POLITECNICO DI TORINO

Master's Degree in Computer engineering - cybersecurity

Master's Degree Thesis

Design and prototype validation of a decision support system for cybersecurity incident mitigation

Supervisors

Prof. Cataldo BASILE

Prof. Víctor A. VILLAGRÁ

Candidate

Francesca PONZETTA

July 2022

Abstract

In the last decades the number of cyber attacks against organizations and enterprises has increased. Cyber attacks are becoming more and more sophisticated, new tactics and techniques are emerging, and the number of exploitable vulnerabilities grows proportionally to the number of new electronic and IoT devices. Moreover, the losses caused by a security incident can be huge, but defending from such attacks can also be costly.

Risk management performed in an organization relies on Intrusion Detection Systems (IDS), able to detect incoming attacks or anomalies, and can be complemented with Intrusion Response Systems (IRS), able to ensure the best response to mitigate the identified attack. But how to identify the best response?

Given all the possible response actions to an attack (also known as Courses of Actions), a common thought is to consider the best response as the one that better mitigates the attack, allowing a complete recovery of the compromised asset. But implementing such a response may generate some negative effects: it may require too much time or too many resources, it may affect the organization services (interfering with the normal behaviour of some components or completely denying their usage for an amount of time), or its monetary cost may be excessive, sometimes higher than the monetary loss provoked by the attack.

In general, to balance the impact of the attack and the impact of the chosen countermeasures, instead of looking for a response that completely mitigates the attack, the objective becomes keeping the risk below an acceptable risk level.

In this scenario, a system able to identify the best Course of Actions according to the organization's needs could be a good way to help recovering from a security incident, considering the security requirements of the organization, the available resources to implement the response the potential impact of the chosen response and the level of importance of the compromised asset.

This thesis proposes a decision support system for an automated intrusion response system: when an attack is detected, this system will identify all the possible Courses of Actions able to mitigate the attack. Moreover a set of security metrics will be defined to characterize the potential impact (in terms of negative effects on the system, but also cost of implementation/deployment, time required, etc.) of each Course of Actions in order to infer the best one.

This will be achieved through an architecture that integrates cyber threat intelligence (in particular the Mitre ATT&CK framework will be taken into consideration as a model for cyber adversary behaviour), the Web Ontology Language (OWL) and a rule based semantic reasoner through the usage of Semantic Web Rule Language (SWRL).

A papà Renato,
perchè il mio percorso universitario è stato anche il suo.
A mamma Claudia,
perchè il calore dei suoi abbracci ha superato i 1200km di distanza.
A Lorenzo e Gabriele,
perchè quando ho nostalgia di casa mi basta pensarli e ritorna il sorriso.

"Le nostre rivoluzioni personali, grandi o piccole che siano, iniziano sempre in un posto che abbiamo dentro. Da qualche parte, tra la testa e il cuore."
Gianluca Gotto, *Le coordinate della felicità.*

Table of Contents

List of Figures

Acronyms

IT
Information Technology

CIA
Confidentiality, Integrity, Availability

ISMS
Information Security Management System

IS
Information Security

PDCA
Plan-Do-Check-Act

RoI
Return of Investment

DFD
Data Flow Diagram

DoS
Denial of Service

CTI
Cyber Threat Intelligence

OSINT
Open Source Intelligence

HUMINT

Human Intelligence

FINTEL

Finished Intelligence

CISO

Chief Information Security Officer

TTP

Tactics, Techniques and Procedures

SOC

Security Operation Center

IoC

Indicator Of Compromise

SDO

STIX Domain Objects

SCO

STIX Cyber Observable Objects

SRO

STIX Relationship Objects

HTTPS

Hyper Text Transfer Protocol Secure

URI

Uniform Resource Identifier

DL

Description Logic

XML

eXstensible Markup Language

DSS

Decision Support System

IDSS

Intelligent Decision Support System

CoA

Course of Action

Chapter 1

Introduction

Information security is a critical issue for many companies. Being the subject of an increasing number of cyber incidents, companies rely on Intrusion Detection Systems and Risk Management processes to detect and manage risks associated with the use of information technology. Even if a good preventive analysis is helpful in limiting the number of attacks that the organization could suffer, it is not always possible to avoid security incidents. In those cases, it is necessary to react as soon as possible in order to limit the damage provoked by the attack. But identifying the right response actions to an attack and choosing the best one is not always simple, especially if employees do not have a certain level of cyber security skills. This contribution proposes a Decision Support System capable of helping employees decide which response action is the most suitable to counter an attack. Many organizations already use decision support systems in security operations; the main objective of the decision support system proposed here is to give the opportunity to set some specific objectives that will be considered in the selection of the best Course of Action. Therefore, the optimal course of action is the one that best reflects the organization's needs.

1.1 Objectives

This research work will be carried out trying to achieve the following objectives:

- implement a decision support system based on **ontologies**, **semantic reasoning** and **SWRL rules** for the identification of an optimal Course of Action in response to a cyber attack;
- model a suitable ontology able to define **Alerts** for the notification of ongoing attacks and **Courses of Actions** to mitigates such attacks;

- model a set of **security objectives** that the user of the system may want to enforce for the identification of the optimal Course of Action;
- identify **security metrics** for:
  - the calculation of the parameters that define a Course of Actions;
  - the identification of the optimal Course of Action on the basis of the defined parameters and of the security objectives;
- specify the security metrics in **SWRL rules** to be executed from a semantic reasoner;
- at the end of the execution of the decision support module developed, receive some recommendations about the Course of Action able to mitigate a defined attack which is optimal according to the objectives specified by the user. The user will be provided also with the optimal Course of Action recommended by the Decision Support module (considering what is best for the compromised system, independently from the specified objectives).

1.2 Structure of the document

This document will present the main security processes and tools related to the identification and management of security incidents (from Chapter 2 to Chapter 6), followed by the description of the implemented decision support system. The chapters are organized as follows:

- **Chapter 2** introduces the key terms in cybersecurity and describes the IT security management systems and the risk management process;
- **Chapter 3** describes the cyber threat modelling process and its frameworks;
- **Chapter 4** presents cyber threat intelligence, its fields of application, sharing platforms and sharing standards;
- **Chapter 5** introduces the concepts of ontology and reasoning;
- **Chapter 6** gives an overview about the general concepts of decision making and decision support system;
- **Chapter 7** introduces the architecture of the decision support system implemented;
- **Chapter 8** describes the ontology created for the decision support system;

- **Chapter 9** presents the security metrics defined for the definition of the parameters of a Course of Action and for the identification of the optimal CoA. It also presents the implementation of the security metrics in SWRL rules;
- **Chapter 10**: describes and validates the results obtained;
- **Chapter 11**: summarizes the work done and proposes future lines of investigation.

Chapter 2

Cybersecurity: general introduction

2.1 Introduction to cyberattacks

In the last decades the number of electronic and connected devices used daily has increased. More than 50% of the population is online and despite digital technology is bringing numerous benefits, it is also bringing many issues, such as an increasing feeling of cyber insecurity.
A study conducted by the World Economic Forum (WEF) has shown how cyber attacks, data fraud, and theft are among the top 10 long-term risks. The goal of cyber attacks is to compromise individuals, but in particular businesses, such that the WEF survey considers them the second most concerning risk for doing business globally over the next 10 years [1]. Organized cybercriminal entities are becoming more powerful, cybercrime as a service represents a real business model, and it is becoming more difficult to detect cyber attacks and implement countermeasures to stop or counter them.
Compromising privacy is one of the main goals of cybercrime, and companies (which constitute a collection of confidential data) are one of the preferred targets. Cyber attacks are targeting in particular small and medium sized businesses, causing damage to the IT assets and infrastructure. The most frequent attacks are:

- phishing and social engineering;
- compromised/stolen devices;
- credential theft.

A data breach causes significant expenses to the company (expenses that the company did not plan in advance). Lost data, business disruption, notification

costs and brand reputation damage are among the main costs that the company has to face.[1]
Many businesses organizations, and governments are collaborating in the development of a secure cyberspace, assessing the impact of cyber attacks and creating an incident response. [2].
It is obvious that implementing a type of system is necessary to manage risks and find countermeasures, especially for companies and organizations. The following sections will provide an overview of the basic concepts of cybersecurity and an introduction to the IT security management process and the risk management process.

2.2 Introduction to cybersecurity

The National Institute of Standards and Technology, **NIST**, has defined the term of **cybersecurity** as *"the ability to protect or defend the use of cyberspace from cyber attacks"* where **cyberspace** is a *"global domain within the information environment consisting of the interdependent network of information systems infrastructures including the Internet, telecommunication networks, computer systems and embedded processors and controllers".*
In order to protect a system, three properties have to be ensured:

- **confidentiality** ensures that data or information are not made available or disclosed to unauthorized persons or processes;
- **integrity** is the security goal that generates the requirement for protection against either intentional or accidental attempts to violate **data integrity** (the property that data have not been altered in an unauthorized manner) or **system integrity** (the quality that a system has when it performs its intended function in an unimpaired manner, free from unauthorized manipulation);
- **availability** ensures timely and reliable access to data and information services for authorized users;

These three properties are known as the **CIA triad**.

2.2.1 Terminology

- **Attack**: is a malicious attempt to gain unauthorized access to system services, resources or information or an attempt to compromise a system integrity, availability or confidentiality.
- **Incident**:is a violation or an imminent threat of violation of computer security policies or standard security practices.

- **Vulnerability**: is a weakness in an information system, system security procedures, internal controls or implementation that could be exploited or triggered by a threat source.

- **Threat**: Any circumstance or event with the potential to harm an information system through unauthorized access, destruction, disclosure, modification of data, and/or denial of service. Threats arise from human actions and natural events and exploit vulnerabilities.

- **Countermeasure**: represents every action, devices, procedures or techniques that meet or oppose a threat, a vulnerability or an attack by eliminating or preventing it, by minimizing the harm that can cause, or by discovering and reporting it so that a corrective action can be taken.

- **Asset**: is any data, device or other component of the environment that supports information-related activities. It includes hardware, software, and confidential information. It should be protected against illicit use, as a malicious attack on the asset can result in a loss to the organization[1].

- **Risk**: is the level of impact on an organizational operation, asset or individuals resulting from the operation of an information system given the potential impact of a threat and the likelihood of that threat occurring.[2]

2.3 IT security management system

IT security management is a process that aims to develop and maintain an appropriate level of computer security for the assets of an organization. This is achieved by preserving the confidentiality, integrity, availability, accountability, authenticity and reliability of the asset. [3]
Implementing an **Information Security Management System (ISMS)** provides various benefits to the organization:

- It allows to have a global and central view of the security of the company;

- it is a model that changes over time in order to adapt to the changes of modern information systems and threats;

[1] `https://en.wikipedia.org/wiki/Asset_(computer_security)`

[2] All the definitions provided above are taken from the NIST glossary present at the following url: `https://csrc.nist.gov/glossary`

- giving a precise view of the current state of security, it allows to optimize the allocation of resources in the implementation of the risk management system.[4]

Numerous standards have been created to provide the guidelines for implementing an ISMS. This paragraph will describe how to set up an ISMS according to the standard **ISO/IEC 27001**. The standard considers 14 thematic areas:

1. context and organization;
2. leadership and commitment;
3. IS objectives;
4. IS policy;
5. roles, responsibilities and competences;
6. risk management;
7. performance monitoring & KPIs;
8. documentation;
9. communication;
10. competence and awareness;
11. supplier relationships;
12. internal audit;
13. incident management;
14. continuous improvement.

The following sections will present briefly the key aspects of some of these areas, in particular, the definition of context, IS objectives, policy and risk management process.

2.3.1 Information security objectives and policy

The first step in the implementation of an ISMS is based on analysing the general objectives and policies adopted by the organization in order to extract its **risk profile**. The risk profile allows us to define the **scope** in which the ISMS will be implemented and the **IT security policy** to be adopted. In defining the

scope, **internal context**, **external context**[3] and the answer to some questions (such as: what tasks of the organization require IT support? Which decisions are based on the security concepts of CIA? What areas may damage the organization if they become the subject of an IT security incident?) should be taken into consideration. Defining the scope at the beginning of the process allows one to better estimate the feasibility and the amount of work to be done to setup the ISMS [5].

The IT security policy defines in particular the security objectives to be achieved, the security strategies to meet these objectives, the roles and responsibilities regarding the ISMS, the risk management approach to be adopted, etc. [3] The company objectives are strictly connected to the IS objectives, such that failing to achieve the IS objectives will have a negative impact in achieving the company objectives. For this reason, it is important to define them properly: they must be aligned with the IS policy, they should be regularly reviewed and integrated in the business process and they should be considered from the beginning to ensure the principle of **security by design**.[5]

2.3.2 Risk management

Risk management is a key point in the ISMS. It consists in analyzing anything that could happen together with the potential impact on the organization. The main objectives of risk management are:

- early identification and elimination of security risks;
- establishing assessment methods for the identified risks;
- clear assignment of responsibilities when dealing with risks;
- clear and standardized documentation of risks including their assessment;
- efficient treatment of risks.

Possible risks for an organization may arise from the exchange of data with the outside, changes in the internal organization, missing updates of the systems and applications used, cooperation with external partners, remote access to the company network, humans as a risk (social engineering), etc.[5]
Ideally, risk management should examine all the assets of the organization and evaluate all possible risks in order to deploy the best countermeasure to reduce risk.

[3]Internal context: relationships between the various departments inside the organization. External context: relationships with external organizations, partners or service providers

In reality, this is not feasible, mainly because IT technologies and threat strategies change rapidly, and the risk assessment could not adapt in time to these changes. Furthermore, the budget that the organization reserves for risk management may not be enough to deal with all the risks that have been detected.
As it is not possible to completely eliminate risks, it is necessary to define an acceptable level of risk. This means that there may be risks that are not considered extremely dangerous or risks where the cost of their impact is lower than the cost of implementing a suitable countermeasure. In this case, the organization can decide not to take any action to eliminate these risks.[3]

2.3.3 Incident management

Once detected the possible risks, an **IS incident management** must be designed. As incidents can have negative impacts on the organization, it is necessary to define in advance an efficient action that can deal with a specific incident if it occurs. Incidents must first be categorized according to their degree of severity (some of them may have a higher priority), then some predefined actions may be implemented. An incident response plan like the one described in the standard ISO/IEC 27035 can be followed as a guideline to deal with security incidents. This plan is iterative and consists of five main actions:

1. **planning and preparation**: design preventive measures;
2. **identifying and adopting**: report security incidents to a central reporting authority;
3. **classifying and deciding**: the reporting authority classifies the incident reported as:
   - Security incident;
   - known error: incident not related to security for which a solution already exists;
   - contingency: incident for which there is a contingency plan in place.

   Incident reports must be properly documented. Once the security incidents are classified, their priorities have to be determined;
4. **incident response**: consists in reacting to the IS incident in accordance with the agreed procedures:
   - containment and initial security of evidence: analysis of the spread of the incident;

- resolution and recovery: measures to restore the desired configuration;
- root case analysis and securing of evidence: determination of the root cause of the incident;

5. **lesson learned/Follow-Up**: the security incidents must be properly documented and subjected to audits in order to see if it possible to improve the way how they are handled.

2.3.4 Internal audit and continuous improvement

When developing an ISMS it is important to schedule some monitoring and review operations to evaluate whether the system meets both the organization's requirements and the standard's requirements. Moreover, the correct implementation and effectiveness of the security measures taken must be reviewed too, because during time they could suffer modifications caused by threats and incidents.
This monitoring is done through an audit program. When planning the audits, it is important to choose the right metrics that define the audit actions: the more the metrics are objective and meaningful, the more it is easy for the analysis to be done, giving useful results.
Since the audit may report some inconsistencies in the ISMS and since it is obvious that the perfect ISMS is impossible to be developed, some actions to allow a continuous improvement of the system must be taken into consideration. Continuous improvement can be done following some iterative methodologies, the one defined by the standard ISO/IEC 27001 is the **PDCA (Plan-Do-Check-Act)**. This approach consists of four phases:

1. **plan**: it establishes the control objectives, the security measures to achieve the control objectives, the performance indicators (to measure the performance against the control objectives) and it defines some corrective actions to keep a security measure within the normal range;

2. **do**: it measures if the objectives are achieved and it implements corrective actions if flaws are identified;

3. **check**: it monitors the implementation of the security measures (checking if they meets the control objectives) and it creates reports to be used for the internal audit;

4. **act**: it makes management decisions to restore the effectiveness of security measures in order to allow continuous improvement.[4] [5]

2.4 Risk management

Following the family of standards ISO/IEC 27000, the risk management process has been defined in detail by **ISO/IEC 27005**. According to this standard, risk management is a process that follows 6 steps. Figure 2.1 illustrates the risk management process and a detailed description of the different activities is provided in the following sections of this paragraph.[6]

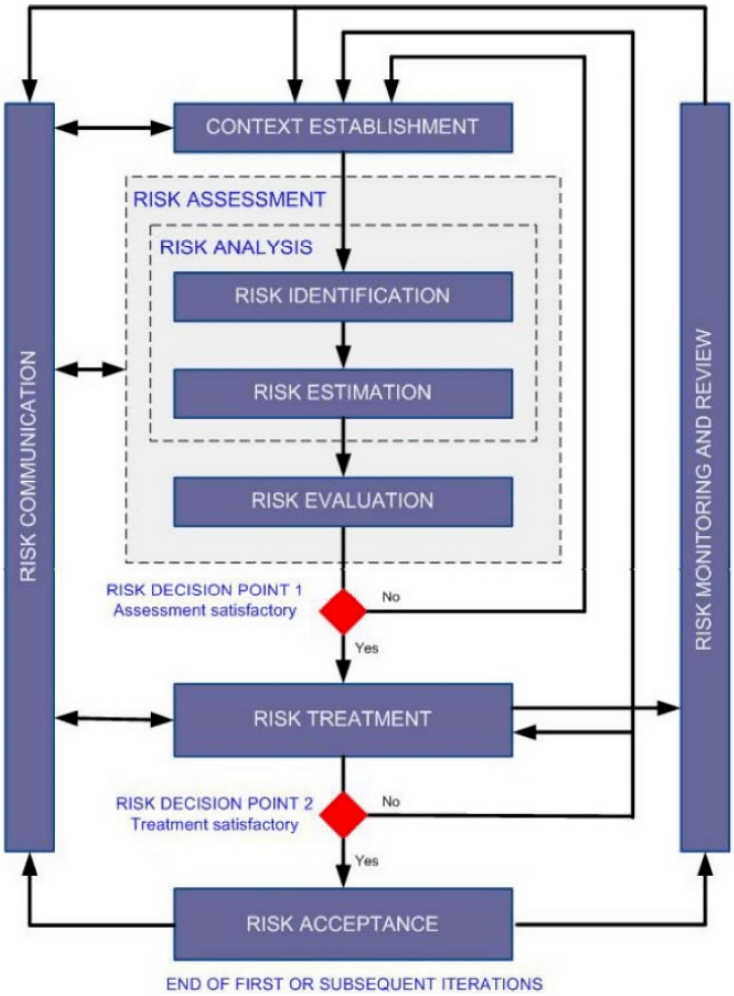

Figure 2.1: IS risk management process according to ISO/IEC 27005:2011

2.4.1 Context establishment

The first step in the iterative process is to establish the context which means determining the purpose of the IS risk management. Establishing the context strictly depends on the ISMS. Different approaches may be followed:

- **Risk management approach**: the risk management has to address basic criteria, such as:
  - risk evaluation criteria: it consists in evaluating the organization's IS risks considering the business process, the criticality of the assets involved, and the legal requirements;
  - impact criteria: it is developed and specified in terms of the degree of damage or costs the organization must face in case of security incidents;
  - risk acceptance criteria: it depends on the organization's policies, objectives and requirements. The organization has to define a precise level of risk acceptance.
- **Scope and boundaries**: the risk management should be developed to a precise scope. This needs to be defined properly to ensure that all relevant assets are taken into consideration
- **Organization for IS security risk management**: the main roles and responsibilities inside the IS risk management system have to be specified.

2.4.2 Information Security risk assessment

A risk can be defined as the consequences (costs) caused by the occurrence of a security incident and the likelihood of the occurrence of that incident. Risk assessment is a process that allows to describe qualitatively and quantitatively a risk and allows to order the risks according to their severity.
It may be an iterative process and consists of the following activities:

1. **risk identification**: it's the process of identifying what events can cause a potential loss and how, where and why the loss might happen. Risk identification should include risks whether or not their source is under the control of the organization, even if the source is not evident. We can consider several types of identifications:
   - asset identification;
   - threat identification;
   - existing controls identification;

- vulnerabilities identification;
- consequences identification;

2. **risk analysis**: two methodologies can be used:

   - qualitative risk analysis: high level analysis, it assigns an attribute to describe the potential consequences and the likelihood (high, medium, low) of a risk;
   - quantitative risk analysis: more precise, it assigns a numeric value to consequences and likelihood using data from various sources.

   Risk analysis executes the following tasks:

   - **assessment of consequences**: it consists in determining the impact on the organization (e.g. costs of recovery, business loss, severity of the compromised assets);
   - **assessment of incident likelihood**: the likelihood is calculated considering data such as how often a threat occurs or how easily a vulnerability can be exploited;
   - **determination of the level of risk**: it is a combination of consequences, likelihood, cost benefit, concerns of stakeholders and other variables related to the risk scenario;

3. **risk evaluation**: It associates a risk level to every risk. and according to this level, more or less priority will be given in the risk treatment phase. The following decisions (on how to manage the risk) depend on the risk level. The risk level is calculated considering parameters such as the likelihood, the consequences, the degree of confidence in the risk identification, the analysis, etc.

2.4.3 Information security risk treatment

First, a risk treatment plan should be defined: it consists of deciding the priority order in which individual risk treatment should be implemented. Priorities are established using various techniques, such as risk ranking according to the risk level or cost-benefit analysis.
Risk treatment provides four possible actions:

- **risk modification**: it consists in selecting the appropriate controls able to correct, eliminate, prevent, minimize the impact, detect, monitor. When

selecting controls, various parameters such as costs of acquisition, implementation, administration, operation, monitoring and maintenance, plus the return of investment (RoI)[4] should be considered;

- **risk retention**: if the risk level meets the risk acceptance criteria, then there's no need to implement additional controls and the risk can be retained;
- **risk avoidance**: it consists in avoiding the activity that creates the risk. This may be done if, for example, the risk is high and the cost of implementing a risk treatment exceeds the benefits;
- **risk sharing**: the risk may be shared with another party that can manage it more effectively (for example an insurance organization).

Once the treatment plan has been defined, the **residual risks**[5] need to be determined. This consists in updating and re-iterating the risk assessment process taking into account the expected effects of the proposed risk treatment. If the residual risk still does not meet the organization risk acceptance criteria, a further iteration of the risk treatment is necessary.

2.4.4 Information security risk acceptance

Risk treatment plans also consider the possibility to accept certain kinds of risk. The risk acceptance criteria are complex, they do not consist only in checking if the risk level falls above or below a single threshold. Furthermore, all decisions to accept the risk and the responsibilities of this decision should be formally recorded.

2.4.5 Information security risk communication and consultation

Information about risks should be shared between the decision-makers or other stakeholders to simplify future decisions in case of similar security incidents.

2.4.6 Information security risk monitoring and review

- **Monitoring and review of risk factors**: as threats, vulnerabilities, values of likelihood and consequences are not static, they should be monitored because their changes could modify the risk level

[4]RoI is a performance measure used to evaluate the efficiency or profitability of an investment. It tries to directly measure the amount of return on a particular investment, relative to the investment cost. Source: `https://www.investopedia.com/terms/r/returnoninvestment.asp`

[5]Residual risk is the risk remaining after risk treatment

- **Risk management monitoring, review and improvement**: monitoring is necessary to ensure that the context, the outcome of the risk assessment and risk treatment and the management plan remain relevant and consistent with the organization's and business' objectives.[6]

Chapter 3

Cyber Threat Modeling: process and common frameworks

In the previous chapter, we have discussed the importance of implementing an ISMS and executing a risk management process. Risk management is crucial to securing the IT infrastructure of an organization, but it is not enough. For this reason, it is combined with the threat modeling process, which addresses threats and attacks in greater detail, answering questions such as who will attack the system, how the attack will be deployed, and outlining possible security measures and controls to stop the threat before it damages the system. Cyber Threat Modeling is strictly related to Cyber Threat Intelligence, such that it is not easy to define where is the line that separates them. Figure 3.1 shows their relationship.

3.1 Cyber Threat Modeling

Threat modeling is a ***strategic process***[1] *aimed at considering possible* ***attack*** *scenarios and* ***vulnerabilities*** *within a proposed or existing* ***application environment*** *for the purpose of clearly identifying* ***risk*** *and* ***impact*** *levels.*[7]
Threat modeling addresses risks with higher precision with respect to risk assessment, because it provides a clear view on how the organization can be compromised and with which probability. By dividing each threat into multiple attacks, it allows to understand easily how each attack scenario is deployed and which is its impact

[1] *Refers to the ability of anticipating threats via simulated attack patterns*

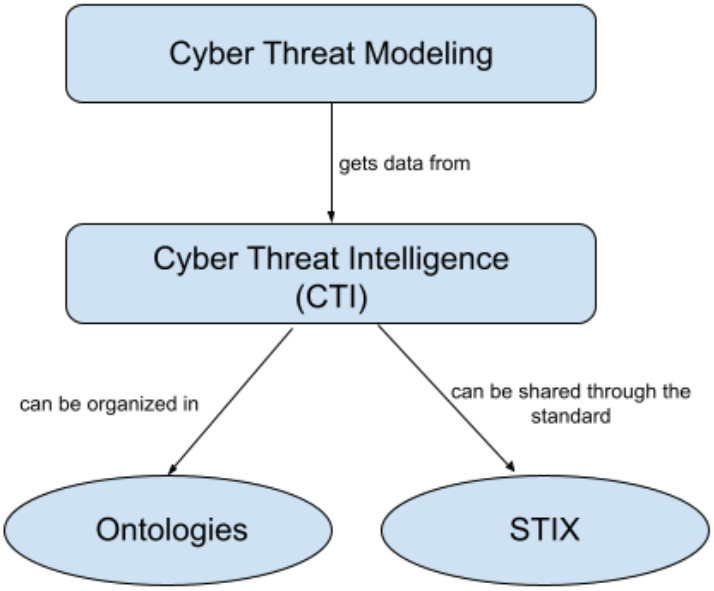

Figure 3.1: Relation between Cyber Threat Modeling and Threat Intelligence

on the organization, allowing to prioritize the mitigation techniques.
Among the advantages of using threat modeling, we can consider the followings:

- it allows finding security bugs early;
- it is a tool for understanding the security requirements better (allowing the identification of which threats are too complex or expensive to be addressed, which ones are not in line with the security requirements and which ones are not taken into account by the requirements);
- it allows the engineering and delivery of a better product (avoiding the need to re-design the system):
- it allows the address of issues that other technologies will not address.[8]

Brainstorming can be a good starting point for creating a threat model. Three different approaches may be followed:

- *Asset-focused*: consists of making a list of assets and then considering how they can be threaten. Possible assets include:

  - things that attackers want, such as user passwords or keys;
  - things you want to protect, such as the reputation of the organization;
  - stepping stones: all the assets already identified may be attacked and exploited to reach other assets.

- *Attacker-focused*: consists of making a list of possible attackers and trying to think like them to understand how the system can be threatened. This approach is risky because engineers may project their knowledge in modeling how an attacker would act, creating a model that does not reflect the actual possible threats.

- *Software-focused*: it is the most efficient approach. Developers (which have the highest knowledge of the software being built) participate actively to help creating a model (usually through diagrams) of the software and identifying possible bugs.[9]

3.1.1 Threat modeling steps

The Threat Modeling Process answers four essential questions:

1. What is the system that could be affected?
2. What can go wrong?
3. What are we going to do about those things that can go wrong?
4. Did we do a good job of analysis?

These questions involve a four-steps approach:

1. Model the system
2. Identify the threats
3. Address the threats
4. Validate the threat model.[10]

3.1.2 Model the system

The first step consists in understanding the system with which we are working. This can be done through **decomposition**, a process that allows to gain knowledge about how the system works and how it interacts with external entities. Decomposition includes:

- the creation of *use-cases*, to identify how the system is used;
- the identification of *entry points*, which allows an attacker to interact with the system;
- the identification of *assets* in which an attacker may be interested;
- the discovery of *actors*, users that interact with the system; they can be internal or external, and should receive some access rights;
- the identification of *trust levels*, which will determine the access rights for external entities.

One of the techniques for decomposing a system is the creation of a **Data Flow Diagram (DFD)**. This type of diagram was introduced in the 1970s to give a visual representation of how data moves from a component to another in a system or an application and where data is modified or stored (temporarly or long term) inside the system. The DFD clearly identifies the *External Entities*, the *End Points* of the system, the *processes*, the *units of functionality*, the *Data Flow (DF)*, and the *Data Store (DS)*. Later, in the early 2000s, the concept of **trust boundaries** was added to improve the DFDs. A trust boundary is a location in the DFD where data changes its level of trust. Trust boundaries are used to isolate trustworthy and untrustworthy elements (for example, if two processes are processing some data, they are divided by a trust boundary).[10][11]

3.1.3 Identify the threats

This is the core element of threat modeling: threats and threat agents[2] may be identified following different **frameworks** or methodologies. Some of these methodologies are:

- **STRIDE**
- **Attack threes**
- **PASTA**
- **CVSS**
- **Security Cards**.

Some of them will be described in more detail in the next section.

