POLITECNICO DI TORINO

Master's Degree in Cybersecurity

Politecnico
di Torino

Master's Degree Thesis

Development of an Ontology-based Tool for Risk Assessment Automation

Supervisors

Prof. Alessandro SAVINO

Dr. Nicolò MAUNERO

Candidate

Christian CASALINI

ACADEMIC YEAR 2022-2023

Abstract

Cybersecurity is an ever-evolving challenge in today's digital landscape. The growing dependence on digital technology, both in personal and business contexts, has resulted in a considerable increase in the potential impact of cyberattacks, with a corresponding increase in the severity of the outcomes of a cyber incident. Thus, the assessment of cybersecurity risks has become increasingly critical in recent times and, in order to effectively improve the cybersecurity posture of organizations, it should be considered as an ongoing process rather than a one-time task.

The goal of this thesis is to develop a tool to assist security teams in identifying and assessing potential vulnerabilities and threats against the system under analysis in a more efficient, faster, and methodical manner. The tool takes as input an ontology that describes the ICT infrastructure under analysis and automatically enriches it with relevant security data. The tool operates on two primary fronts to achieve this objective.

The first front involves retrieving known vulnerabilities and weaknesses that affect the system's assets, as well as attack tactics and techniques that could exploit them. All this information is automatically retrieved from public knowledge bases such as CVE and CWE.

The second front involves leveraging an ontology reasoner and the rules defined within the ontology, along with the detailed information describing the ICT infrastructure, such as assets and data flows, to infer threats affecting the various parts of the system. Additionally, for each identified threat, the tool computes a risk score, which helps the security team prioritize the work required to improve the cybersecurity posture.

This approach offers several benefits that make the vulnerability and threat mapping process considerably more efficient. The tool eliminates the need for manual and time-consuming tasks associated with vulnerability and threat mapping, thus accelerating the process. Routine checks and updates are easy to carry out, ensuring that the analysis remains up-to-date with the latest data from public knowledge bases. This adaptability is particularly useful when changes occur within the system, such as the replacement of some components within the system itself, as updating the base ontology describing the ICT infrastructure is enough to

accurately reflect these changes. Lastly, the proposed solution is highly flexible as it accommodates manual additions of vulnerabilities and threats missing from public knowledge bases, thus allowing for the inclusion of proprietary or system-specific knowledge, further enhancing the overall analysis.

In summary, the tool created in this thesis is a significant step forward in taking a proactive and continuous approach to risk assessment. Automating the collection of security data and using ontology reasoning simplifies the assessment process and guarantees that it stays current with the dynamic threat landscape, thus helping to improve the cybersecurity posture of organizations.

Acknowledgements

I would like to express my gratitude to Professor Alessandro Savino, my thesis advisor. I am immensely grateful to Doctor Nicolò Maunero, whose unwavering dedication, expertise, and encouragement have been instrumental in driving me towards the completion of this thesis. My family has been the bedrock of my strength, and I deeply appreciate their unwavering support and belief in me. A special thanks to Diana, for her enduring love, patience, and support. Lastly, I would like to express my thanks to all those who have contributed to my achievement, whether directly or indirectly.

Table of Contents

List of Tables

List of Figures

Acronyms

ATT&CK
Adversarial Tactics, Techniques, and Common Knowledge

CAPEC
Common Attack Pattern Enumeration and Classification

CPE
Common Platform Enumerations

CVSS
Common Vulnerability Scoring System

CVE
Common Vulnerabilities and Exposures

CWE
Common Weakness Enumeration

HSM
Hardware Security Module

ICT
Information and Communication Technologies

IPS
Intrusion Prevention System

NIST
National Institute of Standards and Technology

NVD

National Vulnerability Database

SOAR

Security Orchestration, Automation, and Response

SWRL

Semantic Web Rule Language

VAPT

Vulnerability Assessment and Penetration Testing

Chapter 1

Introduction

In the current digital landscape, where interconnected digital systems are essential for organizations and individuals, the security of these systems has assumed paramount importance. The rapid acceleration of technological advancements has driven unprecedented innovation along with new and complex cybersecurity challenges. Companies in all industries depend on information technology to carry out essential business functions today, exposing them to cybercriminals, employee mistakes, natural disasters, and other cybersecurity threats [1]. To effectively address these challenges, it is crucial to identify, assess, and mitigate potential threats and risks to digital assets.

