: Security Actuator x protects DigitalArtifact
    ↪  y</definition>
    <rdfs:label>protects</rdfs:label>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="sao:is-protected-by">
    <owl:inverseOf rdf:resource="sao:protects"/>
    <definition>x is protected by y: DigitalArtifact x is protected by
    ↪  Security Actuator x</definition>
    <rdfs:label>is-protected-by</rdfs:label>
</owl:ObjectProperty>
```