POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

Master Degree in Computer Engineering

Master Degree Thesis

ICT System Ontology for Cybersecurity Governance

Author: Martina Trussoni

Advisor: Paolo Ernesto Prinetto
Co-Advisors: Fabio De Rosa, Nicolò Maunero

March, 2022

Abstract

The thesis work takes place in a new regulatory context: following the Decreto Legge No. 105 of 21 September 2019, in fact, the Italian government has established the *Perimetro Nazionale di Sicurezza Cibernetica* that identifies public administrations, public and private entities and operators on which the exercise of an essential function of the State depends. The new legislation requires them to comply with more stringent rules on the prevention and response to cyber attacks. On one hand, they are required to create and periodically update the list of networks, information systems and IT services they are responsible for, and on the other hand they are required to carry out analysis operations to identify potential criticalities and/or vulnerabilities (VAPT) in their ICT infrastructure. The purpose of the thesis is therefore to propose an ontology, a logical description of the components of a specific domain and the hierarchical relationships that bind them, that allows, on one hand a complete and detailed overview of the individual infrastructure. On the other hand the possibility, thanks to the knowledge base created, to provide the base and means to guide VAPT operations.

Attempts to unify knowledge are already in place, such as the *MITRE ATT&CK Framework*, which collects the most common tactics to perform attacks and the techniques used to carry them out, and the *MAEC language*, which contains information on malware in a standardised language; there exist also repositories containing all the known vulnerabilities, the *CVE*, or the weaknesses that may become vulnerabilities, the *CWE*.

Similarly, examples of ontologies in the cybersecurity field such as *UCO*, whose purpose is to create a cybersecurity knowledge base by combining information from different sources, and *IoTSec*, whose purpose is to provide a reference ontology in the IoT, are beginning to emerge.

The starting point of the thesis is the ontology developed by the *Agenzia per la Cibersicurezza Nazionale* that is being used to comply with the laws regarding the *Perimetro Nazionale di Sicurezza Cibernetica*. The ontology has been analysed and compared with the other above-mentioned related works in order to understand its strengths and weaknesses and how to improve it to make it as exhaustive as possible. A new ontology, therefore, has been created describing the infrastructure in a more complete way, considering all the information and details required to describe an IT system, from the end point configuration in terms of hardware, software and protocols adopted, to the network infrastructure and external services used.

This opens up the possibility to create a direct and automated interaction between the ontology and different sources of information about possible vulnerabilities (e.g. *CVE*) or attacks (e.g. *ATT&CK Framework*) in order to obtain a base to start and guide through the

process of VAPT operations. The design of this automated interaction has been proposed as part of this thesis work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The thesis work takes place in a new regulatory context: following the Decreto Legge No. 105 of 21 September 2019, in fact, the Italian government has established the *Perimetro Nazionale di Sicurezza Cibernetica* (*PSNC*) that identifies public administrations, public and private entities and operators on which the exercise of an essential function of the State depends. The Perimetro requires them to comply with more stringent rules on the prevention and response to cyber attacks.

On one hand, they are required to create and periodically update the list of networks, information systems and IT services they are responsible for, and on the other hand they are required to carry out analysis operations to identify potential critical issues and/or vulnerabilities (VAPT, Vulnerability Assessment and Penetration Testing) [18].

The purpose of the thesis is to propose an *ontology* that allows, in the first place a complete and detailed overview of the individual infrastructure, but also the possibility, leveraging on the knowledge base created, to discover potential vulnerabilities and, thereby, prevent possible cyber attacks. The ontology can therefore provide the base and means to guide VAPT operations.

The term *ontology* means a logical description of the components of a specific domain which includes technical features and properties of each element and the hierarchical relationships that bind them [26].

Currently a widespread standard is not available, but several ontologies and other sources of information are proposed and used. A major resource is the *Malware Attribute Enumeration and Characterization Language* created by the MITRE Corporation[1], which describes, using a structured language, the types of malware and their characteristics. This is an important source of information that provides a standard for reporting and describing malware. The MITRE Corporation also developed another important resource, the *MITRE ATT&CK Framework*[2]: it is a collection of the most common tactics to perform attacks and the techniques used to carry them out, each of them identified by a unique ID so that the knowledge can be easily exchanged. For what vulnerabilities are concerned,

[1] https://maecproject.github.io/

[2] https://attack.mitre.org/

the two main resources are the *Common Vulnerabilities and Exposures*[3] and the *Common Weaknesses Enumeration*[4] that are two community-based lists that contain respectively the most common hardware and software vulnerabilities and weaknesses.

All these information should be considered when performing VAPT operations on a computer system and the purpose of this thesis is therefore to create an ontology that allows to integrate security information and information about the infrastructure. An overview of the state of the art shows that there is not a real ontology standard for cybersecurity that includes all the necessary characteristics for a complete and exhaustive representation of all the information needed to describe an information system.

The starting point of this thesis is the ontology developed by the *Agenzia per la Cybersicurezza Nazionale* (*ACN*) that is being used as a reference to comply with the *PSNC* legislation. The ontology is studied and analysed and it is compared with other existing ontologies in the cybersecurity field and the result is a new improved ontology made as exhaustive as possible.

In particular, the two ontologies that are considered to perform this comparison are: the *Unified Cybersecurity Ontology*[33] and the *IoTSec Ontology*[25].

UCO is an extension of the Intrusion Detection System Ontology that aims to have a semantic description, and not just a syntactic description, of information systems; the purpose is creating a cybersecurity knowledge base by combining information from different sources.

IoTSec is a project whose purpose is to create a reference ontology in the Internet Of things field that contains both security and architecture information about the system under consideration.

The analysis of the strengths and weaknesses of the two ontologies shows that IoTSec best outlines the architectural details of the system, while the information about vulnerabilities and threats is static and does not come from external sources; on the contrary, UCO does not provide the possibility to define structural characteristics of the single infrastructure, but it allows interoperability with many external sources.

This last aspect is crucial: by taking this type of information into account and proposing a standardised and unified representation, a simple and effective interaction with external knowledge bases such as Common Vulnerabilities Exposure and Common Weaknesses Enumeration is possible, allowing, by means of specific tools, to automate the search for potential system vulnerabilities.

The new ontology is intended to be a national standard, first of all to allow the entities within the *Perimetro* to easily comply with the new rules for the creation and maintenance of the list of *Beni ICT* under their responsibility, and secondly to be the starting point for the creation of an automated knowledge centralisation system that can be used as a basis for VAPT operations.

The idea is to be able to make all the above knowledge resources interact with the ontology in the most automated way possible. The design of this automated interaction has been proposed as part of this thesis work.

[3] https://cve.mitre.org/

[4] https://cwe.mitre.org/

For what vulnerabilities are concerned, the tool proposed for the interaction is *cve-search*[5]: it offers an API to interact with the CVE and import the data to a local database. It is possible to convert these data in an OWL file in order to make it compatible and mergible with the ontology. For the attacks a similar approach was followed: the choice was to directly connect *CVE* to *ATT&CK*, since for the moment all the information about vulnerabilities come from *CVE*. In particular, the *Mapping ATT&CK to CVE for Impact*[6] project has been followed that connects to each CVE ID the IDs of the techniques that can be used to exploit the vulnerability. The mapping is contained in a CSV file that can be converted in a OWL file and merged with the other existing ontologies.

The thesis is developed in seven chapters: the first gives a general introduction about the work carried out and its context, the second defines the concept of ontology, how to build one and which languages to use, and an overview of *Protégé*, the tool used to build the ontology, is also made. The third chapter clarifies the context of the thesis: an overview is made of the Italian legislation on cybersecurity and the competent units are listed. The fourth chapter provides an outline of the state of the art of current ontologies and taxonomies regarding cybersecurity: standards such as the ATT&CK Framework, MAEC language, CVE and CWE databases are analyzed in detail; in addition, the two reference ontologies *UCO* and *IoTSec* are presented. The fifth chapter presents the ontology of the *ACN* which will provide the basis for the creation of the new ontology. In the sixth chapter the work carried out in all its phases is presented: the rewriting of the *ACN* ontology using *Protégé*, the modifications made to make it more complete and the addition of the concepts of vulnerabilities and attacks; finally, the presentation of the automatic interaction project between the different sources of information is outlined. Finally, the seventh chapter draws conclusions from the work done and contains possible future developments and improvements.

[5] https://cve-search.github.io/cve-search/

[6] https://github.com/center-for-threat-informed-defense/attack_to_cve

Chapter 2

Ontologies

2.1 Ontology

The word *Ontology* comes from the greek word *òntos*, that is the present participle of the verb *to be*, and the word *logos*, that means *logical discourse*; hence ontology literally means *discourse on being*; it is, in fact, a branch of philosophy that studies being and its categories.

This term has become commonly used in the field of computer and information science when talking about knowledge representation. In this sense, an ontology is a formal description of a set of concepts belonging to the same domain. These concepts are divided into *classes* and for each class the hierarchical relations linking it to the others (superclass-subclass) are established. The features and attributes of the different classes are called *properties*, the objects of the class are called *individuals* or *instances* and for each class the range of values that its *instances* can take is defined [27]. In this way, a knowledge base of the domain of interest is created.

All the assumptions within an ontology are represented explicitly and this allows the ontology to be interrogated with automatic reasoning tools in order to obtain further information to enrich the knowledge base; it also allows to check the consistency of data and their properties.

Another important aspect of ontologies is that they create a common language and knowledge that can be shared, integrated and reused. In the Cybersecurity field, in particular, many different sources produce data and information using different standards and languages. Having a deeper integration of data could allow to change the approach that most systems have towards an attack from reactive to proactive or even predictive [33].

2.1.1 How to build an Ontology

The first step in building an Ontology[27] is to define its *domain*: this can be done by delineating some questions, called *competency questions*, the ontology should be able to answer to. Once the domain is established, it is important to research and see if ontologies already exist for it. In fact, it is often more advantageous to reuse and expand existing resources than to start from scratch. Using as an example the creation of a pizza ontology, some competency questions could be:

Do pizzas come in different sizes?

What are the vegetarian pizzas?

Is Margherita a white or red pizza?

What are the possible toppings?

By collecting all the answers to these questions, the list of terms forming part of the domain is created so that it is as exhaustive as possible and the properties that these terms have are listed. These terms need then to be divided in classes, using one of the following approaches:

- **Top-down**: defining the general concepts first and then break down in further sub-classes;
- **Bottom-up**: defining specific concepts first and then group them in super-classes;
- **Combination**: both top-down and bottom-up approaches are used together.

The Combination approach is usually the easiest, but there are no rules that say which one is better to use, mostly it depends on the situation and on the preferences of the developers.

Whichever method is used, starting from the list of terms created earlier, it is possible to define as classes those terms that describe and include other terms and these classes are developed following the hierarchy into sub-classes and super-classes.

In the pizza ontology, the super-classes are: *Pizza*, *PizzaTopping* and *PizzaBase*; some of the sub-classes of PizzaTopping could be *CheeseTopping, VegetableTopping, MeatTopping* and again some sub-classes of CheeseTopping could be *MozzarellaTopping* and *Parmesan-Topping*.

Then, all the properties of the terms must be added in order to obtain a complete knowledge base that allows answering to all competency questions. Each property is associated to a class and all the sub-classes inherit it. Properties can be *intrinsic*, when a term has it of itself, or *extrinsic*, when it depends on the relationships of a term with the others.

In figure 2.1, there is an example of property for the pizza ontology:

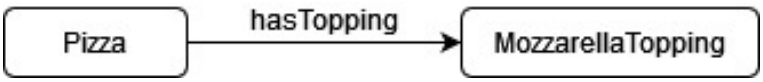

Figure 2.1. Property of the class Pizza

In addition to specifying the properties, their features need also to be considered. Some of the most common are:

- **Cardinality**: defines how many values a property can have; in some cases it can only be single or multiple, in other it is more precise and can be also minimum or maximum (where minimum/maximum N means a property can have minimum/maximum N values). If the maximum is 0, the property can not have any value for a particular subclass;

- **Value-type**: defines the type of values that can be used (e.g string, number, boolean and so on);
- **Domain and Range**: the domain consists of one or more classes that are the subject of a certain property, whereas the range is composed of the classes that are the object of the property.

For example, a pizza has at least one *PizzaBase*, but can have N *PizzaToppings*.

The last step is, then, create the instances of the classes and define their properties with real values.

2.1.2 Semantic Web Standards

The Semantic Web is an extension of the *World Wide Web* whose purpose is to make data on the internet machine-readable and not only human-readable [1]; it relies on the standards set by the *World Wide Web Consortium*[1] (W3C).

For what ontologies are concerned, the Semantic Web Standards proposed by the W3C are the *Resource Description Framework* (RDF) and the *Ontology Web Language* (OWL):

- **Resource Description Framework**: it is a framework for representing information in the Web[22] that exploits *eXtensible Markup Language* (XML) as common language for exchanging and processing data. The core components of the RDF are: the *RDF Model and Syntax* and the *RDF Schema*.

  The basic unit of the RDF Model is the *resource* that can represent everything that is on the Internet; each resource is identified by a unique URI. The other two key elements are the *property* - a feature, an attribute or a relation that describes a resource; each property has a specific meaning, defines its permitted values, defines the types of resources it can describe and its relationship with other properties [23] - and the *values*. Together they create the *statement*. The elements of the statement can be seen as the *subject* (the resource), the *predicate* (the property) and the *object* (the value) of the RDF syntax and together they create a triple. A set of triples forms a RDF oriented graph. In the graph, the *nodes* are the subjects and the objects, whereas the *arc* is the predicate that binds them and the direction of the arc points towards the object [22].

- **Ontology Web Language**: it is a modeling language that is used to express metadata such as concepts, relationships between those concepts and their features in a way that is both machine-readable (for interoperability) and human-readable. Depending on the uses and needs of the application, there are three possible choices of sublanguages that have a growing degree of expressive power, which are *OWL Lite* - classification hierarchy and simple constraint features -, *OWL DL* - maximum expressiveness without losing computational completeness and decidability of the reasoning systems - and *OWL Full* - maximum expressiveness and the syntactic freedom of RDF with no computational guarantees[30].

[1] https://www.w3.org

The first important difference between these two standards is that OWL adds more *vocabulary* for describing properties and classes: for example, it introduces the possibility to express cardinality (e.g. "exactly one"), or more specific relations between classes (e.g. disjointness) and many others [2]. The other difference is that RDF does not perform any consistency check about the logic of the statements so everything can be said, whereas OWL imposes severe constraints about *logic consistency*. For example, in RDF an object can be both an instance and a class; in OWL this is not possible. The third difference is that OWL has a rich variety of annotations (e.g OWL:import, OWL:backCompatibleWith) that make it very easy to link together different ontologies.