Cybersecurity is a field that encompasses a wide range of strategies, practices, and technologies aimed at safeguarding digital assets, networks, and information. As we become increasingly dependent on technology, the risk of malicious actors taking advantage of weaknesses and posing a danger increases. This highlights the need to implement robust cybersecurity measures to protect against data breaches, service disruptions, and other cyberattacks.

In addition to the growing importance of cybersecurity, we are witnessing an increasing trend in formulating regulations and standards by governments and industry bodies. These norms are tailored to address the unique challenges an interconnected world poses. Governments are enacting stringent guidelines to safeguard critical public infrastructures, while private companies are recognizing the importance of adhering to certifications such as ISO standards. These regulations serve as a key driving force, ensuring that cybersecurity practices are not only proactive but also aligned with internationally recognized best practices. They

reinforce the imperative that public and private organizations fortify their digital defenses in the face of an evolving cyber threat landscape.

Cyber risk management, also called cybersecurity risk management, is the process of identifying, prioritizing, managing, and monitoring risks to information systems. Companies across industries use this process to protect information systems from cyberattacks and other digital and physical threats [1]. In the most widely adopted Risk Management frameworks, one of the critical parts of the whole process is Risk Assessment.

Risk assessment evaluates various stages of the life cycle of a system, from design to implementation, to gauge its cybersecurity posture and readiness to withstand potential cyberattacks. The Risk Assessment process can be broken down into three main steps: (i) *Threat Modeling*, which looks for potential threats to the system, such as possible attack vectors and any weaknesses or flaws that attackers could exploit; (ii) *Vulnerability Assessment* (VA), which involves the repeated process of identifying vulnerabilities; and (iii) *Penetration Testing* (PT), which involves a thorough analysis of the identified vulnerabilities and potential problems by attempting to exploit them. These three phases assess the system's exposure to cyber risk through their combined efforts.

As the complexity of digital infrastructures and the number of potential vulnerabilities and threats increase, manual approaches to Risk Assessment become overwhelming and prone to error. In this context, automation emerges as a powerful tool that accelerates the process and serves as a crucial defense against human errors. Automation drastically reduces the time required to evaluate cyber risks, enabling organizations to assess large and intricate systems in a fraction of the time it would take using manual methods. This scalability is crucial to stay ahead in a world where technology deployment is rapid and extensive.

The efficiency gained through automation optimizes resource allocation and allows cybersecurity professionals to focus on more complex and strategic security aspects. It enables organizations to make the most of their human expertise.

Another noteworthy benefit is the consistency and standardization that automation brings to the Risk Assessment process. Automated tools adhere rigorously to predefined methodologies and guidelines, eliminating the risk of human error and ensuring that Risk Assessments comply with industry standards. This level of consistency is particularly vital when assessing a multitude of assets or systems.

This thesis presents an automated tool designed to simplify the threat modeling and

risk assessment process, providing a practical solution to organizations struggling with cybersecurity issues. The main function of the tool relies on the use of an ontology, which is a structured representation that includes everything related to the ICT infrastructure, such as resources, networks, data flows, and more. This ontology also contains carefully defined SWRL rules that serve as guiding principles for a powerful reasoner. These rules, in concert with the ontology, enable the tool to establish and map Common Attack Pattern Enumeration and Classification (CAPEC) entities affecting the intricate ICT system delineated within the ontology.

Moreover, the tool leverages the vast repository of resource-specific Common Platform Enumerations (CPEs) to query renowned databases, notably NIST and MITRE, extracting and integrating the latest Common Vulnerabilities and Exposures (CVEs) and Common Weakness Enumerations (CWEs), as well as ATT&CK Tactics and Techniques, along with their mitigations.

The automated process ends up returning a meticulously populated ontology containing essential information about CVEs, CWEs, CAPECs, and ATT&CK Tactics and Techniques that could affect the components of the ICT infrastructure. This comprehensive overview of security-related data empowers organizations to take informed steps in identifying and mitigating potential security risks.

To emphasize the practical utility of the tool, this thesis includes a case study illustrating its real-world application. Through this case study, we demonstrate how the tool can be effectively employed to detect and address security vulnerabilities in an authentic operational environment. Furthermore, the ontology's adaptability extends to the incorporation of additional data gleaned from Vulnerability Assessment and Penetration Testing (VAPT), further enriching the wealth of information at the user's disposal.

The developed tool greatly improves the process of threat modeling and risk assessment. By automating and streamlining these crucial tasks, organizations can better protect their digital assets and information against potential security breaches. This tool serves as a powerful ally for cybersecurity professionals, providing them with a comprehensive and efficient solution.