[2]Threat agent: individual or group interested in exploiting a vulnerability and carrying out a threat against the asset

3.1.4 Address the threats

Once identified the threats, it is necessary to understand how to deal with them, which ones are more urgent to be addressed, and which security countermeasures are needed to mitigate their impact.
A useful technique is to create a **threat traceability matrix**: the attacks are enumerated on the base of the danger associated with them. Taking into account the level of risk associated with each vulnerability during the risk management process, threats are classified from most severe to less severe, and stakeholders and risk owners can analyze them to find the appropriate countermeasures and mitigation techniques. The risks will be treated according to what is defined in the risk management process.
A common framework used to evaluate the existing vulnerabilities is **DREAD**.[12]

3.1.5 Validate the threat model

This last step consists of checking if the threat model is complete (if it identifies all possible threats) and if all threats are adequately mitigated. For threats that have not been completely mitigated, the residual risk is calculated and analyzed (is the residual risk consistent with the acceptable level of risk?).[13]
A common strategy for validating the threat model is using **tests**. Tests can be automatic or manual and different techniques may be applied. For example, **penetration testing** may be used to assess the vulnerabilities of the system (the vulnerabilities found are compared to the ones identified by the threat model), or the mitigation techniques identified by the model are validated by simulating an attack and applying those techniques to check if they are really able to mitigate the attack.[14]

3.2 Frameworks used in Cyber Threat Modeling

3.2.1 STRIDE

It was created by Microsoft for three main purposes:

1. as a systematic approach to analyze the possible cyber threats against each system component based on its technical knowledge;

2. to provide a comprehensive analysis of the security properties;

3. to identify the impact of a component vulnerability on the entire system.[11]

STRIDE is the acronym for the type of threats it covers:

- **Spoofing**: violation of the authentication property. The attacker pretends to be someone else, such as a process, an external entity, or a person and compromises something inside the system. A common scenario may be: the attacker exploits a weak authentication system (for example, by sniffing the key from APIs that use single-key authentication requests). Once the key is stolen and access to the system is gained, the attacker pretends to be an innocuous process and creates or maliciously modifies a file before the real process.

- **Tampering**: violation of integrity property. The attacker modifies something (a file, the code, or some data) in an unauthorized way. This attack could be detected by checking log files and notifications.

- **Repudiation**: violation of the property of non-repudiation. It ensures that bad behavior cannot be proven. Some non-repudiation mechanisms could be auditing and tracing (but always considering that also tracing files could be tampered).

- **Information disclosure**: violation of the property of confidentiality. Some confidential information could be accidentally disclosed (for example, through error messages) or be exposed to an attack (such as buffer overflow).

- **Denial of Service (DoS)**: violation of the property of availability. A system becomes unreachable by exploiting its resources maliciously and preventing it from being used for legitimate purposes. Some examples may be DoS affecting processes (the attack absorbs memory or CPU and the process is not able to run) or databases (they are filled with useless information and are not able to receive useful data).

- **Elevation of privilege**: violation of the property of authorization. The attacker claims to be an authorized user with high privileges (such as admin instead of a common user). For example, by corrupting a process, the attacker may gain read or write access rights to some sensitive memory locations.[15][16]

Once DFD is created, the STRIDE-based threat modeling can be performed in two ways:

- **STRIDE per-element**: for each threat covered by STRIDE, every component of the system is analyzed to check if it may be subject to this threat.

- **STRIDE-per-interaction**: the system components are considered in tuples (origin, destination, and interaction) and their interaction is analyzed to check if it may be subject to one or more threats covered by STRIDE.[11]

3.2.2 Attack trees

An attack tree is a conceptual diagram that is used to describe the security of a system. In particular, in threat modeling, it describes threats and the possible attack paths to realize those threats. The **root** of the tree represents the attack goal, while the various **leaf nodes** represent the different ways of achieving the goal. Each leaf node is connected to the root through a **path of nodes** which represent the subgoals that must be achieved to reach the root (completing the attack). The various nodes may receive a **value** (possible or impossible, easy or difficult, expensive or inexpensive, etc.). The node values can be combined to have a more complete scenario of the vulnerabilities of the system, allowing to distinguish between different attacks (for example, the cheapest attack with the highest probability of success or the best low-skill attack, etc.). Figure 3.2 provides a simple example of an attack tree designed to show how to gain access to a database. To create a good attack tree, it could be useful considering the different types of attackers, their knowledge and capabilities, and the tools they could use. In addition, what-if scenarios and possible countermeasures could be considered.
The procedure to create an attack tree is the following:

1. Decide a representation of the attack three: there are **AND trees** (the state of a node is true if all the nodes below it are true) and **OR trees** (the state of a node is true if any of its nodes is true).

2. Create the **root node**: this corresponds to the identification of the possible **attack goals**: to each goal corresponds a different tree. In threat modeling, multiple trees are created for a single system.

3. Create the **subnodes**: with the actions that allow to fulfill the attack goal (they are connected by AND or OR relationship and may be identified through brainstorming).

4. Determine if the set of attack trees is complete. This may be done by iterating on the different nodes and checking if there are other ways of performing the same attack. STRIDE or attack libraries can be used.

5. Prune the tree: there may be multiple nodes that represent the same action, or there are actions that may never happen because mitigation techniques have already been implemented. The corresponding nodes should be marked to specify that they do not need to be analysed.[17][18]

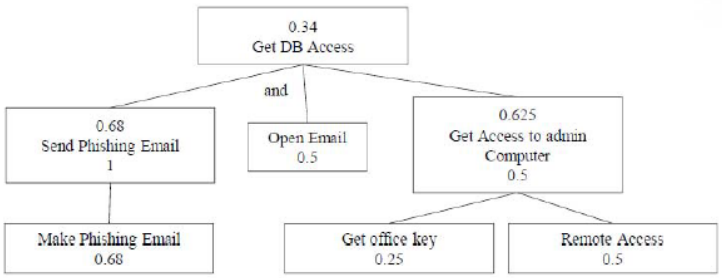

Figure 3.2: Example of an attack tree[19]

3.2.3 PASTA

PASTA stands for **Process of Attack Simulation and Threat Analysis**. It is a **risk-centric** threat modeling methodology and was co-founded in 2015 by VerSprite[3] CEO Tont UcedaVélez and security leader Marco M. Morana.
PASTA provides an accurate threat model, considering both business and IT stakeholders and allowing one to have a complete view of the risks of a system, the likelihood of attacks, and the business impact in case an attack is performed.[20]
The **seven stages** at the base of this framework are:

1. **Define the objectives**: after defining the business objectives (which may be internally or externally driven), the people in charge of Information Security have to understand those objectives and support them with technology and security measures (for example, if the business requirement is *high availability for the application to process simultaneous requests*, the corresponding security requirements will be *DoS mitigation techniques, and High Availability architecture*).

2. **Define the Technical Scope**: consists of defining the attack surface. This is done by identifying all the IT assets (platforms/systems, relevant databases, application servers, service configurations, API endpoints, etc.) that are necessary to fulfill the requirements defined in the previous stage. For understanding what to protect, it is necessary to consider also the dependencies with third party services and which technologies are being used.

3. **Decompose the application**: starting from the technical scope identified in the previous stage, a Data Flow Diagram is built to provide a map of how

[3]Cybersecurity consulting firm

information moves around the system and across the trust boundaries. This is a good starting point for the subsequent threat analysis.

4. **Analyse the threats**: the overall threat scenario is analysed, many different sources of data (threat intelligence gathered from internal and external sources) are considered, the threat agents that are more likely to perform an attack are enumerated and the most likely and relevant threats identified, together with a value of likelihood. In order to create a good threat model, credible evidence of the possible threats must be provided. The threats identified will be collected in a **threat library** and all the information gathered are used to build threat trees.

5. **Vulnerability Analysis**: in this stage all possible flaws, weaknesses, and vulnerabilities (of the code, of the system design, and of the trust model developed so far) are identified. They are then mapped to the threats identified in the previous step.

6. **Attack analysis**: the vulnerabilities found in the previous step must be proven to be viable. Each of them is mapped to a node in the **attack trees** in order to determine its likelihood.

7. **Risk and Impact Analysis**: this last stage is about identifying the countermeasures able to mitigate threats. The threat model is finalized by reviewing all the information collected so far and calculating the **impact** of each threat (this is done by simulating the corresponding attack and considering possible countermeasures). Knowing the impact of each threat allows us to take security decisions (such as where to implement mitigation techniques) that will help the organization reduce the risk and save time and money.[20][21]

3.2.4 DREAD

DREAD is a threat model framework that is used to assess threats, determine their relative priorities, and define their mitigation strategies. DREAD was developed by Microsoft and was published for the first time in the second edition of *Writing Secure Code* in 2002 by David LeBlanc and Michael Howard.[22]
The term DREAD is the acronym of the five criteria used to assess threats to software:

- **Damage**: corresponds to the most critical part of threat modeling. It relies on considering every type of damage (data loss, hardware failure, performance loss, business reputation loss, and every other similar measure applied to the environment considered).

- **Reproducibility**: indicates how often the same attack will succeed. If a vulnerability is easily exploited, it is more likely that a threat to this vulnerability exists.
- **Exploitability**: assesses the efforts and knowledge needed to carry out an attack. A threat that can be attacked without a great level of expertise is more likely to be exploited.
- **Affected users**: it identifies the number of users that could be affected by an attack. If the number of affected users is low, this measure is given a low value. On the other hand, if an attack affects many users (for example, a DoS which affects thousands of users at the same time), this measure receives a high value.
- **Discoverability**: it is the likelihood that a vulnerability will be exploited and a threat implemented against it. It is not easy to assess this value: the safest approach is to consider that any threat will be discovered and exploited eventually. As a consequence, the ranking of threats mainly relies on other measures.

The measures described are values from 1 to 10. Once their values have been assessed, an average is calculated by adding all values and dividing by 5 (number of criteria). The result is a numerical score between 1 and 10 for each threat and is used to classify threats (a high score indicates a serious threat, a low score indicates a less important threat).[23]

Chapter 4

Cyber Threat Intelligence

If we think about the processes described so far (risk management, threat modeling, etc.) in the context of a company, it is evident that the security operation teams have to deal with many challenges: continuously checking the state of the system for threats or vulnerabilities detection, analyzing a large volume of alerts coming from Intrusion Detection Systems and filtering the ones that are truly valuable, evaluating which threats have to be managed, in which order of importance, how should they be treated, how much time and how many resources are necessary to answer to a security incident, etc.
Many organizations have started to collect relevant information about threats and actors, with the objective of providing intelligence for effective cyber security decisions. This intelligence, called Cyber Threat Intelligence (CTI), has also been collected in platforms that can be purchased by organizations for supporting their cyber security operations.

4.1 CTI: What is and how it is produced

As defined by Gartner[1]:
"Cyber Threat Intelligence is evidence-based knowledge, including context, mechanisms, indicators, implications and action-oriented advice about an existing threat or hazard to assets. This intelligence can be used to inform decisions regarding the subject's response to that menace or hazard."[24]

[1]Gartner is a technological research and consulting firm based in Stamford, Connecticut that conducts research on technology and shares this research both through private consulting as well as executive programs and conferences.

In simple terms, CTI is a collection of information about threats and cyber attacks (occurred in the past or that have not occurred yet), and it can be extremely useful for an organization to improve its cybersecurity infrastructure and to deal with future cyber attacks. For example, the security team of a company may use CTI to gain information about what are the most concerning threats or the most common attacks against similar companies (same business area or similar company structure) and how those companies have dealt with particular attacks. Having this knowledge allows the security team to have a better view of what kind of defensive strategies to implement without wasting too much time in research and planning. But how does cyber threat intelligence be produced?

4.1.1 Threat Intelligence lifecycle

Threat intelligence is the product of a six-phase cycle of data collection, processing, and analysis. The six phases are:

1. Direction
2. Collection
3. Processing
4. Analysis
5. Dissemination
6. Feedback.[25]

Planning and Direction

The first phase requires setting the objectives of the threat intelligence program. The high level needs are defined (which consist of a deep understanding of what is the asset that needs protection, what the consequences are if an incident occurs, what type of threat intelligence the organization requires, and the priorities about what to protect). Then the organization can formulate more precise questions to create concrete requirements.[26]
It is important to consider who will benefit from the threat intelligence product (for example, if it will be used by security teams with technical expertise to deal with cyber attacks or by executives that only need a broad overview of the security trends for their investment decisions). [25]

Collection

Data gathering is performed following the requirements established in the previous phase. It is important to collect data from a wide range of sources in order to provide a more comprehensive product. The sources can be divided into **internal** (such as IDS logs) and **external** (such as data coming from cybersecurity blogs, dark web,etc.).[26]
It is possible to distinguish five major categories of CTI data sources:

1. **Internal Intelligence**: is the most common and traditional CTI data source and it provides significant information about the organization. Data consists of network logs collected through packet sniffers, ports scans, vulnerability assessments, and other network devices on the network of the organization.

2. **Open Source Intelligence (OSINT)**: comes from external sources of data (traditional social media sites, but also Darknet with its hacker forums, Darknet Marketplace, etc.). These sources provide an overview of the cyberthreats that may exist within relevant industries.

3. **Human Intelligence (HUMINT)**: data can come from both internal and external sources. Precise knowledge about specific threats is collected from manual researches (e.g. direct interaction with hackers).

4. **Counter Intelligence**: data can come from both internal and external sources. Data collection is based on approaches that allow to interact safely with the attacker (such as through honeypots) with the goal of identifying tools and methods used during attacks.

5. **Finished Intelligence (FINTEL)**: data can come from both internal and external sources. It consists in finished intelligence ready for dissemination (e.g. it may come from commercial data).[27]

Processing

Processing consists of transforming the information collected into a format that can be understood and used by the organization. All the raw data need to be sorted and organized into metadata tags and filtered (it is very common to have redundant data or false positives/negatives). Since data are collected through different methods, there are different means of processing them.
In this phase, automation is extremely important: if we think, for example, that small organizations collect data from millions of log events every day, it becomes obvious that security analysts cannot afford this work load. In this scenario, tools (such as machine learning) may help in the classification of large amounts of data.[25]

Analysis

Analysis is a human process that transforms processed information into threat intelligence. The way how threat intelligence is extracted depends on the goal defined at the beginning of the process and on who will receive and use this intelligence. In general, the goal is to get the data into a format that the audience can understand. For this reason, it is important to be concise, to avoid confusing technical terms, to articulate the issues in business terms and include a recommended course of action. Depending on the audience, it may be necessary to deliver intelligence in different formats, but it is important for security teams to be able to exchange CTI successfully.[26]

Dissemination

Dissemination consists of distributing the finished intelligence to its intended consumers. To maintain some continuity between one CTI cycle and the next one, it is necessary to track the intelligence produced so far. This may be done through a ticketing system (tickets may be submitted for each step of the intelligence cycle and reviewed by different teams).[25]

Feedback

It is the last phase of the cycle: the security team that requested the CTI with the specified requirements is now in charge of reviewing the intelligence product received and checking if it is compliant with the request. Receiving feedbacks is extremely important for driving the objectives of the next intelligence cycle.[26]

4.1.2 Types of threat intelligence

Based on the requirements specified in the Direction phase and on the criteria followed in the process of creation of the CTI product, it is possible to break down threat intelligence into three categories:

1. Strategic
2. Tactical
3. Operational

Strategic Threat Intelligence

Strategic threat intelligence provides high-level information about threats, threat teams, current or future cyber risks, the financial impact of a security incident on a

company, etc. that may interest the vital asset of organizations (IT infrastructure, employees, customers, and applications).
It is usually intended for the high-level executives of a company (such as the IT management and the CISO[2]) to make business decisions considering the risks associated with those decisions, as well as the current tactics and targets of the threat actors.
This type of threat intelligence is less technical and it is presented in the form of reports and briefings. It is produced by collecting data from multiple sources (such as OSINT, CTI vendors, policy documents from nation states or nongovernmental organizations, papers and researches produced by security organizations, etc.).[25][29]

Tactical Threat Intelligence

Tactical Threat Intelligence outlines the **Tactics, Techniques and Procedures (TTP)** used to carry out attacks. It allows the security personnel of the organization a deep understanding of which area of the organization is more likely to be attacked, which preventive measures should be taken to avoid such attacks, what the capabilities of the attackers are, and what are the current attack vectors.
Since it is more technical, this intelligence is intended for cyber security professionals (IT managers, security architects, etc.). Among the sources there are security vendors reports, malware incident reports, attack group reports, and human intelligence.
It is mainly used to help improve the security controls and help in the incident response process of a company. For this reason, it is crucial to integrate internal data (e.g., within the organization's network).[25][29]

Operational threat intelligence

Operational threat intelligence collects knowledge about specific cyber attacks that may compromise the organization. In particular, it is intended to point out the potential risks and the vulnerable assets of an organization, the goals and the methodologies used by attackers, and their capabilities.
Since it may provide also more technical information, such as the resources used by attackers (in terms of tools, command and control channels, vulnerabilities exploited,etc.), it is also considered as **Technical Threat Intelligence**.

[2]The CISO (Chief Information Security Officer) is a senior-level executive responsible for developing and implementing an information security program, which includes procedures and policies designed to protect enterprise communications, systems and assets from both internal and external threats.[28]

Common sources for obtaining operational threat intelligence are humans or social media. Knowledge may come from analyzing human behavior or threat teams. On the other hand, sources for technical threat intelligence come from technical information (such as threat data feeds which focus on indicators like malware hashes or suspicious domains). [25][29]

4.2 CTI: fields of application

Threat intelligence represents an important resource for organizations, not only because it helps preventing attacks, but also because it can be applied in other security processes and operations (it helps filtering false positive/false negative alerts, it may be part of risk analysis, vulnerability management and it may help in decision making for responding to cyber intrusion). In the following part of this section, a brief description of the various fields of application of Cyber Threat Intelligence will be provided.[25]

4.2.1 Security Operations

Monitors and threat detection tools used by organizations to detect anomalous or suspicious behaviors in their network produce large volumes of alarms and security alerts, which should then be analyzed and prioritized by security analysts or by the **Security Operation Center (SOC)**. However, considering the high number of alerts produced, it is impossible for analysts to manage all of them as they should, and many of them are ignored.
CTI enriches the context of alerts by adding relevant information which help SOC analysts perform triage by quickly identifying the most significant alerts (ignoring the ones related to incidents that do not represent a relevant problem for the company or for which control and defense measures have already been developed). [30]

4.2.2 Incident Response

The huge number of cyber incidents, the higher complexity of threats, and the need to analyze data coming from too many resources, place incident response teams under unbearable pressure, making it difficult to properly manage and respond to cyber incidents.
CTI helps incident response teams: false positive alerts are automatically identified and not taken into account; moreover, alerts are enriched with real-time context from external sources (such as the dark web), enabling prioritization of threats according to the organization's specific needs. Thanks to CTI, organizations can apply a proactive approach: possible threats can be identified in advance, enabling

incident response teams to develop a response plan before the incident occurs and to make faster and better decisions.[31]

4.2.3 Vulnerability Management

One thing to be highlighted in vulnerability management is that not all vulnerabilities need to be patched. Research has shown that

- Zero-day vulnerabilities will not necessarely be exploited: even if new vulnerabilities are discovered, many new threats are simply variations of existing threats, continuing to exploit the same vulnerabilities;
- The avarage time needed to develop an exploit for a new vulnerability is approximately 15 days. As a consequence: security teams should speed up the process of patching vulnerabilities, if they do not have a patch in 15 days, they should develop some mitigation technique. Moreover, if a vulnerability is not exploited within two weeks to three months, it is unlikely that it will be exploited in the future;
- Ranking vulnerabilities according to the severity of a threat[3] may be misleading, because they do not consider real scenarios of exploitation (for example, if currently performed attacks are exploiting those vulnerabilities).

Threat intelligence helps identify vulnerabilities that really represent risks to the organization and provides information about the likelihood of exploitation by combining knowledge from internal vulnerability scanning data and additional external context (such as TTPs of threat actors).[32]

4.2.4 Risk Analysis

Risk analysis is used for risk assessment, but also to estimate how much to invest in cybersecurity and with which priority. However, usually these estimations are not precise. Threat intelligence provides the additional context necessary to make precise risk measurements and predictions based on transparent assumptions, variables, and outcomes. It may be useful posing questions such as: which are the attacks that are currently compromising enterprises in our sector? Do they exploit the vulnerabilities present in our company? What is the damage provoked in those enterprises?[25]
To help creating a proper risk model, the **FAIR (Factor Analysis of Information Risk)** framework has been developed. [33]

[3] as we can see in CVE (Common Vulnerabilities and Exposures) and CVSSs (Common Vulnerability Scoring Systems)

4.2.5 Fraud Prevention

Many of the existing attacks compromise the confidentiality of data and may lead to the theft of money or identity using techniques such as phishing or social engineering. Organizations need protection from fraudulent uses of their data or brand. This is another field where cyber threat intelligence plays a fundamental role because it helps prevent:

- Payment fraud
- Compromised data: monitoring sources where criminals upload caches of usernames and passwords helps identifying the leak of credentials of corporate data
- Typosquatting[4]: monitoring newly registered phishing and typosquatting domains prevents criminals from impersonating the domain of the organization.[25]

4.2.6 Support Security Leaders

CTI provides an overall picture of what is the current threat scenario: which are the most common threats, what assets and types of companies they are targeting, what are the consequences of these attacks (in terms of monetary loss), what courses of actions have the compromised companies taken to respond to the security incident, etc.
This knowledge helps security leaders to perform risk assessment, identify strategies to mitigate the identified risks, create a business plan to manage them, and justify particular investments in defensive measures.[25]

4.3 ICT: Frameworks and Sharing Platforms

4.3.1 FAIR

FAIR (Factor Analysis of Information Risk) is a framework that can be used by organizations for economically driven cyber risk management, since it associates risk with specific probabilities and monetary losses.
To identify the risk scenario, FAIR starts by considering four key components:

[4]Typosquatting is a type of social engineering attack which targets internet users who incorrectly type a URL into their web browser rather than using a search engine. Users may be tricked into entering sensitive details into these fake sites[34]

- Threats: defined as everything that can compromise the asset. FAIR provides a detailed profile of the potential threats (goal, motivation, risk tolerance, side effects);
- Asset: defined as everything (in terms of hardware, software, data) that can be compromised by a threat.
- Organization: FAIR analyzes the environment of the organization, the risks to which it is exposed, the origin of these risks, and what their impact on the company.
- External Environment: FAIR also analyzes the risks coming from external factors, such as industry competitors, regulatory frameworks, etc.

Stages of FAIR Risk Assessment

1. **Identification of inherent components of the risk scenarios**: risk scenarios are evaluated using a probability-based model. Assets at risks and possible threats against them are identified considering the probability of occurrence of these risks.
2. **Evaluation of loss event frequency**: this stage estimates:
   - Threat Event Frequency (TEF): which is the probable frequency, within a precise time frame, that a threat may cause a loss in the organization;
   - Threat Capability: is the ability of threat agent to create a loss;
   - Control Strength: is the difficulty threshold that the threat agent must overcome to provoke a loss;
   - Vulnerability: assess the probability that a threat results from a precise vulnerability;
   - Loss event frequency: is a standard of measurements used to determine how often losses are likely to happen, in a specific time period and from the actions of different threat agents.
3. **Evaluating Probable Loss Magnitude (PLM)**: evaluates the loss that the organization expects from a loss event (considering the most probable scenario, the worst case scenario, etc.).
4. **Deriving and articulating risks**: in the end of the process, a reasonable articulation of risks is provided to decision makers. Detailed information about risk factors, how they are related, level of loss, simulation models for analysing the risk scenarios are provided.[35]

4.3.2 Lockheed Martin Cyber Kill Chain

The Cyber Kill Chain, developed by Lockheed Martin in 2011, is a threat intelligence framework inspired by the military concept of the Kill Chain. It consists of breaking an attack in 7 sequential stages and it can help creating a defense model for blocking an attack in specific stages of the kill chain. Even though it represents a good starting point in the development of cyber attacks defence strategies, it presents some limitations: for example, the fact that there are attacks that do not follow extactly all the stages of the kill chain. [36]

7 Steps of the Cyber Kill Chain

As mentioned, the Cyber Kill Chain considers attacks as composed by seven stages:

1. **Reconnaissance**: Attackers start by gathering information about their target. Tools such as search engines, packet sniffers, port scanners, etc. are used. Defence strategy: secure confidential data of the company and perform analysis to find possible attacks.

2. **Weaponization**: attackers can now choose one or several attack vectors, which will allow to access the system and begin the intrusion. The most common attack vectors are social engineering against employees, weak or stolen credentials, DoS, Man-In-The-Middle attacks, trojans, SQL injection, etc. After gaining access to the system, attackers need to find a way to move freely without being discovered. Defence strategy: collect information about possible attack vectors, detect the attack and analyse it, considering its potential impact.

3. **Delivery**: Once inside the system, the malicious payload or whatever program created for attacking the system can be delivered to the target (it may be malware, ransomware, spyware,etc.). **Defence strategy**: investigate attack considering its attack vector and try to understand the intentions of the attacker.

4. **Exploitation**: Once the payload is delivered, the exploitation of the system begins. Usually it consists into identifying the presence of exploitable vulnerabilities. Obfuscation capabilities may be used to hide malicious activities and avoid detection. Based on the type of payload, there may be different types of exploitation. Defence strategy: use penetration testing to detect the vulnerable areas of the system (currently attacked or under risk of attack).

5. **Installation**: the attacker can install a backdoor to be able to access easily the system in the future, in case it is interested in performing future attacks against

the same system. Detecting the presence of a backdoor is not easy. Defence strategy: use attack prevention measures, such as the use of appropriate certificates, examining the signatures, etc.

6. **Command and Control**: At this point, the attacker can take remote control of the system. Defence strategy: identification of the attack vectors and security flaws to understand where possible risks are.

7. **Persistance**: the attacker continues the attack by infiltrating the system step by step. At this point, being capable of detecting the attack is crucial. Defence strategy: This is the worst scenario, a response action to the attack has to be developed rapidly in order to stop it and mitigate its negative impact on the system.[37] [38]

The Cyber Kill Chain works together with CTI since the information needed in the various phases (both for performing the attack and for protecting from the attack) comes from the CTI sources of data.

4.3.3 MISP

MISP (Malware Information Sharing Platform) is a platform that collects IoC (Indicators Of Compromise)[5] and other threat information about attacks, threat actors, attack techniques, attack consequences, etc.
The main goals of MISP are:

- Storing precise information about existing malwares and their attributes;
- Storing data in a structured format so that it can be used automatically by other tools (such as detection systems and forensic tools);
- Using the data collected to create rules for Network IDSs;
- Sharing malware and threat attributes with other parties and trusted groups to help detect malware and avoid duplicate works;
- Creating a trust platform (containing trusted information from trusted partners).[39]

[5]IoC is a piece of digital forensics that suggests that an endpoint or network may have been breached.This information is used to detect attacks.

4.3.4 The Diamond model

The diamond model was created in 2013 to classify the different elements of an attack and improves the accuracy of intrusion analysis. Figure 4.1 shows a graphical representation of the Diamond Model.

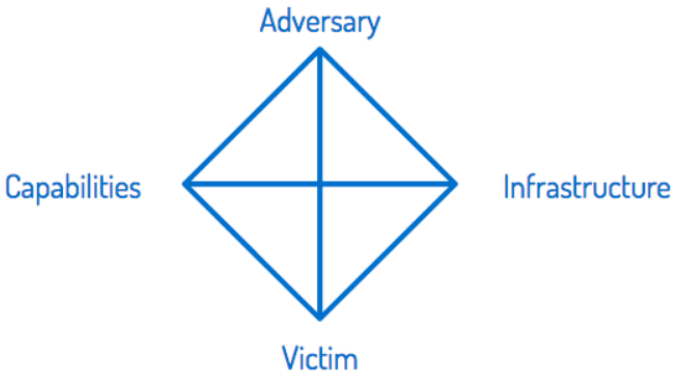

Figure 4.1: The Diamond Model

Figure 4.1 can be interpreted in this way: "For every intrusion event, there exists an adversary taking a step toward an intended goal by using a capability over an infrastructure against a victim to produce a result".[40]
In the threat intelligence domain, the diamond model is used by intrusion analysts to derive relationships between the existing pieces of information coming from intelligence sources. It helps to clearly identify the goals, tactics, and techniques used by attackers and identify security measures against threats.
The diamond model can also be used in the context of threat information sharing: it can be easily integrated with other frameworks and used in the development of courses of actions and security countermeasures, providing a good foundation for cyber taxonomies and ontologies.[41]

4.3.5 MITRE ATT&CK

According to MITRE, the not-for-profit company that developed the ATT&CK framework, *"ATT&CK (Adversarial Tactics, Techniques and Common Knowledge) is a global-accessible knowledge base of adversary tactics and techniques based on real world observation. It is used for the development of threat models and methodologies with the goal of having a more effective cybersecurity"*.[42]
ATT&CK framework is organized into three different matrices. Each matrix represents a field of application of the framework, and for each matrix, the related attack tactics and techniques are identified. The matrices are:

- **Enterprise Matrix**: collects the tactics and techniques of an attacker inside a corporate network. This matrix can be used by the organization to prioritize defense actions.
- **Mobile**: it is based on the NIST mobile Threat Catalogue and it collects tactics and techniques used to compromise mobile devices.
- **PRE-ATT&CK** collects the activities performed by attackers before attacking a particular target network or system. It helps security teams understand how attackers enter the target system and monitor their activities outside the boundaries of the corporate network.[43]

In the matrix, the columns represent the **Tactics**, while the rows represent the **Techniques**. This is a new way of describing attacks, in fact instead of looking at the final result, the framework highlights the motivations of the attack and the actions performed starting from the IoC.
Since the Enterprise Matrix has been taken into consideration for the development of the decision support system in this research work, the description of ATT&CK framework will focus on the Enterprise Matrix (which is shown in Figure 4.2).

Figure 4.2: The ATT&CK Enterprise Matrix

Tactics

The Tactics may be considered as categories of techniques, and they specify the objectives of attackers. The Enterprise Matrix has 14 tactics:

1. Reconnaissance
2. Resource Development

3. Initial Access
4. Execution
5. Persistence
6. Privilege Escalation
7. Defense Evasion
8. Credential Access
9. Discovery
10. Lateral Movement
11. Command and Control
12. Collection
13. Exfiltration
14. Impact

Techniques

The Techniques specify how attackers accomplish those objectives. In particular, every technique represents a single step in a complex multi-step attack. New techniques are continually added, identified through a four-digit code, and contain specific information about how the attacker operates (privileges required, platform used), how they can be detected and mitigated.[43]

How to use the matrix

An attack sequence would usually involve at least one technique per tactic, but multiple techniques can also be used for the same tactic. Looking at the matrix, an attack sequence could be read as moving from left to right. An attack does not have to involve all the types of tactics, but only the minimum number for achieving the attacker's goal. Figure 4.3 shows the tactics and the related techniques used by an attacker to steal sensitive files from the CEO in a 3-steps attack.