2.2 Protégé

The most comprehensive tool for building and modifying an ontology is *Protégé*: it is an OWL ontology development environment created by the *Stanford Center for Biomedical Informatics Research* at the Stanford University School of Medicine. It is a free, open-source platform[2] that provides a suite of tools to construct domain models and knowledge-based applications with ontologies [3]. It supports development of plug-ins that permit to build both simple and complex ontology-based applications. It can be used either online with the Web-Protégé version or locally by downloading the Protégé Desktop version.

It has two different approaches for modeling the ontologies: *Protégé-Frames*, that follows the OKBC (Open Knowledge Base Connectivity) Protocol and creates models frame-based, and *Protégé-OWL*, that supports the Semantic Web Languages, in particular OWL. For the purpose of this thesis, only the Protégé-OWL approach will be considered.

2.2.1 Protégé-OWL

[29] Protégé OWL allows users to edit ontologies in the Ontology Web Language (OWL) and to use description logic classifiers to maintain consistency of their ontologies. The most important features are:

- **Graphical Interface and API**: Protégé OWL is built upon the Protégé frame-based knowledge model and uses the Protégé GUI, in Figure 2.2, for editing classes, properties and instances;
- **Graphical editor for OWL expressions**: Protégé OWL provides a convenient expressions editor that allows users to quickly assemble expressions. It also uses an object-oriented graphical display of primitive and defined classes, where primitive classes mean classes that have only necessary conditions, while defined classes have necessary and sufficient conditions. The editor supports drag/drop and copy/paste;
- **Wizards to streamline complex tasks**: there are wizards to support common ontology-engineering patterns, such as creating groups of classes, making a set of classes disjoint, creating a matrix of properties in order to set many property values, and creating value partitions;

[2] https://protege.stanford.edu/

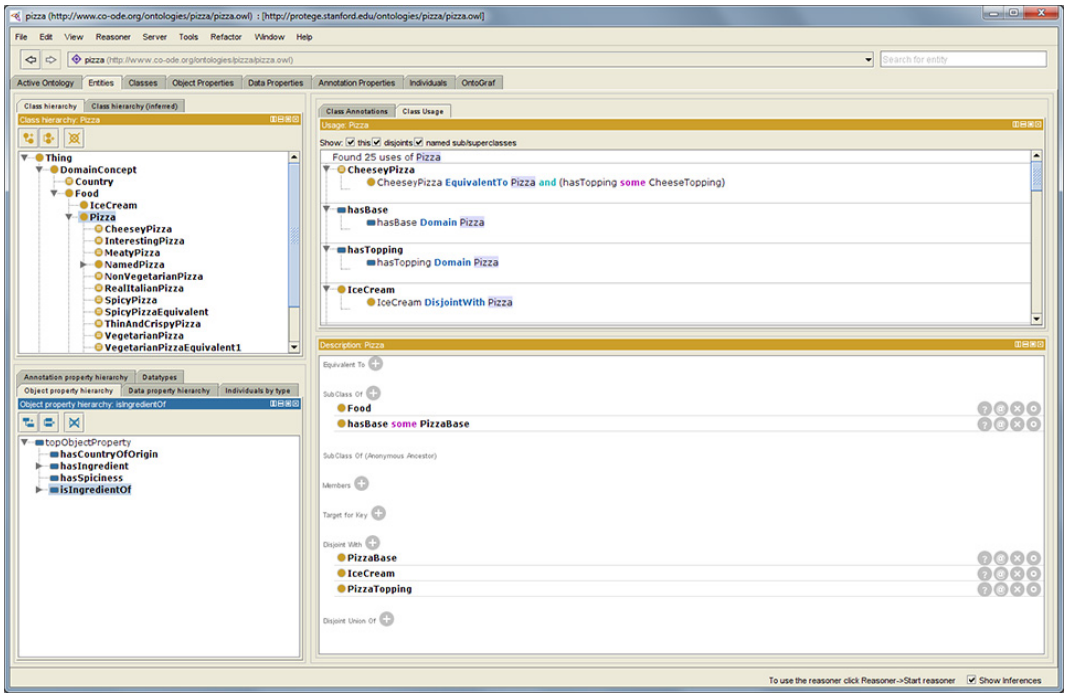

Figure 2.2. Protégé Desktop GUI

- **Direct access to reasoners**: it provides direct access to high-performance reasoners such as *HermiT* that determines if the ontology is consistent, identifies subsumption relationships between classes, and much more [4].

It also provides a number of widgets that allow ontologies to be visualised graphically, such as:

- **OntoGraf**: as seen in Figure 2.3, it gives the possibility to navigate through the relations in the ontology and to interact with them. It also allows to filter node types and relationships so that it is possible to create customized views [5];

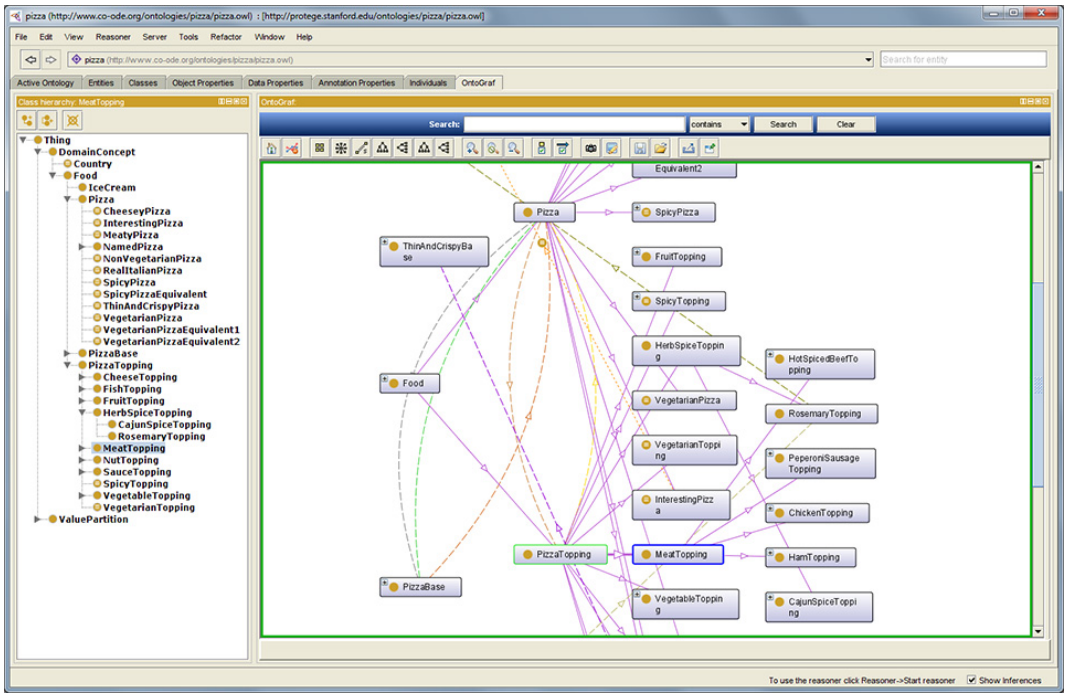

Figure 2.3. Protégé Desktop OntoGraf Widget

- **OWLViz**: it enables class hierarchies in an OWL ontology to be viewed and incrementally navigated, allowing comparison of the asserted class hierarchy and the inferred class hierarchy [6].

2.3 Ontology Evaluation

Once an ontology is complete, it is important to evaluate it. As the use of ontologies has become more widespread, several evaluation methods and metrics have started to spread in recent years, which can be grouped into four macro-categories, namely *gold standard-based, corpus-based, task-based* and *criteria-based* approaches [28]. This section will provide an overview of these approaches to understand their advantages and disadvantages.

In general, there are two main aspects that must be considered when performing the evaluation that are *quality* and *correctness*; they cover different criteria:

- **Accuracy**: it states if the definitions, descriptions of classes, properties, and individuals in an ontology are correct;
- **Completeness**: it measures if the domain of interest is dealt with comprehensively in the ontology;
- **Conciseness**: it concerns the relevance of the elements included in the ontology with regards to the domain to be covered;

- **Adaptability**: it measures how far the ontology anticipates its uses. An ontology should offer the conceptual foundation for a range of anticipated tasks;
- **Clarity**: it measures how effectively the ontology communicates the intended meaning of the defined terms;
- **Computational**: it measures the ability of the used tools to work with the ontology, in particular the speed that reasoners need to fulfil the required tasks;
- **Consistency**: it checks that the ontology does not include or allow for any contradictions.

Based on the criteria that are assessed, four methods of ontology evaluation can be distinguished:

- **Gold Standard-based**: this type of approach evaluates an ontology comparing it with another previously created ontology that is called the *gold standard*. This reference ontology represents the ideal result that should be obtained, but it is very difficult to have one created according to the same rules as the ontology under examination. In some cases, taxonomies created by experts in the domain of interest are used instead of ontologies because they are easier to obtain. This method can evaluate accuracy, correctness and completeness.
- **Corpus-based**: this approach also uses comparison, but this time not with an existing ontology but with a corpus, i.e., a text that covers the domain in question in a comprehensive and exhaustive manner. The comparison can be performed in many ways, the simplest one is extracting the terms from the text and counting how many of them are present in the ontology. The criteria evaluated by this methodology are the same as the ones evaluated by the golden-standard method, accuracy, correctness and completeness; the criticalities are also very similar, even if finding a complete corpus is easier than finding an existing ontology.
- **Task-based**: this type of approach focuses on how well the ontology performs in accomplishing the task for which it was created without evaluating its structural aspects. The criteria that can be evaluated are adaptability, by applying the ontology to several tasks and evaluate its performance, and consistency; additionally, it can compute the speed to accomplish the task.
- **Criteria based**: this last approach evaluates the extent to which an ontology meets certain criteria; it can be further broken down in two categories:
  - **Structure-based**: these are the most objective measures concerning the structure of the ontology, so it is very easy to automate the evaluation; for example, the number of nodes, the relational density of nodes, the depth, etc. can be calculated and on the basis of these measures conclusions can be drawn on the correctness of the ontology.
  - **Complex and expert based**: these are metrics developed specifically to evaluate different features of the ontology together and create new and complex methodologies; for example, *OntoClean* relies on philosophical notions such as essence to evaluate properties and relationships of the ontology.

In general, criteria-based approaches can best evaluate the clarity of an ontology; using more complex approaches it is possible to easily evaluate also consistency by analyzing the axioms of the ontology.

Starting from this division of methods, it is then possible to combine them to create a customised approach according to the type of ontology and its intended use. In any case, it is important to have a formalisation of the evaluation of the ontology to make the knowledge base created more scientific and reliable.

Chapter 3

The Italian Cybersecurity Legislation

Having defined what an ontology is and how it is constructed, it is important at this point to outline the context in which the thesis fits.

In recent years, the field of cybersecurity in Italy has undergone considerable changes from a regulatory and legislative point of view. The development of legislation and the corresponding creation and reorganisation of competent authorities is therefore analysed.

3.1 The NIS Directive

With the Decree Law of 18 May 2018, n.65, published on the *Gazzetta Ufficiale* n.132 of 9 June 2018, Italy implemented the *NIS Directive*, an EU-wise cybersecurity legislation[1], to define the measures necessary to achieve a high level of security of networks and information systems. The Order applies to the *Operatori di Servizi Essenziali (OSE)* and the *Fornitori di Servizi Digitali (FSD)*.

- **Operatori di Servizi Essenziali**: those entities, public or private, that provide essential services to society and economy in the fields of health, energy, transport, banking, financial market infrastructure, drinking water supply and distribution, and digital infrastructure;
- **Fornitori di Servizi Digitali**: legal persons providing e-commerce, cloud computing or search engine services, with a principal place of business, registered office or designated representative on the national territory. The obligations for FSDs do not apply to companies which European legislation defines 'small' and 'micro' companies, i.e. those which have fewer than 50 employees and an annual turnover or annual balance sheet not exceeding 10 million Euros.

According to the Decree Law, both the OSEs and the FSDs are required to adopt technical and organizational appropriate and proportionate measures to prevent and minimize

[1] https://www.enisa.europa.eu/topics/nis-directive

the impact of incidents on the security of networks and information systems, in order to ensure continuity of service. In addition, they have an obligation to notify, without undue delay, incidents that have a significant impact on service continuity and service provision respectively, to the Italian *Computer Security Incident Response Team* (CSIRT), also informing the relevant competent NIS Authority.

The tasks of the CSIRT are defined by Decree Law No. 65 of 18 May 2018 and the Decree of the President of the Council of Ministers of 8 August 2019 Article 4. They include [7]:

- defining procedures for the prevention and management of computer incidents;
- receiving incident notifications, informing the DIS, as a single point of contact and for the activities of prevention and preparation of possible crisis situations and activation of alert procedures entrusted to the Cyber Security Unit;
- providing the notifying party with information that may facilitate the effective management of the event;
- informing other EU Member States that may be affected by the incident, protecting the safety and commercial interests of the OSE or FSD as well as the confidentiality of the information provided;
- ensuring cooperation in the CSIRT network, by identifying forms of operational cooperation, exchange of information and sharing of best practices.

3.2 The *Perimetro di Sicurezza Nazionale Cibernetica*

The establishment of the *Perimetro di Sicurezza Nazionale Cibernetica* takes place within the framework of the NIS Directive.

The *PSNC* was established pursuant to Article 1, comma 1, of Decree Law No. 105 of 21 September 2019, converted with amendments by Law No. 133 of 18 November 2019 (in the *Gazzetta Ufficiale* No. 272 of 20/11/2019) in order to ensure a high level of security of the networks, information systems and IT services of public administrations, public and private entities and operators having an office in the national territory, on which the exercise of essential State functions depends.

The Decree clarifies that:

- An entity exercises an essential function of the State, henceforth an essential function, wherever the legal system assigns it tasks aimed at ensuring the continuity of the action of the Government and the Constitutional Authorities, internal and external security and the defence of the State, international relations, security and public order, the administration of justice, the functionality of the economic and financial systems and transport;
- An entity, whether public or private, provides an essential service for the maintenance of civil, social or economic activities that are fundamental to the interests of the State, hereinafter referred to as an essential service, where it carries out activities

instrumental to the exercise of essential State functions; activities necessary for the exercise and enjoyment of fundamental rights; activities necessary for the continuity of supplies and the efficiency of infrastructures and logistics; research activities and activities relating to productive realities in the field of high technology and in any other sector, where they have economic and social significance, also for the purposes of guaranteeing national strategic autonomy, competitiveness and the development of the national economic system.