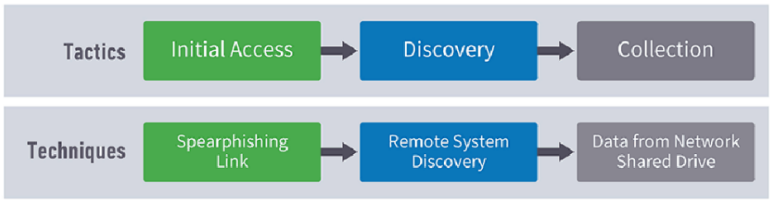

Figure 4.3: A simple attack to steal sensitive files from the CEO can be accomplished in three steps using three tactics and techniques.

Comparison between MITRE ATT&CK and the Cyber Kill Chain

We have already discussed the Lockheed Martini Cyber Kill Chain and how it divides the attack into seven steps. On the other hand, the ATT&CK framework considers an attack chain made of ten steps (in the tactics list, from initial access to command and control), and since for every tactic it lists the techniques that can be executed, it allows to delineate an attack with more granularity.[43]

Mapping MITRE ATT&CK to CVEs for vulnerabilities impact

Security teams performing vulnerability and threat management often struggle to integrate vulnerability and threat information. To enable a standardized way to describe the impact of vulnerabilities, MITRE has carried out a project whose objective is to map ATT&CK to CVE (Common Vulnerabilities and Exposures)[6] and to provide a consistent view of how attackers use vulnerabilities to achieve their goals. Knowing the techniques that can be can be used against a vulnerability, defenders can characterize rapidly the impact of vulnerabilities and prioritize them properly.
Before introducing the methodology used to map ATT&CK and CVE, it may be helpful describing the exploitation of a vulnerability.
Let's suppose that an attacker is exploiting a vulnerability where the credentials are sent in clear text. The steps (in terms of ATT&CK techniques) followed by the attacker can be divided in three categories:

- **Exploitation Technique**: is the method used to exploit the vulnerability. In the example we are considering, the technique would be "Sniff the network" (T1040).

[6]The CVE system provides a reference-method for publicly known information-security vulnerabilities and exposures. Each vulnerability is identified by a CVE identifier. CVE is typically used for the vulnerability management process in enterprises.[44]

- **Primary Impact**: represents the initial benefit obtained by exploiting the vulnerability. In the example, the technique would be "access to unsecured credentials" (T1552).
- **Secondary Impact**: represents what the adversary can do after gaining the benefits from the primary impact. In the example, the technique would be to "get access to a valid account" (T1078).

Note that, since ATT&CK describes attacks from a high level of abstraction, it is not always possible to classify a technique in one of these categories.
To map ATT&CK techniques to vulnerabilities, three methods have been proposed:

- **Vulnerability Type**: this method provides a mapping for the three categories described above. It consists of grouping vulnerabilities with common vulnerability types (e.g., cross-site scripting, SQL injection, etc.) that have common technique mappings.
- **Functionality**: this method provides a mapping only for the primary and the secondary impact categories. It consists in grouping vulnerabilities according to the type of capability that the attacker acquires by exploiting the vulnerability.
- **Exploit Technique**: this method provides a mapping only for the exploitation technique category. It consists in grouping techniques by common steps taken to exploit a vulnerability. It is usually applied when a vulnerability has too many possible exploitation scenarios to list in the vulnerability type method.

Using this mapping methodology, there are cases where mapping is not possible for the available categories and there are cases in which a group of vulnerabilities may be associated with more than one technique for the same category. It is evident that this methodology is only a starting point to help security teams describe vulnerabilities in a standardized way, since it does not provide a specific mapping for all the possible ways in which a system can be exploited.
One of the benefits provided by including ATT&CK technique references in vulnerability reports is that defenders are quicker in performing risk assessment and creating mitigation plans for new vulnerabilities, since ATT&CK includes also information about detection and mitigation methods for each technique.[45]

4.4 CTI: Sharing Standards

As mentioned so far, the increasing number of threats and incidents burdens security teams with the necessity of finding remediation techniques rapidly to minimize the negative consequences of attacks. If organizations proactively share the threat intelligence they have, it can help create a common knowledge for making resilience

and reactivity to cyber attacks stronger. Information sharing would allow gaining a deeper understanding of the threat landscape: for example, an attack that is affecting organization A, may affect organization B after some months (especially if the two organizations are in the same targeting scope of an attacker). If the threat intelligence information collected from organization A about the attack (goals. techniques used, vulnerabilities exploited, etc.) is shared, organization B could take advantage of it to develop proactive measures or to take faster reactive decisions. These requirements were the basis of the development of STIX (a standard language to represent threat information) and TAXII (a protocol for information sharing).

4.4.1 STIX

STIX (Structured Threat Information Expression) is a standardized language developed by MITRE and the OASIS Cyber Threat Intelligence Technical Committee. As stated in the official documentation of the language, "STIX is a language and serialization format used to exchange cyber threat intelligence. It enables organizations to share CTI with each other in a consistent and machine-readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively." [46]
The development of STIX was driven by the idea of creating a fully expressive, flexible, extensible, automated and human-readable language to represent structured threat information.

STIX implementation

STIX Architecture tries to put together different cyber threat information, so that it can be used in many different use cases (flexibility). As a consequence, STIX combines different existing standards for representing information, some of them are: **CybOX** (Cyber observable eXpression), **IndEX** (Indicator Exchange eXpression), **CAPEC** (Common Attack Pattern Enumeration and Classification), **MAEC** (Malware Attribute Enumeration and Characterization), etc.
The first version of STIX uses **XML Schema**[7] as a structured and easily understandable way for representing information.[48]
The subsequent versions of STIX support **JSON**[8] serialization, which is simpler to use, more lightweight and sufficient to express CTI information.[49]

[7]XML (Extensible Markup Language) is a standard for delivering content on the Internet. It is a markup language used to describe the content and structure of data in a document through the use of tags (easily extensible).[47]

[8]`https://www.javatpoint.com/what-is-json`

STIX Architecture

STIX is composed by three main components:

- **STIX Domain Objects (SDOs)**: provide a description of the main concepts of CTI (such as indicators or course of actions, etc.).
- **STIX Cyber Observable Objects (SCOs)**: provide a description of facts about the assets involved in the cyber attack (such as IP addresses, keys, files, etc.).
- **STIX Relationship Objects (SROs)**: allow the connection between SDOs and SCOs.

A STIX CTI product is presented as a graph, where SDOs and SCOs are the nodes and SROs are the links that connect the nodes (Figure 4.4).[48]

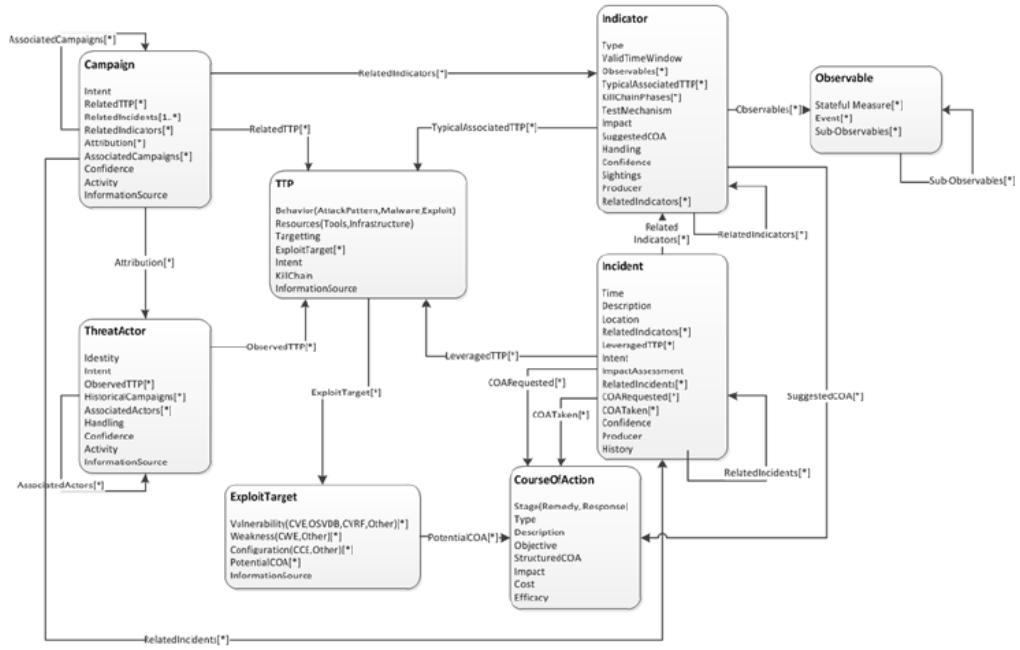

Figure 4.4: STIX Architecture v0.3

Each object is described through a set of properties of different types (from file hashes to describe malicious activities, to contextual information to describe tactics, techniques, and procedures used by attackers).[50]
The latest version of STIX (2.1) defines 18 Domain Objects:

- **Attack Pattern**: type of TTP used for describing how adversaries try to compromise specific targets.

- **Campaign**: used to describe malicious activities carried out against a specific set of targets.
- **Course of Action**: used to describe the actions to be taken in response to a security incident
- **Grouping**: used to assert that STIX objects have a shared context.
- **Identity**: used for describing individuals, organizations, or groups.
- **Indicator**: provides a pattern used to detect suspicious cyber activities.
- **Infrastructure**: type of TTP, used to describe tools (software, systems or services) which support malicious activities.
- **Intrusion Set**: used to describe a group of malicious behaviors and resources created by a single organization and with common characteristics.
- **Location**: used for describing a geographic location.
- **Malware**: type of TTP.
- **Malware Analysis**: used to describe metadata and results of the analysis performed on malwares.
- **Note**: used to provide additional context to STIX objects.
- **Observed Data**: used to describe entities related to cyber security (file, systems, networks, etc.) through the Cyber-Observable Objects (SCOs).
- **Opinion**: used to describe information produced by a different entity to assess the correctness of the produced STIX object.
- **Report**: provides a description of CTI focused more on topics (description of threat actors, malware, attack techniques, including context).
- **Threat Actor**: provides a description of individual, group or organization considered to be responsible of malicious activities.
- **Tool**: used to describe the legitimate software used by an attacker to perform the attack.
- **Vulnerability**: used to describe weaknesses in software that can be exploited to gain access to a system or a network.

These objects are represented in JSON. Figure 4.5 provides an example of a Campaign Object in JSON format.[51]

```
{
    "type": "campaign",
    "id": "campaign--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "spec_version": "2.1",
    "created": "2016-04-06T20:03:00.000Z",
    "modified": "2016-04-06T20:03:23.000Z",
    "name": "Green Group Attacks Against Finance",
    "description": "Campaign by Green Group against targets in the financial
services sector."
}
```


Figure 4.5: STIX 2.1 Campaign Object in JSON fromat

4.4.2 TAXII

TAXII (Trusted Automated Exchange of Intelligence Information) is an application layer protocol that supports the exchange of CTI over HTTPS (Hyper Text Transfer Protocol Secure). By defining an API that supports common sharing models, TAXII enables organizations to exchange threat intelligence data. It has been designed for supporting CTI in STIX format, but it is not limited to STIX (other formats are allowed).[52]
As described in the official OASIS documentation of TAXII, two primary services are defined:

- **Collection**: A collection is an interface to a logical repository of CTI objects provided by a TAXII Server. Here, a producer can host his CTI data, which can be requested by a consumer.
- **Channel**: a channel allows producers to push data to many consumers and consumers to receive data from many producers.

TAXII Clients and Servers exchange information through a request-response model, while TAXII Clients exchange information with other TAXII Clients through a publish-subscribe model.
Collections and channels may be organized in different ways as instances of the TAXII API. [52]

API Roots and interaction between TAXII Clients and Servers

API Roots are instances of the TAXII API created by logically grouping Collections and Channels. They are accessible via different URLs, and the API Root is the "root" URL of that particular instance. The same TAXII server may host many API Roots that allow the division of content and access control among different trust groups.
Each API Root contains a set of **Endpoints** (which consists of a specific URL

and an HTTP method) that allows interaction between TAXII Clients and Servers. There are different types of interaction:

- Server Discovery: used to learn which API Roots are hosted by a TAXII Server.
- Client - Collection interaction: allows a client to discover which type of CTI is contained in a collection, to push new CTI or retrieve CTI from the collection. Each piece of CTI in a collection is considered as an object.
- API Root - Channel interaction: an API Root may host zero or more channels.
- API Root - Client interaction: allows a client to check on the status of specific requests (e.g., request to submit a new CTI).[52]

Sharing Models

TAXII has been designed so that it allows integrating existing sharing agreements and access control limitations. When possible, it leverages existing protocols, with native support for HTTP and HTTPS. It provides three sharing models:

- **Hub and Spoke**: the organization takes a central role (Hub) and coordinates the information exchange between partner organizations (Spoke), which produce or consume information from the Hub.
- **Source/Subscriber**: the organization is the only source of information and sends this information to subscribers.
- **Peer to Peer**: two or more organizations share information directly with each other.[53]

Chapter 5

Ontologies

5.1 Introduction

This thesis is centered on the development of a decision support system for the mitigation of cybersecurity incidents based on the use of ontologies and reasoning. Ontologies are defined through formal languages and rule languages and they are able not only to represent knowledge (which can be easily shared and reused) but also to interpret this knowledge in order to extract additional information (reasoning). Reasoning is what will allow the identification of response actions for incident mitigation. This chapter will give an overview of what an ontology is, the languages used to describe it, and the languages used to reason over the knowledge provided.
Derived from a philosophical concept, the term ontology in computer science refers to a data model used for a formal representation of knowledge about the world. A complete definition of the concept of ontology has been provided in [54], where it has been defined as "an **explicit** and **formal** specification of a **shared conceptualization**". In fact, an ontology:

- specifically defines the concepts, properties, relationships, functions, taxonomies, axioms, and restrictions or rules from which it is composed (explicit);
- it is specified by a machine-interpretable language (formal);
- it provides a simplified view of the represented domain (conceptualization);
- the information provided is first agreed among different groups of experts (shared).

Ontologies provide a common and homogeneous knowledge that can be shared and reused by humans and machines (artificial intelligence agents), overcoming the

problem of interpretation coming from the use of different languages and ways of representing information.[55]
The use of ontologies has increased with the introduction of Semantic Web [1]. In this context, ontologies are the combination of

- a **Taxonomy**, which consists of:
  - **Individuals** or **Instances**: represent the objects (concrete, such as people, animals, things in general, and abstract, such as numbers or words) of the domain in which the ontology is centered. The ontology's general purpose is providing means for classifying individuals.
  - **Attributes or Properties**: represent features, characteristics, or parameters that individuals can have and share.
  - **Classes** or Concepts: are sets, collections, or types of objects. The objects of a class are defined by specific characteristics that represent the constraints to be members of that class. Classes contain individuals and may be organized hierarchically (superclass - subclass hierarchy).
  - **Relations**: show how objects can be related one another. The set of relations describes **semantics of the domain**. Types of relations are "is-a", "is-a-subclass-of", "is-a-superclass-of".[57][58][59]
- a set of **Inference rules**, used to express restrictions and conditions on the taxonomy's objects. In ontology language, restrictions and rules are expressed in the form of **axioms**. There may be several types of axioms (subclass axioms, equivalence axioms, property restriction axioms, etc.).[55]

5.2 Ontology Languages

Ontology languages are formal languages used to construct ontologies. As defined by IGI Global, *"an ontology language is based on a logic paradigm that can represent concepts and the constraints between them. Reasoning capabilities of the language depend on the paradigm in which the language is based on"*[2].
In the last years, several ontology languages have been developed. They differ in the syntax on which they are based and can be classified into three groups:

[1]Semantic Web can be seen as an extension of the World Wide Web, which provides additional concept (in terms of machine-interpretable metadata) to data present on the web. From the definition of rules handling data, meaningful interpretation and reasoning can be carried out from data provided by Semantic Web. [56]

[2]`https://www.igi-global.com/dictionary/ontology-language/21127`

- **Traditional languages**: they were developed before the birth of Web Semantic and were based on first-order logic and frames.
- **UML (Unified Modeling Language) and OCL (Open Constraint Language)**: UML is based on diagrams and is used in software engineering. OCL has been introduced to specify restrictions on the information defined using UML.
- **Languages for Web Semantic**: use a markup scheme to encode knowledge, commonly **XML**. Among these languages, there are: RDF, RDFS, DAML+OIL, OWL, OWL2, etc.

The following part of this section will be focused on the description of the Web Ontology Languages, in particular OWL2 which is the one that has been used for the research work developed. Many of the developed Web Ontology Languages and Standards belong to W3C (Word Wide Web Consortium). Figure 5.1 shows some of the Web Ontology Languages developed from the introduction of the Semantic Web.

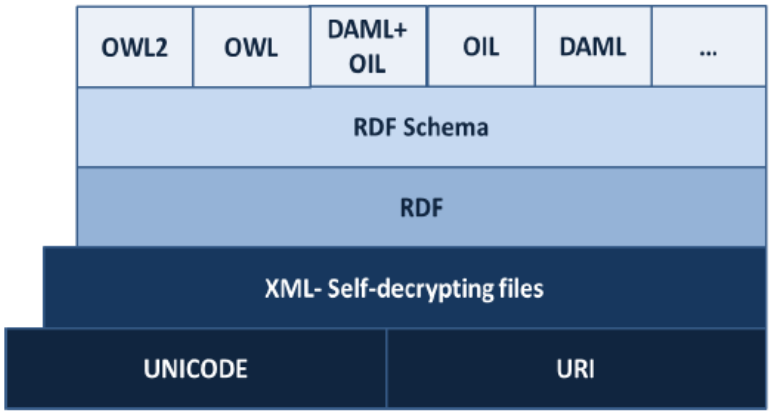

Figure 5.1: Web Ontology Languages

5.2.1 XML

As we have mentioned, XML (eXtensible Markup Language) is at the bases of the Web Semantic Languages. It cannot be considered an ontology language, as it does not enforce any semantic restriction. XML categorizes and structurally organizes information and allows the unique identification of elements of a document using *XML Namespaces*. An XML namespace can be considered as a string used like a prefix to resolve the ambiguity between elements with the same name. A namespace

name is a **URI** (Uniform Resource Identifier), defined as a unique sequence of characters that identifies a logical or physical resource on the Semantic Web.[3][55]

5.2.2 RDF

RDF (Resource Description Framework) is a standard developed by W3C and designed as a data model to describe and exchange metadata on the World Wide Web. Being the first language based on formal semantics (which can be interpreted by machines), it is now the basis of many ontology languages (such as RDFS and OWL), created with the objective of describing RDF data.
RDF semantic is based on statements that describe resources in the form of triplets **Subject-Predicate-Object**:

- **Subject**: denotes the resource that is being described.
- **Predicate**: denotes a property or a relation about the subject (it binds the subject to the object).
- **Object**: is the value of the property. It may be another resource or a literal value (for which a data type has to be specified).

Since resources are identified through URIs, subjects, predicates, and objects (when they are not literal) are also identified by URIs. The connection of triplets forms an RDF directed graph, where subjects and objects are nodes, and predicates are arcs from subjects to objects. The formal semantics of RDF could allow inferring new information through reasoning. But since this semantic is simple and the triplets are not sufficiently expressive, using RDF for reasoning is not the best option. RDF provides several data serialization formats. The most common are **N-Triplets**, **Turtle** and **RDF/XML**. RDF/XML is the most widely used, but Turtle is easier to be interpreted and understood by humans.[55]
Figure 5.2 shows the different ways of representing RDF triplets: in Triple format (also called N-triples format), each component is a URI (or a data value in the case of a literal object); XML/RDF and Turtle represent information in the same way, but Turtle is more human readable since it uses prefixes and allows grouping triples with the same subject into blocks.[60]

For example, let us consider an RDF graph showing information about Bob Marley. The following figures show how the same RDF graph can be presented in N-Triplets (Figure 5.3), Turtle (Figure 5.4) and RDF/XML formats (Figure 5.5).[61]

[3]XML Namespace definition: `https://en.wikipedia.org/wiki/XML_namespace`
URI definition: `https://en.wikipedia.org/wiki/Uniform_Resource_Identifier`

Graphical form	
Triple	`subject predicate object`
Relational form	`predicate(subject,object)`

RDF/XML	`<rdf:Description rdf:about="subject">`
	`  <ex:predicate>`
	`  <rdf:Description rdf:about="object"/>`
	`  </ex:predicate>`
	`</rdf:Description>`

Turtle	`subject ex:predicate object.`

Figure 5.2: RDF triplets in different forms

(1) <http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
(2) <http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/2000/01/rdf-schema#label> "Bob Marley"@en .
(3) <http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/2000/01/rdf-schema#label> "Bob Marley"@fr .
(4) <http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/2000/01/rdf-schema#seeAlso> <http://dbpedia.org/resource/Rastafari> .
(5) <http://dbpedia.org/resource/Bob_Marley> <http://dbpedia.org/ontology/birthPlace> <http://dbpedia.org/resource/Jamaica> .
(6) <http://dbpedia.org/resource/Jamaica> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Country> .
(7) <http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2000/01/rdf-schema#label> "Jamaica"@en .
(8) <http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2000/01/rdf-schema#label> "Giamaica"@it .
(9) <http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2003/01/geo/wgs84_pos#lat> "17.9833"^^<http://www.w3.org/2001/XMLSchema#float> .
(10) <http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2003/01/geo/wgs84_pos#long> "-76.8"^^<http://www.w3.org/2001/XMLSchema#float> .
(11) <http://dbpedia.org/resource/Jamaica> <http://xmlns.com/foaf/0.1/homepage> <http://jis.gov.jm/> .

Figure 5.3: N-Triplets format

5.2.3 RDFS

RDFS (Resource Description Framework Schema) is a Web Ontology Language created by W3C as an extension of RDF. The structure of RDFS is similar to that of RDF, but the semantics is different: RDFS allows us to define additional restrictions on RDF resources and properties. In particular, RDFS introduces the following:

- *Class* and *Type* to define an instance (individual) and the class to which it belongs (considered as a collection of resources);
- the property **subclassOf** (used to create class hierarchies);
- the concepts of **Range** and **Domain** (to specify the class range and the class domain of a property).

In RDFS, the system of classes and properties is similar to the type systems of object-oriented programming languages, with the difference that RDFS describes properties in terms of the classes of resources to which they apply (and not vice versa). RDFS allows the construction of vocabularies that can be considered as ontologies.[55][62]

```
@prefix dbr:    <http://dbpedia.org/resource/> .
@prefix dbo:    <http://dbpedia.org/ontology/> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix geo:    <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <http://schema.org/> .

dbr:Bob_Marley
   a foaf:Person ;
   rdfs:label "Bob Marley"@en ;
   rdfs:label "Bob Marley"@fr ;
   rdfs:seeAlso dbr:Rastafari ;
   dbo:birthPlace dbr:Jamaica .

dbr:Jamaica
   a schema:Country ;
   rdfs:label "Jamaica"@en ;
   rdfs:label "Giamaica"@it ;
   geo:lat "17.9833"^^xsd:float ;
   geo:long "-76.8"^^xsd:float ;
   foaf:homepage <http://jis.gov.jm/> .
```

Figure 5.4: Turtle format

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:foaf="http://xmlns.com/foaf/0.1/"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:ns0="http://dbpedia.org/ontology/"
       xmlns:schema="http://schema.org/"
       xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">

 <foaf:Person rdf:about="http://dbpedia.org/resource/Bob_Marley">
  <rdfs:label xml:lang="en">Bob Marley</rdfs:label>
  <rdfs:label xml:lang="fr">Bob Marley</rdfs:label>
  <rdfs:seeAlso rdf:resource="http://dbpedia.org/resource/Rastafari"/>
  <ns0:birthPlace>
   <schema:Country rdf:about="http://dbpedia.org/resource/Jamaica">
    <rdfs:label xml:lang="en">Jamaica</rdfs:label>
    <rdfs:label xml:lang="it">Giamaica</rdfs:label>
    <geo:lat rdf:datatype="http://www.w3.org/2001/XMLSchema#float">17.9833</geo:lat>
    <geo:long rdf:datatype="http://www.w3.org/2001/XMLSchema#float">-76.8</geo:long>
    <foaf:homepage rdf:resource="http://jis.gov.jm/"/>
   </schema:Country>
  </ns0:birthPlace>

 </foaf:Person>

</rdf:RDF>
```

Figure 5.5: RDF/XML format

5.2.4 OWL

OWL (Ontology Web Language) is one of the most widely used languages on the Semantic Web (together with OWL2) to define ontologies. It is an object-oriented language, it describes a domain in terms of classes, properties, and individuals, and it is able to provide a detailed description of these objects.

Sintaxes

The OWL family of languages support two categories of syntax:

- **High level syntax**: it is used to specify the structure and semantics of the ontology. In this category there is **OWL abstract syntax**, which describes the ontology as a sequence of *annotations* (carrying machine and human-oriented meta-data), *axioms* (used to create relations among classes, individuals and properties), and *facts* (which state data about individuals). URI references are used to identify classes, properties, and individuals.
- **Exchange syntax**: it is more suitable for general use. In this category there is the *RDF syntax* and *OWL/XML syntax*.

RDF/XML syntax and **OWL/XML syntax** are based on **model theory**, which can be considered as a technique to officially specify the semantics of a formal language. It assumes that a language is a world and specifies the minimum set of conditions that the world must satisfy to be meaningful in each language expression. The model theory at the base of OWL specifies that OWL language is purely descriptive: the sentences being part of the ontology do not determine the existence of an element, but they just describe such an element (the ontology simply describes what it is assumed to be an existing world).

Semantics

Regarding the semantics, two formal semantics are proposed: one is compatible with RDFS and it is described by a **model theory**. The other is based on **Description Logic (DL)**, a family of logics that originated with first-order logic with computational properties.

OWL sublanguages

The OWL specification includes three variants with three different levels of expressiveness, which make OWL adaptable to the user's needs and to existing applications. The three variants are:

- **OWL Lite**: its main feature is simplicity. It is defined as a subset of all existing OWL structures (establishing also restrictions on their use) and it is considered an extension of RDFS (but it is not compatible with all RDF/RDFS documents).
- **OWL DL** (OWL Description Logic): provides maximum expressiveness ensuring computational completeness and inference in finite time. Its semantic is an extension of RDF, it contains all existing OWL structures but with some restrictions on the properties. For this reason it is not compatible with documents that use the maximum expressiveness of the RDF/RDFS.
- **OWL Full**: provides maximum expressiveness and is compatible with RDF/RDFS. As a consequence, the models built in OWL Full can freely use RDF, RDFS, and OWL structures, but a finite time for reasoning is not guaranteed. [55][63]

5.2.5 OWL2

OWL2 is an extension of OWL, developed by W3C. Like OWL, its goal is to create a standard language for describing ontologies in order to facilitate information exchange on the Web and to make it accessible and machine-readable. Figure 5.6 shows the main building blocks of OWL2 and how they are related to each other. The ellipse in the center represents the ontology, which can be considered as an abstract structure or an RDF graph. In the upper part of the figure, there are the concrete syntaxes (used to share the information represented in the ontology), while in the lower part, there are the two supported semantic specifications.

Syntaxes

OWL2 supports 2 categories of syntaxes:

- **Concrete syntax**: allows ontologies to be stored and exchanged among tools and applications. The main syntax, which must be supported by all OWL2 tools, is RDF/XML. Other concrete syntaxes to be used are: Turtle, OWL/XML, and Manchester syntax (which is more readable).
- **Functional syntax**: it is mainly used for specifying the structure of the language.[64]

Semantics

For OWL2, two semantic models are defined: **Direct semantics** and **RDF-Based semantics**. They represent two ways of assigning a meaning to the ontology and

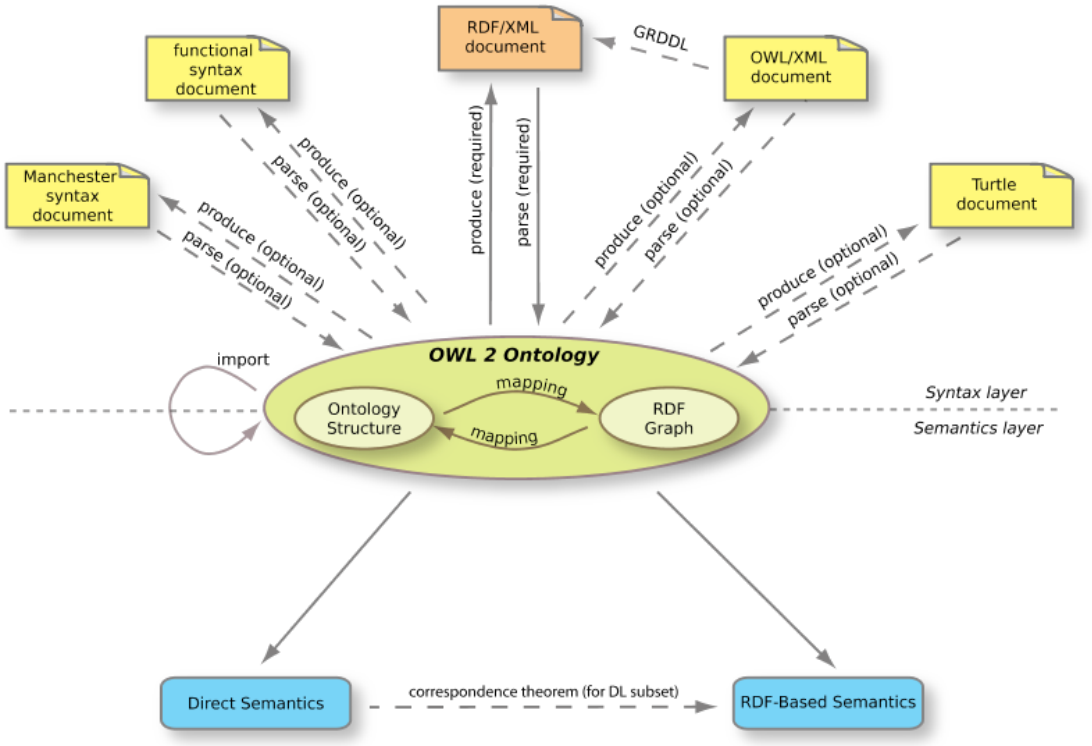

Figure 5.6: OWL2 structure

they are both used by reasoners to check the consistency of the ontology or to make queries about individuals. The difference is that Direct Semantics assign meaning directly to the ontolgy, while RDF-Based Semantics assign meaning to the RDF graph on which the ontology is built.

OWL2 Profiles

OWL2 profiles are sublanguages of OWL2. Each profile can be applied in a specific scenario and can be considered as a *syntactic restriction* of OWL2 (this means that each profile contains a subset of the entire set of structural elements used for the description of the ontology). There are three different profiles of OWL2:

- **OWL2 EL** (Existencial Language): it is suitable for applications using very large ontologies. It allows performing reasoning tasks in polynomial time, but still guaranteeing performance.
- **OWL2 QL** (Query Language): it is suitable for applications using relatively lightweight ontologies, with a great number of individuals, requiring direct access to data via relational queries (such as SQL).

- **OWL2 RL** (Rule Language): it is suitable for applications using relatively lightweight ontologies, with a great number of individuals that require to operate directly on data in the form of RDF triplets. Reasoning tasks are performed in polynomial time.

OWL vs OWL2

OWL2 structure is very similar to the OWL structure: RDF/XML syntax plays a central role in the definition of these languages, and both use direct and RDF-based semantics. OWL2 is backword compatible, which means that OWL ontologies are valid OWL2 ontologies.
Anyway, some differences exist between OWL and OWL2:

- OWL2 adds some functionalities that allow us to increase the expressiveness of the ontology. Some examples are keys, new properties (such as asymmetric, reflexive, and disjoint), qualified cardinality restrictions, etc.
- OWL2 defines three new profiles (the ones described above) and a new syntax (Manchester syntax).
- Both are based on DL (Description Logic), but OWL2 overcomes some restrictions of OWL, resulting in a more expressive language.[55][64]

5.3 Reasoning and Rule Languages

One of the most important benefits derived from the use of ontologies and their formal specification is the possibility to infer new knowledge from the known facts already present in the ontology. This concept is known as reasoning. In this research work, reasoning will be used to infer the optimal course of action in response to a cyber incident.
Some inference may derive directly from the ontology (OWL2 allows defining some axioms and restrictions used in the reasoning process), but to perform implicit inference (such as expressing relations among individuals referenced by their properties), it is necessary to introduce **Semantic Rules**.
Semantic Rules belong to a layer that is on top of (and integrates coherently with) the ontology: rules are referred to the elements present in the ontology and they allow to reason with them.
The authors of Semantic Web have developed various Rule Languages, such as RuleML (Rule Markup Language), SWRL (Semantic Web Rule Language), RIF (Rule Interchange Format), etc. The following paragraphs of this section will focus on the description of RuleML and SWRL. Particular attention will be paid to SWRL, since it is used in this research work.[65]

5.3.1 SPIN

SPIN (SPARQL Inferencing Notation) is a standard created by W3C to represent SPARQL rules[4] and constraints on Semantic Web models[5]. It describes object behavior on the Semantic Web by combining object-oriented languages, rule languages, and rule-based systems. The goal is to link ontology classes with SPARQL queries, allowing the definition of the rules which regulate the expected behaviour of the members of the class. It is made up of:

- **SPIN Templates**: are parametrized queries that are available in RESTful web services and can reuse common SPARQL patterns.
- **SPIN Rules**: allowing to create (infer) new information based on existing data and are implemented using SPARQL CONSTRUCT or SPARQL UPDATE queries.
- **SPIN Constraints**: used to check the validity of the data, they are specified using SPARQL ASK or CONSTRUCT queries.
- **SPIN Functions**: are custom query elements typically implemented as SPARQL subqueries or as Java/Javascript programs.
- **RDF Syntax for SPARQL**: SPIN elements are declarative and stored in RDF (allowing them to be queried, shared, and discovered).

SPIN rules and constraints are used in several contexts: to dynamically calculate the value of a property based on other properties, to check constraints and perform data validation, or to isolate a set of rules to be executed under certain conditions. Since it operates directly on RDF, it is simple and fast (rules are executed without the need to transform data in another format).[66]

5.3.2 RuleML

RuleML (Rule Markup Language) is a language created for defining and interchanging Web Semantic Rules. It is based on XML/RDF, such that rules are defined by means of XML, and it is supported by many applications managing rules and inference. RuleML allows the definition of different types of rules, which can be divided into two top-level categories: **Deliberation Rules** (based on facts and queries, including integrity constraints) and **Reaction Rules** (used to specify

[4]SPARQL is the standard query language and protocol for Linked Open Data and RDF databases. Ref: `https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/`

[5]`https://spinrdf.org/`

conditions under which precise actions must be taken). The entire set of rules includes:

- Production Rules (Condition - Action rules)
- Event - Condition - Action Rules: which are executed when a particular event occurs
- Integrity Rules: which define conditions that always have to be satisfied
- Derivation Rules (implication - inference rules)

One of the disadvantages of RuleML is that it does not allow integration with OWL and OWL2.[55][67]

5.3.3 SWRL

SWRL (Semantic Web Rule Language) is a rule language proposed in 2014 by W3C for the Semantic Web. It combines OWL-DL and OWL-Lite with the Unary/Binary Datalog sublanguage of RuleML. SWRL semantics is an extension of OWL-DL and does not depend on RDF/RDFS. The most relevant element is the introduction of a new type of axiom called rule.

Rule axioms

A rule axiom consists of an **antecedent** (called body) and a **consequent** (called head); body and head correspond to a conjunction of atoms, but can also be empty. A rule axiom can be identified by a URI reference. In human-readable syntax, a rule can be expressed in the form of implication:

$$antecedent \Rightarrow consequent$$

Informally, a rule may be interpreted as: if the antecedent is true, then the consequent must also be true (this implies that all the atoms creating the antecedent and consequent must be true). If the antecedent is empty, it is automatically considered true, while if the consequent is empty it is automatically considered false.
To provide an example of SWRL rule, suppose that we have an OWL ontology with the class *Person* and the properties *hasParent*, *hasBrother*, *hasUncle* that relate individuals belonging to the class *Person*; we want to create a rule which asserts that the properties *hasParent* and *hasBrother* imply property *hasUncle*. The rule (in human-readable syntax) would be:

$$Person(?x1) \wedge Person(?x2) \wedge Person(?x3) \wedge hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \Rightarrow hasUncle(?x1, ?x3)$$

The antecedent declares that there are three individuals belonging to the class Person. Two of them (x1 and x2) are related through the property *hasParent* (x1 has a parent, which is x2), and x2 is related to the third individual x3 through the property *hasBrother* (x2 has a brother, which is x3). This rule affirms that if an individual "Parent" has a son "Son" and a brother "Brother", then the individual "Brother" is uncle of the individual "Son" and they are related through the property *hasUncle*.

The same rule is then expressed in XML concrete syntax (which combines *OWL XML Presentation Syntax* and *RuleML XML syntax*), where specific labels are used to describe the various components of the rule:

- *<ruleml:imp>*: expresses the implication between body and head;
- *<ruleml:_body>*: lists the atoms of the body;
- *<ruleml:_head>*: lists the atoms of the head;
- *<ruleml:var*: defines the variables used to evaluate the rule;
- *<swrlx:individualPropertyAtom>*: allows the definition of axioms referred to existing properties. It is also possible to define atoms referred to classes, ranges, mathematical functions, etc.

Figure 5.7 shows the rule presented previously in concrete XML syntax. It is important to highlight that only the variables present in the antecedent can be present in the consequent. This condition is called **security condition** and it is important because it implies that conclusions can be drawn only from existing knowledge.

Atoms

Atoms are predicates in the form of *C(x), P(x,y), sameAs(x,y), differentFrom(r,x), builtIn(x,y,...)*, where C is a class or a data type and P is a property, both present in the OWL/OWL2 ontology; r is a *built-in* relation (pre-programmed); x and y can be variables or individuals in OWL/OWL2 or datatypes.
SWRL allows the creation of different types of atoms: for example, they can be used to express that an individual belongs to a class, to specify the relation between two objects (ObjectProperty) or between an object and a literal (DataProperty) of the ontology, etc.[55]

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom  swrlx:property="hasParent">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x2</ruleml:var>
   </swrlx:individualPropertyAtom>
   <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
     <ruleml:var>x2</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
   <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Figure 5.7: SWRL rule expressed in XML concrete syntax

Built-ins

Built-ins are SWRL elements that allow one to execute many operations on different data types of the XML schema. SWRL built-ins can be divided into seven categories according to the function they execute and to the type of data to which they are applied:

- Built-ins for comparison: equal, notEqual, lessThan, lessThanOrEqual, greaterThan, greaterThanOrEqual.
- Math built-ins: add, subtract, multiply, divide, mod, pow, etc.
- Built-ins for boolean values: booleanNot
- Built-ins for strings: stringEqualIgnoreCase, stringConcat, stringLength, contains, etc.
- Built-ins for date, time, and duration: yearMonthDuration, dayTimeDuration, dateTime, date, time, etc.
- Built-ins for URIs: resolveURI, anyURI.
- Built-ins for lists: listConcat, listIntersection, listSubtraction, length, etc.

Other characteristics of SWRL

Adding SWRL rules to an existing ontology means adding new implicit information. This can have two consequences:

- The new implicit information is in conflict with pre-existing information: the new ontology created by the combination between OWL and SWRL is not consistent.
- The new implicit information is not in conflict with the pre-existing information: the new ontology is consistent. In this case, two things may happen:
  - The new information corresponds to the information explicitly defined in OWL. In this case, SWRL simply proves that the information is consistent and that the axioms and restrictions are satisfied.
  - The new information does not correspond to the explicit one, so it is actually adding new knowledge. Since this new information is implicit, it is not added to the ontology in the form of axioms in the same way as the axioms created explicitly in OWL. However, the interpretation of the new ontology must satisfy the implicit restrictions, even if they are not expressed explicitly. The only way to make this information explicit is through inference.[55]

Some of the rules built with SWRL can also be defined in OWL. However, SWRL allows for the definition of specific restrictions that cannot be defined in OWL or RDF. Among the most important characteristics of SWRL we can highlight the following:

- SWRL allows the definition of complex conditions, through the use of built-ins, AND, and atoms referred to classes, properties, mathematical functions, etc. Restrictions can also be defined using variables in the atoms (not supported by RDF or OWL).
- SWRL is based on **First Order Logic** (which uses basic logic operators such as AND, OR, NOT, etc. in both the antecedent and the consequent), although only AND can be used in SWRL.
- SWRL offers a high level of abstraction and is recommended by the Semantic Web community due to its compatibility with OWL/XML and RDF.
- It can be integrated in ontology editors (such as Protégé)
- It is characterized by a high level of integration with OWL ontologies, since it is able to define rules directly on the elements represented in OWL.

SWRL also presents some drawbacks:

- it is limited to OWL classes and binary predicates;
- It does not include the logic operators NOT and OR for the definition of atoms;
- It does not allow specifying meta-information on rules (such as priorities, field of application, etc.), which could be useful for remotely managing the rules or for conflict resolution;
- It does not allow one to represent events explicitly [55]

5.4 Semantic Reasoners

In this research work, apart from defining the domain of Courses of Actions in response to intrusions and the rules to be applied on this domain, it was necessary to use reasoning in order to infer new knowledge. Executing reasoning is possible thanks to semantic reasoners. A **Semantic Reasoner** (also called reasoning engine, rule engine, or simply **reasoner**) is a software specifically created to infer logical consequences from a set of asserted facts or axioms. Inference rules are usually specified through rule languages, which (as already discussed) are a combination of an ontology language and a description logic language. Many reasoners use a **first-order predicate logic** to perform reasoning, others are probabilistic reasoners. The main difference among reasoners is the rule language they support, the mechanism used to perform reasoning and inference, the efficiency and the performances of the reasoner.
Reasoners are mainly used to perform 3 types of reasoning:

- **Deductive reasoning**, also called **progressive reasoning** or **Forward Chaining (FC)**: considers formal logic and observations to prove a theory or hypothesis. Starting from assumptions, deductive reasoning makes observations to validate or refute those assumptions. Considering rules, a reasoner based on forward chaining proceeds looking for inference rules until it finds one in which the antecedent is true, so that it can execute the consequent and obtain new information. Reasoners based on this type of reasoning usually use a RETE algorithm (able to infer knowledge starting from a base of facts, with the objective of speeding up the process of rule execution).
- **Inductive reasoning**, also called **regressive reasoning** or **Backward Chaining (BC)**: uses theories and assumptions to validate observations. It may be considered as the opposite of deductive reasoning since it starts from

a specific case (initial hypothesis) in order to derive a general rule (the reasoning chain retrieves data and evidence starting from the conclusions). This type of reasoner analyses rules until it finds one which consequent corresponds to the desired objective.

- **Mixed, hybrid chaining**: it is a mechanism in which deductive and inductive reasoning are mixed.

The following part of this chapter will focus on the description of the semantic reasoner used in this research work. It is called Pellet and is based on SWRL rules.[55][68]

5.4.1 Pellet

Pellet is an open-source OWL2 semantic reasoner developed in Java. Since it is open source, it is widely used in various projects, from research to industrial settings.
Pellet is based on Description Logic, which allows it to completely support OWL-DL. It provides standard and advanced reasoning services on OWL ontologies, various optimization techniques based on DL, capability of reasoning and provides responses to complex queries, and incremental reasoning. Among Pellet most important functions we can find:

- Ontology consistency checking: Pellet allows stating that the ontology does not contain inconsistent facts.

- Ontology validation through the satisfiability concept: Pellet allows to determine whether a class can have instances or whether it causes some inconsistency.

- Classification of classes: Pellet allows creating a complete class hierarchy by evaluating the relations among subclasses. This hierarchy can be used to formulate queries (for example SPARQL queries, supported by Pellet).

- Definition of concepts in the hierarchy after executing classification (for example, finding the most appropriate class to which an individual belongs).

- Ontology debugging: Pellet allows to identify the specific inconsistent axioms, which cause the ontology to be inconsistent.

Pellet is in a continuous state of development. The last published version is *Pellet 3.0*

Formats and languages supported

Pellet allows performing reasoning on data expressed in different formats and languages:

- OWL-DL: as mentioned, Pellet is based on Description Logic. As a consequence, it supports OWL-DL, including nominal reasoning.
- OWL-Full: Pellet allows reasoning on the reverse functionality properties of the OWL-Full datatypes.
- OWL2: starting from version 1.5.2, Pellet supports OWL2. With the development of newer versions of Pellet, the compatibility of reasoning with OWL2 has improved.
- XML-S: Pellet allows reasoning on data expressed in XML-Schema (numbers, strings, date-time).
- SPARQL/SPARQL-DL: Pellet includes a query engine that supports the answer to a conjunction of queries.
- SWRL: Pellet allows combining DL-Safe rules with OWL-DL. This is done through an SWRL rules parser that loads the DL-safe rules encoded in SWRL. Integration between Pellet and SWRL rules improved with the development of newer version of Pellet (for example, at the beginning, Pellet supported only built-ins comparison rules, while now it supports all built-ins). However, DL-safe implementation is not recommended for small/medium ontologies. SWRL rules can be managed not only directly from Pellet, but also through Java libraries such as Jena and **OWLAPI** (used in this research work).

Interfaces

Pellet reasoner can be accessed through several interfaces, such as:

- **CLI (Command Line Interface)**: users can easily access all Pellet functionalities through a GNU-Style interface; a set of commands is provided for consistency check, classification, queries/answers, and inferred information extractions.
- A programmable API that can be used in a remote application.
- **Jena** and **OWL-API** interfaces
- Direct integration with the SWOOP ontology editor.
- Through protocol DIG, which allows clients (such as Protégé) to use Pellet.[55]

Chapter 6

Decision making and Decision Support System

As revealed in the title, the research work in this thesis proposes the prototype of a Decision Support System (DSS) for the mitigation of cybersecurity incidents. This chapter will give a general introduction to decision-making and Decision Support System. Then it will focus on the description of specific DSSs developed for supporting cybersecurity.

6.1 Decision-making

Decision-making is the process that individuals or computer systems carry out to determine the best option when they need to make a decision. It is characterized by two main components: a set of *alternatives* and a set of *goals* to be satisfied by selecting of one of the alternatives. This process is widely applied in organizational and managerial activities, especially for making business decisions.[69]
A decision can be defined as a **Course of Action** selected from a set of alternatives to achieve some specific objectives or goals. The alternative chosen to achieve some specific goal can be defined as **Course of Action (CoA**.

6.1.1 Decision-making process

When approaching a decision-making problem, it is extremely important to have a good understanding and knowledge of the type of decision to make, what the goals are and what the possible consequences of the decision made, among which alternatives to decide and their properties, etc. As a consequence, decision making is considered as a sequential process that executes the following steps:

1. Identification of the type of decision to make.
2. Gathering of relevant information and data that can help create alternatives.
3. Identification and development of the alternatives.
4. Evaluation of the alternatives after assigning them a weight.
5. Selection of the best possible option among the alternatives (based on the weights).
6. Execution of the alternative chosen.
7. Revision of the decision made (checking its consequences and repeating the decision-making process in case the option chosen does not meet the needs).[69][70]

This process requires a lot of time to execute, but there are many fields of application where the speed with which decisions are made is extremely important to achieve the predefined goal. **Decision support systems** softwares have been developed with the objective of helping decision makers make better and faster decisions.

6.2 Decision Support System

A Decision Support System is a computer-based system created to help with decision-making activities by accessing and analyzing large volumes of information collected from various information systems. A DSS can be human-powered, automated, or a combination of both. A DSS can be formally defined as an application or a program that combines and analyzes raw data and documents of various types and from various sources (depending on the scope of the DSS) to identify problems and determine their solution to facilitate optimal decision making. The biggest benefit of using a DSS is the fact that it helps overcome the problems related to decision making: first, the need to make fast decisions (often in a context that is rapidly changing), but also lack of knowledge, wrong calculations, not considering alternatives, etc.[71]

6.2.1 DSS components

A DSS is made up of three main components:

- **Model Management System**: stores models used for decision-making. A model is the representation of some concept, event, or situation. It is a way to simplify the context in which a decision must be made. There are many types of model according to the field of application of the DSS (for example, in business, models are used to represent variables and their relationships).

- **Knowledge base**: stores and maintains the information that the DSS will use in the decision-making process. This information can come from internal or external sources.
- **User interface**: is the set of tools that allow a user to communicate with the system.[72]

6.2.2 DSS categories

According to the technology used to be developed, Decision Support Systems can be divided into the following categories:

1. **Model-driven DSS**: built on a decision model, it helps analyse different scenarios that meet a set of predefined user requirements.
2. **Data-driven DSS**: it makes decisions based on data coming from internal databases and external databases. It may use *data mining* techniques to identify patterns and trends.
3. **Communication-driven and group DSS**: using various communication tools (such as emails and messages), it allows greater collaboration between users and systems because it allows more users to work on the same task.
4. **Document-driven DSS**: it is a type of information management system that uses data from documents: this type of DSS helps users to access webpages and databases in order to retrieve documents which include policies or procedures, corporate records, etc.
5. **Knowledge-driven DSS**: it uses data coming from a knowledge base which is continuously updated by a *knowledge management system*. This type of DSS has an interface enabling users to query the knowledge base, an inference engine, that interacts with the knowledge base to derive new knowledge used to support decisions, and the knowledge base, containing knowledge encoded in rules.
6. **Web-based DSS**: it extends its capabilities by using the World Wide Web and the Internet.

Intelligent Decision Support Systems

A DSS that uses **Artificial Intelligence** is known as Intelligent Decision Support System (IDSS). Artificial intelligence is used to mine and process large amounts of data, enabling better decision-making and making better recommendations. An IDSS has been designed to act similarly to a human (by analyzing data, it identifies

patterns and trends to emulate human decision making capabilities): it is able to identify troubleshooting issues and evaluates possible solutions. It may include other advanced capabilities, such as **machine learning** and **data mining**.[73]

6.2.3 Advantages and Disadvantages of using a DSS

As already mentioned, Decision Support Systems have been introduced to help companies in strategic decision-making activities. DSS are evolving and more and more efficient systems have been developed over the years. Anyway, these systems are not perfect.

6.2.4 Advantages of a DSS

Using a Decision Support System creates a competitive advantage for the organization:

- It reduces the time needed to perform a **decision-making cycle**: some of the objectives of a DSS are simplifying things and saving time. The DSS reduces the time needed to analyze the data and compare the possible courses of actions and quickly makes decisions analyzing the pros and cons;
- It increases **Data accuracy**: a DSS analyzes and interprets data objectively (without being influenced by human bias), leading to better decision-making;
- It **reduces the cost of decision making**: since the steps of decision making are collected in a unique computer system, the DSS reduces significantly the costs of gathering, processing and analyzing data.[74]

6.2.5 Disadvantages of a DSS

Among the disadvantages of a Decision Support System there are:

- Information overload: the efficiency of a DSS could increase if only useful and necessary information is considered in the decision-making process.
- Too much dependence on DSS: it can happen that decision makers completely rely on DSS and stop using their knowledge for decision-making.
- Devaluation of subjectivity: a DSS allows making objective decisions, but subjectivity should not be completely rejected.
- Cost of development: the cost of decision-making decreases when a DSS is used, but the development and implementation of a DSS require a large amount of money.[75]

6.2.6 Examples of DSS

Decision Support systems can be built in any knowledge domain and there are many industries that use them. Before we focus on the use of DSSs in cybersecurity, some examples of other DSSs are provided.

Clinical Decision Support System

A clinical DSS (CDSS) was created with the objective of improving health and health care by making medical decisions (to give advice) using targeted clinical knowledge and patient information, intelligently filtered or presented at the appropriate time. A CDSS has been defined as *DSS that links health observations with health knowledge to influence health choices by clinicians for improved health care*[1]. Among the tools used in the decision-making process, there are computerized alerts and reminders to care patients, clinical guidelines, patient data records and summaries, diagnostic support, etc. The first CDSS was simply used to make decisions for the clinician: the clinician had to input some information about a patient and wait for the CDSS to output the right choice. Modern CDSS are intended to analyse patients' data and make suggestions. The clinician then uses his knowledge to select the best suggestion.
There are two main types of CDSS:

- *knowledge-based*: it is composed by a knowledge base, an inference engine, and a mechanism to communicate. The system retrieve data and evaluates it by applying rules (such as IF-THEN statements) in order to produce an action or output;
- *non-knowledge-based*: the system still requires a data source, but decisions are made using Artificial Intelligence, Machine Learning or statistical pattern recognition (it does not require rules or expert inputs).[76]

Agricultural Decision Support System

The Agricultural DSS (ADSS) has the purpose of improving the efficiency of agricultural activities. It is a human-computer system that analyzes data from various sources to provide farmers with advice to support them in their decision-making under different circumstances. It does not give direct instructions or commands, it just suggests possible actions, giving the farmer the possibility to make the final choice. The suggestions provided may be for ongoing activities but also for future tasks (to improve performance). Among the technologies used,

[1]CDSS definition provided by Robert Hayward of the Center for Heath Evidence

we can find Artificial Intelligence, Internet of Things, and Cloud Computing. An ADSS may also be able to estimate the trend of the price in the trading markets; As a consequence, things such as the time for irrigation, pollination, fertilization, harvesting, and sales are controlled in order to ensure the highest profit.[77]

GPS route planning Decision Support System

This type of DSS compares different routes, considering factors such as distance, driving time, cost, and even traffic in real time. As in the other DSS described, the GPS route planning DSS not only gives the best option, but shows all the possible routes to let the user decide which one to use.

ERP dashboards Decision Support System

The ERP (Enterprise Resource Planning) is a software that is used to manage and integrate business functions (such as finance, human resources, supply chain, inventory management, etc.). ERP dashboards can use a DSS to help managing business processes, monitoring the current business performances against selected goals and identify areas of improvement.[73]

6.3 Decision Support Systems in cybersecurity

DSSs are widely applied also in the field of cybersecurity. They can be used for several tasks, such as vulnerability assessment and risk assessment or countermeasure selection to prevent cyber attacks or to react to ongoing cyber attacks, etc. When based on Artificial Intelligence, they are particularly effective in monitoring and identifying suspicious events, allowing a major protection of computer systems.

6.3.1 Examples of DSS for security countermeasures selection

DSS for cybersecurity investment challenge

One of the challenges faced by security managers is how to allocate resources and how much to invest in securing the organization. Several research studies have been carried out to develop a DSS capable of helping managers select suitable countermeasures without exceeding a predefined budget.
In [78], the objective is to maximize the expected benefit from the implementation of security measures and minimize the total expected loss (due to the security incident