The above-mentioned entities and operators are found in different sectors: internal affairs, defence, space and aerospace energy, telecommunications, economy and finance, transport, digital services, critical technologies and social security/labour.

For each sector, the competent Minister must identify entities whose exercise of essential function or whose provision of essential service depends on networks, information systems and computer services whose malfunctioning, interruption, even partial, or improper use may result in a threat to national security. The result is a list of entities falling within the *PSNC*.

All of these subjects, according to the Decree, are required to prepare and update, at least once a year, a list of the networks, information systems and IT services for which they are responsible, including their architecture and components. They must, then, identify the ICT assets necessary to perform the essential function or service, in order to assess the impact of an incident on the ICT asset, in terms of its operability and the compromise of data availability, integrity or confidentiality (CIA triad) and assess dependencies with other networks, information systems, IT services or physical infrastructures pertaining to other entities. Finally, entities must identify the ICT assets that, in the event of an incident, would cause total disruption of the essential function or service.

The list compiled is transmitted to the to the Presidency of the Council of Ministers and to the Ministry of Economic Development and they will transmit it to the *Dipartimento delle Informazioni per la Sicurezza* for the prevention, preparation and management of cyber crises entrusted to the Cyber Security Unit, as well as to the the organ of the Ministry of the Interior for security. The transmission of ICT asset lists takes place via a digital platform set up at the DIS.

The Decree of the President of the Council of Ministers (DPCM) of 15 June 2021 identifies the categories of ICT assets, systems and services intended for use in the *PSNC*. Those categories are based on the technical criteria of the art. 13 of the Decree of the President of the Republic, no. 54, of the 5 February 2021, identified based on the execution of these functions:

- Switching or intrusion protection and detection of cyber threats in a network, including the application of security policies;
- Command, control and implementation in an industrial control network;
- Monitoring and configuration control of an electronic communication;
- Network security with respect to the availability, authenticity, integrity or confidentiality of services offered or data stored, transmitted or processed;
- Authentication and allocation of resources of an electronic communication network;

- Implementation of an IT service by means of the configuration of an existing software program or the development, partial or total, of a new software program, constituting the application relevant part to the provision of the IT service itself.

Based on these criteria, the implementing Decree established the categories referring to the hardware and software components that enable the implementation of these functions [24]. The following are the four identified categories:

1. Hardware and software components performing telecommunications network functions and services (access, transport, switching);

2. Hardware and software components performing security functions for telecommunications networks and the data they process;

3. Hardware and software components for data acquisition, monitoring, supervision, control, implementation and automation of telecommunications networks and industrial and infrastructure systems;

4. Software applications for the implementation of security mechanisms.

The details of each category are specified in tables 3.1 and 3.2.

Following Article 5, comma 1, of Decree Law No. 82 of 14 June 2021, converted with amendments by Law No. 109 of 4 August 2021, the *Agenzia per la Cybersicurezza Nazionale* is established for the protection of national interests in the field of cybersecurity, with headquarters in Rome.

The Agency shall have legal personality under public law and shall have regulatory administrative, patrimonial, organisational, accounting and financial autonomy within the limits of the provisions of the Decree.

The Agency becomes the national authority for cybersecurity and the single contact point for network and information system security, for the purposes of the NIS Decree Law, so it assumes all the responsibilities the DIS had with respect to the implementation of the *PSNC*.

The *ACN* has created an ontology following the *PSNC* laws to describe the subject infrastructure which will be used in this thesis as a starting point to create a more comprehensive one that also includes security information needed for VAPT operations.

Categoria	Bene, sistema, servizio
Componenti hardware e software che svolgono funzionalità e servizi di rete di telecomunicazione (accesso, trasporto, commutazione)	- Router - Switch - Repeater - Bilanciatori di carico - Traffic shaper - Proxy - Ponte radio - Access Network per reti radiomobili 2G, 3G, 4G, 5G - Gateway Wifi - Network Function Virtualization(NFV): o vSwitch o vRouter o Application Function (5G) - Optical transmission board - Multiservice Provisioning Platform (MSPP) - Automotive ECU switch (Ethenret, CAN, LIN) - IoT Edge Gateway
Componenti hardware e software che svolgono funzionalità per la sicurezza di reti di telecomunicazione e dei dati da esse trattati	- Firewall - Security Gateway - Hardware Security Module (HSM) - Intrusion Detection System (IDS) - Intrusion Prevention System (IPS) - Network Function Virtualization (NFV) o Authentication Server Function (5G) o Whitelisting dei processi - Virtual Private Network (VPN) - Trusted Platform Module

Table 3.1. Allegato DPCM 15 June 2021 part 1

Categoria	Bene, sistema, servizio
Componenti hardware e software per acquisizione dati, monitoraggio, supervisione controllo, attuazione e automazione di reti di telecomunicazione e sistemi industriali e infrastrutturali	- Sistemi SCADA (Supervisory Control And Data Acquisition) - Manufacturing Execution Systems (MES) - Software Defined Network (SDN) Controller - Sistemi Artificial Intelligence (AI) e Machine Learning (ML) per gestione reti/sistemi - 5G Mobile Edge Computing (MEC) - NFV: o Network Slice Selection Function (5G) o Application Function (5G) o Policy Control Function (5G) o Unified Data Management (5G) o Session Management Function (5G) - Management and Orchestration (MANO) - IoT orchestrator
Applicativi software per l'implementazione di meccanismi di sicurezza	- Applicazioni informatiche per la sicurezza o Public Key Infrastructure (PKI) o Single Sign-On (SSO) o Controllo Accessi - Moduli software che implementano Web Service mediante API, per protocolli di comunicazione

Table 3.2. Allegato DPCM 15 June 2021 part 2

Chapter 4

State of the Art

In the Cybersecurity field, the data that are used to analyse threats and prevent attacks come from many different sources, both external and internal to the systems being monitored. Moreover, they are not necessarily structured, and there is no common language or vocabulary that allows interoperability between different tools.

A common knowledge base of cybersecurity information would permit to have a very large amount of data and a more predictive approach when defending a system. Many organisations are working to create this knowledge base using different methods.

4.1 Standards

The MITRE Coorporation has developed both the *MITRE ATT&CK Framework* and the *MAEC language*:

- **MITRE ATT&CK Framework**: MITRE's Adversarial Tactics, Techniques, and Common Knowledge is a curated knowledge base and model for cyber adversary behavior, reflecting the various phases of an adversary's attack lifecycle and the platforms they are known to target[31]. The basic elements of ATT&CK are *techniques* and *tactics*. The relationship between them is represented in the *ATT&CK Matrix*, in figure 4.1: techniques represent actions that adversaries can perform to accomplish objectives and those objectives are the tactics. Every technique fall under a tactic category and in this way they create a Matrix.

  To complete the ATT&CK Model there are other two key concepts that are part of the Framework and that are *groups* -sets of related intrusion activity that are tracked by a common name - and *software* - an instantiation of a technique. Software is broken out into three high-level categories: tools, utilities, and malware [8].

Figure 4.1. A snippet of the ATT&CK Matrix [8]

There are three main reasons why ATT&CK has been very important and useful and it's continuing to grow [17]:

  - Adversary's perspective: it keeps the adversary's perspective; this makes it easier to understand attacks and potential countermeasures;
  - Empirical use: the knowledge base for the attacks comes from real incidents that have actually happened and have been reported and not from theoretical attacks that could happen;
  - Level of abstraction: it is a mid-level adversary model that can link together low level concepts (e.g. malware and vulnerabilities repositories and databases) and high-level models (e.g. Microsoft STRIDE - a model for identifying computer security threats and divide them in very high level categories).

- **MAEC**: the Malware Attribute Enumeration and Characterization wants to provide a common language to share structured information about Malware [9]; its core components, in figure 4.2, are the *Schema* (grammar), the *Enumerations* (vocabulary) and the *Cluster* (output format). The vocabulary is composed by malware attributes and the grammar defines the relationship between them and their structure [21].

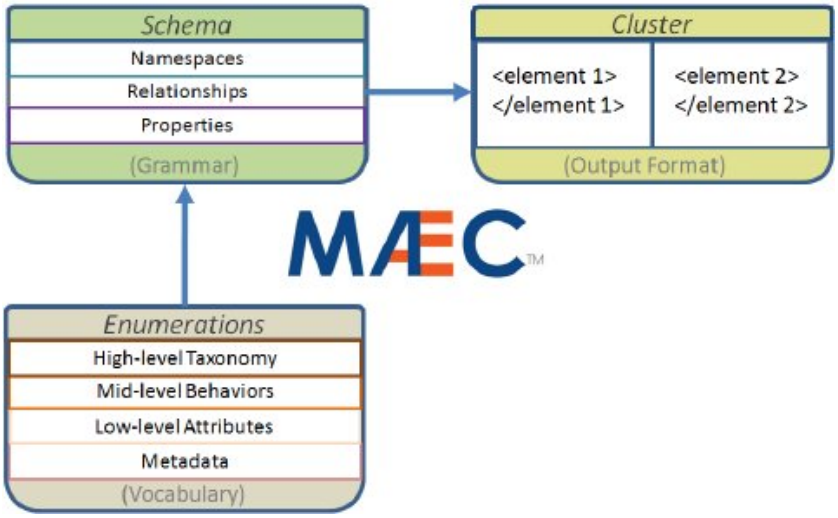

Figure 4.2. Maec Overview [21]

  - The Enumerations of attributes are divided into three levels, each of which in turn is divided in sub-levels: this allows to have a low-level description of the activities performed by the malware, but also to regroup them in high-level categories and have a complete knowledge of the malware behaviour;
  - The Schema defines a syntax for the vocabulary and describes the relationships between the attributes that are not already made explicit by the three-levels model;

  - The Cluster represents a standard output format that is used to store and then share all the knowledge about the malware. Initially, MAEC will support XML as data interchange format, but the idea is to support in the future also RDF and JSON.

The MITRE Corporations operates also other two repositories that are widely known and used when performing Vulnerability Assessment and Management: the *Common Vulnerabilities and Exposures* and the *Common Weaknesses Enumeration*.

- **CVE**: it is an international community-based repository of publicly known cybersecurity vulnerabilities; these vulnerabilities are stored in the *CVE list* and they are uniquely identified by a *CVE identifier*. New vulnerabilities can be submitted by anyone by forwarding the request to a *CVE Numbering Authority* or directly to the MITRE Corporation that is the root CNA. When the request is received, the CVE ID is assigned and its status is canditate, if the request is accepted the status becomes entry. Other than the CVE ID and the description of a vulnerability, another important information that can be retrieved from the CVE is the severity of the vulnerability computed using the *Common Vulnerability Scoring System*.
  - **CVSS**: it is an open framework for communicating the characteristics and severity of software vulnerabilities [10]. It is composed of three metrics: Base, computes the value based on the intrinsic characteristics of the vulnerability and assumes the worst case scenario, Temporal, considers also factors that may change over time, and Environmental, adjusts the score based on information about the specific environment. The Base Metric is the most common and its value range is from 0, less severe, to 10, more severe.
- **CWE**: it is an international community-developed list of common software and hardware weaknesses type, where weakness means bug, flaw, software or hardware implementation error that can lead to a vulnerability if it is not solved [11]. Each weakness is associated with a unique ID and with an abstraction level; in this way we have a tree representation of weaknesses where the layers are [11]:
  - Class: most abstract, language and technology independent;
  - Base: more specific than class, it is a child of a class weakness;
  - Variant: most specific type of weakness, it is linked to a certain type of product and it involves a specific language or technology.

  In each entry there are many information: description, relationships with other CWEs, applicable platforms, consequences, some demonstrative examples, potential mitigations, detection methods and a list of related CVEs.

4.2 Existing Ontologies

The first two analysed standards (*ATT&CK* and *MAEC*) are very important because they standardise the way information on attacks and malware is communicated and stored, but they are not intended to be a complete taxonomy on computer security. The other two

standards (*CVE* and *CWE*), on the other hand, are crucial when performing Vulnerability Assessment and Management of a system because they are a valuable and complete source of information for detecting possible vulnerabilities and weaknesses. However, these information alone are not sufficient to perform a complete security assessment of a system; the next step is to put this forming detailed data into a broader context involving all aspects of cybersecurity. This can be done through the use of ontologies.

As mentioned above, one important aspect of ontologies is that they can, and must, be reused: it is always recommended to start from existing ontologies and merge them or add to them some new information. For the purpose of the thesis, two existing cybersecurity ontologies were studied and analysed to be improved and made suitable to become a single ontology to be used as a standard: the *Unified Cybersecurity Ontology* (UCO) and the *IoTSec Ontology*.

4.2.1 Unified Cybersecurity Ontology

The *Unified Cybersecurity Ontology* [33] is an extension to Intrusion Detection System ontology that wants to provide a common structure to describe the cybersecurity domain. The main idea of this project is to try to change the approach of the cybersecurity standards from a syntactic representation to a more semantic representation. To do so, a large number of existing standards and ontologies have been studied and reviewed and the most common ones have been selected to be integrated with UCO. In particular, UCO is intended to be the semantic version of *STIX* and it includes references to many external standards, such as *CVE, CWE, CVSS, CAPEC, CYBOX, KillChain* and *STUCCO*.

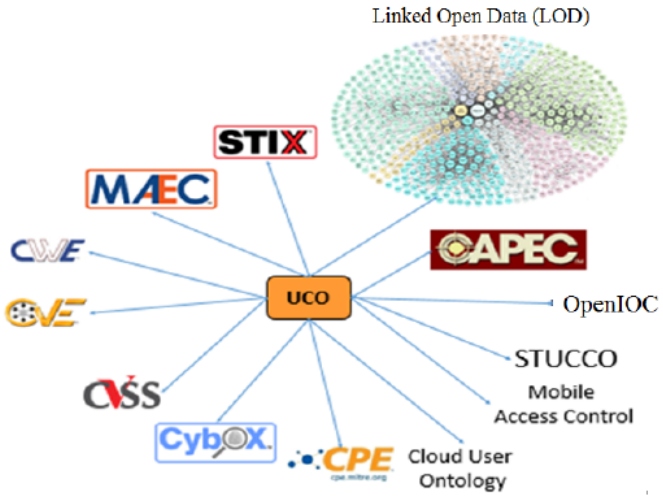

Figure 4.3. UCO structure [33]

As shown in figure 4.3, another important external reference is to the *Linked Open Data*: a dataset of general world knowledge in the Linked Data format used in this context

to support different use cases.