and the negative side effects caused by the defense mechanism implemented[2]). The solution proposes a quantitative risk assessment considering two factors: *Direct Cost*, which corresponds to the cost of implementing a defense and must be within the organization's budget, and *Indirect Cost*, which corresponds to the impact that defense has on business. The DSS has been developed considering three different methodologies: **game theory** (to model the interaction between (*Defender (D)* and *Attacker (A)*), the **Knapsack algorithm** and a **hybrid** solution based on the combination between game theory and knapsack.
In [79], the implemented solution takes inspiration from another DSS ([80]), but it proposes a different approach since risk is measured using financial engineering percentile measures of *Value-at-Risk VaR* and *Conditional-Value-at-Risk CVaR*. Two approaches are considered to select the countermeasure portfolio: a **single objective approach**, where the objective is to minimize the expected loss (plus the required budget) or the expected worst-case loss (plus the required budget), and a **bi-objective approach**, where both values must be minimized to balance the required budget and expected cost with risk tolerance.

DSS for optimal countermeasures selection

Another usage of DSS in cybersecurity is related to optimal countermeasure selection, not from the point of view of investments, but considering the its effectiveness. In [81], a probabilistic attack graph is used: nodes represent the stages in which the attacker can go, and edges represent vulnerabilities that, if exploited, allow the attacker to reach a precise edge. The proposed solution is based on the *Bayesian Stackelberg games* over probabilistic attack graphs to select the optimal set of countermeasures. Two optimization modules are created: one for selecting preventive countermeasures to minimize the potential security risk before an attack is performed, and the other for selecting countermeasures to minimize the ongoing security risk when an attack occurs.

6.3.2 Ontologies and Decision Support Systems

Decision Support Systems for cybersecurity incident mitigation need to perform a rapid and reliable security assessment and countermeasure generation. For this reason, some solutions propose representing the information used for decision-making in the form of ontologies. Threat intelligence is usually modeled through ontology, and DSSs often use data coming from threat intelligence. In addition,

[2]Defense mechanisms may have a negative impact on the system, such as reducing the speed in which normal tasks are executed, or preventing users performing their usual tasks for a period of time, etc.

ontologies are used for reasoning, which can also be applied in decision-making. Several examples of ontology used in the field of information security exist: López de Vergara et al. use ontology for mapping alerts to attacks and consequently to identify the policies to be applied for mitigating the threats. Razzaq et al. propose an ontology for intrusion detection, able to detect zero-day attacks with low false positive rates. Granadillo et al. propose an ontology to represent the information manipulated by SIEM systems and use this model to perform reasoning based on rule templates. Kotenko et al. propose an ontological approach to create a SIEM data repository, used for logic reasoning and to produce countermeasures for security incidents.[82]
In [82], the developed DSS is based on a new approach, where the ontology not only models alerts, attacks, and countermeasures, but focuses on the security metrics used in the countermeasure selection process. A security metric can be defined as a tool used to facilitate decision-making since it reports performance-related data, in particular security goals and objectives[3]. Many metrics have been defined, which can be related to the attacker, the attack, the countermeasures, the network, vulnerabilities, etc. During the countermeasure selection process, the DSS evaluates the instances of class *SystemMetric* (containing the metrics related to the attack) and the instances of the class *CostMetric* (containing the metrics related to the countermeasures). Then, using logical reasoning, conclusions are drawn about the attack, the resources attacked, and possible countermeasures (considering costs and benefits).

6.3.3 Metrics

Among the existing security metrics that have been defined, we can find the followings: attack impact, success rate of the attack, skills level of the attacker, vulnerability criticality, attack probability, expected loss from an attack, Return of Investment from the reaction to the attack, effectiveness and side effects of the countermeasure, etc.[82] Metrics have been largely investigated and used in this research work, for this reason, the last part of this chapter will describe some of the metrics taken into consideration and then used for the definition of the SWRL rules for inferring the optimal course of action. All metrics considered have been defined and described in detail in the doctoral thesis developed by Verónica Mateos ([55]), who proposed an automatic intrusion response system based on ontologies. Among the metrics defined in [55], we can find:

- **Alert reliability**: it is a value used to describe how reliable the received alert is, so that the alert truly corresponds to the specified attack. It is calculated

[3]Source: NIST SP 800-55

considering the type of attack included in the alert, the anomalies detected on the network, and the reliability of the IDS that sends the alert.

- **Asset importance level**: it specifies how important is the attacked asset and how the attack compromises its confidentiality, availability, integrity, and authenticity. This parameter is calculated considering also the cost that the organization has to face if this asset is partially or completely compromised. It may have three values: *High, Medium and Low*. This parameter will play an important role in the optimal countermeasure selection process.

- **Intrusion Impact**: it describes the potential damage caused by a threat to an asset. It may be expressed quantitatively or qualitatively; in [55] has been calculated considering the severity of the attack (which corresponds to the threat level associated to the intrusion), the effect of the intrusion on the four security dimensions (confidentiality, integrity, availability and authenticity) and the exposure factor (how long the asset will be under threat).

- **Cost of the response action**: it is given by the sum of two values:

  - **Deployment Cost**: it represents the cost that the organization has to face to develop the response action (in terms of resources consumed). It is calculated considering the cost of usage of the resource, the time in which the resource is used and the number of resources used.

  - **Impact**: it is the damage caused by the response to the asset. It is calculated considering the severity of the response, the effect of the response on the four security dimensions, and the number of assets affected by the response.

- **Response action efficiency**: this value is calculated once the response has been inferred and executed. Specifies whether the chosen response has been able to mitigate the attack or not. It is calculated considering the number of times the same response has been inferred and the partial efficiency associated with the i-th execution of the response.

Apart from these metrics used to define parameters that characterize the asset, the intrusion and the response, [55] defines other metrics for the inference of the optimal response. Several metrics have been proposed with the goal of applying them dynamically according to the specific intrusion scenario. In particular, what allows one to select the most appropriate metric is the level of importance of the compromised asset.

Damage reduction metric

This metric does not depend on the level of importance of the asset, so it must always be applied. Allows one to balance the attack impact and the response impact. Applying this metric, the system automatically discards the responses where the the response impact is higher than the attack impact multiplied by the intrusion reliability and the IDS reliability. Since this metric does not depend on the asset's importance level, it can be applied together with one of the following metrics.

Minimum cost metric

This metric can be applied only if the level of importance of the asset is low (or maybe medium), so if the compromised asset is not as relevant for the organization. If the level of importance is high, this metric should not be used.
The goal is to minimize the total cost related to the development and execution of the response action. The total cost is the sum of the impact (damage caused on the resources of the system) and the development cost (in terms of the number of resources needed for the development of the response) of the response. The response cost is strictly related to the concept of response complexity; for this reason, if two responses have the same cost, the one with the lowest complexity is chosen.

Maximum severity or maximum efficiency metric

This metric is applied when the level of importance of the asset is high (critical to the operations of the organization). The goal of the metric is to maximize the severity and the efficiency of the response action, in order to infer the response with the highest efficiency as the optimum response.
This metric depends on the results of the previous executions of the response, the severity of the intrusion and the severity of the response.If the level of importance of the asset is high, first the damage reduction metric is applied, then the maximum severity rule is applied on the subset of responses selected with the damage reduction rule.

Chapter 7

Proposal for a Decision Support System architecture based on ontologies

7.1 Project proposal

This chapter will start presenting the Decision Support System developed in this thesis project: the goal is to create a module that, given an attack, a set of Courses of Actions, and a security objective, is able to select:

- one or more optimal courses of actions, identified considering the objective specified by the security manager of the organization;
- one optimal course of action recommended by the system, identified considering specific metrics. This course of action is the one that best mitigates the attack considering the values of the parameters describing the attack, the compromised asset and the courses of actions, but not the necessities of the organization.

A **Course of Action (CoA)** is a sequence of actions (countermeasures) that are meant to react to the attack notified by the alert in order to mitigate its malicious effects on the system. A CoA can have the objective of mitigating an ongoing attack, or it can be executed during a vulnerability management process to prevent future attacks. In this project, CoAs are considered as reactions to ongoing attacks. Each CoA is described through parameters (such as deployment cost, complexity, side effects, effectiveness, etc.).

7.2 Architecture

As already mentioned in previous chapters, **ontologies** and **rule languages** have been considered for the development of this DSS. The decision to use ontologies was dictated by several factors: they represent information formally and in a standardized way that can be understood and shared among different applications, they allow reuse of knowledge (in fact, new ontologies can be built from existing ones), and they can be used to perform automatic inference and create new knowledge from existing one. The developed ontology has been specified through **OWL2 ontology language** and structured in a way that allows the use of the knowledge about cyber attacks collected in the Cyber Threat Intelligence framework **MITRE ATT&CK** (presented in Chapter 4). To select the optimal course of action, the DSS uses **Pellet reasoner**, which executes the inference rules specified in **SWRL**. This choice was dictated by several reasons: SWRL has a high level of integration with OWL/OWL2, it allows defining every type of rule in a general way and it has a higher level of expressiveness with respect to other languages (for example, antecedent and consequent of the rule can contain variables, so the condition is more specific and it allows to select exactly the interested individuals) and, last but not least, it is a widespread and constantly evolving language, compatible with the current and future tools used in semantic web.
All these tools have been largely discussed in Chapter 5. The tools used and the way they are related are shown in Figure 7.1.

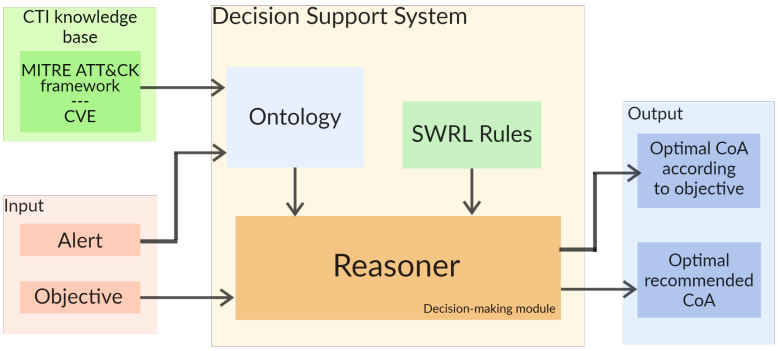

Figure 7.1: DSS architecture

7.3 Requirements

The decision support system is a logical module designed to provide the most efficient Course of Action in response to a determinate incoming alert. For this purpose, it considers as input data a list of candidate reactions which are described through the following parameters:

- Deployment Cost
- Deployment Effort
- Installation Complexity
- Operation Complexity
- Required Human Skills
- Time to be up and run
- Effectiveness
- Severity
- Side Effects

As it is intended to provide the optimal CoA according to some user defined objectives, it also considers as input an Objective object which contains the following parameters:

- ID of the alert to which this objective is associated
- a boolean value for each of the following sub-goals:
  - Damage reduction
  - Maximum complexity allowed
  - Maximum deployment cost allowed
  - Maximum human skills allowed
  - Maximum time-to-run allowed
- an integer/float/string to specify the maximum value allowed for every sub-goal:
  - Maximum installation complexity value
  - Maximum operation complexity value

- Maximum deployment cost value
- Maximum deployment effort value
- Maximum human skills value
- Maximum time-to-run value

7.4 Workflow

The developed program can be divided into two modules: the first module has the goal of creating CoAs, while the second infers the optimal course of action. These processes are described more in detail by the workflow diagram in Figure 7.2

7.4.1 Courses Of Actions creation module

As we will see in Chapter 8, which provides a detailed description of the ontology and how the data are related, the idea is that each alert notifies a list of attack techniques (which together build the attack) and that each attack technique can be mitigated by one or more countermeasures. Initially, the ontology does not contain CoAs, but only countermeasures that can mitigate each technique. The steps performed in this module are the following:

1. The alerts are retrieved from the ontology. To simplify the problem, we consider that a list of alerts is already saved in the ontology. Although in a realistic environment, the alerts are generated by some external module (such as the intrusion detection system).
2. For each alert:
   (a) The list of notified attack techniques is retrieved.
   (b) For each attack technique, the list of countermeasures that can mitigate it is retrieved.
   (c) Countermeasures are combined to create a list of CoAs that will be associated with the current alert.
3. Once the CoAs have been created and added to the ontology, the decision-making module is executed.

Function that creates CoAs

This is the main function of this module. Given an alert, the function receives one list of countermeasures for every notified attack technique (as we have said, each attack technique can be mitigated by one or more countermeasures). The idea is to

combine the various countermeasures to create a CoA that is able to mitigate all the attack techniques notified by the alert. Since to mitigate an attack technique, there may be more countermeasures, several CoAs will be created.
The algorithm used to create the list of CoAs takes inspiration from the Cartesian product between arrays. In order to explain how it works, an example will be provided.
Let us suppose that we have the alert *A1* that notifies an attack composed of three attack techniques *{T1,T2,T3}*. We know that:

- *T1* can be mitigated by countermeasures *{C1,C2,C3}*
- *T2* can be mitigated by countermeasures *{C4}*
- *T3* can be mitigated by countermeasures *{C5,C6}*.

The function that creates CoAs will iterate on each array of countermeasures and define the following CoAs:

CoA1: *{C1,C4,C5}*
CoA1: *{C2,C4,C5}*
CoA1: *{C3,C4,C5}*
CoA1: *{C1,C4,C6}*
CoA1: *{C2,C4,C6}*
CoA1: *{C3,C4,C6}*

As we can see, each CoA is a set of countermeasures in which each countermeasure is able to mitigate one of the attack techniques.

7.4.2 Decision-Making module

This module is the most relevant, as it is the one in which the optimal CoA is inferred. The steps performed are the following:

1. A list of objectives is received from user input. Each objective is associated to an alert and it is used to infer the optimal CoA to react to that alert.

2. For each alert the system checks if an objective has been specified. As the objective is an object that contains several sub-objectives (the user can specify one or more sub-objectives), if the objective exists, the function checks which sub-objectives have been specified and executes the corresponding rules to infer the optimal CoA for that alert and that sub-objective.

3. Once the optimal CoA for the user objective has been inferred, some general rules are executed in parallel on all the alerts to infer the optimal CoA recommended by the DSS.

4. The system provides as output:

    - Zero or more optimal CoAs depending on the sub-objectives specified by the user: zero if none of the CoAs associated to that alert is compliant with the objective specified by the user, one if the user has specified only one sub-objective and there is at least one CoA compliant with the sub-objective, more than one if the user has specified several sub-objectives and there is at least one CoA compliant with each of the sub-objectives.
    - An optimal CoA recommended by the DSS. It is identified considering the characteristics of the compromised asset, of the attack, and of the CoAs.

At the end of the execution, the user receives one or more optimum CoAs, and by comparing the optimum CoAs obtained and considering his needs, can decide which one to execute to mitigate the attack. The final decision is always taken by the user, the system only provides a suggestion to help the user choose the best CoA.

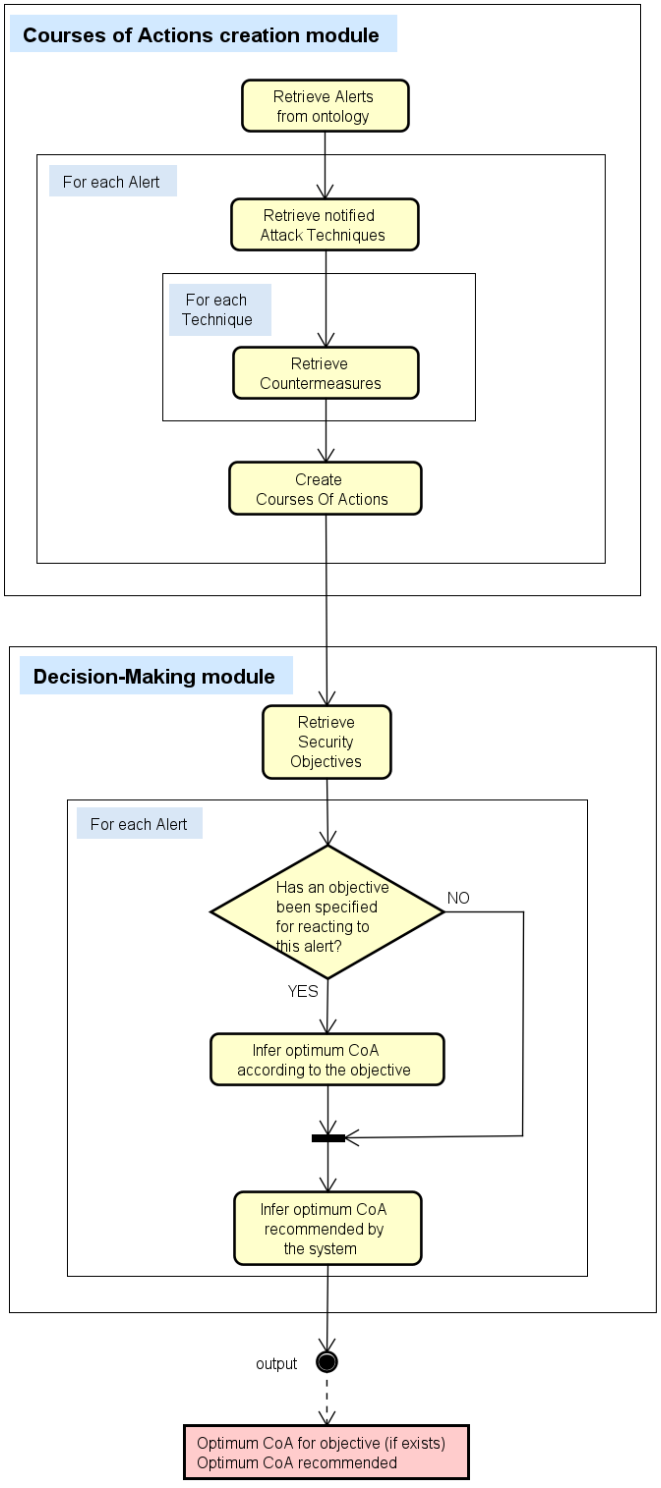

Figure 7.2: DSS workflow

Chapter 8

Proposal for a Decision Support Ontology

The benefits provided by representing knowledge using ontologies have been extensively discussed in previous chapters. The use of ontologies in cyber security is not something new, and in the last years many research works using them to model security domains have been carried out. In the development of this thesis, in-depth research has been conducted to identify existing cyber security ontologies and taxonomies to take inspiration for modeling the realm of interest.
In order to develop an ontology, several methodologies can be followed. In general, since an ontology is meant to describe a reality or a domain, it is important to define the domain of interest in detail: first, an objective is determined, and then a detailed description of the domain of interest according to the defined objective is provided; from this description, the most important terms are highlighted and considered as keywords, which will be translated into concepts related to each other through object properties. Regardless of the methodology used, all ontologies must comply with the following requirements:

- it must be clear and objective in specifying the represented domain;
- it must be coherent and consistent (this aspect can be checked through inference);
- it must be complete, containing both necessary and sufficient conditions in its specification;
- it must be extensible, so that new axioms can be added or it can be used in other contexts;
- its classes must be disjoint;

- it must be composed by independent modules;
- it must describe attributes and properties in a standardized way.[55]

8.1 Tools used to build the ontology

Many tools, applications, and languages are available for building and specifying ontologies. In this section, a brief description of the ones used in this thesis will be provided.

8.1.1 OWL2 and OWL API

As already described, OWL and OWL2 are the most expressive languages to formally describe knowledge on the Semantic Web. Due to its advantages already described in Chapter 5, OWL2 in RDF/XML format is the language used to implement the ontology. The interaction beteween the program developed and the ontology was possible through Java OWL API.

8.1.2 Protégé

Protégé[1] is a free open-source editor and framework to build intelligent systems. It was developed by a team at Stanford University with the objective of providing an efficient tool for building ontology-based applications. It supports various specification formats (such as RDFS, OWL, and OWL2) and provides two options for modeling ontologies:

- Editor *Protégé-Frames*: used to build ontologies based on frames following the OKBC protocol (Open Knowledge Base Connectivity);
- Editor *Protégé-OWL*: used to build ontologies in OWL, OWL2, or RDF. This editor allows editing OWL ontologies that include descriptions of classes, properties, and individuals; it is also possible to install additional plug-ins, such as the one used for executing SWRL rules, use semantic reasoners to perform inference, and evaluate the consistency of the ontology.

Protégé can be easily configured, and its functionalities can be extended through the numerous plug-ins available. The proposed ontology was built using version 5 of Protégé (which already includes some reasoners, such as Pellet, the one used in this project) and the following additional plug-ins:

[1] `https://protege.stanford.edu/`

- **SWRLTab**: it provides a development environment to work with SWRL rules and SQWRL queries.
- **OntoGraph**: it is software created for the graphic visualization of OWL ontologies. It allows for the display of classes and subclasses of the ontology and their relationships (but not their data properties).

8.2 Domain of the decision support ontology

As mentioned, the first step in creating a new ontology is the definition of the domain to be modeled. The domain on which this project focus corresponds to a Decision Support System, which helps security managers in the selection of the optimal Course of Action to mitigate a cyber intrusion (taking into account the necessities of the organization). In particular, the focus is on what happens right after an attack is detected (so the part of discovering a vulnerability and analyzing the system to detect anomalies is not part of this scope). Once an attacker breaks into the system of the organization, an **Alert** is produced to notify that an intrusion as occurred and to give information about the intrusion. This alert is received by the **Decision Support Module** of the system and taken into consideration in the process of inference of the optimal **Course of Action** (action that can mitigate the damage caused by the attack). An attack is described through the set of **Techniques** used in its implementation. A list of attack techniques is created by taking inspiration from the currently known attack techniques collected in the MITRE-ATT&CK knowledge base. A CoA executes a set of **Countermeasures** (one countermeasure for each attack technique to be mitigated) that is taken from a list of countermeasures.
From this description of the domain it was possible to identify some keywords that were used to model the classes of the ontology and the relationships between them.

8.3 Implementation of the decision support ontology

Once described the domain in natural language, it has to be translated into a formal language (OWL2). This section provides a detailed description of the classes, subclasses, and properties that make up the ontology. The ontology contains 3 classes: *Alert*, *AttackTechnique*, *MitigationTechnique*. The Mitigation Technique class contains two subclasses: *Countermeasure* and *CourseOfAction*. Each class presents several properties which define the individuals and instances being part of the class. The ontology developed for the Decision Support System can be seen in Figure 8.1.

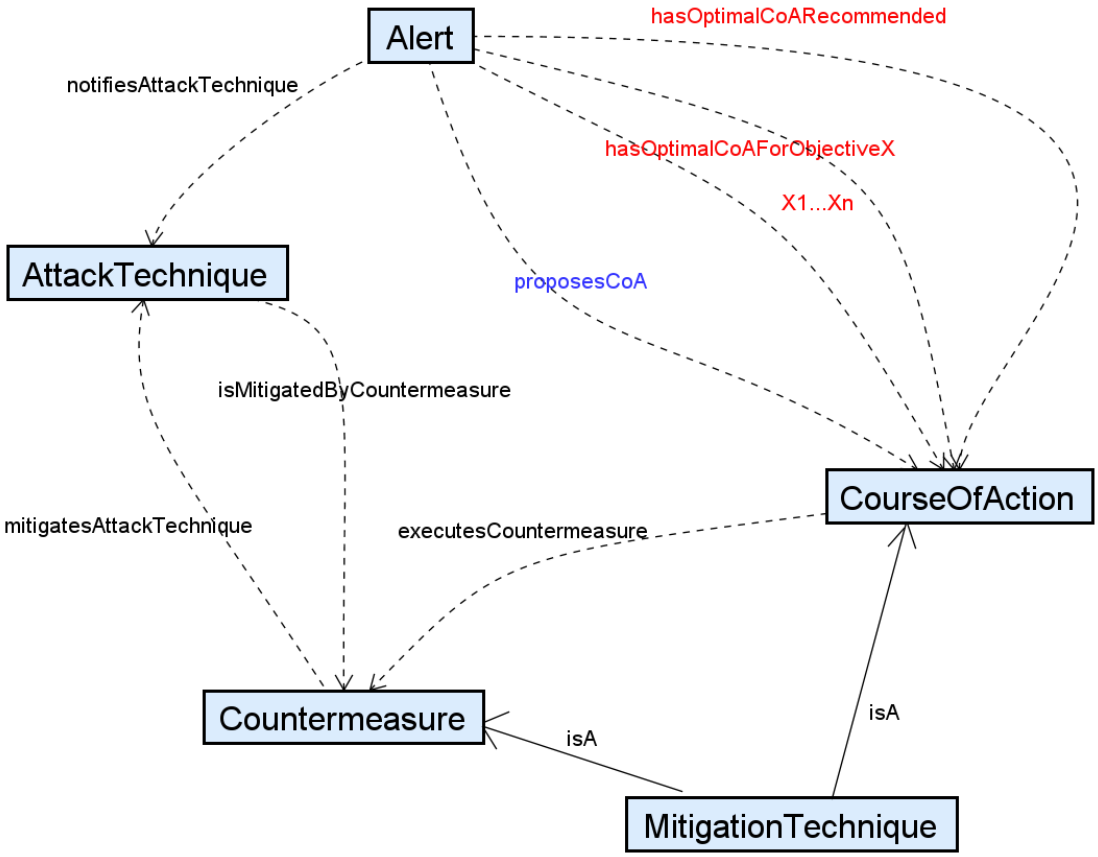

Figure 8.1: Ontology for Decision Support System

8.3.1 Alert

Class used to represent the alerts that notify an attack. Each instance of this class contains information about the attack detected, described through the *datatypes* and *object properties* as shown in Figure 8.2.

- *hasAlertID*: attribute of type string, it is used to uniquely identify the individual of type Alert;
- *hasAssetImportanceLevel*: attribute of type String, describes how important is the compromised asset for the organization; the possible values are restricted to *{Low, Medium, High}*;
- *hasAttackImpact*: attribute of type Integer, describes the damage provoked by the attack on the system; the possible values are restricted to the interval *[0,10]* (0 means no damage, 10 means very high damage);

Alert		
Datatype Properties		
hasAlertID	String	
hasAssetImportanceLevel	String	
hasAttackImpact	Integer	
hasAttackSeverity	String	
hasAttackSeverityNum	Integer	
hasNumberOfPossibleCoAs	Integer	
hasAlertReliability	Float	
hasVulnerabilityExploited	String	
Object Properties		
notifiesAttackTechnique	Instance*	AttackTechnique
proposesCoA	Instance*	CourseOfAction
hasOptimalCoAForComplexityMetric	Instance*	CourseOfAction
hasOptimalCoAForDamageReductionMetric	Instance*	CourseOfAction
hasOptimalCoAForHumanSkillsMetric	Instance*	CourseOfAction
hasOptimalCoAForMonetaryCostMetric	Instance*	CourseOfAction
hasOptimalCoAForTimeMetric	Instance*	CourseOfAction
hasOptimalCoARecommended	Instance*	CourseOfAction

Figure 8.2: Alert class of the DSS ontology

- *hasAttackSeverity*: attribute of type String, it represents the severity associated to the attack. It can take the values *{Low, Medium, High, Critical}*;

- *hasAttackSeverityNum*: attribute of type integer; it is simply the translation of the severity string value into integer (to help in the calculation to identify the optimal CoA). It can take the values *[1,4]*;

- *hasNumberOfPossibleCoAs*: attribute of type integer, at the beginning it is empty and it is filled with the number of CoAs able to mitigate the attack notified by this alert once they are created;

- *hasAlertReliability*: attribute of type float, it indicates the percentage of reliability in the alert (does the alert really corresponds to the detected attack?). The value is restricted to the interval *[0,1]*;

- *hasVulnerabilityExploited*: attribute of type string, it contains the CVE identifier of the expolited vulnerability;

- *notifiesAttackTechnique*: one or more. It is a relation with range *AttackTechnique*. The attack notified by the alert is considered the set of techniques used

by the attacker to compromise the system[2];

- *proposesCoA*: one or more. It is a relation with range *CourseOfAction* and represents all possible CoAs that can mitigate the attack notified by this alert. When a a new CoA for the current alert is created, it is linked to the alert through this relationship;

- *hasOptimalCoAForComplexityMetric*: zero or one. It is a relation with range *CourseOfAction*. When the user specifies that the CoA's complexity doesn't have to exceed a certain value, this property is the one that links the alert to the optimal CoA according to this objective.;

- *hasOptimalCoAForDamageReductionMetric*: zero or one. It is a relation with range *CourseOfAction*. When the user specifies that the objective is to reduce as much as possible the damage provoked by the attack, this property is the one that links the alert to the optimal CoA according to this objective;

- *hasOptimalCoAForHumanSkillsMetric*: zero or one. It is a relation with range *CourseOfAction*. When the user specifies that the CoA's value for the required human skills does not have to exceed a certain value, this property is the one that links the alert to the optimal CoA according to this objective;

- *hasOptimalCoAForMonetaryCostMetric*: zero or one. It is a relation with range *CourseOfAction*. When the user specifies that the CoA's cost doesn't have to exceed a certain value, this property is the one that links the alert to the optimal CoA according to this objective;

- *hasOptimalCoAForTimeMetric*: zero or one. It is a relation with range *CourseOfAction*. When the user specifies that the time necessary to build the CoA does not have to exceed a certain value, this property is the one that links the alert to the optimal CoA according to this objective;

- *hasOptimalCoARecommended*: exactly one. It is a relation with range *CourseOfAction*. Regardless of what the user's objective is, the system infers the optimal CoA simply considering the parameters of the compromised asset, the CoA, and the attack. This property is the one that links the alert to the optimal CoA recommended by the system;

[2]This way of describing an attack comes from the CTI framework MITRE-ATT&CK

8.3.2 MitigationTechnique

Class that models actions intended to mitigate a single attack technique, and a list of attack techniques (attacks). It contains two subclasses (**Countermeasure** and **CourseOfAction**), as shown in Figure 8.3.

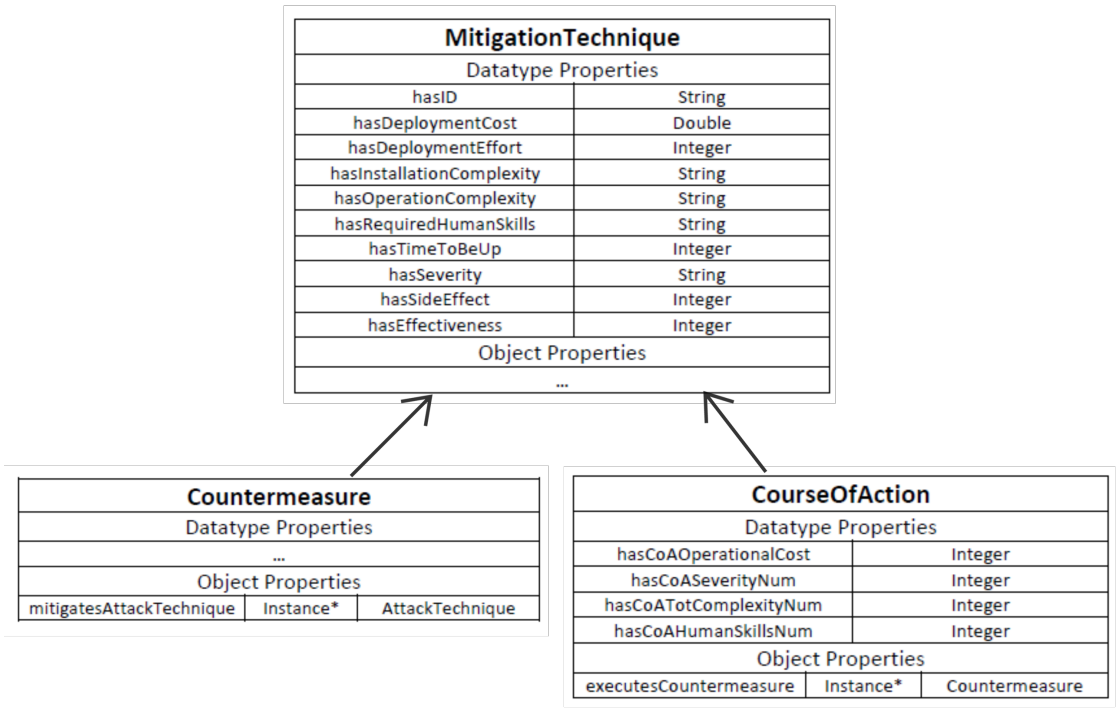

Figure 8.3: MitigationTechnique class of the DSS ontology

- *hasID*: attribute of type string, it is used to uniquely identify the individual of type MitigationTechnique (which can be a Countermeasure or a CourseOfAction.
- *hasDeploymentCost*: attribute of type double, it represent the cost (in terms of money spent) to develop the mitigation technique. Taking into account a CoA, it is calculated by summing the deployment costs of the countermeasures it executes.
- *hasDeploymentEffort*: attribute of type integer, it represents the effort made to develop the mitigation. Its value is restricted to the range *[0,10]*. Taking into consideration a CoA, it is calculated by executing an average of the efforts of the countermeasures it executes.

- *hasOperationComplexity*: attribute of type string, it describes the complexity of the operation performed by the mitigation technique. It can take the values *{None, Low, Medium, High, Extreme}*. Taking into account a CoA, this value is calculated by transforming the complexity of the countermeasures it executes into integers, executing their average, and then transforming the value into the corresponding string.

- *hasInstallationComplexity*: attribute of type string, it describes the complexity of the installation of the mitigation technique. It can take the values *{None, Low, Medium, High, Extreme}*. Taking into account a CoA, this value is calculated by transforming the complexity of the countermeasures it executes into integers, executing their average, and then transforming the value into the corresponding string.

- *hasRequiredHumanSkills*: attribute of type string, it specifies what level of exepertise the security managers need to have in order to be able to execute this mitigation technique. It can take the values *{None, Medium, High}*. Taking into account a CoA, this value is calculated by transforming into integer the required human skills of the countermeasures it executes, executing their average, and then transforming the value into the corresponding string.

- *hasTimeToBeUp*: attribute of type integer, it represents the time needed to develop and install the mitigation technique.

- *hasSeverity*: attribute of type string, it represents the severity associated with the mitigation technique. It can take the values *{Low, Medium, High, Critical}*. Taking into account a CoA, this value is calculated by transforming the severity of the countermeasures it executes in integer, executing their average and then transforming the value into the corresponding string.

- *hasSideEffect*: attribute of type integer, it represents the negative impact that the mitigation technique has on the system when it is executed. Its value is restricted to the range *[0,10]*. Taking into consideration a CoA, this value is calculated by performing an average of the side effect of the countermeasures it executes.

- *hasEffectiveness*: attribute of type integer, it represents the ability of the mitigation technique to successfully mitigate the intrusion. Its value is restricted to the range *[0,10]*. Considering a CoA, this value is calculated by performing an average of the side effect of the countermeasures it executes.

- The subclass **Countermeasure** has an additional *object property*:

  - *mitigatesAttackTechnique*: zero or more. It is a relation with range *AttackTechnique* and it is used to specify which attack techniques can be mitigated by each countermeasure.

- The subclass **CourseOfAction** has some additional *datatype properties* and *object properties*:

  - *hasCoAOperationalCost*: attribute of type integer, it represents an overall cost in terms of money and resource used. It is calculated by performing an average between the deployment cost (previously transformed into a value in the range [0,10]) and the deployment effort. This value is restricted to the range *[0,10]* and is used in the security metrics to select the optimal CoA.
  - *hasCoASeverityNum*: attribute of type integer; it is the translation of the severity into integer value, to simplify the calculation of some parameters used in security metrics.
  - *hasCoATotComplexityNum*: attribute of type integer, represents an overall complexity associated with CoA, and is calculated by performing an average of installation and operation complexity. Its value is restricted to the range *[1,5]*.
  - *hasCoAHumanSkillsNum*: attribute of type integer,it is simply the translation of the required human skills in integer value, in order to simplify the calculation of some parameter used in security metrics.
  - *executesCountermeasure*: one or more. It is a relation with range *Countermeasure* and it links the Course of Action to the countermeasures it executes.

8.3.3 AttackTechnique

Class that models the techniques that are executed to perform an attack. An attack is described in terms of the techniques it executes. This class contains only one data property (*hasAttackTechniqueID*) and one inferred object property (*isMitigatedByCountermeasure*).

8.4 Specification of restrictions and conditions through axioms

Axioms allow to specify rules and restriction on the modeled domain. Several types of axioms exist: subclass axiom, equivalence axiom, conjunction or disjunction

axioms, axioms to specify restriction over values of datatype properties, etc. In this section, some of the axioms of the DSS ontology will be presented in RDF syntax.

8.4.1 Subclass axiom

Axiom used to state that a class C1 is a subclass of C2. As an example, Figure 8.4 shows that class *Countermeasure* is a subclass of *MitigationTechnique*. All the subclasses included in the ontology are expressed in the same way.

8.4.2 Disjoint classes axiom

Axiom used to express the disjunction between two classes. In the DSS ontology, all classes at the same hierarchical level are disjoint. The example in Figure 8.4, in addition to the subClassOf axiom, also contains the disjointClasses axiom, to specify that the classes *Countermeasure* and *CourseOfAction* are disjoint.