The UCO is made of classes, terms and rules. In figure 4.4 all the classes of UCO are shown, the top-level classes are:

- **Attack**: it describes cyber attacks;
- **Attacker**: it describes the adversary;
- **AttackPattern**: it describes from the attacker point of view the most common ways of implementing an attack;
- **Consequences**: it describes the possible effects of an attack;
- **Exploit**: the description of an individual exploit;
- **Means**: it describes the ways in which an attack can be performed.

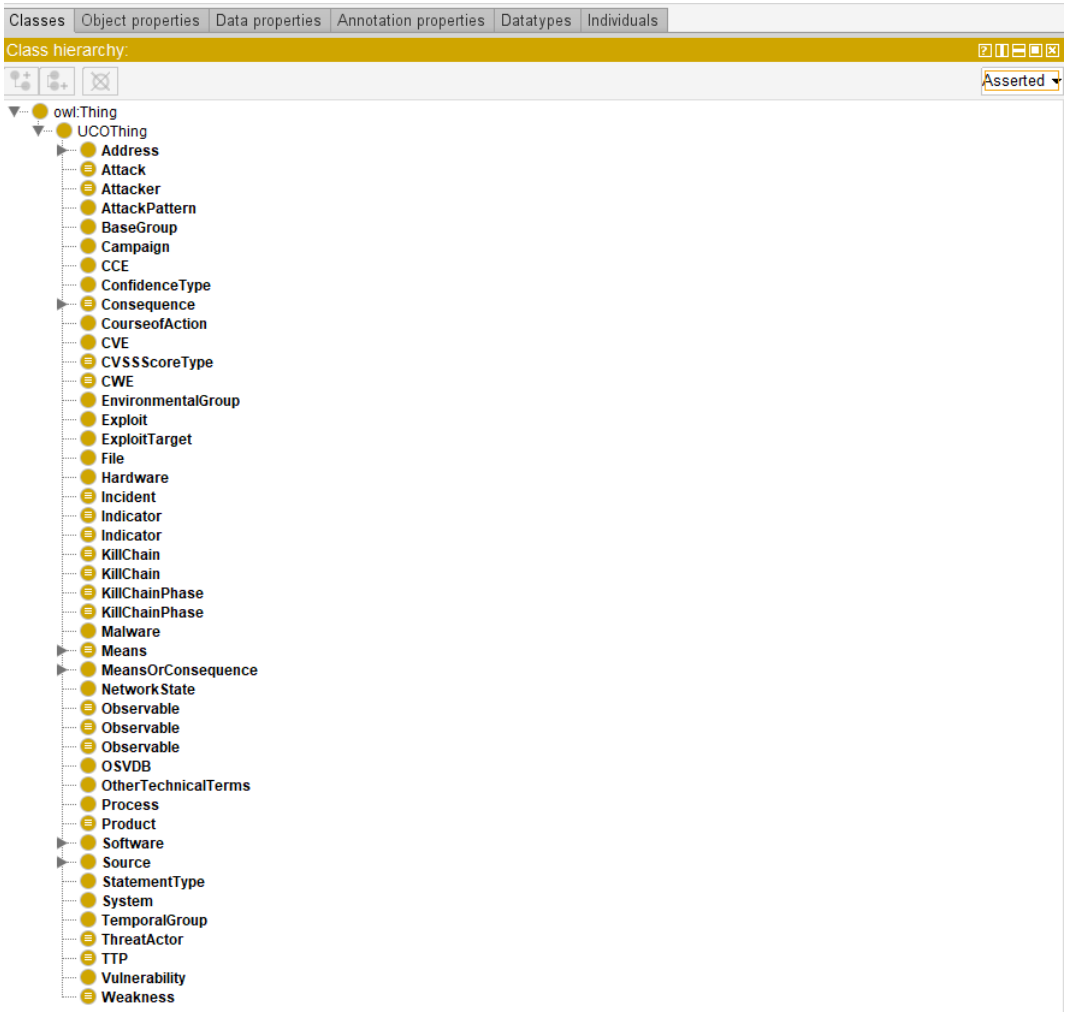

Figure 4.4. UCO classes overview

Then there are other classes resulting from *STIX*[1]. It is a structured language for describing cyber threat information that consists in 9 key entities and the relationships between them [16]:

- **Observables**: describe stateful properties or measurable events in cyber;
- **Indicators**: combine observable patterns with contextual information to identify a possible behavior;
- **Incidents**: describe instances of specific adversary actions;

[1] https://stixproject.github.io/about/

- **Adversary Tactics, Techniques, and Procedures**: describe attack patterns, malware, exploits, kill chains, tools, infrastructure, victim targeting, and other methods used by the adversary; in UCO the correspond is the *TTP* class;
- **Exploit Targets**: describe vulnerabilities, weaknesses, or configurations that might be exploited;
- **Courses of Action**: describe actions to address a threat that may be preventative or reactive;
- **Campaigns**: describe sets of incidents and/or TTPs pursuing a shared intent;
- **Threat Actors**: describe identification and/or characterization of the adversary;
- **Reports**: collect related STIX content and give them shared context; this class is not present in UCO.

So as far as attacks and threats are concerned, the information are written and shared using a structured language and this is a great advantage for interoperability.

The same applies for vulnerabilities: the figure 4.4 shows the presence of the classes that allow to query external databases and store information, such as *CVE, CWE, CVSScoreType* and *OSVDB*. The implementation choice made is to represent external references as parent classes; this allows ontologies to expand independently and makes them easier to manage.

As with other ontologies, in addition to classes also the properties of both the objects and the data are defined; in particular, object properties represent the relationships between classes. An example is the property *hasCVE_ID* that has as domain the class *Vulnerability* and as range the class *CVE*; so, an individual of the class *Vulnerability hasCVE_ID* an individual of the class *CVE*. Not all the possible properties of parent classes of external sources are defined: this may become a drawback because each implementation must specify and correctly maintain the structure of the data.

UCO is at the moment the most comprehensive and exhaustive ontology for cybersecurity related information and it has been mapped to all the most common publicly available ontologies, but for the purpose of this thesis one fundamental aspect is missing: the description of the infrastructure. In fact, the focus of UCO is to give an overview about all general concepts in cybersecurity, including possible attacks, attack patterns, means and consequences, but only very few classes are dedicated to the description of the infrastructure. The classes *Address, File, Hardware, NetworkState, Product, Software* and *System* are present but they do not allow for the detailing of the different components of the infrastructure and the relationships between them.

The use cases [33] for which UCO is designed are for example the discovery of all the vulnerabilities of a certain vendor or the search of the least vulnerable software of a specific category, so the idea of collecting knowledge from different sources remains very important but it is not intended for finding out all the vulnerabilities of an specific infrastructure.

4.2.2 IoTSec Ontology

Another important existing ontology in the domain of cybersecurity is the *Internet Of Things Security Ontology* [25]. The goal of the authors of IoTSec is to have a resource that

helps gaining a big picture of the information security in the Internet of Things field.

It was created following the Methodology for Enterprise Reference Ontology Development (MENTOR) that is composed of two phases: *Lexicon Settlement*, that consists of the gathering and acquisition of all the information and the knowledge necessary to create a lexicon of the domain of interest, and *Reference Ontology Build*, that is the creation of the reference ontology followed by the mapping with existing ontologies.

The Reference Ontology is created following three steps: first, all the keywords belonging to the domains of IoT Technologies and Information Security and similar domains are analysed in order to find a correlation between them. Then, related works concerning the concepts under consideration, such as already existing ontologies, taxonomies, scientific articles, are studied and compared to understand what are the differences and similarities and what can be added and improved. The last step consists in mapping and harmonizing existing ontologies; this can be done using a glossary and a thesaurus of the domain that can help finding mismatches and inconsistencies in the semantic of the Reference Ontology. The final result consists of the top level classes shown in figure 4.5 and their relationships.

As shown in figure 4.5, IoTSec introduces the concepts of *Asset* and *SecurityMechanism* that were not present in UCO and that help outline an overview of a single infrastructure. In fact, IoTSec was created to ensure that companies in the Internet of Things field use their devices in the most secure possible way and to help them identify vulnerabilities and criticalities. Being IoT oriented, many subclasses of *Asset* are very specific, like *UMTSTechnology, LTETechnology* and *BluetoothTechnology*, but the idea of having the possibility to specify this kind of characteristics is considered for this thesis as a good starting point.

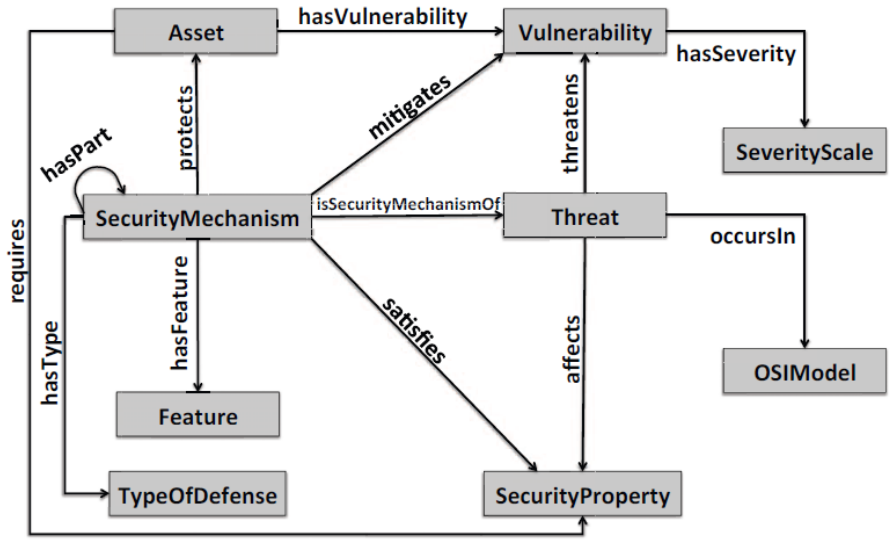

Figure 4.5. IoTSec classes overview [25]

The class *SecurityMechanism* represents as well a positive aspect of this ontology because it allows to represent concepts like cryptographic algorithms, firewalls, protocols and

other mechanisms that are being used. By making use of the reasoner, it is possible to infer for example the necessity to introduce a new security mechanism to protect a specific asset or to find out that a security mechanism already present is not sufficient to defend a *SecurityProperty*.

The negative aspect of IoTSec is that the knowledge of the classes *Vulnerability* and *Threat* is static: it does not come from external sources and it must be managed and maintained directly when implementing the ontology. In particular:

- **Vulnerability**: it contains a set of sub-classes listing the possible vulnerabilities for the different technologies (for example *BluetoothVulnerability, WebVulnerability, SoftwareVulnerability*) and for some of these sub-classes a set of individuals is specified (for the *WebVulnerability* some of the instances are *BrokenAuthenticationSession, SensitiveDataExposure*);

- **Threat**: it contains the threats corresponding to the vulnerabilities previously defined (for example *BluetoothAttack, WebAttack*); also in this case some instances are defined such as for the case of *WebAttack* where there are *BruteForce, BufferOverflow, XSS*.

So, the idea is to already have a complete and detailed list of the possible vulnerabilities and attacks for the cybersecurity field in the ontology. This is helpful when performing assessment of an infrastructure, but it makes it more difficult to always have an up-to date list; as a matter of fact, the updates must be performed manually by the user because the language used is not standard, so it can become a long process and can produce errors.

The relationships between the classes are defined in figure 4.5, so the *Asset* class is directly connected to the *Vulnerability* class through the object property *hasVulnerability* and the *SecurityMechanism* class is directly connected to both the *Threat* and the *Vulnerability* classes respectively through the object properties *isSecurityMechanismOf* and *mitigates*. This means that all the information in the ontology is strongly connected and any change made must be consistent; if the amount of information is significant, it may become difficult to manage.

4.3 Vulnerability Assessment and Penetration Testing

The new ontology is created with the intention of becoming a basis for VAPT operations and, consequently, retains the characteristics of the above ontologies that make it useful for this purpose. So, to better understand some implementation choices that will be addressed later in the thesis, it is important to have a general idea of what *Vulnerability Assessment and Penetration Testing* operations are.

The first step consists in performing *Vulnerability Assessment* [18]: a system, a network or a software is scanned to identify and report possible vulnerabilities. Many techniques can be adopted:

- *static analysis*: no test case is executed but only the code structure and the components of the system are analysed;

- *dynamic analysis*: tests are executed either manually directly by the tester without the use of any tool or automatically by using some testing tools that can produce more accurate results.

Generally, vulnerability assessment is not an expensive operation, it can be performed with automated tool so it is not time-consuming and can be performed at any development stage, especially if conducted in a static way. The negative aspect is that it reports all found vulnerabilities and this means both true positives and false positives. The way to discriminate between them is the next step: *Penetration Testing*.

It consists in trying to exploit the found vulnerabilities from an attacker point of view, so using a black-box approach that means that the attacker does not have any privileges and knows only information that are publicly available (such as the public IP address). The positive note is that it reports only vulnerabilities that can be exploited, but it is mostly based on manual activities so it requires high expertise and it can be very expensive to perform.

The standard for PT defines 7 stages [12]:

- *Pre-engagement Interactions*: defining the aims and constraints;
- *Intelligence Gathering*: knowing the assets and their values;
- *Threat Modeling*: defining how to threat assets;
- *Vulnerability Analysis*: performed with black-box approach;
- *Exploitation*: creating exploits for vulnerabilities;
- *Post Exploitation*: understanding what can be gained by the exploit;
- *Reporting*: reporting the obtained results.

Chapter 5

ACN Ontology

For the purpose of this thesis, the starting point of the research has been the ontology created by the *Dipartimento delle Informazioni per la sicurezza*, now *Agenzia per la Cybersicurezza Nazionale* of the Italian Government. The goal was to study it and compare it with the above-mentioned existing ontologies and sources of knowledge in order to integrate it with more information and to improve it.

The *ACN* Ontology follows the guidelines of the DPCM mentioned in 3.2 and it allows to comply with the law by describing in a structured manner all the relevant components of the *Bene ICT* and their relationships. In particular, it provides a description of the architecture of the infrastructure, of the relationships with other *Beni ICT*, with the *Funzioni essenziali* and/or the *Servizi essenziali* and of any possible incoming dependencies.