```
<owl:Class rdf:about="http://www.semanticweb.org/franc/ontologies/2022/1/decisionSupportSystem4#Countermeasure">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/franc/ontologies/2022/1/decisionSupportSystem4#MitigationTechnique"/>
    <owl:disjointWith rdf:resource="http://www.semanticweb.org/franc/ontologies/2022/1/decisionSupportSystem4#CourseOfAction"/>
</owl:Class>
```

Figure 8.4: Example: subClassOf and disjointClasses axioms

8.4.3 Data property range axiom

Axiom used to specify that the value of a data property is restricted to a precise range of values. For example, this axiom has been used to specify that the effectiveness value of a countermeasure is an integer in the range [0,10] (as shown in Figure 8.5).

8.5 Evaluation

Once the conceptual model has been specified in formal language, it is necessary to check the consistency and coherence of the ontology. In order to evaluate these two properties, the plug-ins available in Protégé have been used. In particular, the Pellet reasoner was used to check the consistency of the ontology and the potential unsatisfiability of classes and properties. The results have confirmed that the ontology satisfies the requirements.

```
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/franc/ontologies/2022/1/decisionSupportSystem4#hasCountermeasureEffectiveness">
    <rdfs:domain rdf:resource="http://www.semanticweb.org/franc/ontologies/2022/1/decisionSupportSystem4#Countermeasure"/>
    <rdfs:domain rdf:resource="http://www.semanticweb.org/franc/ontologies/2022/1/decisionSupportSystem4#MitigationTechnique"/>
    <rdfs:range>
        <rdfs:Datatype>
            <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
            <owl:withRestrictions rdf:parseType="Collection">
                <rdf:Description>
                    <xsd:minInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</xsd:minInclusive>
                </rdf:Description>
                <rdf:Description>
                    <xsd:maxInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10</xsd:maxInclusive>
                </rdf:Description>
            </owl:withRestrictions>
        </rdfs:Datatype>
    </rdfs:range>
</owl:DatatypeProperty>
```

Figure 8.5: Example: data property range axiom

Chapter 9

Proposal for security metrics and their specification by SWRL rules

In the Decision Support System process described so far, there are several parameters that need to be evaluated; they are mainly related to the alert, the attack and, in particular, the course of action. These parameters are contained in the ontology as datatype properties of individuals of the various classes. It is possible to distinguish between two types of metric:

- Metrics executed externally, automatically or manually by the system administrator. The result of these metrics corresponds to the parameters used by the reasoner in the execution of the inference rules. Examples of these metrics are the level of importance of the compromised asset, the impact and severity of the attack, the cost, complexity, etc. of the courses of action.

- Metrics that influence the behaviour of the reasoner: they are not simply parameters used in the rule, but they take part in the construction of the rule. Examples of these metrics are the ones used to infer the optimal course of action.

The metrics described in Chapter 6 have been taken into consideration for the definition and the creation of the metrics used in this project. This chapter will provide a presentation and a description of the metrics used and their translation in SWRL rules.

9.1 Parameters used by the SWRL rules

Some of the datatype properties of the classes Alert, Countermeasure and Course of Action are used as parameters in the SWRL rules and, according to the values they take, influence the selection of the optimal CoA. The scope of this work does not include the calculation of these parameters (the system was supposed to receive directly the countermeasures and the list of courses of actions with the parameters already calculated and set). But despite this, research work has been conducted to understand how these parameters can be measured.

Asset Importance Level

This parameter plays an important role in the selection of the optimal course of action: the level of importance of the attacked asset defines how the intrusion compromises the confidentiality, integrity, availability and authenticity of the asset. The metric used to evaluate this property was taken from [55] where a scale of three values is proposed: **High** (serious or very serious damage), **Medium** (significant damage) and **Low** (minor or insignificant damage). This metric is present in the ontology through the property *hasAssetImportanceLevel* in the class *Alert*.

Attack Impact

This value can be expressed with qualitative or quantitative values. Various metrics have been proposed for the calculation of the attack impact:

- It can be calculated in terms of system resources availability after the attack: the lower the availability, the higher the impact.
- It can be calculated in terms of level of importance of every compromised asset and the threat level of the intrusion;
- It can be considered as the sum of the products between the impact of the attack on the four security dimensions (confidentiality, integrity, availability, and authenticity) and the level of importance of the asset.
- It can be considered as the product between attack severity (level of threat associated with the intrusion), the effect of the intrusion on the compromised asset, and the exposure factor (period of time in which the asset is under threat).[55]

The metric used to specify the attack impact in this project is quantitative and is based on the scale of integer values [0,10], where 0 corresponds to the lowest impact and 10 to the highest. It is represented in the ontology through the property *hasAttackImpact* of the class *Alert*.

Side effect

It is the damage caused by the countermeasure or the CoA in the system. It can be expressed in the same way of the attack impact. In this work, there is a distinction between the side effect of the countermeasures and of the CoAs. In particular, since the CoA executes a set of countermeasures, its side effect is calculated by performing an average of the side effects of the countermeasures. In both cases, this metric takes the integer values in the interval [0,10] and it is represented in the ontology through the property *hasSideEffect* of the class *MitigationTechnique*.

Deployment Cost

It can be considered as the financial cost associated to the deployment of the countermeasure or the CoA. The deployment cost of the countermeasure is already set, while the deployment cost of the CoA is calculated as the sum of the costs of the countermeasures it executes. This metric is presented in the ontology through the property *hasDeploymentCost* of the class *MitigationTechnique*.

Operational Cost

This value is a general cost associated with the CoA and is calculated considering the deployment cost and the deployment effort of the CoA. In particular, the deployment cost is first scaled to a value in the interval [0,10], then the average between this cost and the deployment effort is executed. This metric is restricted to the integer values in the interval [0,10] and is presented in the ontology through the property *hasCoAOperationalCost* of the class *CourseOfAction*.

Total Complexity

Parameter that combines the two properties of installation complexity and operation complexity to calculate a unique complexity value for the CoA. It is given by the average between installation and operation complexity (after their values in string are translated into an integer in the range [1,5]). This metric is presented in the ontology through the property *hasCoATotComplexity* and *hasCoATotComplexityNum* (which is the corresponding value in integer, useful for the execution of the inference rules) of the class *CourseOfAction*.

Effectiveness

This value represents the ability of a mitigation technique to successfully mitigate an attack. It assumes integer values in the range [0,10] (0 if the technique is not able to mitigate the attack, 10 if it can mitigate completely the attack). The countermeasures effectiveness is already set, while the effectiveness of a CoA is calculated

by performing an average of the effectiveness of the countermeasures it executes. This metric is presented in the ontology through the property *hasEffectiveness* of the class *MitigationTechnique*.

Required Human Skills

Parameter that indicates the level of expertise required in order to be able to manage the execution of the countermeasure or the course of action selected. This value is already set for countermeasures, while for CoAs created, it is calculated by performing an average of the required human skills of the countermeasures, and it can take three possible values (*Low, Medium, High*). This metric is presented in the ontology through the properties *hasRequiredHumanSkills* of the class *MitigationTechnique* and *hasRequiredHumanSkillsNum* (which is the corresponding value in integer) of the class *CourseOfAction*.

Time to be up and run

Parameter that describes how long it takes to create and install the mitigation technique. This value is already set for the countermeasures, while for the CoAs it is calculated by summing the time to be up of the various countermeasures it executes. This metric is presented in the ontology through the property *hasTimeToBeUp* of the class *MitigationTechnique*.

9.2 Metrics used to infer the optimal CoA

The metrics described in this section are the ones that directly influence the behaviour of the reasoner. They can be distinguished in two classes: metrics created to identify the optimal course of action taking into account the user security objectives and metrics proposed by the system to identify the recommended optimal course of action, independently of the specified security objective.

9.2.1 Metrics for the optimal CoA according to security objectives

Damage Reduction metric

Metric that establishes a balance between the impact of the attack and the side effects of CoA (corresponding to the negative impact of the mitigation technique). The equation that satisfies this metric is the following:

$$AttackImpact \geq CoASideEffect \tag{9.1}$$

This metric is similar to the damage reduction metric proposed by Stackhanova in [83], but here the alert reliability is not used since it is supposed that the alert received by the DSS is for sure associated to the ongoing attack. When this metric is applied, the Decision Support System discards all CoAs whose side effects are greater than the attack impact. The metric depends on two parameters (values of the properties *hasAttackImpact* of the class *Alert*, and *hasCoASideEffect* of the class *CourseOfAction*), which were previously calculated according to the corresponding metrics. If several CoAs have the same side effect, the one with the greatest effectiveness (value of the property *hasCoAEffectiveness* of the class *CourseOfAction*) will be selected.

Minimum deployment cost metric

Metric that has the objective of minimizing the deployment cost associated with a CoA. It satisfies the following equations:

$$deploymentCost \leq maxDeploymentCost \tag{9.2}$$

$$min\,\{deploymentCost\} \tag{9.3}$$

When this metric is applied, first the CoAs with a deployment cost value greater than the maximum deployment cost are discarded, then the CoA with the minimum deployment cost is identified among the remaining CoAs and considered as the optimal. The parameter *maxDeploymentCost* is defined in the security objective specified by the user and it represents the maximum value of financial resources that the organization is able to allocate to mitigate the attack. If various CoAs have the same deployment cost, the one with the lowest deployment effort will be selected. This value is contained in the property *hasCoADeploymentEffort* of class *CourseOfAction* and it is previously calculated according to the corresponding metric.

Minimum complexity metric

Metric that has the objective of minimizing the overall complexity associated with a CoA. It satisfies the following equations:

$$\begin{aligned}(installationComplexity + operationComplexity) \leq \\ (maxInstallationComplexity + maxOperationComplexity)\end{aligned} \tag{9.4}$$

$$min\,\{installationComplexity + operationComplexity\} \tag{9.5}$$

When this metric is applied, first the CoAs with a total complexity value greater than the maximum total complexity are discarded, then the CoA with the minimum

total complexity is identified among the remaining CoAs and considered as optimal. The parameters *maxInstallationComplexity* and *maxOperationComplexity* are defined in the security objective specified by the user and they represent the maximum value of installation and operation complexity that the system of the organization is able to handle to mitigate the attack. The parameters used are contained in the data properties *hasCoAInstallationComplexity* and *hasCoAOperationComplexity* of the class *CourseOfAction* and are calculated according to the corresponding metrics; their sum is presented in the ontology through the properties *hasCoATotComplexity* and *hasCoATotComplexityNum* (which is the total complexity translated into integer). It is important to specify that since these parameters are all strings (values *{None, Low, Medium, High, Extreme}*), the calculations are made after they are translated into integers (values *{[1,5]}*). If various CoAs have the same total complexity, the one with the lowest deployment cost will be selected.

Minimum required human skills metric

Metric that has the objective of minimizing the level of expertise required to manage the system and execute the CoA. It satisfies the following equations:

$$requiredHumanSkills \leq maxHumanSkills \tag{9.6}$$

$$min\{requiredHumanSkills\} \tag{9.7}$$

When this metric is applied, first the CoAs wich require a human skills value greater than the maximum allowed are discarded, then the CoA with the lowest required human skills value is selected as optimal. The parameter *maxHumanSkills* is defined in the security objective specified by the user and it represent the maximum level of expertise of the users of the organization that will manage the system and the execution of the CoA. The other parameter of the equation is contained in the property *hasCoARequiredHumanSkills* of the class *CourseOfAction* and represents the level of expertise that the user should have to be able to execute a precise CoA. It is important to specify that since these parameters are all strings (values *{None, Medium, High}*), the calculations are made after they are translated into integers (values *{[1,3]}*). If various CoAs have the same human skills required, the one with the lowest deployment cost will be selected as optimal.

Minimum time to be up and run metric

Metric that has the objective of minimizing the time required to counter an attack. It satisfies the following equations:

$$timeToBeUpAndRun \leq maxTimeToBeUpAndRun \tag{9.8}$$

$$min\,\{timeToBeUpAndRun\} \tag{9.9}$$

When this metric is applied, first the DSS discards the CoA which require a time to be up and run that is higher than the maximum. Then the CoA with the lowest time to be up is selected as optimal. The parameter *maxTimeToBeUpAndRun* is defined in the security objective specified by the user and it represents the maximum time that the organization is willing to wait before responding to an attack. The other parameter of the equation is contained in the property *hasCoATimeToBeUpAndRun* of the class *CourseOfAction* and is calculated according to the corresponding metric.

9.2.2 Metrics for the optimal recommended CoA

The following metrics are always applied independently on the user's security objective. The goal of these metrics is to consider the importance level of the compromised asset as the main parameter in the selection of the optimal CoA, so, based on the importance of the asset, a different metric is applied. For example, if the asset importance level is low, it is not very relevant how much the CoA is able to mitigate the attack impact, so the cost of the CoA will have higher priority with respect to the CoA effectiveness in the selection of the optimal; on the contrary, if the asset importance level is high, the effectiveness will have higher priority with respect to the cost in the selection process, because it is necessary for the organization to mitigate as much as possible the attack impact on that asset. The metrics proposed have been created by considering the ones presented in the doctoral thesis [55], which have been modified and adapted to the context of this project.

Minimum cost metric

The Decision Support module applies this metric if the level of importance of the compromised asset is *low* or *medium*. If the level of importance is *high*, the system should not use this metric to infer the best CoA. The cost considered by this metric depends on the operational cost and on the side effects. It is possible to distinguish between two cases:

- When the **asset importance level** is **low**, the metric satisfies the following equation:

$$cost = (operationalCost + sideEffect) \tag{9.10}$$

$$min\,\{cost\} \tag{9.11}$$

- When the **asset importance level** is **medium**, the equation 9.11 is supplemented by the following condition:

$$effectiveness \geq attackImpact \tag{9.12}$$

This metric differs from the one defined in [55]: here the *cost* depends on the side effect caused by CoA execution, moreover if the level of importance of the asset is medium, the CoA effectiveness is used to filter the CoAs.
The objective of this metric is to minimize the specified equation, where the overall cost associated with a CoA is the sum of its operational cost (previously computed according to the corresponding metric) and its side effect on the system. If various CoAs have the same cost, the one with the lowest complexity is considered optimal.

Minimum residual impact or maximum effectiveness metric

If the level of importance of the compromised asset is *high* (if it is very relevant or critical to the organization), the decision support module applies the maximum effectiveness metric. This metric allows the system to select the CoA that best mitigates the attack and satisfies the following equations:

$$residualImpact = attackImpact - effectiveness \tag{9.13}$$

$$min\,\{residualImpact\} \tag{9.14}$$

Even if the concept is the same, this metric differs from the one used in [55], because instead of maximizing the CoA severity, here the objective is to maximize its effectiveness by minimizing the residual impact specified in the equation. If various CoAs have the same residual impact, the one with the lowest side effect is considered optimal.

9.3 Specification of security metrics with SWRL rules

As already mentioned, the Decision Support System proposed uses the technologies of the Semantic Web, such as ontologies, rule languages and reasoner. The reasoner is the most important component since it is the basis of the decision support module and is responsible for inferring the optimal CoA as a response to an attack. The reasoner applies the metrics to infer the optimal CoA proposed in the previous section of this chapter, but in order to be able to interpret and execute them, the metrics must be specified through SWRL rules.

9.3.1 SWRL rules to infer the optimal CoA according to security objectives

Damage reduction metric

To specify the damage reduction metric in SWRL it is necessary to define a set of rules. In particular, the ontology includes five rules. Two of them will be presented as an example in Figure 9.1 and Figure 9.2.

```
Alert(?a) ^ hasAlertID(?a, alertID) ^
hasAlertAttackImpact(?a, ?atImp) ^ hasAlertReliability(?a, ?aReliab) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1)  ^ hasCoASideEffect(?coa1, ?sEf1)^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^ hasCoASideEffect(?coa2, ?sEf2) ^
swrlb:lessThan(?sEf1, ?atImp^ swrlb:lessThan(?sEf2, ?atImp) ^
swrlb:lessThan(?sEf1, ?sEf2) -> hasOptimalCoAForDamageReductionMetric(?a, ?coa1)
```

Figure 9.1: Damage reduction metric - Example1

```
Alert(?a) ^ hasAlertID(?a, alertID) ^
hasAlertAttackImpact(?a, ?atImp) ^ hasAlertReliability(?a, ?aReliab)^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^
hasCoASideEffect(?coa1, ?sEf1) ^ hasCoAEffectiveness(?coa1, ?effectiv1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^
hasCoASideEffect(?coa2, ?sEf2) ^ + "hasCoAEffectiveness(?coa2, ?effectiv2) ^
swrlb:lessThan(?sEf1, ?atImp) ^ swrlb:lessThan(?sEf2, ?atImp) ^
swrlb:equal(?sEf1, ?sEf2) ^ swrlb:greaterThan(?effectiv1, ?effectiv2)
->hasOptimalCoAForDamageReductionMetric(?a, ?coa1)
```

Figure 9.2: Damage reduction metric - Example2

The two rules are very similar, they establish that if the side effect of the CoA is less than the attack impact, then this CoA can be inferred as optimal. In the first example (Figure 9.1), the lines 5 and 6 are the most important because they specify the metric: they infer as optimal the CoA with the minimum side effect value by comparing the values of the *hasCoASideEffect* property of two individuals of the class *CourseOfAction*.
In the second example (Figure 9.2), the metric is specified in lines 7 and 8. This rule considers the case in which several CoAs have the same side effect value, so inference is made by comparing the values of the property *hasCoAEffectiveness* of two individuals of the class *CourseOfAction*.

Minimum deployment cost metric

To specify the minimum deployment cost metric, the ontology includes a set of 4 SWRL rules. One of them will be presented as an example in Figure 9.3.

```
Alert(?a) ^ hasAlertID(?a, alertID ) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^ hasCoADeploymentCost(?coa1, ?depCost1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^ hasCoADeploymentCost(?coa2, ?depCost2) ^
swrlb:lessThanOrEqual(?depCost1, maxDepCost) ^
swrlb:lessThanOrEqual(?depCost2,  maxDepCost ) ^
swrlb:lessThan(?depCost1, ?depCost2)
-> hasOptimalCoAForMonetaryCostMetric(?a, ?coa1)
```

Figure 9.3: Deployment cost metric - Example

In this case, the most relevant lines that define the metric are lines 4, 5, and 6: CoAs are filtered considering only those with deployment cost lower than the maximum deployment cost allowed, then the CoA with the lowest value of deployment cost is inferred as optimal by comparing the values of the property *hasCoADeploymentCost* of two individuals of the class *CourseOfAction*.

Minimum complexity metric

This metric is specified by a set of five SWRL rules in the ontology. Figure 9.4 shows one of those rules as an example. The most relevant lines that specify

```
Alert(?a) ^ hasAlertID(?a,  alertID ) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^ hasCoATotComplexityNum(?coa1, ?compl1)^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^ hasCoATotComplexityNum(?coa2, ?compl2) ^
swrlb:lessThanOrEqual(?compl1, maxCompl) ^
swrlb:lessThanOrEqual(?compl2, maxCompl ) ^
swrlb:lessThan(?compl1, ?compl2)
-> hasOptimalCoAForComplexityMetric(?a, ?coa1)
```

Figure 9.4: Complexity metric - Example

the metric are lines 4, 5, and 6: CoAs with a complexity value greater than the maximum complexity allowed are discarded, and then the one with the minimum complexity is inferred as optimal by comparing the values of the property *hasCoATotComplexityNum* of two individuals of the class *CourseOfAction*.

Minimum required human skills metric

This metric has been specified by a set of five SWRL rules. Figure 9.5 provides one of them as an example. Lines 4, 5, and 6 define the metric: the CoAs with required

```
Alert(?a) ^ hasAlertID(?a, alertID ) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^ hasCoAHumanSkillsNum(?coa1, ?hs1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^ hasCoAHumanSkillsNum(?coa2, ?hs2) ^
swrlb:lessThanOrEqual(?hs1, maxHumanSkills ) ^
swrlb:lessThanOrEqual(?hs2, maxHumanSkills ) ^
swrlb:lessThan(?hs1, ?hs2)
-> hasOptimalCoAForHumanSkillMetric(?a, ?coa1)
```

Figure 9.5: Human skills metric - Example

human skills value greater than the maximum allowed are discarded, then the one with lowest value is inferred as optimal by comparing the values of the property *hasCoARequiredHumanSkillsNum* of two individuals of the class *CourseOfAction*.

Minimum time to be up and run metric

To specify the minimum time metric in SWRL, a set of 3 rules has been specified. Figure 9.6 shows one of them as an example. Lines 4, 5, and 6 define the metric: the

```
Alert(?a) ^ hasAlertID(?a, alertID ) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^ hasCoATimeToBeUp(?coa1, ?time1)^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^ hasCoATimeToBeUp(?coa2, ?time2) ^
swrlb:lessThanOrEqual(?time1, maxTime) ^
swrlb:lessThanOrEqual(?time2, maxTime) ^
swrlb:lessThan(?time1, ?time2)
-> hasOptimalCoAForTimeMetric(?a, ?coa1)
```

Figure 9.6: Time metric - Example

CoAs are filtered in order to consider only those with time to be up and run lower than the maximum allowed, then the CoA with the lowest time to be up and run is inferred as optimal by comparing the values of the property *timeToBeUpAndRun* of two individuals of the class *CourseOfAction*.

9.3.2 SWRL rules to infer the optimal recommended CoA

Minimum cost metric

The minimum cost metric is defined by four SWRL rules:

- Two rules are applied when the **asset importance level** is **low**. These can be seen in Figures 9.7 and 9.8. In particular, the rule in Figure 9.7 defines the metric in lines 7-12: considering only alerts with a low asset importance level, the costs associated with two individuals in the class *CourseOfAction* are calculated (sum of the values contained in properties *hasCoAOperationalCost* and *hasCoASideEffect)* and compared. The objective is to infer as optimal the CoA with lower cost. The rule in Figure 9.8 is simply a variation of the first rule, it is applied when several CoAs have the same cost, so the optimal is inferred by comparing the values of the property *hasCoATotComplexityNum*.
- Two rules are applied when the **asset importance level** is **medium**. In figure 9.9, the lines 10-17 specify the metric: considering only alerts with a medium asset importance level, the CoAs associated with these alerts are first filtered (only those whose effectiveness is greater than the attack impact are taken into account); then, the costs associated with two individuals of the class *CourseOfAction* are calculated and compared in order to infer as optimal the one with lower cost. The other rule, which is not shown here, is a simple variation of the one in Figure 9.9, it handles the case when several CoAs have the same cost by inferring the CoA with lower complexity as optimal.

```
Alert(?a) ^hasAlertAssetImportanceLevel(?a, ?impLev) ^
hasAlertNumberOfPossibleCoAs(?a, ?numCoas) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^
hasCoASideEffect(?coa1, ?sidEf1) ^ hasCoAOperationalCost(?coa1, ?operCost1) ^
hasCoASideEffect(?coa2, ?sidEf2) ^  hasCoAOperationalCost(?coa2, ?operCost2) ^
swrlb:add(?cost1, ?operCost1, ?sidEf1) ^
swrlb:add(?cost2, ?operCost2, ?sidEf2) ^
swrlb:equal(?impLev, "low") ^
swrlb:greaterThan(?numCoas, 1) ^
swrlb:lessThan(?cost1, ?cost2) ^
-> hasOptimalCoARecommended(?a, ?coa1)
```

Figure 9.7: Asset importance level low, Minimum cost metric - Example1

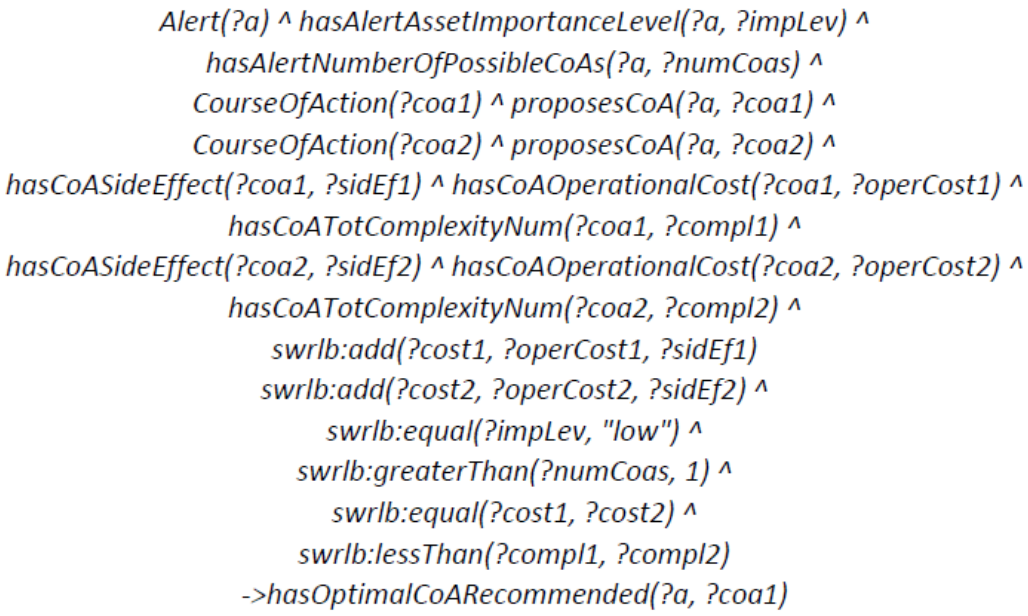

Figure 9.8: Asset importance level low, Minimum cost metric - Example2

Minimum residual impact or maximum effectiveness metric

The SWRL rule that defines this metric is the one in Figure 9.10. In order to apply this rule, the importance level of the asset must be high. In particular, we can see this condition and the entire specification of the metric in the lines 8-13 of the rule: the residual impact (saved in variable x) is calculated by subtracting from the attack impact the effectiveness of the CoA. The objective of the rule is to minimize the residual impact. This is done by calculating and comparing the residual impacts of two individuals of the class *CourseOfAction* to infer the one with the lowest value as optimal. This metric also includes another rule (Figure 9.11) that proposes a variation of the previous rule in order to handle the case where several CoAs have the same residual impact. In that case, the optimal is inferred by considering the CoA with the lowest side effect.