The *ACN* ontology and the tool developed to use it are analysed below.

5.1 *PSNC* Ontology

The main classes, here called entities, of the ontology are:

- **Soggetto**: it is the entity that compiles the description and is inserted in the *Perimetro* or a third party that provides a *ServizioEsterno*;
- **FunzioneEssenziale**: the *Funzioni Essenziali* provided by the *Soggetto*;
- **ServizioEssenziale**: it represents the service provided by the *Soggetto*;
- **BeneICT**: the ICT and OT component necessary to exercise a *FunzioneEssenziale* or to provide a *ServizioEssenziale*;
- **DatiDigitali**: the typology of data processed by the *Bene ICT*;
- **Rete**: it represents a network that composes one or more *Beni ICT* and possibly one or more *ServiziInformatici*;
- **SistemaInformativo**: it represents one or more computer services that composes one or more *Beni ICT* and possibly one or more *ServiziInformatici*;

- **ServizioInformatico**: it represents a computer service that composes one or more *Beni ICT*;
- *Risorsa*: it represents both the hardware and the software components of a *Rete* and of a *SistemaInformativo*;
- **Nodo**: the spacial clustering of one or more *Risorse* located in a *Riferimento-Geografico* that can depend on one or more *ServiziEsterni*;
- **RiferimentoGeografico**: it represents a geographical location;
- **ServizioEsterno**: it is an incoming service on which depends or that depends on a component of the *Bene ICT*.

Each entity is associated with a name, a description, a set of properties (along with the type, the description and the possible valorisation obligation), a set of relationships (along with the second entity of the relation, the direction, the description and the cardinality) and a unique alphanumeric ID. The relationships between the entities are represented in figure 5.1

5.2 *PSNC* Tool

Along with the ontology, the *ACN* developed a web platform to allow those who are part of the *PSNC* to create and submit the list of *Beni ICT* they are responsible for. The platform is made of a web portal and a compilation software: from the portal, after authentication is performed, it is possible to access the software.

The tool supports the user in the creation and the definition of a new model: the first step is to choose the entity and it can be done through a proposed list of the above-mentioned entities or through a search by name; then, when the entity is selected, it is possible to specify some properties. For example, if the selection is *Bene ICT*, the insertable properties are *nome, descrizione* and *tipoBeneICT*. In some cases the property is compulsory and a value must be specified, otherwise it can be left empty; the values can be plain text or they must be chosen from a proposed list.

Once at least two entities have been created, it is possible to specify the relationship that binds them. When selecting a pair the tool suggests the alternatives: only one kind of relationship available, more than one (so a selection is needed) or no relationship available.

The last important step is the validation. At any time the model can be validated: the tool verifies the compliance of the data that have been inserted and the output contains both the errors (non-compliance that causes invalidity of the data) and the notices (non-compliance that does not cause invalidity of the data).

If the validation is successful, the file can be exported and delivered to the competent authority following the set of rules provided.

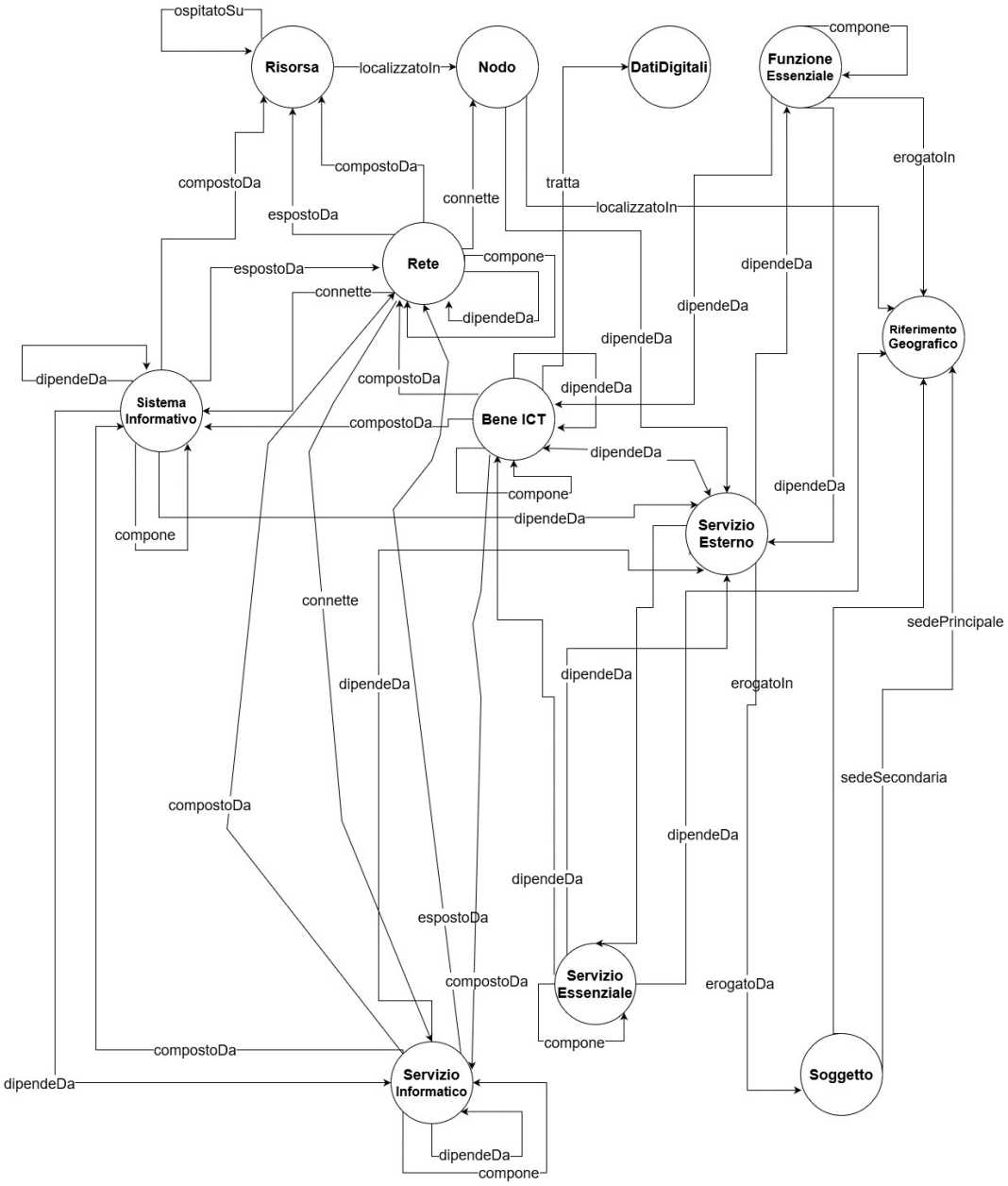

Figure 5.1. ACN ontology overview

Chapter 6

Contibution

In this chapter, the work that has been performed will be presented: the work started with the study of the specifications of the *ACN* ontology, so that it could be correctly reproduced in the *Protégé* environment and its expansion could begin. The objective of this extension is to add classes and relations to the ontology that are useful for specifying, firstly, the security mechanisms present in the infrastructure described and, secondly, the vulnerabilities of the various assets and the attacks that may derive from them. These changes are made to the ontology with a view to preparing it to include not only information entered manually by the user, but also coming from external sources. Two examples are presented in this respect: the first using as an external resource *CVE* and the second using the *ATT&CK Framework*. The work will conclude with the proposal of a high-level implementation of the communication between the created ontology and the different information sources.

6.1 The Cyber Ontology Creation

As previously mentioned, the new ontology is the expansion and improvement of the *ACN* ontology so, first, it is necessary to convert the *ACN* ontology in owl format using the *Protégé* tool. For each entity, two tables are given describing respectively:

- Entity properties with: name, datatype and if that property is mandatory or not for the entity;
- Entity relationships with: name, the entities involved and the cardinality of the relationship.

Each entity is represented as a class and they are all sub-classes of the `owl:Thing` class. The properties of the entities become `data properties` of the classes and for each of them the domain and the range are defined: the *domain* represents the one or more classes that have the property and the *range* represents the value that the property can take (it can be a datatype or a customised range defined by the user). The relationships between the entities become the `object properties` of the classes and, also in this case, both domain and range are represented: the *domain* is composed of the union of the one or more classes from which the relationship starts and the *range* is the one or more classes

towards which the relationship ends. For the `object properties` it is possible to specify also the cardinality.

For example, for the entity `DatiDigitali` the properties are:

- `descrizione`: its datatype is string and it is mandatory;
- `nome`: its datatype is string and it is mandatory.

In this case there is only one relationship that is `tratta` from `BeneICT` to `DatiDigitali` with `BeneICT` that may be connected to some entity `DatiDigitali` and `DatiDigitali` that must be connected to at least one entity `BeneICT`. To express the cardinality, sub-classes are defined so in this case the class `BeneICT` is a sub-class of `tratta some DatiDigitali`.

For some properties of certain classes, the *PSNC* ontology specifications define a list of values that these can take. For example, the class `Risorsa` has the three following properties: `tipoRisorsa, tipoFunzionalità` and `sottoTipoFunzionalità`. Initially, the properties were considered as three different sub-classes of the class `Risorsa`, and the values as individuals of each sub-class or directly as sub-classes themselves because the hierarchy between the properties was not made explicit. However, it is reasonable to think that the same `Risorsa` may have all the three properties, so the final implementation choice is to define only the `Risorsa` class with the three data properties and in each of them specify as range the set of values defined in the specification. In this way, when creating an individual of the class `Risorsa`, it is possible to assign to it a `tipoRisorsa`, a `tipoFunzionalità` and a `sottoTipoFunzionalità`. The reasoner will verify that the value assigned is within the range.

The UCO and the IoTSec ontology were already available in the owl format, so we now have all three ontologies in the same format and can compare them using *Protégé*, which will also be used to create the new ontology.

The first part of the development of the new ontology will focus more on the description of the single infrastructure and its security implementation information, then the second part will address issues related to the inclusion of information about possible vulnerabilities and attacks in the ontology. Lastly, an implementation model will be proposed to make different information sources interact with the ontology so that the knowledge about vulnerabilities and attacks is not only static, so for example manually inserted by the user, but also dynamic, i.e. automatically derived from external databases.

First of all, therefore, the ontology has been modified so that it can fully describe the architecture of the infrastructure and the security mechanisms adopted.

The first change that is made to the *ACN* ontology concerns the introduction of new classes to modify the hierarchy between the existing ones. In particular, some new concepts taken from IoTSec are introduced:

- `Asset`: it is a direct sub-class of `owl:Thing` and it is a general concept that includes any information, device or component that must be protected;
- `SecurityProperty`: it contains a set of individuals representing the main security properties that assets may require (AccessControl, Accountability, Anonimity, Authentication, Authorization, Availability, Confidentiality, Integrity, NonRepudiation);
- `SecurityMechanism`: it is a general class that contains all the possible mechanisms put in place to protect assets.

The underlying idea is the one of IoTSec, but it is simplified because the concepts of threat and defense are not addressed. This is only the first step towards the new ontology, as the concepts of vulnerability and attack will also be introduced, but later in the chapter. The result achieved so far is the schema in Figure 6.1 that represents a general overview of the new ontology.

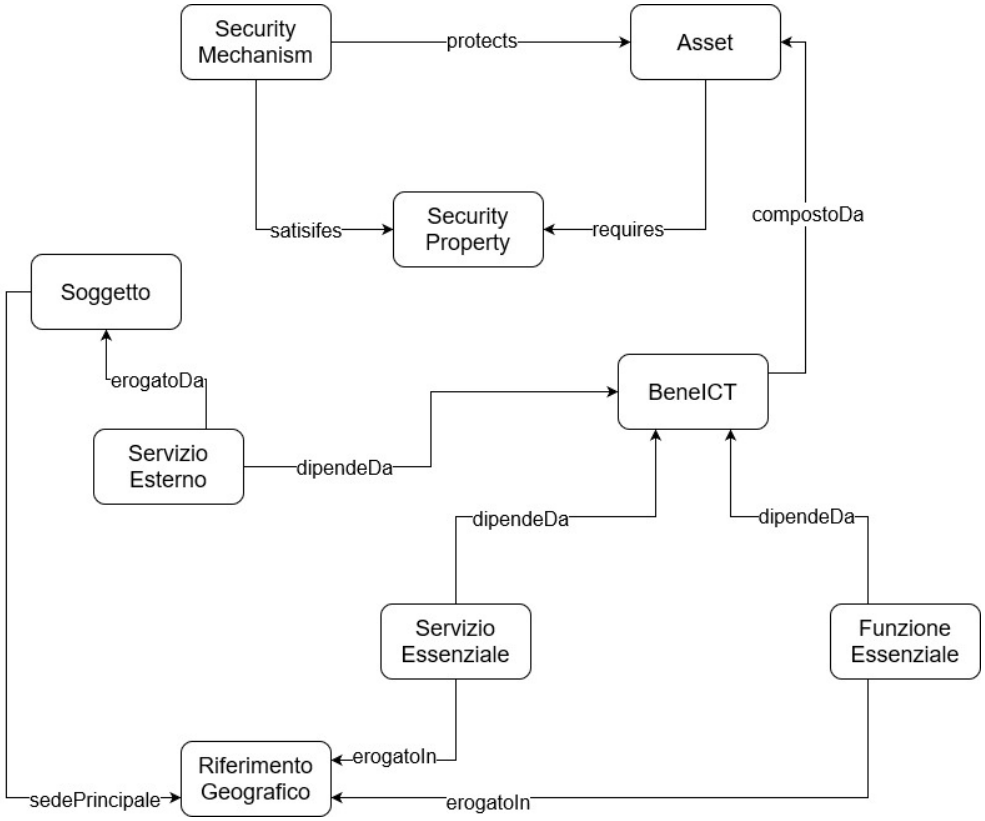

Figure 6.1. Overview of the new ontology

Analyzing more in detail the `Asset` class, it is made up of three sub-classes:

- `Information`: it is the super-class of the entity `DatiDigitali` which has been further specialised with the two sub-classes `AtRest` and `InTransit`; this was not present in the original ontology but it was considered appropriate to add it because data in transit and at rest may need different security levels or approaches;
- `Nodo`: it was left unchanged, with the possibility to specify its type through the data property `tipoNodo` and choosing between a set of values given by the *ACN*;
- `Technology`: it is a super-class that groups `ServizioInformatico`, `SistemaInformativo`, `Risorsa` and `Rete`; only the last class was broken in four sub-classes depending on the technology adopted for the network (`DMZTechnology`, `EthernetTechnology`, `VPNTechnology` and `WiFiTechnology`).

The `Asset` class is related to the `SecurityMechanism` class through the object property `hasSecurityMechanism` whose domain is `Asset or BeneICT` and whose range is `SecurityMechanism`. An individual of the class `Asset` can be made of one or more individuals of the class `Asset` so it is added in the description of the domain and range of the relationships `compostoDa` and `compone`. The same reasoning applies to the class *SecurityMechanism*.

The `SecurityMechanism` class introduces some security concepts that it was not possible to specify in the original ontology, but that are present in the IoTSec ontology and that are important when wanting to assess the security of an infrastructure. Starting from IoTSec, only the more general information are taken into consideration, leaving aside specific classes of the IoT field which are not of interest for the purpose of this thesis. All the new information are divided in the following sub-classes:

- `CryptographicConcept`: it introduces in the ontology three key concepts of cryptography that are `DigitalSignature`, `HashFunction` and `KeyManagement`; their implementation can be represented by the user through individuals of the classes and each of them can allow to satisfy specific security property. The data property `hasByteLength` is created so it is possible to add the length of the key or of the hash;

- `EncryptionAlgorithm`: it specifies the algorithm used when an encryption is performed; it is possible to specify if it is asymmetric or symmetric and, if so, if it is a block cipher or a stream cipher through the presence of the corresponding sub-classes;

- `NetworkManagementSecurityMechanism`: it allows to detail the security mechanisms present in the infrastructure that are dedicated to networks; the main sub-classes are `IntrusionPreventionSystem, IntrusionDetectionSystem, Proxy, ReverseProxy, Antivirus, NetworkManagementSecurityMechanismTool` and `ServerSecurityMechanism`;

- `SecurityPropertyMethod`: it contains the methods used to satisfy the main security properties that have not been addressed by other classes; the sub-classes are `AccessControlMethod`, that contains `Login/Password`, `AuthenticationMethod`, that contains for example `AuthenticationServer, MutualAuthentication`, and `IntegrityMethod`, that contains `ChecksumAlgorithm` and `MessageAuthenticationCode`;

- `Protocol`: it wants to regroup all the individuals that represent a protocol under the same class.

6.1.1 Vulnerability Assessment

As seen in Chapter 4, UCO and IoTSec use two different approaches for vulnerabilities. The purpose of this thesis is to obtain a trade-off of the two so that a complete and easily integrated solution can be adopted for performing vulnerability assessment.

In detail, the idea is adding to the present ontology the class `Vulnerability` as subclass of `owl:Thing`; to simplify the ontology in comparison with UCO, only this new class is introduced to describe vulnerabilities. Its main relationships with other classes

are defined, so that all its individuals and sub-classes inherit the same relationships. The approach followed is to use it as a parent class of any other classes that contain information about vulnerabilities, such as *CVE*. The main relationships are the ones of IoTSec:

- `Asset hasVulnerability Vulnerability`;
- `Vulnerability isVulnerabilityOf Asset`;
- `SecurityMechanism mitigates Vulnerability`;
- `Vulnerability isMitigatedBy SecurityMechanism`.

At this point it is necessary to add some other classes that represent the link with the external databases *CVE* and *CWE*: in this way it is possible to integrate the ontology with new information. This is done following the UCO approach: a class for each external source is defined, but they are grouped so as to create a hierarchy between the classes and not have them all on the same level. In particular, the sources used are those discussed and explored in chapter 4: *CVE* and *CWE*.

Both *CVE* and *CWE* are implemented as sub-classes of `Vulnerability`, in this way they inherit the existing relationships of the class `Vulnerability` with the other classes in the ontology.

The `CVE` class has the data property `CVE_ID` where the ID of the vulnerability can be stored as a string; the `CWE` has an equivalent data property called `CWE_ID`. Both the classes have the data property `CVSS` where the severity score of the vulnerability, if present, can be saved. In a vulnerability assessment perspective this can be of great importance because it offers the possibility to create a vulnerability rating and, when fixing them, it allows to prioritize and address the most severe ones first.

The new schema of the ontology is depicted in Figure 6.2.

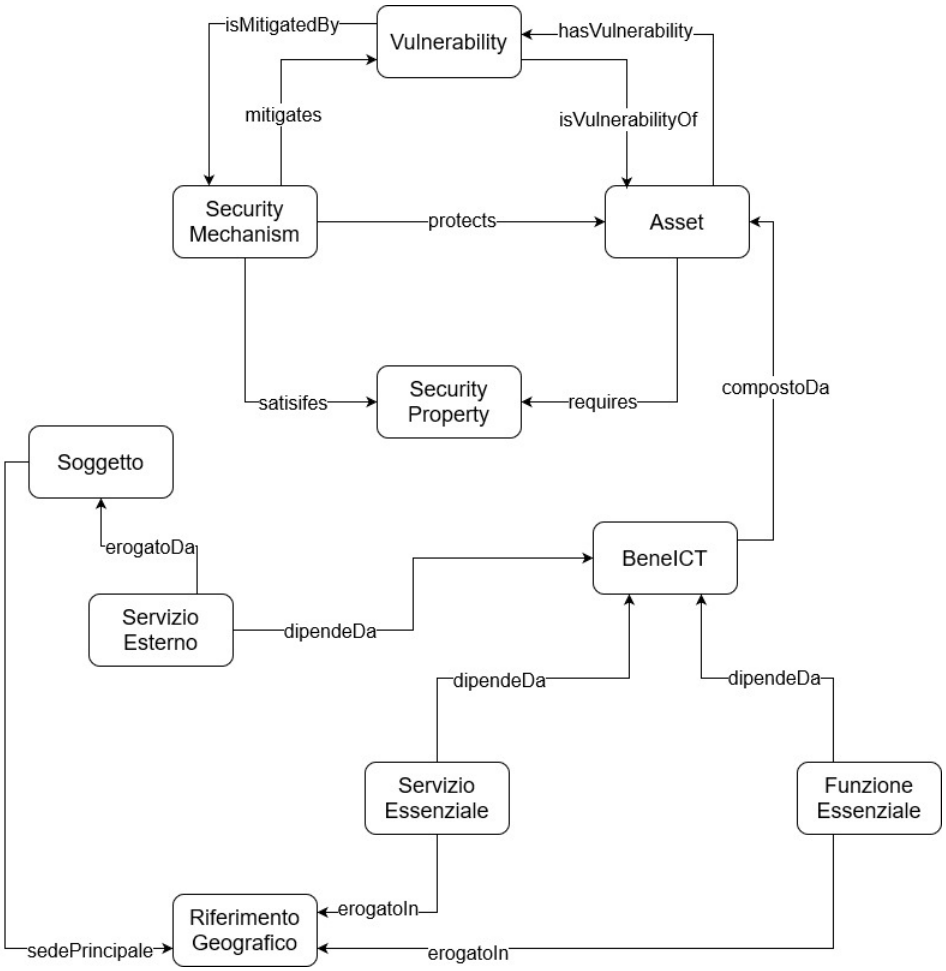

Figure 6.2. Overview of the new ontology with the Vulnerability class

Having made these changes, the ontology now also contains dynamic security information. In fact, vulnerabilities can be added manually or automatically by scanning the assets to identify known vulnerabilities; at this point the IDs of these vulnerabilities are saved and all the needed information to represent actual relationships between vulnerabilities and assets in the infrastructure are derived from the external sources. The details on how to implement this process will be explained later in the chapter.

6.1.2 Penetration Testing

The details of how to carry out penetration testing and which techniques to use are beyond the scope of this document, but the intention with the developed ontology is to provide a basis that can serve as a guide for these operations. For this reason, the ontology was further developed by introducing the `Attack` class.

The *Attack* class is not present in IoTSec; in this case the relationships are not the ones

of IoTSec, but they are reduced so that the classes result less connected between them. The `Attack` class is linked only to the `Vulnerability` class through the following properties:

- `Vulnerability causes Attack`;
- `Attack exploits Vulnerability`.

Both the properties can be directed to the class itself: a vulnerability can cause another vulnerability and an attack can exploit another attack.

The deployment of the class is similar to the one of `Vulnerability`: its sub-classes are the parent classes of external sources that inherit the relationships defined for `Attack`. For the moment, the only sub-class is the one representing the *ATT&CK Framework* described in 4, but other resources may be added in the future.

The `ATT&CK` class has the data property `techniqueID` to store the identifier of the technique derived by the `ATT&CK Matrix`. In addition, a direct connection between the `ATT&CK` class and the `CVE` class is created including in the ontology three new object properties.

According to a study carried out to map *ATT&CK* to *CVEs* [13], when analysing the possible consequences of a vulnerability it is sufficient to stop after three steps: *exploitation technique, primary impact* and *secondary impact*.

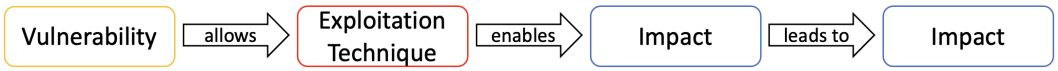

Figure 6.3. Consequences of a vulnerability [13]

These three steps are reported in the ontology as object properties with domain the `CVE` class and with range the `ATT&CK` class, so that given a specific CVE it is possible to identify its consequences maintaining the chronological information; then, as previously mentioned, an attack can cause another attack so also this information can be stored.

The specifics on how the mapping between the two resources works will be addressed later in the chapter.

The final aspect of the ontology is the one in the figure 6.4.

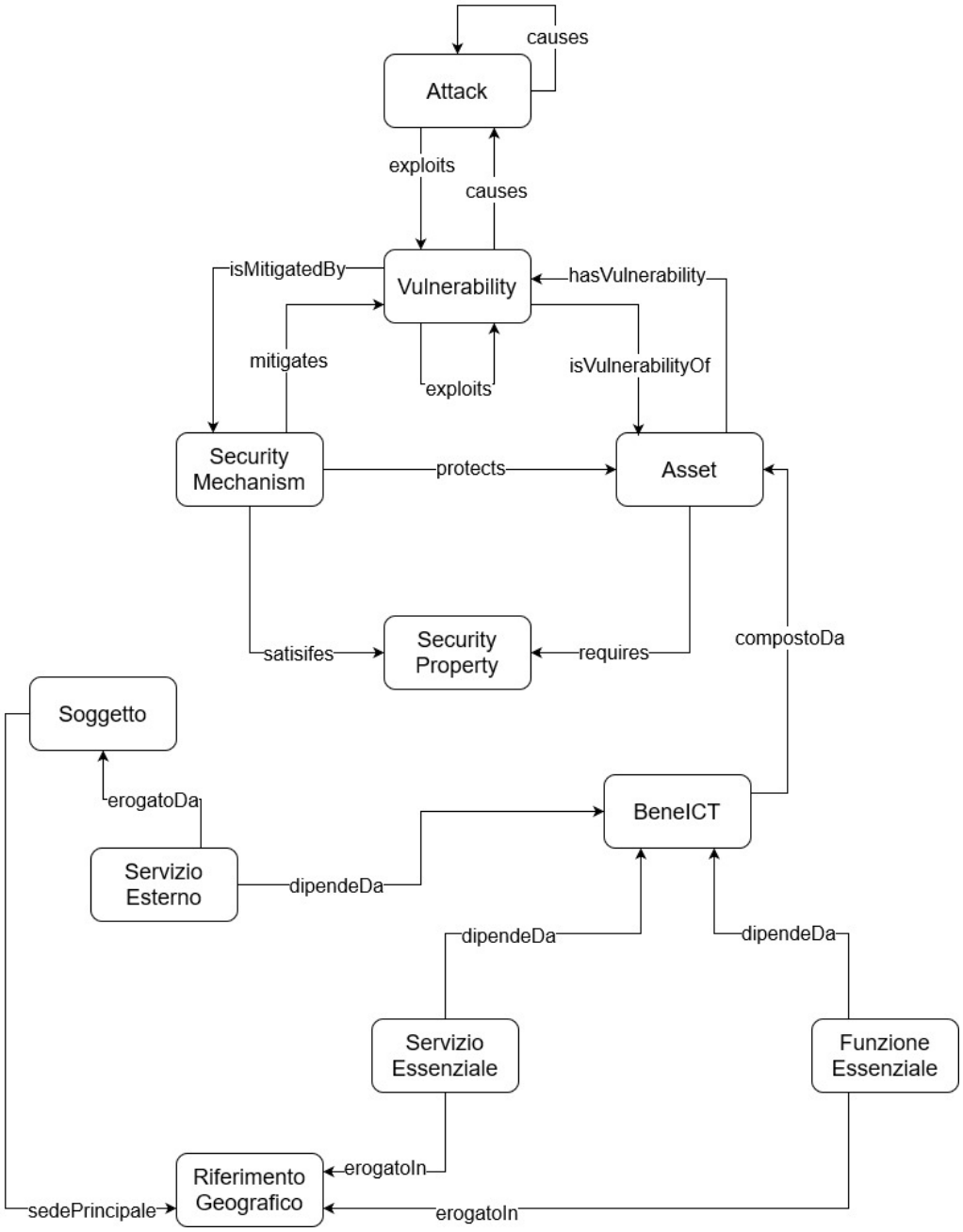

Figure 6.4. Overview of the final ontology

6.2 Project for automated interaction

At this point, the ontology is as complete as possible: it contains information on the architecture of the infrastructure, the security mechanisms implemented and has been prepared to receive information from external resources. It is now necessary to elaborate on this last point, that is, how to make the different sources of information interact with each other.

The ontology must be completed in steps: the first part, which concerns the description of the infrastructure, the relationship between its components and the security mechanisms adopted for each asset, is carried out manually by the subject or the person acting on his behalf. After this initial phase, a more detailed picture of the state of the infrastructure is obtained.

It is possible, in fact, to use some features offered by *Protégé* to conduct an initial form of security assessment. The first one is the *reasoner*: it can verify if the ontology is consistent and can add some new inferred information about properties and classes. For example, if two classes are *disjoint* and the same individual was created as belonging to both classes, the error in figure 6.5 will occur. In the ontology created, the two sub-classes of `DatiDigitali` are disjoint so the same individual can be either belonging to the class `AtRest` or `InTransit`.