```
Alert(?a) ^ hasAlertAssetImportanceLevel(?a, ?impLev) ^
hasAlertNumberOfPossibleCoAs(?a, ?numCoas) ^
hasAlertAttackImpact(?a, ?atImp) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^
hasCoATotComplexityNum(?coa1, ?compl1) ^ hasCoAEffectiveness(?coa1, ?effect1) ^
hasCoAOperationalCost(?coa1, ?operCost1) ^ hasCoASideEffect(?coa1, ?sidEf1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^
hasCoATotComplexityNum(?coa2, ?compl2) ^ hasCoAEffectiveness(?coa2, ?effect2) ^
hasCoAOperationalCost(?coa2, ?operCost2) ^ hasCoASideEffect(?coa2, ?sidEf2) ^
swrlb:add(?cost1, ?operCost1, ?sidEf1) ^
swrlb:add(?cost2, ?operCost2, ?sidEf2) ^
swrlb:greaterThan(?effect1, ?atImp) ^
swrlb:greaterThan(?effect2, ?atImp) ^
swrlb:lessThan(?cost1, ?cost2) ^
swrlb:greaterThan(?numCoas, 1) ^
swrlb:equal(?impLev, “medium")
-> hasOptimalCoARecommended(?a, ?coa1)
```

Figure 9.9: Asset importance level medium, Minimum cost metric - Example3

```
Alert(?a) ^ hasAlertAssetImportanceLevel(?a, ?impLev) ^
hasAlertAttackImpact(?a, ?atImp) ^
hasAlertNumberOfPossibleCoAs(?a, ?numCoas) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^
^ hasCoAEffectiveness(?coa1, ?effect1) ^
hasCoAEffectiveness(?coa2, ?effect2) ^
swrlb:subtract(?x1, ?atImp, ?effect1) ^
swrlb:subtract(?x2, ?atImp, ?effect2) ^
swrlb:lessThan(?x1, ?x2) ^
swrlb:equal(?impLev, "high") ^
swrlb:greaterThan(?numCoas, 1)
-> hasOptimalCoARecommended(?a, ?coa1)
```

Figure 9.10: Maximum effectiveness metric - Example1

```
Alert(?a) ^ hasAlertAssetImportanceLevel(?a, ?impLev) ^
hasAlertAttackImpact(?a, ?atImp) ^
hasAlertNumberOfPossibleCoAs(?a, ?numCoas) ^
CourseOfAction(?coa1) ^ proposesCoA(?a, ?coa1) ^
CourseOfAction(?coa2) ^ proposesCoA(?a, ?coa2) ^
^ hasCoAEffectiveness(?coa1, ?effect1) ^
hasCoAEffectiveness(?coa2, ?effect2) ^
swrlb:subtract(?x1, ?atImp, ?effect1) ^
swrlb:subtract(?x2, ?atImp, ?effect2) ^
swrlb:equal(?x1, ?x2) ^
swrlb:lessThan(?sEff1, ?sEff2)^
swrlb:equal(?impLev, "high") ^
swrlb:greaterThan(?numCoas, 1)
-> hasOptimalCoARecommended(?a, ?coa1)
```

Figure 9.11: Maximum effectiveness metric - Example2

Chapter 10

Proposal validation and analysis of the results

The validation of the Decision Support System prototype proposed in this research work follows two paths: first an ontology is created to test the system, then, after checking its consistency, this ontology is used as a knowledge base to execute the DSS process. This chapter will describe the ontology used, propose a set of test cases (use cases to test the behaviour of the DSS), and provide an analysis of the results obtained.

10.1 Ontology: design and validation

Chapter 8 provided a detailed description of the ontology proposed to implement the Decision Support System. This section will present how the ontology is actually filled in order to test the DSS prototype. As mentioned, the ontology contains the classes *Alert, AttackTechnique* and *MitigationTechnique*, and the latter contains two subclasses (*Countermeasure* and *CourseOfAction*). At the beginning, the class *CourseOfAction* is empty (it will be filled during the process), while the other classes are populated with generic instances (their names are fictitious, and the values of their data properties are assigned randomly, taking into account their data ranges).

Alert

Even if in a real environment the DSS should receive alerts dynamically (as soon as they are generated by an intrusion detection system), this research work considers a static list of alerts. Individuals in class *Alert* are created in advance and added to the ontology, to be processed later one by one. In particular, the Alert class

is filled with a set of six instances (Figure 10.1), where each instance has a set of data properties initialized with random values (an example is provided in Figure 10.2) and is related to one or more *AttackTechnique* instances through the object property *notifiesAttackTechnique* (an example is provided in Figure 10.3). Furthermore, as soon as the CourseOfAction individuals are created, the Alert is related to the CoAs through the object property *proposesCoA*; when the optimal CoA (recommended by the system or inferred according to the user objective) is identified, other relations are created (for example, through object properties such as *hasOptimalCoAForDamageReductionMetric* or *hasOptimalCoARecommended*). These properties are shown in Figure 10.4.

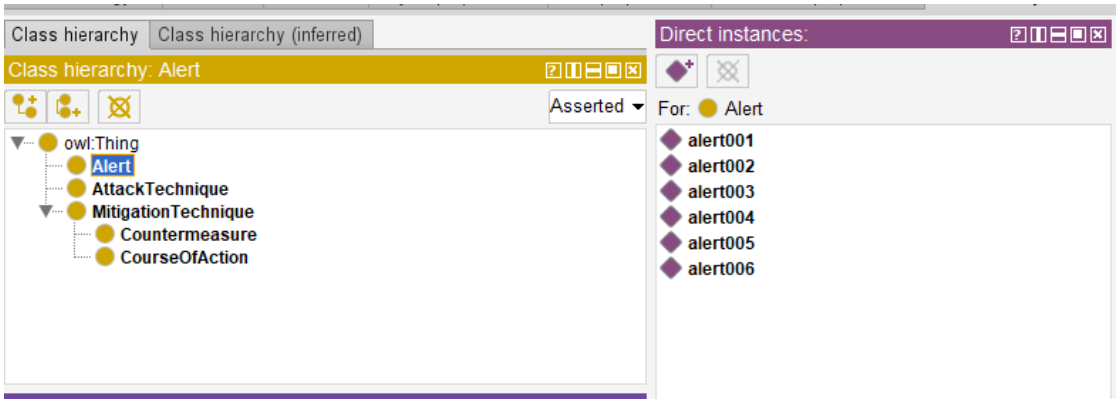

Figure 10.1: List of instances of the Alert class

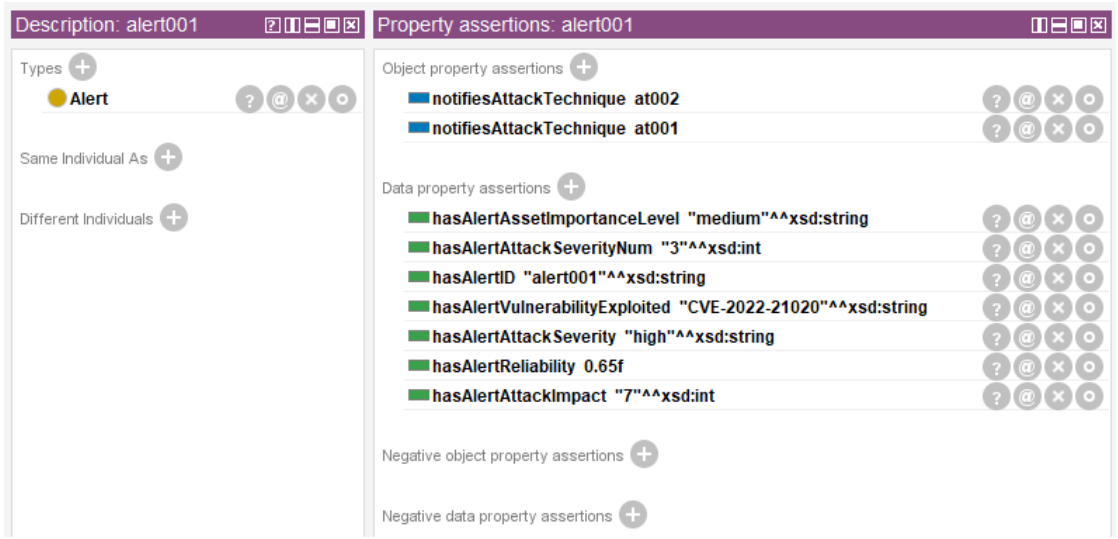

Figure 10.2: Property assertions of instance *alert001*

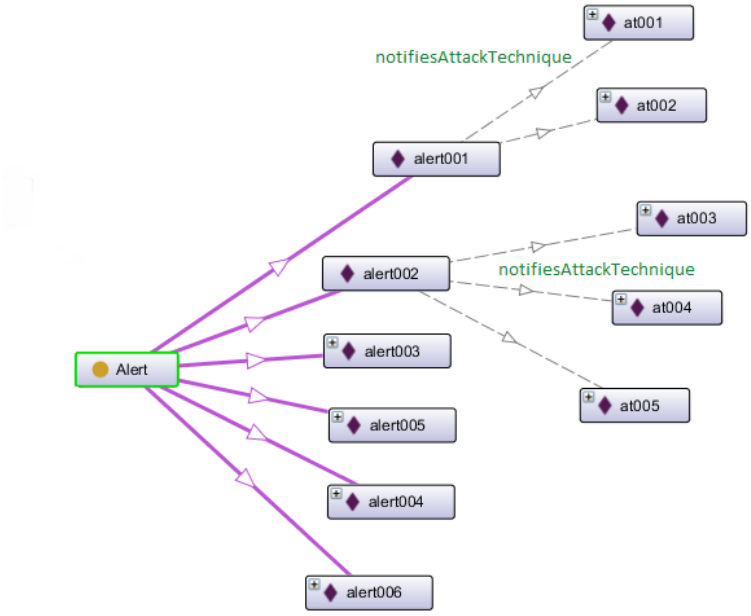

Figure 10.3: OntoGraf representation of Alert class and its instances

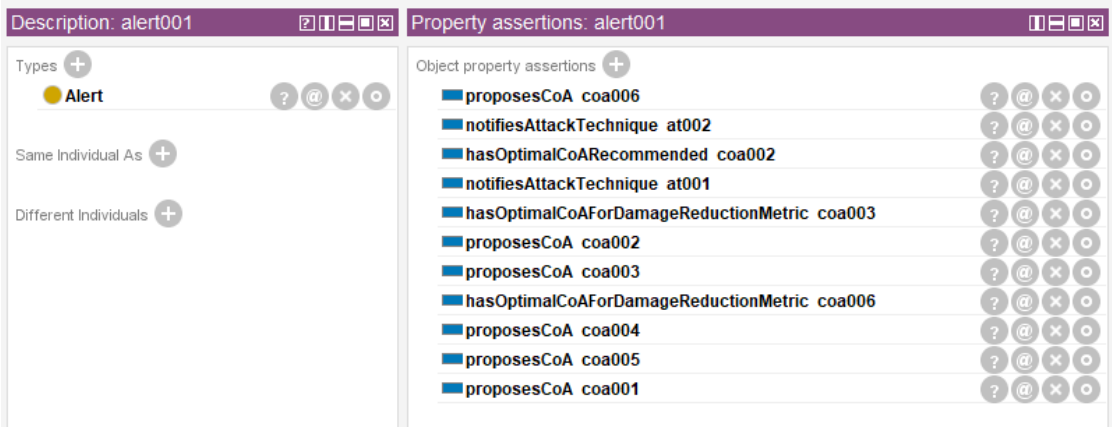

Figure 10.4: Object properties between Alert instances and CourseOfAction instances

AttackTechnique

The class AttackTechnique is populated by a list of seven instances (Figure 10.5). The only data property they have is the ID, while they are related to one or more *Countermeasure* instances through the object property *isMitigatedByCountermeasure*. An example is provided in Figure 10.6

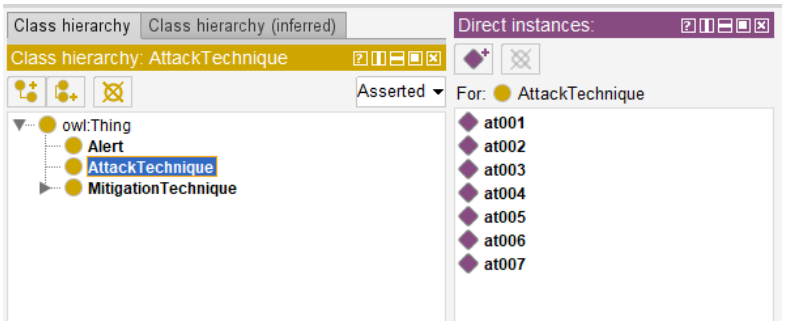

Figure 10.5: List of instances of the AttackTechnique class

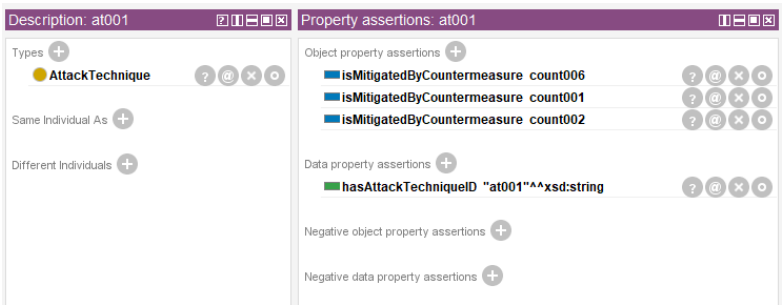

Figure 10.6: Property assertions of instances *at001*

Countermeasure

The class Countermeasure is populated by six instances (Figure 10.7). Each Countermeasure instance has a set of data properties initialized with random values (an example is provided in Figure 10.8) and is related to one or more *AttackTechnique* instances through the object property *mitigatesAttackTechnique* (an example is provided in figure 10.9).

CourseOfAction

The class CourseOfAction at the beginning is empty. According to the workflow of the program described in Chapter 7, the CoA instances are created in *Courses of Action creation module*, after having read and processed the various alerts, attack techniques, and countermeasures included in the ontology.
To create the CoAs of an alert, the Java method combines in different ways the various countermeasures associated with each notified technique. For example, considering *alert001*, to retrieve its CoAs the steps are the following: it is known that this alert notifies three attack techniques (*at001, at002*) and that these techniques are mitigated by the following sets of countermeasures:

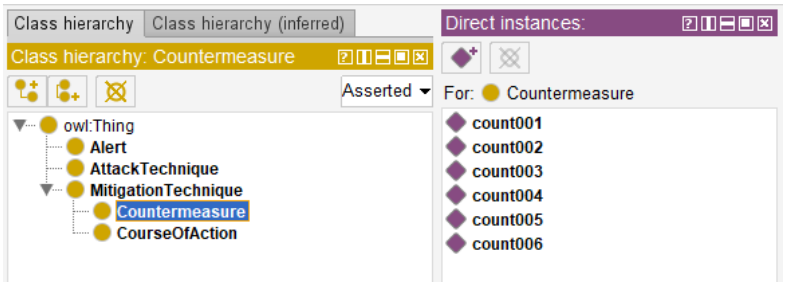

Figure 10.7: List of instances of the Countermeasure class

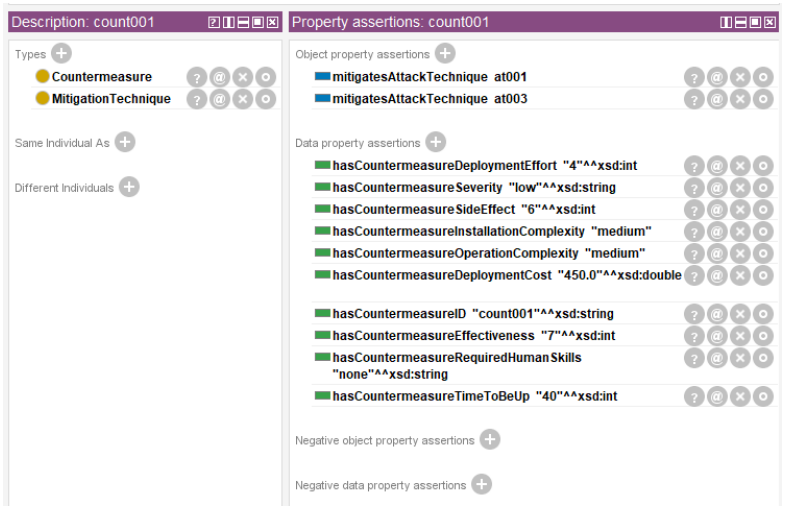

Figure 10.8: Property assertion of instance *count001*

- *at001* is mitigated by 3 countermeasures (*count001, count002, count006*);
- *at002* is mitigated by 2 countermeasures (*count003, count005*).

By applying the method for the CoA creation (which takes inspiration from algorithm of the Cartesian product between arrays), in order to mitigate each attack technique it is expected to obtain `3 * 5 = 6` courses of actions. In particular, following six combinations should be obtained:

- (*count001, count003*);
- (*count001, count005*);
- (*count002, count003*);
- (*count002, count005*);
- (*count006, count003*);

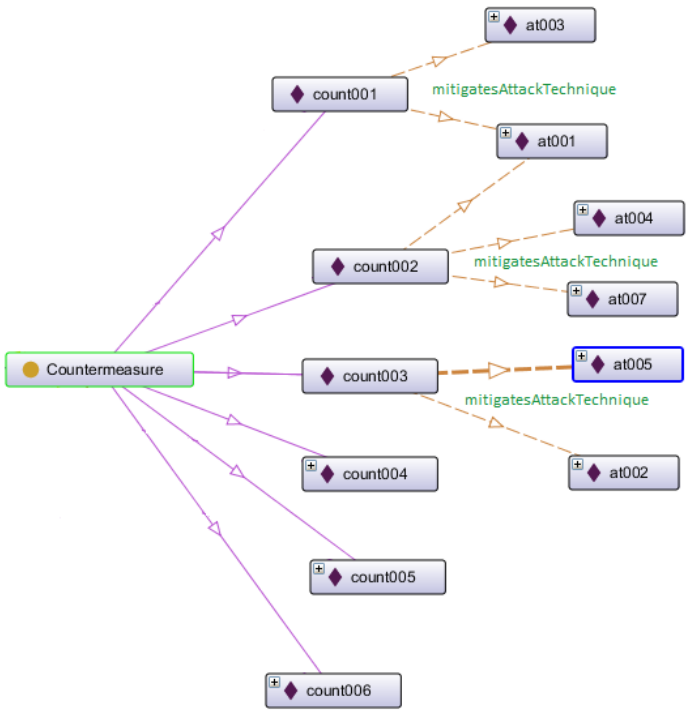

Figure 10.9: OntoGraf representation of Countermeasure class and its instances

- (*count006, count005*);

Looking at the ontology after executing the CoAs creation module, the CoAs for *alert001* are exactly as expected (Figure 10.10).

At the end of the execution of this module, the class CourseOfAction is populated by 21 instances (Figure 10.11). Each instance has a set of data properties whose values have been calculated according to the parameters of the countermeasures it executes; each CoA instance is related to one or more Countermeasure instances through the object property *executesCountermeasure* (Figure 10.12 shows one of the CoA and its properties).

10.1.1 Validation

The ontology created is validated by verifying its consistency through the Pellet reasoner. After the ontology is populated, the reasoner is invoked in order to prove its consistency. Since no inconsistencies are found, the ontology is assumed to be consistent and can be used as a knowledge base for the Decision Support System.

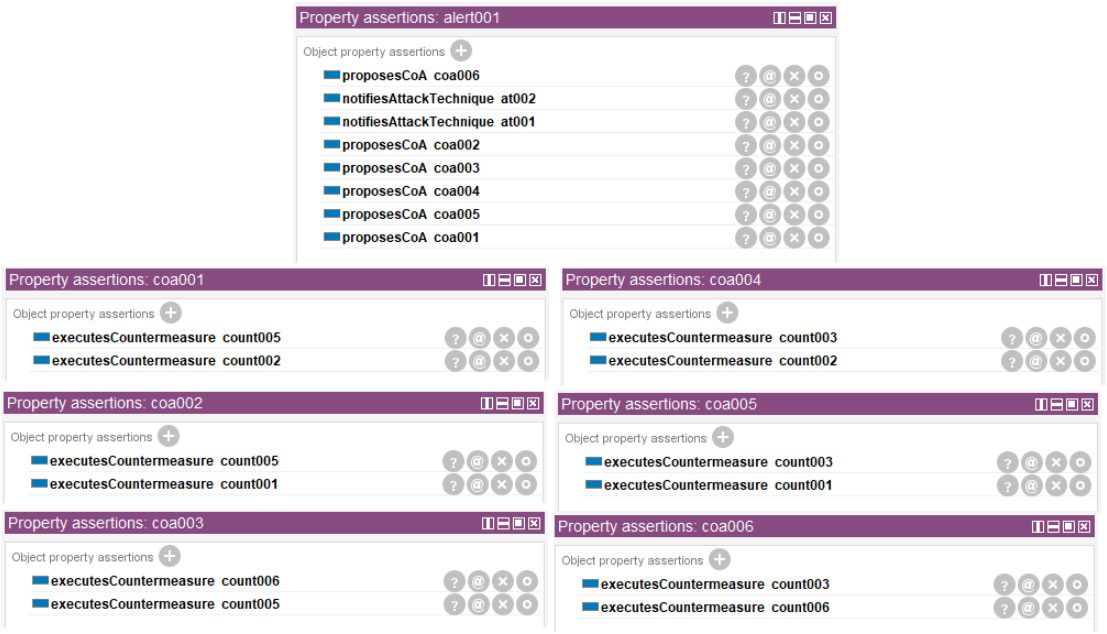

Figure 10.10: CoAs created for *alert001*

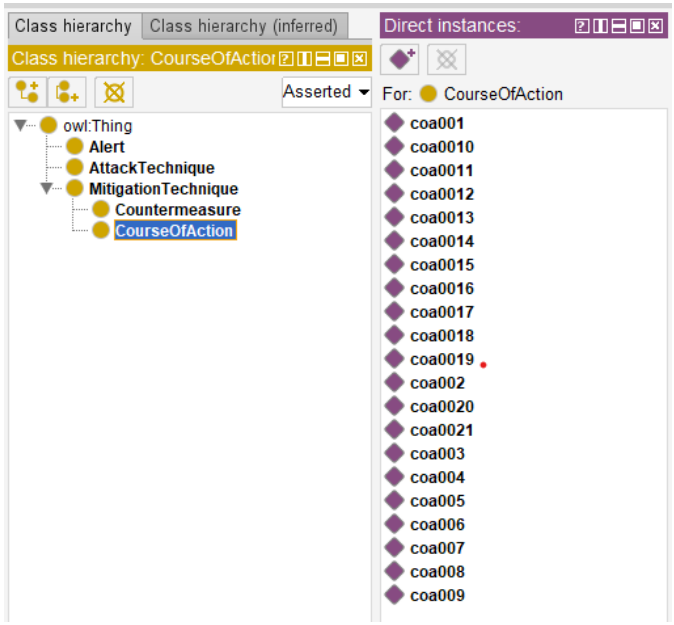

Figure 10.11: List of instances of the CourseOfAction class

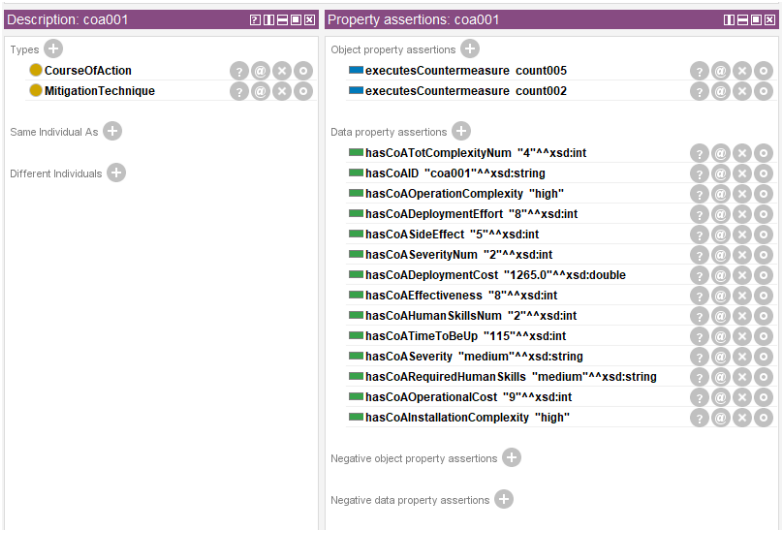

Figure 10.12: Property assertions of instance *coa001*

10.2 Test cases and analysis of the results

The functioning of the DSS prototype is tested by creating specific use cases and comparing the expected results with the results obtained by running the system. This section will provide a description of the test cases and a brief analysis of the results obtained.

User defined security objectives

The prototype developed does not really handle an interaction between the user and the system. The various user objectives associated with each alert are taken from a configuration file specifically written in JSON (Figure 10.13 provides an example). This choice was taken considering the idea of integrating the system with a front-end as future work, in order to allow the user to dynamically define the security objectives for each alert. Several use cases are used to test the system are all related to the different security objective the user may specify and to how the system deals with these objectives in the decision support process.

10.2.1 Test case 1: The user does not specify a security objective

As already mentioned, the goal of the DSS proposed is to help the security manager of an organization decide which CoA to execute in case of a cyber attack, taking into account the precise needs of the organization (resources that can be allocated,