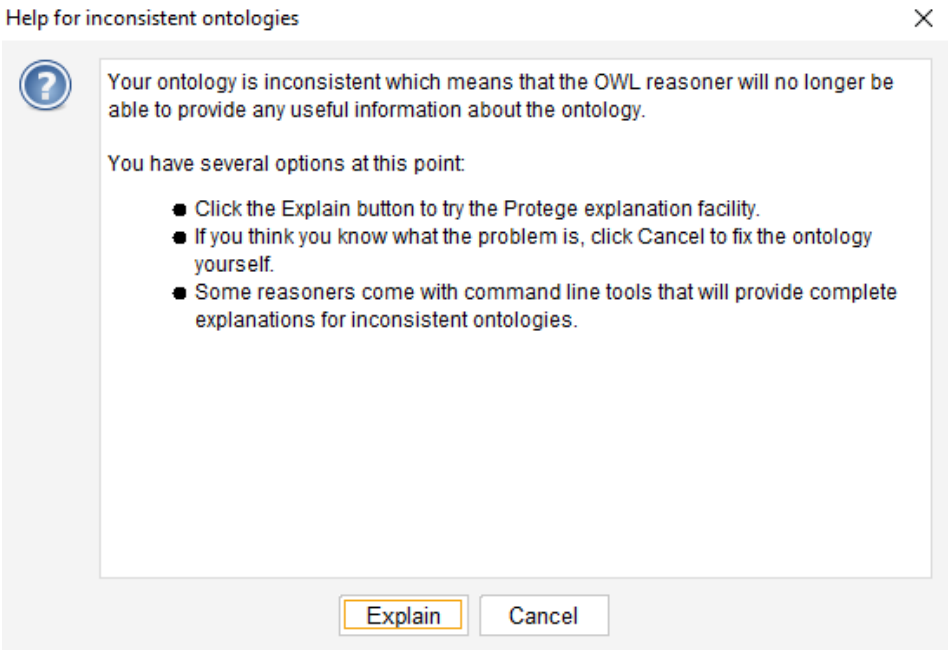

Figure 6.5. Example of inconsistency error in *Protégé*

Other inferred information can be relationships that were not manually inserted but that exist in the infrastructure. The reasoner can also check that the cardinality of the specifications is respected.

After running the reasoner, the ontology is said to be *classified* and another important feature of *Protégé* can be used: the *DL Query* tab. This plugin is particularly interesting because it allows to query the ontology in order to get information about a particular class, property or individual [14]. For example, it is possible to obtain the list of all the instances of the class `Risorsa` that are connected to an individual in the `SecurityMechanism` class. The query would be:

Risorsa and isProtectedBy some SecurityMechanism

Another possible scenario is wanting to know if some encryption algorithm uses a key that is not long enough to be considered secure. So a list with all the `EncryptionAlgorithm` instances whose key is for example 256 byte can be obtained with the following query in figure 6.6:

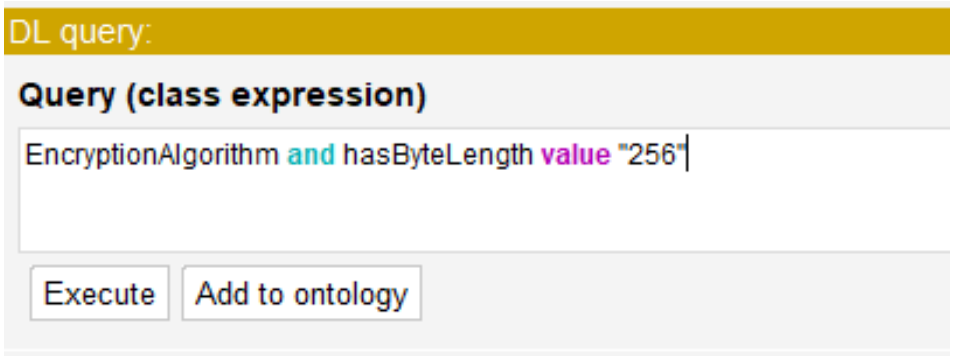

Figure 6.6. Example of query performed in *Protégé*

6.2.1 CVE-search

Having made these considerations on the basis of the static description of the infrastructure, the interaction part with *CVE* can be examined. The main tool that can be used to interrogate *CVE* is the *cve-search* tool[1]: it is a project that has been developed in several stages, starting with a simple script to fetch CVEs from MITRE and save them in a database and ending with the current state of a set of tools consisting of a web interface and several features. The *cve-search project* main objective is to import *CVE* to a local database in order to avoid doing public and direct lookups to limit the exposure of sensitive data and to allow doing local lookups that are generally faster.

It includes a back-end to store vulnerabilities and related information, an intuitive web interface for searching and managing vulnerabilities, a series of tools to query the system and a web API interface.

It is developed for a Linux environment [15] and it uses two different databases, the *MongoDB* and the *Redis*. *MongoDB* is by default the cvedb and at the moment of the installation it must be populated: the JSON files from *CVE* and *CPE* are fetched, but it is also possible to import an own file; the database can also be updated and repopulated. To interact with *cve-search*, a web interface is present; users are able to:

- *View recent CVEs*: it contains an overview of all the CVEs, ordered from recent to old;
- *Browse per vendor*: search CVEs per vendor; the result is ordered from new to old, sorted by Last Major Update;
- *CVE overview*: it contains all the information that is in the CVE-Search database.

In particular, for every CVE these are the fields that are present:

- **ID**: the identifier of a CVE;
- **Summary**: the description with an explanation of the attack vector and the result;
- **References**: links to other websites with information about the CVE; it can be vendor statements, explanations etc..;
- **Vulnerable Configuration**: list of products vulnerable to the CVE; it can be an empty field;
- **CVSS**: the severity score;
- **Last Major Update**: the latest update where information is added or changed;
- **Published**: the date of the publication of the CVE;
- **Last Modified**: the date the CVE got last modified; modifications can be spelling changes, changes in wording etc.

[1] https://www.cve-search.org/about/

The use of *cve-search* becomes particularly interesting for the purpose of this thesis because it allows to interact with CVE and save information locally. This provides another level of analysis of the information in the ontology: the tool can be used to populate the class `CVE` of the ontology. In particular, the ID is saved as a data property of the individual of the class `CVE` as well as the `CVSS`; then, from the vulnerable configuration or the references, information about the products affected by the vulnerability can be derived and saved in the ontology in the form of relationships (e.g. `isVulnerabilityOf`).

The details of the implementation of the import from the MongoDB database to the ontology are beyond the scope of this thesis, but there exist, currently, tools that allow automatic conversion from MongoDB to owl files such as the *M2Onto: MongoDB to Ontology* tool [19]. It uses transformation rules to map MongoDB constructs into OWL ontology. The approach is composed of five steps: first, the ontology skeleton must be created (classes, sub-classes and relationships) and, on the database side, every collection is transformed in an ontology class and from every document a sub-class relationship is extracted. The second step consists in finding out the data and object properties: the firsts from the fields of the document, the latter either from embedded document or from references with DBRef. The third step is about identifying the individuals: they correspond to the values of the fields in the database. Fourth, class axioms (equivalence and disjoining), property axioms (inverseOf) and constraints (cardinality constraints, value constraints) are deduced. Finally, the ontology is enriched with classes definition operators (union, intersection, complement). All these five steps are executed by the tool.

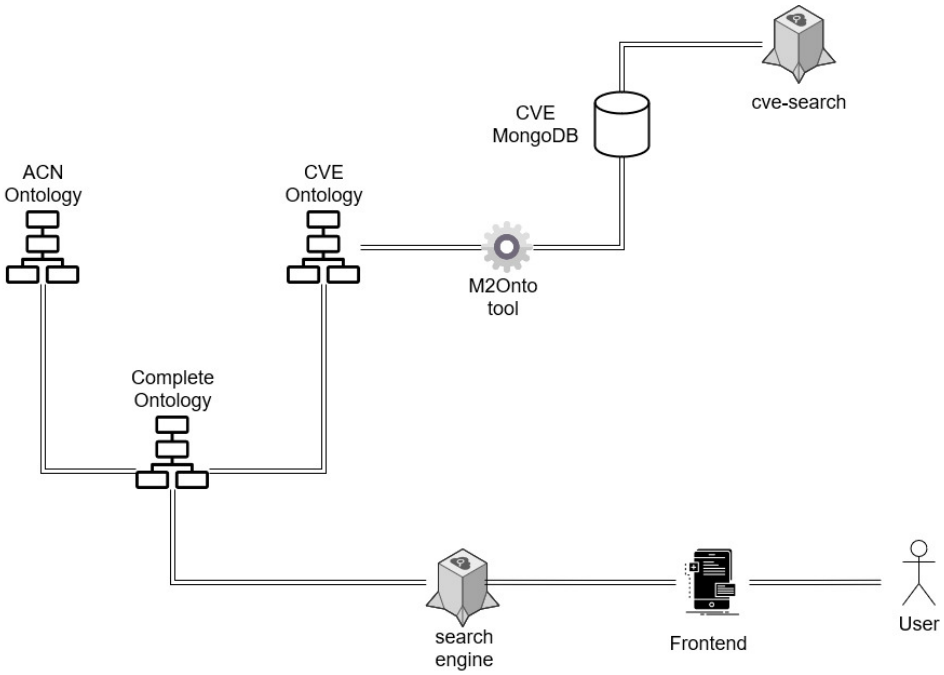

Figure 6.7. Overview of the resources interaction

The figure 6.7 represents the idea of interaction between the different resources explained so far: the user, i.e. the subject, can interact with the knowledge base via a search engine and obtain at the same time static information on the architecture of the infrastructure and information on vulnerabilities deriving from *CVE*; the latter are periodically updated as provided by *cve-search*.

In this way, vulnerability assessment is made more automatic and easier to perform; naturally, following the process outlined, it is possible to add other external resources to obtain an even more complete ontology. In particular, if a guide for penetration testing operations is required, the knowledge deriving from the *ATT&CK Framework* can be added.

6.2.2 AT&CK to CVE mapping

As previously said in 4, the *ATT&CK Framework* contains all the tactics that can be pursued by an attacker and the techniques that can be used to achieve the objective. It gives an idea of the chronological order of events during an attack and how a certain exploit can lead to another one. This kind of information becomes of particular importance when it comes to conducting Penetration Testing operations: it allows the subject to obtain a list of techniques to exploit a specific vulnerability. Also, *ATT&CK* includes detection and mitigation information, which can be used to investigate whether the mitigations the subject has in place are adequate for addressing the vulnerability or whether additional mitigations are needed [20]. For this reason, it has been considered appropriate to add the `Attack` class within the ontology, which, for the time being, only contains the `ATT&CK` subclass, but may be extended in the future with other forms of knowledge.

In the current state of the ontology, vulnerability information is almost entirely derived from *CVE*, so it is chosen to directly link individuals of this class with those of the `ATT&CK` class. For this purpose, the *Mapping ATT&CK to CVE for Impact* [13] project has been followed. The goal of this project is to standardize the way the impacts of vulnerabilities are described so that defenders will be able to use this ATT&CK-based impact information to better guide their risk models; so, what this methodology wants to create is a direct connection between vulnerability management and threat modeling. In detail, as anticipated in 6.1.2, the attack process is divided in three steps that are found to be sufficient enough to describe it:

- **Exploitation Technique**: the method used to exploit the vulnerability;
- **Primary Impact**: the initial benefit gained through exploitation of a vulnerability;
- **Secondary Impact**: what the adversary can do by gaining the benefit of the primary impact.

Three methods are defined for mapping ATT&CK techniques to vulnerabilities: *Vulnerability Type, Functionality* and *Exploit Technique*; of these three, the *Vulnerability Type* method is the one that has the most mappings for all the three categories above-mentioned, so it will be the one deepened.

Vulnerabilities that have the same type often also have the same attack steps; if that type has a common set of techniques used to exploit the vulnerability or that can be executed when the vulnerability is exploited, then the mapping will be included, but not

all the vulnerability will necessarily have a mapping for all the three available categories. An example of mapping is the following in table 6.1:

Vulnerability Type	Primary Impact	Secondary Impact	Exploitation Technique	Notes
Cleartext Transmission of Sensitive Information	T1552 (Unsecured Credentials)	T1078 (Valid Accounts)	T1040 (Networks Sniffing)	A sub-technique can be chosen where applicable

Table 6.1. Example of Vulnerability Type mapping

As seen in table 6.1, all the vulnerabilities that belong to the *Cleartext Transmission of Sensitive Information* type will have as primary impact the technique T1552, as secondary impact the technique T1078 and as exploitation technique the T1040.

At this point, there is still one step missing: the connection with *CVE*. The project is very recent but there is already a large number of vulnerabilities that have been mapped and can be found at their repository [2]. This file contains a table where to each CVE ID corresponds a technique ID, where possible, for each of the three categories; the format of the file is *csv*.

Having this file available, it is possible to convert it into the owl format using different tools and thus obtain an ontology formed by the `ATT&CK` and `CVE` classes; the first has as individuals the various IDs of the techniques, while the second contains as individuals the CVE IDs. The individuals of the two classes are bound by the object properties `Primary Impact, Secondary Impact` and `Exploitation Technique`. This new ontology is consistent with the previously created ontology so they can be merged, obtaining the final ontology. At this point the final interaction between all the resources is the one represented in figure 6.8.