```
{
  "objective":{
    "id": 1,
    "alertID": "alert001",
    "damageReduction" : true,
    "maxComplexityAllowed" : false,
    "maxDeploymentCostAllowed" : false,
    "maxHumanSkillsAllowed" : false,
    "maxTimeToRunAllowed" : false,
    "maxInstallationComplexity": "",
    "maxOperationComplexity" : "",
    "maxCostInEuros" : -1,
    "maxDeploymentEffort" : -1,
    "maxHumanSkills" : "",
    "maxTime" : -1
  }
},
```


Figure 10.13: Definition of a security objective in JSON format

level of expertise in cyber security of the employees, maximum time allowed to block the attack, etc). However, there are cases where the organization does not specify specific requirements for dealing with the attack, so the user of the system can simply avoid specifying a security objective. In this case, the system should provide all CoAs that can mitigate the attack notified by the current alert. Furthermore, the system suggests to the user one or more recommended CoAs (based on the security metrics for the recommended COA described in Chapter 9). This use case is tested by setting to false all sub-objectives of the objective related to *alert001*. The result obtained can be seen in Figure 10.14 and it is compliant to the expected result.

10.2.2 Test case 2: The user specifies an objective, but no optimal CoA is found

If an organization using the DSS is small and has a limited number of resources to manage all its activities, in case of a cyber incident the user of the system may be interested in setting as security objective the maximum financial cost that the organization is able to spend or the maximum complexity it is able to handle for the CoA implementation. However, it may happen that among the available CoAs, there is not even one that complies with the defined security objective. In this case, the system is expected to notify the user with a message; moreover, the system should still provide the user with all the possible CoAs associated with

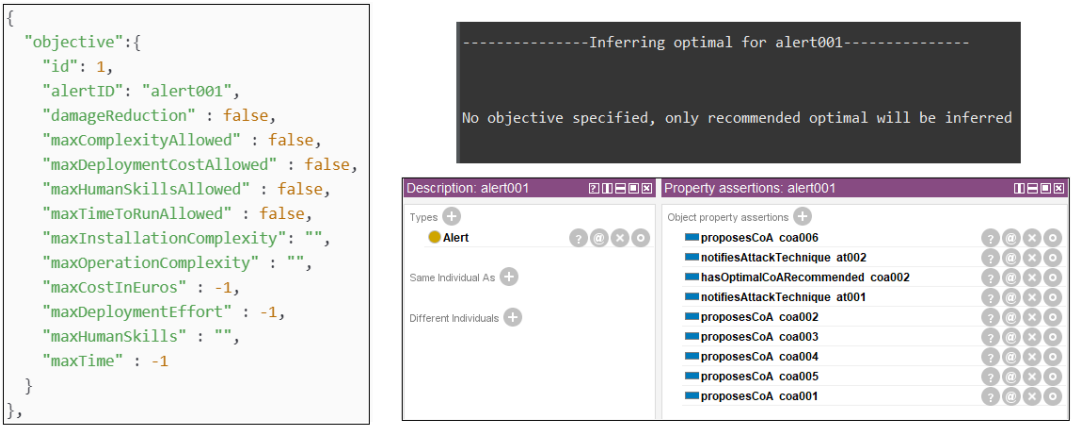

Figure 10.14: Use case 1: No objective specified

the current alert and suggest him a recommended optimal CoA. This use case is tested by setting the sub-objective *maxDeploymentCostAllowed* to true and the *maxCostInEuros* to *500* for *alert002*; this objective implies the application of the *minimum deployment cost metric*.
The courses of actions associated with *alert002* are *coa011* and *coa012*, which both have a deployment cost greater than 500, so the system should not be able to find an optimal CoA for this objective. The result obtained can be seen in Figure 10.15 and it is compliant with the expected result.

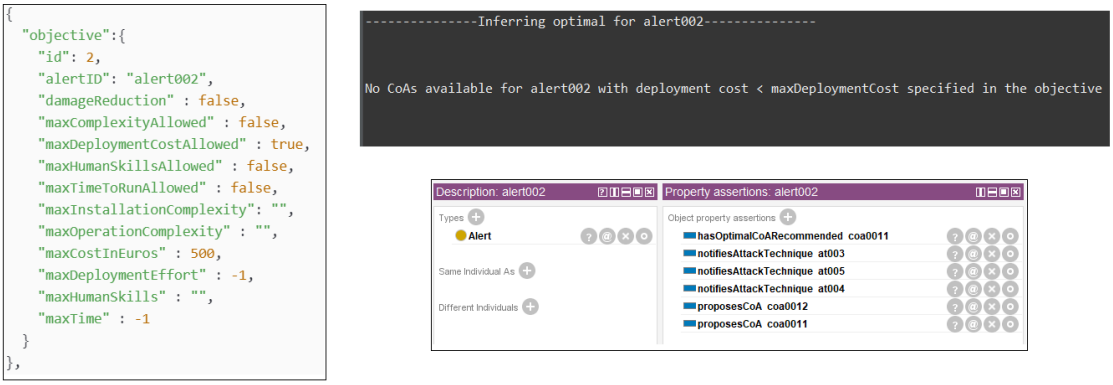

Figure 10.15: Use case 2: No optimal found for maximum deployment cost allowed objective

10.2.3 Test case 3: The user specifies an objective and the optimal CoA is found

The user may be interested in receiving suggestions about one or more security objectives; if several optimal CoAs are suggested, the user can compare and analyze them to decide which is the preferred one.

One single sub-objective is specified

When a single sub-objective is specified, the system is expected to propose one (or sometimes more) optimal CoAs identified according to that sub-objective. The system also provides the user with all the possible CoAs associated to the current alert and suggests him a recommended optimal. This use case is tested by setting the security sub-objective *maximumHumanSkillsAllowed* to true and *maximumHumanSkills* to *high* for *alert004*; this objective implies the application of the *minimum required human skill metric*. *Alert004* is associated with the courses of actions *coa019* (which has *requiredHumanSkills = medium*) and *coa020* (which has *requiredHumanSkills = high*), so the expected optimal CoA is be *coa019* because it has the lowest value of required human skills. The result obtained can be seen in Figure 10.16 and is compliant with the expected result.

Figure 10.16: Use case 3a: One optimal CoA found for objective *maximum human skills allowed*

Multiple sub-objectives are specified

When multiple sub-objectives are specified, the system is expected to propose one (or sometimes more) optimal CoAs for each sub-objective. This use case is applied to *alert002* and tested by setting to true the security sub-objectives

maximumComplexityAllowed and *maximumTimeToRunAllowed* with corresponding values for *maximumInstallationComplexity* and *maximumOperationComplexity* set to *medium* (so the total resulting complexity is medium) and *maxTime* set to *120* minutes. The metrics used to satisfy these security objectives are *minimum complexity metric* and the *minimum time to be up and run metric*. The courses of actions associated with *alert002* are *coa011* (*totComplexity = medium*, *timeToBeUp = 110*) and *coa012* (*totComplexity = medium*, *timeToBeUp = 130*). Considering complexity, since the two CoAs have the same total complexity value, the optimal is expected to be *coa012* because it is the one with the lowest deployment cost. Considering the time to be up, the optimal is expected to be *coa011* because it is the only one that satisfies the security sub-objective of maximum time allowed. The results can be seen in Figure 10.17 and are compliant with the expected results. This example also allows us to highlight that according to the sub-objective specified, the inferred optimal CoA can change (in fact here we have two different optimal CoAs).

Figure 10.17: Use case 3b: Multiple optimal CoAs found for sub-objectives *maximum complexity allowed* and *maximum human time to run allowed*

10.2.4 Test case 4: The system provides the recommended optimal CoA

As described in the previous chapters, the DSS not only helps the user identify the optimal CoA according to the organization's needs, but also provides the user with some suggestions about the optimal CoAs that best mitigate the attack considering the level of importance of the compromised asset. This is done for three main reasons:

- The organization may not have specific needs to identify the optimal CoA (so no security objectives are specified), but the user still needs some suggestions.
- As shown in case 2, it may happen that no one of the available CoAs satisfies the security objective set by the user. In this case, the system still provides the user with an optimal CoA.
- Since the optimal CoA recommended by the system is the one that best mitigates the attack because it considers the criticality of the involved asset, the user may be interested in receiving this information in order to compare and analyze the various possibilities he has before choosing which CoA to execute.

In order to test this use case, an example for each value of asset importance level will be provided.

Low asset importance level

In the created ontology, the alerts with *asset importance level = medium* are *alert002* and *alert006*. Since *alert006* is associated with a single CoA (which will automatically be inferred as optimal), *alert002* will be taken into account. In this case it is supposed that the security metric *minimum cost* for low asset importance level is applied. According to this metric, the CoA with the lowest value of *operationalCost + sideEffect* should be inferred as optimal. The CoAs associated with *alert002* have the following values:

- *coa011*: *operationalCost = 7* and *sideEffect = 5*;
- *coa012*: *operationalCost = 7* and *sideEffect = 6*;

This means that *coa011* is expected to be the optimal CoA. The result can be seen in Figure 10.18 and it is compliant with the expected result.

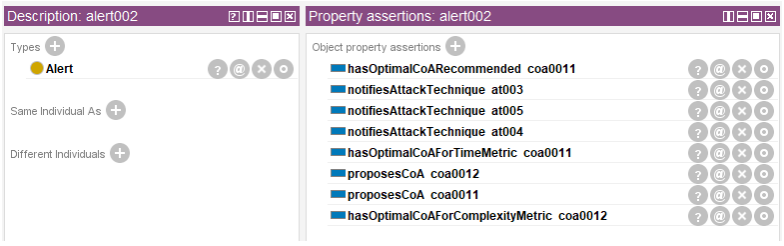

Figure 10.18: Use case 4a: Optimal recommended CoA for asset importance level = low

Medium asset importance level

In the created ontology, the alerts with *asset importance level = medium* are *alert001* and *alert004*. In this case it is supposed that the security metric *minimum cost* for medium asset importance level is applied. According to this metric, the CoA with the lowest value of *operationalCost + sideEffect* should be inferred as optimal after discarding CoAs that have a value of *effectiveness* lower than the value of *attack impact*.
Alert004 (which proposes the courses of actions *coa019* and *coa020*) is considered as an example. The parameters used in the metric are the following:

- *attackImpact = 2*;
- *coa019*: *operationalCost = 7* and *sideEffect = 5*, *effectiveness = 8*;
- *coa020*: *operationalCost = 5* and *sideEffect = 6*, *effectiveness = 9*;

Both CoAs can be considered, since their effectiveness is greater than the attack impact. The optimal is expected to be *coA020* since the calculated cost is the lowest. The result obtained by running the DSS program is showed in Figure 10.19 and is compliant with the expected result.

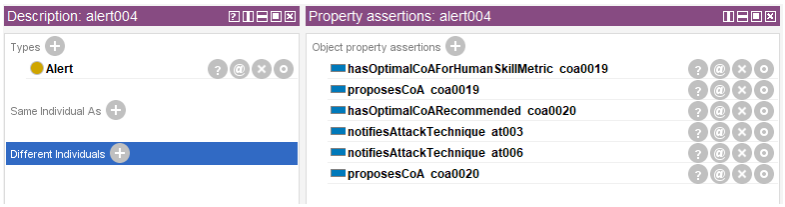

Figure 10.19: Use case 4b: Optimal recommended CoA for asset importance level = medium

High asset importance level

In the ontology created, alerts with *asset importance level = high* are *alert003* and *alert005*. In this case it is supposed that the *minimum residual impact or maximum effectiveness* security metric is applied. According to this metric, the CoA with the lowest value of *attackImpact - effectiveness* should be inferred as optimal.
To test this use case, we will take into account *alert003* (which proposes the courses of actions *coa007, coa008, coa009,* and *coa010*. The parameters used in the metric are the following:

- *attackImpact = 8*;
- *coa007*: *effectiveness = 8*;

- *coa008*: *effectiveness* = 7;
- *coa009*: *effectiveness* = 9;
- *coa010*: *effectiveness* = 8.

The optimal is expected to be *coa009*, as it is the one with the highest effectiveness value. The result obtained is shown in 10.20. It can be noticed that many optimal recommended CoAs are inferred (in particular, the only excluded is the one with the lowest value of effectiveness). This happens because SWRL rules do not provide a method to compare all values of a list at the same time (for example, to identify the minimum or the maximum among them), but the comparison must be specified manually in the rule by selecting and comparing two, three, or more elements. In the SWRL rule created for this metric, the comparison is done between two CoAs individuals, the effectiveness values of the CoAs are compared in couples of two, so all CoAs with effectiveness greater than the the effectiveness of *coa008* are considered optimal. Even though this result does not propose one single optimal CoA, it cannot be considered wrong since the security policy described in the metric is correctly observed and the system is still helping the user in deciding which CoA to execute.

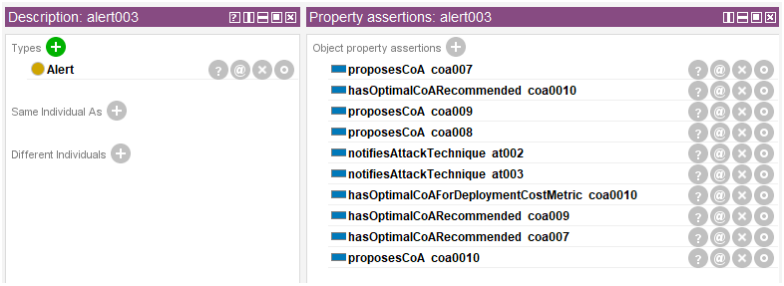

Figure 10.20: Use case 4c: Optimal recommended CoA for asset importance level = high

Chapter 11

Conclusion and future work

This chapter will propose a summary of the work presented in the previous chapters and the future investigations that can arise from this work.

11.1 Summary of the work carried out

The main objective of this thesis was the development of a decision support system based on ontologies and SWRL rules; such a decision support system aimed to infer the optimal response to mitigate a cyber attack taking into account the needs of the organization (considering the availability of financial, technological, and human resources). The project included the definition of:

- an ontology for the description of alerts, attacks, and mitigation techniques;
- some security metrics to specify the parameters to be used in the identification of the optimal CoA;
- some inference rules to be executed by a semantic reasoner.

The ontology was defined using the OWL2 language and created with the help of Protégé ontology editor. For the creation of the ontology (classes and their properties), several existing ontologies were analyzed and used as a cue; in particular, the ontology contained the classes *Alert, AttackTechnique, Countermeasure* and *CourseOfAction*, where an attack was described in terms of attack techniques (as suggested by MITRE ATT&CK framework), each countermeasure mitigated one or more attack techniques and one or more course of actions (set of countermeasures) were specifically created to mitigate an attack.
For the definition of security metrics, some existing metrics were used as source of inspiration. In particular, two classes of metrics were defined:

- metrics for the definition of the parameters used in the SWRL rules: the parameters defined were *asset importance level, attack impact, side effect, deployment cost, operational cost, total complexity, effectiveness, required human skills*, and *time to be up and run*;
- metrics for the inference of the optimal CoA, divided into:
  - *metrics for the optimal CoA according to security objectives*: damage reduction metric, minimum deployment cost metric, minimum complexity metric, minimum required human skills metric, minimum time to be up and run metric;
  - *metrics for the optimal CoA recommended by the system*: minimum cost metric (for asset importance level low or medium) and minimum residual impact or maximum effectiveness metric (for asset importance level high).

The metrics to identify the optimal CoA were translated in SWRL rules in order to be executed by the Pellet reasoner and infer the optimal CoA. In the end, first the consistency of the ontology was verified, and then the developed system was validated by executing some test cases and comparing the results obtained with the expected results. As presented in Chapter 10, the results obtained in the test cases were compliant to the expected results.

11.2 Future work

Although the developed DSS works as intended, there are points that can be improved. SWRL has difficulties in dealing with complex rules; the fact that it supports only AND statements (OR and NOT are not supported) and that the built-ins provided do not consider certain types of operation (for example, identifying the minimum/maximum value inside a list of values) limits the type of rules that can be created. In this project, the limitations of SWRL had some impact on the inference of the optimal CoA: a built-in able to compare the values of a specified parameter for all the CoAs in a list is missing (for example, to identify the CoA with the minimum deployment cost, or complexity, or time to be up, etc.), and in order to perform this comparison it was necessary to create several rules (one for each possible list of CoAs, distinguishing the cases in which the list was composed of 1, 2, 3, 4, etc. CoAs). Since managing all possible lists is not feasible, only lists containing 1, 2, or 3 CoAs were considered to create the rules, extending the rule for a list of 3 CoAs to all lists with a number of CoAs greater than or equal to 3. As a consequence, when the list contained more than 3 CoAs, in some cases several optimal CoAs were inferred for the same objective. Although this result is not precise as expected, it cannot be considered wrong since the objective of the

DSS is to help the user identifying the optimal response, and even when several CoAs are inferred as optimal, the system is still providing some help to the user. It could be useful to refine this solution in order to overcome the limitations resulting from SWRL and ensure the inference of a single optimal CoA for the specified objective. In particular, a proposal for future work could be to modify the proposed DSS to use SPARQL instead of SWRL. SPARQL is the semantic web query language capable of extracting and manipulating RDF data. SPARQL supports three types of queries: ASK queries, SELECT queries and CONSTRUCT queries. **CONSTRUCT queries** are the ones that allow inference to be performed; in particular, after identifying the matches of interest through ASK and SELECT queries, CONSTRUCT queries return an RDF graph by substituting the variables in those matches in a set of triple templates. In Chapter 5 an example of the SWRL rule was provided to infer the object property *hasUncle*. The same property can be inferred using SPARQL as showed in Figure 11.1.[84]

```
Construct { ?x dd:hasUncle ?y}

        WHERE { ?x dd:hasParents ?z . ?z dd:Gender
"Male"^^xsd:string ; dd:hasBrother ?y}
```

Figure 11.1: Inference with SPARQL

Thanks to the several clauses supported by SPARQL, filtering CoAs according to the specified objective and identifying the best one (for example the one with lower complexity or lower cost) would be easier than with SWRL.
In this project, a static list of alerts was considered (as they were all received in advance and saved in the ontology), but this was just a way to make the work easier. In order to make the system usable, it is necessary to include an Intrusion Detection System (so that the DSS can receive real-time alerts every time an anomaly is detected). As future work, it may be useful to create a module able to translate the alert from the format supported by the IDS to the format supported by the DSS, in order to allow the interaction between IDS and DSS. Finally, the framework MITRE ATT&CK could be integrated as a knowledge base (since in the investigation carried out it was only used as a starting point for modeling the attack and the courses of actions in the ontology) and a front-end application could be developed for the interaction between user and DSS.

Bibliography

[1] Embroker Team. *2021 Must-Known Cyber Attacks Statistics and Trends*. Nov. 2021. URL: https://www.embroker.com/blog/cyber-attack-statistics/ (cit. on pp. 4, 5).

[2] WEF - World Economic Forum. *The Global Risks Report 2020*. 2021. URL: https://www3.weforum.org/docs/WEF_Global_Risk_Report_2020.pdf (cit. on p. 5).

[3] L. Brown W. Stallings. «Computer Security: Principles and Practice». In: Harlow, United Kingdom: Pearson, 2018. Chap. 14 (cit. on pp. 6, 8, 9).

[4] Cristian Fornaciari. *ISMS: cos'è, a cosa serve e come strutturare un Information Security Management System*. Cybersecurity360. Jan. 2020. URL: https://www.cybersecurity360.it/soluzioni-aziendali/isms-cose-a-cosa-serve-e-come-strutturare-un-information-security-management-system/ (cit. on pp. 7, 10).

[5] A. Holl G. Funk J. Hermann et al. *Implementation Guideline ISO/IEC 27001:2013 - A practical guideline for implementing an ISMS in accordance with the international standard ISO/IEC 27001:2013*. ISACA Germany Chapter e.V. 2016 (cit. on pp. 8, 10).

[6] ISO - International Organization for Standardization. *International standard ISO/IEC 27005- Information technology, Security techniques, Information security risk management*. ISO office. 2011 (cit. on pp. 11, 15).

[7] Tony Ucedavélez and Marco M. Morana. «RISK CENTRIC THREAT MODELING -Process for Attack Simulation and Threat Analysis». In: Hoboken, New Jersey: John Wiley and Sons, 2015. Chap. 1, p. 29 (cit. on p. 16).

[8] Adam Shostack. «Threat Modeling - Designing for security». In: Indianapolis, Indiana: John Wiley and Sons, 2014, pp. xxiii–xxiv (cit. on p. 17).

[9] Adam Shostack. «Threat Modeling - Designing for security». In: Indianapolis, Indiana: John Wiley and Sons, 2014. Chap. 2, pp. 34–43 (cit. on p. 18).

[10] Josh Fruhlinger. *Threat modeling explained: A process for anticipating cyber attacks*. Apr. 2020. URL: https://www.csoonline.com/article/3537370/threat-modeling-explained-a-process-for-anticipating-cyber-attacks.html (cit. on pp. 18, 19).

[11] Kieran McLaughlin Rafiullah Khan et al. *STRIDE-based Threat Modeling for Cyber-Physical Systems*. 2017. URL: https://ieeexplore.ieee.org/document/8260283 (cit. on pp. 19–21).

[12] OWASP - CheatSheets Series Team. *Threat Modeling Cheat Sheet*. Nov. 2021. URL: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html (cit. on p. 20).

[13] Sebastien Deleersnyder. *Threat modeling in 4 steps*. URL: https://www.toreon.com/threat-modeling-in-4-steps/ (cit. on p. 20).

[14] Adam Shostack. «Threat Modeling - Designing for security». In: Indianapolis, Indiana: John Wiley and Sons, 2014. Chap. 10, pp. 190–193 (cit. on p. 20).

[15] Giorgio Giacinto. *Threat modeling*. 2020. URL: https://www.unica.it/static/resources/cms/documents/16CS.ThreatModeling.pdf (cit. on p. 21).

[16] Adam Shostack. «Threat Modeling - Designing for security». In: Indianapolis, Indiana: John Wiley and Sons, 2014. Chap. 3, pp. 62–63 (cit. on p. 21).

[17] Adam Shostack. «Threat Modeling - Designing for security». In: Indianapolis, Indiana: John Wiley and Sons, 2014. Chap. 4, pp. 88–90 (cit. on p. 22).

[18] B. Schneier. *Attack Trees*. Dec. 1999. URL: https://www.schneier.com/academic/archives/1999/12/attack_trees.html (cit. on p. 22).

[19] N. Alhebaishi et al. «Threat modeling for cloud data center infrastructures». In: *International Symposium on Foundations and Practice of Security*. 2016 (cit. on p. 23).

[20] Tony UcedaVélez. *What is PASTA Threat Modeling?- Step Into the Kitchen As the Co-Founder of PASTA Threat Modeling Breaks Down the 7 Steps*. Nov. 2021. URL: https://versprite.com/blog/what-is-pasta-threat-modeling/ (cit. on pp. 23, 24).

[21] Tony Ucedavélez and Marco M. Morana. «RISK CENTRIC THREAT MODELING -Process for Attack Simulation and Threat Analysis». In: Hoboken, New Jersey: John Wiley and Sons, 2015. Chap. 7, pp. 343–344, 368–369, 419–420, 439–441, 457 (cit. on p. 24).

[22] Cyral Team. *Threat Modeling with DREAD*. URL: https://cyral.com/glossary/threat-modeling-with-dread/ (cit. on p. 24).

[23] Microsoft Team. *Threat modeling for drivers*. Dec. 2021. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers#the-dread-approach-to-threat-assessment (cit. on p. 25).

[24] Webroot featuring Gartner. *Threat Intelligence: What is it, and how can it protect you from today's Advanced Cyber-Attacks?* 2014. URL: https://www.gartner.com/imagesrv/media-products/pdf/webroot/issue1_webroot.pdf (cit. on p. 26).

[25] Recorded Future. *What is Threat Intelligence?* URL: https://www.recordedfuture.com/threat-intelligence/ (cit. on pp. 27–33).

[26] Adam Shostack. «The Threat Intelligence Handbook. A practical Guide for Security Teams to unlocking the Power of Intelligence». In: 1997 Annapolis Exchange Parkway Suite 300 Annapolis, CyberEdge Group, LLC, 2018. Chap. 1, pp. 1–10 (cit. on pp. 27–29).

[27] S. Samtani M. Abate V. Benjamin W. Li. *Cybersecurity as an Industry: A Cyber Threat Intelligence Perspective*. 2019. URL: https://www.researchgate.net/publication/335688262_Cybersecurity_as_an_Industry_A_Cyber_Threat_Intelligence_Perspective (cit. on p. 28).

[28] Emily McLaughlin. *CISO (chief information security officer)*. 2021. URL: https://www.techtarget.com/searchsecurity/definition/CISO-chief-information-security-officer (cit. on p. 30).

[29] Infosavvy. *Types of Threat Intelligence*. URL: https://info-savvy.com/types-of-threat-intelligence/ (cit. on pp. 30, 31).

[30] Adam Shostack. «The Threat Intelligence Handbook. A practical Guide for Security Teams to unlocking the Power of Intelligence». In: 1997 Annapolis Exchange Parkway Suite 300 Annapolis, CyberEdge Group, LLC, 2018. Chap. 2, pp. 11–19 (cit. on p. 31).

[31] Adam Shostack. «The Threat Intelligence Handbook. A practical Guide for Security Teams to unlocking the Power of Intelligence». In: 1997 Annapolis Exchange Parkway Suite 300 Annapolis, CyberEdge Group, LLC, 2018. Chap. 3, pp. 19–31 (cit. on p. 32).

[32] Adam Shostack. «The Threat Intelligence Handbook. A practical Guide for Security Teams to unlocking the Power of Intelligence». In: 1997 Annapolis Exchange Parkway Suite 300 Annapolis, CyberEdge Group, LLC, 2018. Chap. 4, pp. 31–42 (cit. on p. 32).

[33] Adam Shostack. «The Threat Intelligence Handbook. A practical Guide for Security Teams to unlocking the Power of Intelligence». In: 1997 Annapolis Exchange Parkway Suite 300 Annapolis, CyberEdge Group, LLC, 2018. Chap. 6, pp. 51–59 (cit. on p. 32).

[34] Kaspersky. *What is Typosquatting? – Definition and Explanation.* URL: https://www.kaspersky.com/resource-center/definitions/what-is-typosquatting (cit. on p. 33).

[35] RSI security. *WHAT'S A FACTOR ANALYSIS OF INFORMATION RISK ASSESSMENT?* 2020. URL: https://blog.rsisecurity.com/whats-a-factor-analysis-of-information-risk-assessment/ (cit. on p. 34).

[36] Adam Shostack. «The Threat Intelligence Handbook. A practical Guide for Security Teams to unlocking the Power of Intelligence». In: 1997 Annapolis Exchange Parkway Suite 300 Annapolis, CyberEdge Group, LLC, 2018. Chap. 8, pp. 67–75 (cit. on p. 35).

[37] Hornet security blog team. *Cyber Kill Chain Increasing IT security in companies step-by-step.* URL: https://www.hornetsecurity.com/en/knowledge-base/cyber-kill-chain/?_adin=02021864894/ (cit. on p. 36).

[38] Netskope blog team. *What is the Cyber Security Kill Chain?* URL: https://www.netskope.com/security-defined/cyber-security-kill-chain/ (cit. on p. 36).

[39] Circl Luxemburg team. *MISP - Open Source Threat Intelligence Platform.* URL: https://www.circl.lu/services/misp-malware-information-sharing-platform/ (cit. on p. 36).

[40] CRIS CARREON. *Applying Threat Intelligence to the Diamond Model of Intrusion Analysis.* 2018. URL: https://www.recordedfuture.com/diamond-model-intrusion-analysis/ (cit. on p. 37).

[41] Cyware blog team. *What is the Diamond Model of Intrusion Analysis?* 2021. URL: https://cyware.com/educational-guides/cyber-threat-intelligence/what-is-the-diamond-model-of-intrusion-analysis-5f02 (cit. on p. 37).

[42] MITRE team. *MITRE ATT&CK.* URL: https://attack.mitre.org/ (cit. on p. 37).

[43] Tim Matthews. *What is the MITRE ATT&CK Framework? A Complete Guide.* 2019. URL: https://www.exabeam.com/information-security/what-is-mitre-attck-an-explainer/ (cit. on pp. 38–40).

[44] Balbix team. *What Is a CVE?* 2021. URL: https://www.balbix.com/insights/what-is-a-cve/ (cit. on p. 40).

[45] Jon Baker. *Using MITRE ATT&CK® to Describe Vulnerabilities*. 2021. URL: https://github.com/center-for-threat-informed-defense/attack_to_cve/blob/master/methodology.md (cit. on p. 41).

[46] Oasis team. *Sharing threat intelligence just got a lot easier! STIX and TAXII*. 2021. URL: https://oasis-open.github.io/cti-documentation/ (cit. on p. 42).

[47] Tutorials Point team. *XML Overview*. URL: https://www.tutorialspoint.com/xml/xml_overview.htm (cit. on p. 42).

[48] MITRE. *STANDARDIZING CYBER THREAT INTELLIGENCE INFORMATION WITH THE STRUCTURED THREAT INFORMATION EXPRESSION (STIX™)*. Tech. rep. MITRE, 2012 (cit. on pp. 42, 43).

[49] Oasis team. *Comparing STIX 1.X/CybOX 2.X with STIX 2*. 2021. URL: https://oasis-open.github.io/cti-documentation/stix/compare.html (cit. on p. 42).

[50] Oasis team - Chris Lenk. *Introduction to Structured Threat Information Expression (STIX)*. 2021. URL: https://github.com/oasis-open/cti-documentation/blob/master/docs/Introduction_to_Structured_Threat_Information_Expression.pdf (cit. on p. 43).

[51] Oasis team. *Introduction to STIX*. 2021. URL: https://oasis-open.github.io/cti-documentation/stix/intro (cit. on p. 44).

[52] Oasis CTI technical committee. *TAXII™ Version 2.1*. 2021. URL: https://docs.oasis-open.org/cti/taxii/v2.1/os/taxii-v2.1-os.pdf (cit. on pp. 45, 46).

[53] Oasis CTI technical committee. *About TAXII (Archive)*. URL: https://taxiiproject.github.io/about/ (cit. on p. 46).

[54] D. Fensel. R. Studer V. R. Benjamins. «Data & Knowledge Engineering.» In: *Knowledge Engineering: Principles and Methods*. Vol. 25. 1998, pp. 161–197 (cit. on p. 47).

[55] Verónica Mateos Lanchas. «Contribuición a la automatización de sistemas de respuesta frente a intrusiones mediante ontologias.» PhD thesis. Madrid: ETSIT-Universidad Politécnica de Madrid, Nov. 2013. URL: https://oa.upm.es/21885/ (cit. on pp. 48, 50, 51, 54, 56, 58, 59, 61–64, 72, 73, 83, 94, 99, 100).

[56] Ontotex team. *What Is the Semantic Web?* URL: https://www.ontotext.com/knowledgehub/fundamentals/what-is-the-semantic-web/ (cit. on p. 48).

[57] IGIGlobal team. *What is Ontology in Computer Science*. URL: https://www.igi-global.com/dictionary/ontology-in-computer-science/21124 (cit. on p. 48).

[58] Matthew Horridge. *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools*. Tech. rep. Edition 1.3. University of Manchester, 2011 (cit. on p. 48).

[59] Wikipedia. *What Is the Semantic Web?* 2022. URL: https://en.wikipedia.org/wiki/Ontology_components (cit. on p. 48).

[60] Guy Lapalme. «XML: Looking at the Forest Instead of the Trees». In: Département d'informatique et de recherche opérationnelle Université de Montréal, 2019. Chap. 7. URL: https://www.iro.umontreal.ca/~lapalme/ForestInsteadOfTheTrees/HTML/ch07s01.html (cit. on p. 50).

[61] Angus Addlesee. *Understanding Linked Data Formats- Turtle vs RDF/XML vs N-Triples vs JSON-LD*. 2018. URL: https://en.wikipedia.org/wiki/Ontology_components (cit. on p. 50).

[62] R.V. Guha Dan Brickley. *RDF Schema 1.1*. 2014. URL: https://www.w3.org/TR/rdf-schema/ (cit. on p. 51).

[63] Wikipedia. *Web Ontology Language*. 2022. URL: https://en.wikipedia.org/wiki/Web_Ontology_Language (cit. on p. 54).

[64] OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. 2012. URL: https://www.w3.org/TR/owl2-overview/ (cit. on pp. 54, 56).

[65] Sarika Jain Sonia Mehla. *Rule Languages for the Semantic Web: Proceedings of IEMIS 2018, Volume 1*. 2019. URL: https://www.researchgate.net/publication/329605032_Rule_Languages_for_the_Semantic_Web_Proceedings_of_IEMIS_2018_Volume_1 (cit. on p. 56).

[66] TopQuadrant team. *SPIN SPARQL Inferencing Notation*. 2013. URL: https://www.topquadrant.com/technology/sparql-rules-spin/ (cit. on p. 57).

[67] Adrian Paschke Harold Boley and Omair Shafiq. *RuleML 1.0: The Overarching Specification of Web Rules*. 2014. URL: http://cs.unb.ca/~boley/papers/RuleML-Overarching.pdf (cit. on p. 58).

[68] Indeed Editorial Team. *7 Types of Reasoning: Definitions and Examples*. 2021. URL: https://www.indeed.com/career-advice/career-development/types-of-reasoning (cit. on p. 63).

[69] UMass Dartmouth team. *Decision-making process*. URL: https://www.umassd.edu/fycm/decision-making/process/ (cit. on pp. 65, 66).

[70] Prachi Juneja. *What is Decision Making ?* URL: https://www.managementstudyguide.com/what-is-decision-making.htm (cit. on p. 66).

[71] Prachi Juneja. *Decision Support Systems – Introduction, Categorization and Development*. URL: https://www.managementstudyguide.com/decision-support-systems.htm (cit. on p. 66).

[72] blogspot team. *Components of Decision Support System*. 2010. URL: http://dsssystem.blogspot.com/2010/01/components-of-decision-support-systems.html (cit. on p. 67).

[73] TechTarget contributor. *Decision Support System (DSS)*. 2021. URL: https://www.techtarget.com/searchcio/definition/decision-support-system (cit. on pp. 68, 70).

[74] Prachi Juneja. *Gaining Competitive Advantage with Decision Support Systems*. URL: https://www.managementstudyguide.com/gaining-competitive-advantage-with-decision-support-systems.htm (cit. on p. 68).

[75] Prachi Juneja. *Limitations & Disadvantages of Decision Support Systems*. URL: https://www.managementstudyguide.com/limitations-and-disadvantages-of-decision-support-systems.htm (cit. on p. 68).

[76] Wikipedia. *Clinical Decision support system*. 2022. URL: https://en.wikipedia.org/wiki/Clinical_decision_support_system (cit. on p. 69).

[77] José Fernán Martíne et al. Zhaoyu Zhai. *Decision support systems for agriculture 4.0: Survey and challenges*. 2020. URL: https://www.sciencedirect.com/science/article/pii/S0168169919316497 (cit. on p. 70).

[78] Pasquale Malacariac et al. Andrew Fieldera Emmanouil Panaousisb. *Decision support approaches for cyber security investments*. 2016. URL: https://www.researchgate.net/publication/272521196_Decision_support_approaches_for_cyber_security_investment (cit. on p. 70).

[79] Tadeusz Sawik. *Selection of optimal countermeasure portfolio in IT security planning*. 2012. URL: https://www.sciencedirect.com/science/article/pii/S0167923621001093?via%3Dihub (cit. on p. 71).

[80] Terry Rakes Jason K. DeaneLoren P. ReesLoren P. Rees. *IT security planning under uncertainty for high-impact events*. 2012. URL: https://www.researchgate.net/publication/227419287_IT_security_planning_under_uncertainty_for_high-impact_events (cit. on p. 71).

[81] Pasquale Malacaria Yunxiao Zhang. *Bayesian Stackelberg games for cybersecurity decision support*. 2021. URL: https://www.sciencedirect.com/science/article/pii/S0167923621001093?via%3Dihub (cit. on p. 71).

[82] I. Kotenko O. Polubelova I. Saenko E. Doynikova. *The Ontology of Metrics for Security Evaluation and Decision Support in SIEM Systems*. 2013. URL: https://www.researchgate.net/publication/261332134_The_Ontology_of_Metrics_for_Security_Evaluation_and_Decision_Support_in_SIEM_Systems (cit. on p. 72).

[83] J.Wong N.Stackhanova S.Basu. «A Cost-Sensitive Model for Preemptive Intrusion Response Systems». In: *Advanced Information Networking and Applications, 2007. AINA '07. 21st International Conference*. 2007 (cit. on p. 97).

[84] Owais Qayyum Asad Ali. *Inference New Knowledge Using Sparql Construct Query*. 2019. URL: https://ieeexplore.ieee.org/document/8673494 (cit. on p. 125).