[2] https://github.com/center-for-threat-informed-defense/attack_to_cve/blob/596de4aea26b05871e4f3197a3a4fb768c1f8321/Att&ckToCveMappings.csv

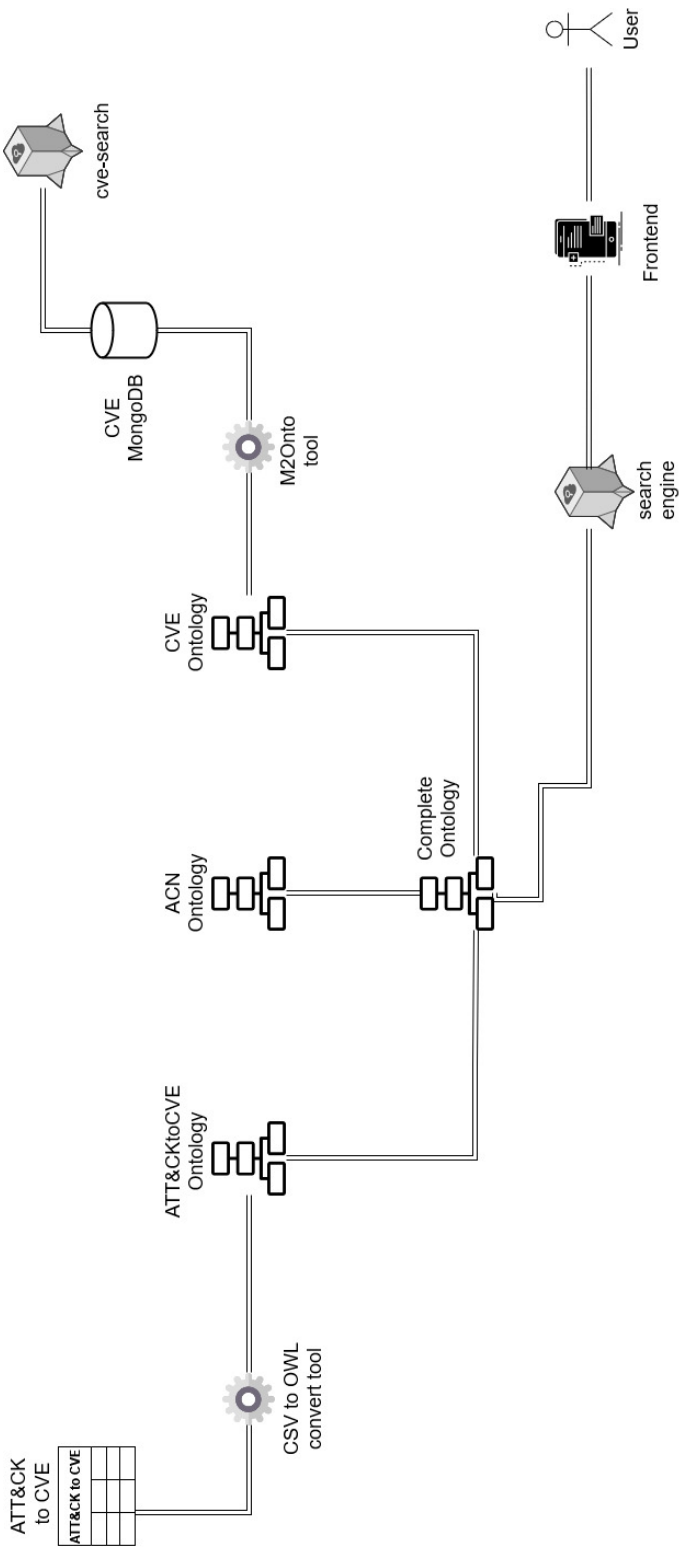

Figure 6.8. Overview of the final resources interaction

Chapter 7

Conclusion

In this final chapter the first part will provide a summary of the work that has been carried out and identify the objectives that have been achieved, and the second part will propose ideas for future improvements.

7.1 Conclusion

The design of the automatic interaction between the various entities involved concludes the work of this thesis. The following goals have been achieved:

- The most comprehensive existing ontologies in cybersecurity, *UCO* and *IOTSec*, have been extensively studied and their strengths and weaknesses identified. This gave a clearer idea of, on one hand, what the requirements of the ontology to be created should be and, on the other hand, what are the most troublesome aspects to not be included in the new ontology. What emerged was that UCO is a more general ontology that allows the possibility to integrate information from different sources but does not allow the infrastructure to be described in detail; on the contrary, IoTSec contains several classes to specify to a high degree of detail the components of a system, but these classes tend to be very specific to the IoT domain. Both ontologies, although using different approaches, contain information on vulnerabilities and attacks that inspired the creation of the new ontology. This analysis provided a good understanding of the current state of ontologies and made it possible to become more familiar with the *Protégé* tool.
- The new Italian laws on cybersecurity were introduced in order to outline the context and thus the objectives of the thesis. The starting point for the actual work was the analysis of the ontology created by the ACN. This ontology allows the technical description of the infrastructure and the relationships between its components. From the study and comparison with other ontologies, it emerged that different nuances of relationships can be described using this ontology and many structural details can be added, but the negative side is that the different entities do not have a real hierarchy. Moreover, only a few details on the security mechanisms implemented can be specified and it is not designed either to obtain information from external databases or to contain information on vulnerabilities and attacks.

- Trying to combine the strengths of each ontology and improve the weaknesses, a new ontology has been created which allows to describe in considerable detail both the architecture of the infrastructure and the level of security required by the various assets and the consequent security method applied; in this way it is possible to identify the first weaknesses and vulnerabilities with only a static description of the system. This is an important innovation because the ontology becomes a fundamental tool for vulnerability assessment and management. With regard to vulnerability assessment, to make the ontology even more complete, but above all more dynamic, the *Vulnerability* and *Attack* classes have been introduced to contain knowledge from external sources. In particular, for the moment, the external sources considered have been the *Common Vulnerabilities and Exposures* and the *Mitre ATT&CK Framework*, but it would certainly be important to increase their number in the future.
- A high-level implementation project of how these different sources of information should communicate with each other has been proposed. In particular, the process was organised in two stages, first addressing vulnerabilities and then attacks. A set of APIs was identified to interface with *CVE* in order to obtain a local database with all the known vulnerabilities; starting from the database, the idea was to create a real ontology of vulnerabilities that is consistent with the ontology created so that they can be merged together. For the second step, concerning attacks, the starting point was a mapping project between *CVEs* and *ATT&CK*, which envisages matching each CVE ID with at least one technique contained in *ATT&CK* that exploits the aforementioned vulnerability.

The final result can be considered satisfactory as it fulfils all the objectives set: an ontology describing an infrastructure in all its details has been created and can be used as a basis for guiding VAPT operations. In addition, it has been designed as an extension of the *ACN* ontology and therefore respects the guidelines of the *PSNC* and can be used by entities that are part of it.

With this tool it is possible to start thinking about making vulnerability assessment and management more ontology oriented. Most of the needs highlighted by the analyses conducted have been covered, but there are still several aspects that can be deepened.

7.2 Future Developments

The work developed in this thesis offers several ideas for future developments. First of all, the final architecture that has been described to allow the user to interface with the ontology containing information from different sources is only an high-level implementation idea. The next step is to write a program that allows the different entities to actually interact with each other so as to assess the actual feasibility of the project and to understand whether the proposed strategies are the best possible or whether there are more efficient alternatives.

Once this step has been completed, the way is open to other expansions, which are examined in detail below.

7.2.1 Evaluation Metrics for Ontology

The ontology that has been created in the course of this work was originated from other existing ontologies but is in fact a new ontology in its own right; for this reason, one of the aspects to be explored concerns its evaluation.

As explained in section 2.3, there are several evaluation methodologies depending on the criteria to be considered. In the case of the ontology created, it is important to assess its *accuracy, completeness* and *consistency*, so the approach to follow for its evaluation should be a *gold-based* approach. However, it is difficult to find a single gold standard that includes all the details that should be reported in the ontology, so it is necessary to compare the ontology with more than one source. In order to assess more correctly and comprehensively, also considering consistency, a *criteria-based* approach should also be used, in particular by adopting the complex and expert based approach in order to create customised metrics.

7.2.2 Integration of new external sources

A significant part of the work presented in this thesis is the integration of information from different sources and in different formats. One of the main purposes of the created ontology is to give access through a single interface to potentially all knowledge on vulnerability assessment. Until now, the focus has been on integrating structured, official and more authoritative resources that could provide this information, but, in particular with regard to vulnerabilities, many researches have shown that vulnerabilities are often spread first through social media and only later through official sources [34]. This is also due to the bureaucratic complexity of the process of entering a new vulnerability into an official database such as *CVE*.

For this reason, including social media among the external sources of the project in figure 6.8 would constitute a major innovation. In this regard, projects with this intention are beginning to appear in the literature, such as the one proposed by Romilla Syed [32], which can be considered a good starting point for the continuation of the thesis work.

The project presented in the article [32] consists of a *Cyber Intelligence Alert* system that, starting from a formal knowledge representation called *Cybersecurity Vulnerability Ontology*, automatically creates alerts to inform the user about vulnerabilities in his infrastructure and possible countermeasures. The *CVO* is composed of five main classes that are *Vulnerability, Threat, Countermeasure, Product* and *Intelligence*; in general, the ontology collects vulnerability information from various sources such as *CVE, NVD* and *vendor sites*, but it is the last class that is the most interesting one because it stores information coming from *Twitter*.

The process to obtain this type of information is the following: using a *Twitter* public API, all the tweets containing the hashtag *#CVE* are collected and for each of them also information about the author are retained; then, CVE IDs extracted from the tweets are used to obtain information about the vulnerability from *CVE*. Within the *CIA* system there is the *Social Media Intelligence Extractor-Tagger (SMIET)* that labels the tweets dividing them into five content categories: *alert*, they alert about a new vulnerability, *advisory*, they contain advisory information, *patch*, they propose a patch, *exploit*, they provide an exploit, and *root cause*, they give technical details about the vulnerability. Using the information

collected on the authors of tweets, it is possible to classify them as *security professionals, firm* or *other*; the first two categories are considered more reliable.

The labelling procedure is performed first manually by tagging a set of tweets and authors and then applying *Natural Language Processing* techniques. In particular, a supervised NLP approach is adopted and the manually tagged tweets are used as training set; the technique followed is the *ensemble learning* which is a machine learning technique that combines the output of several different classifiers to improve the accuracy of classification and that is widely used for extracting data from social media.

Information from *Twitter* and information from *CVE* or vendor sites are then merged together by a vulnerability mapper to avoid duplicates and then they are included in the *Cybersecurity Vulnerability Ontology*.

This research is just one example of how social networks can become a valuable source of information on vulnerabilities. Not only twitter, but also forums and other social networks can be used to get information on the latest vulnerabilities and possible countermeasures.

Bibliography

[1] URL: https://en.wikipedia.org/wiki/Semantic_Web.

[2] URL: https://cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/rdfs-vs-owl/.

[3] URL: https://protege.stanford.edu/.

[4] URL: https://protegewiki.stanford.edu/wiki/HermiT.

[5] URL: https://protegewiki.stanford.edu/wiki/OntoGraf.

[6] URL: https://protegewiki.stanford.edu/wiki/OWLViz.

[7] URL: https://www.sicurezzanazionale.gov.it/sisr.nsf/wp-content/uploads/2018/06/La-NIS-in-pillole.pdf.

[8] URL: https://attack.mitre.org/.

[9] URL: https://maecproject.github.io/documentation/overview/.

[10] URL: https://www.first.org/cvss/specification-document.

[11] URL: https://cwe.mitre.org/about.

[12] URL: http://www.pentest-standard.org/index.php/Main_Page.

[13] URL: https://github.com/center-for-threat-informed-defense/attack_to_cve.

[14] URL: https://protegewiki.stanford.edu/wiki/DLQueryTab.

[15] URL: https://cve-search.github.io/cve-search/.

[16] Sean Barnum. «Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX™)». In: (Feb. 2014), p. 22.

[17] Matt Bromiley. «Contextualizing the MITRE ATT&CK Framework». In: (Apr. 2021).

[18] Jai Narayan Goel and B.M. Mehtre. «Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology». In: *Procedia Computer Science* 57 (2015). 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), pp. 710–715. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.07.458. URL: https://www.sciencedirect.com/science/article/pii/S1877050915019870.

[19] Simin Jabbari and Kilian Stoffel. «Ontology extraction from MongoDB using formal concept analysis». In: Oct. 2017, pp. 178–182. DOI: 10.1109/ICKEA.2017.8169925.

[20] Jon Baker Jonathan Evans and Richard Struse. «CVE + MITRE ATT&CK® to Understand Vulnerability
Impact». In: *MITRE-Engenuity* (Oct. 27, 2021). URL: https://medium.com/mitre-engenuity/cve-mitre-att-ck-to-understand-vulnerability-impact-c40165111bf7 (visited on 10/27/2021).

[21] Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. «Malware Attribute Enumeration and Characterization». In: (Nov. 2021).

[22] Graham Klyne and Jeremy Carroll. «Resource Description Framework (RDF): Concepts and Abstract Syntax: W3C Recommendation 10 February 2004». In: (Feb. 2004).

[23] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. W3C, Feb. 1999. URL: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[24] Andrea Lucariello Lorenzo Damiano Luisa Franchina. «Perimetro di sicurezza nazionale cibernetica, in Gazzetta ufficiale tutti i beni e servizi inclusi». In: *cybersecurity360* (Aug. 2021).

[25] Bruno A. Mozzaquatro, Ricardo Jardim-Goncalves, and Carlos Agostinho. «Towards a reference ontology for security in the Internet of Things». In: *2015 IEEE International Workshop on Measurements Networking (M N)*. 2015, pp. 1–6. DOI: 10.1109/IWMN.2015.7322984.

[26] Dietmar P.F. Möller. *Cybersecurity in Digital Transformation. Scope and Applications*. Springer, 2020. ISBN: 978-3-030-60570-4.

[27] N. Noy and Deborah Mcguinness. «Ontology Development 101: A Guide to Creating Your First Ontology». In: *Knowledge Systems Laboratory* 32 (Jan. 2001).

[28] Joe Raad and Christophe Cruz. «A Survey on Ontology Evaluation Methods». In: *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Lisbonne, Portugal, Nov. 2015. DOI: 10.5220/0005591001790186. URL: https://hal.archives-ouvertes.fr/hal-01274199.

[29] Daniel Rubin, Holger Knublauch, Ray Fergerson, Olivier Dameron, and Mark Musen. «Protégé-OWL: Creating Ontology-Driven Reasoning Applications with the Web Ontology Language». In: *AMIA Annual Symposium Proceedings* 2005 (Jan. 2005).

[30] Michael Smith, Chris Welty, and Deborah Mcguinness. «OWL Web Ontology Language Guide». In: (Jan. 2004). DOI: 10.1007/978-3-642-15970-1_5.

[31] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. «Mitre att&ck: Design and philosophy». In: *Technical report* (2018).

[32] Romilla Syed. «Cybersecurity vulnerability management: A conceptual ontology and cyber intelligence alert system». In: *Information & Management* 57.6 (2020). ISSN: 0378-7206. DOI: https://doi.org/10.1016/j.im.2020.103334. URL: https://www.sciencedirect.com/science/article/pii/S0378720620302718.

[33] Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews, and Anupam Joshi. «UCO: A Unified Cybersecurity Ontology». In: Feb. 2016.

[34] Slim Trabelsi, Henrik Plate, Amine Abida, M. Marouane Ben Aoun, Anis Zouaoui, Chedy Missaoui, Sofien Gharbi, and Alaeddine Ayari. «Mining social networks for software vulnerabilities monitoring». In: *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. 2015, pp. 1–7. DOI: 10.1109/NTMS.2015.7266506.