Chapter 2

Background

Before we dive into the details of this project, it is crucial to gain a better understanding of some key concepts. In this chapter, we will explore Risk Management, as well as what an ontology is and why it is a suitable choice for our objectives. Additionally, we will provide a brief overview of the common knowledge bases used by the tool to extract known vulnerabilities, threats, attack patterns, and adversary tactics and techniques to populate the ontology.

2.1 Risk Management

Risk is a foundational concept that underlies virtually all decision-making processes, and in the world of cybersecurity, it takes on a particularly critical role. At its core, risk represents the likelihood and potential consequences of undesirable events occurring. In cybersecurity, these events typically involve the compromise of an organization's information systems, data, or digital assets.

The U.S. NIST (National Institute of Standards and Technology) defines cybersecurity risk as *"an effect of uncertainty on or within information and technology. Cybersecurity risks relate to the loss of confidentiality, integrity, or availability of information, data, or information (or control) systems and reflect the potential adverse impacts to organizational operations (i.e., mission, functions, image, or reputation) and assets, individuals, other organizations, and the Nation"* [2].

What we can get out of that definition, is that cybersecurity risk is a constant concern that must be addressed with utmost seriousness. With the ever-changing and dynamic nature of cyber threats, organizations must acknowledge that they face numerous risks such as cyberattacks, data breaches, malware infections, and other malicious activities. These threats can potentially cause significant damage, ranging from financial loss to reputation damage, legal consequences, and operational disruptions. That's why it's imperative for companies, organizations, and governments to prioritize cybersecurity risk management as a fundamental process.

Risk management is a complex and multifaceted process that requires the participation of the entire organization. From senior executives who provide strategic vision and objectives, to midlevel leaders who plan, execute, and manage projects, to individuals on the front lines who operate information systems, risk management must be addressed holistically. It is a comprehensive process that requires organizations to: (i) frame risk (i.e., establish the context for risk-based decisions); (ii) assess risk; (iii) respond to risk once determined; and (iv) monitor risk on an ongoing basis using effective organizational communications and a feedback loop for continuous improvement in the risk-related activities of organizations. Risk management must be integrated into every aspect of the organization, from the strategic level to the tactical level [3].

In summary, the concept of risk in cybersecurity is fundamental, as it underscores the need for proactive and adaptive measures to safeguard an organization's digital assets and operations from an ever-evolving landscape of threats. By comprehensively understanding and managing these risks, organizations can reduce their exposure to potential harm and improve their resilience in the face of cyber threats.

2.1.1 Risk Assessment

Risk assessment is a crucial process that involves identifying, evaluating, and ranking potential information security threats. To accurately assess risk, it is necessary to carefully analyze threat and vulnerability data to determine the degree to which an organization could be impacted by negative events or circumstances, as well as the probability that such events or circumstances will occur. This information is essential for organizations to develop effective strategies and implement appropriate security measures to mitigate risk and protect against potential security breaches [4].

The purpose of Risk Assessment is to identify: (i) threats to the system under analysis (i.e., operations, assets, or individuals); (ii) vulnerabilities internal and external to the system; (iii) the harm (i.e., consequences/impact) to the system that may occur given the potential for threats exploiting vulnerabilities; and (iv) the likelihood that the given harm will occur. The final result is an evaluation of the potential risk involved, taking into account the level of harm that could be caused and the probability of occurrence [3].

It is crucial to understand that while a risk evaluation serves as a helpful guide when devising an incident response plan, it does not represent the ultimate state of an organization's cybersecurity posture. A cyber risk assessment is not a one-time event; rather, it should be conducted continuously as a means of assessing an organization's cybersecurity measures and refining them as new technologies and methodologies are developed and adopted. This ongoing process is essential to ensure that an organization's cybersecurity posture remains robust and effective against the ever-evolving threat landscape it faces. Therefore, it is imperative that organizations make cyber risk assessments a regular practice to keep their security measures up-to-date and effective at all times [5].

As mentioned in Chapter 1, the Risk Assessment process can be broken down into three main steps: (i) *Threat Modeling*, (ii) *Vulnerability Assessment* and (iii) *Penetration Testing*. Since the tool object of this work mainly focuses on automating the first of these steps, we will only discuss Threat Modeling in detail.

2.1.2 Threat Modeling

Threat modeling is an approach to analyzing the security of a system (e.g., device, application) or a system of systems (e.g., multi-component system) so that vulnerabilities can be identified, enumerated, and prioritized. Threat modeling typically employs a systematic approach to identifying assets most desired by an attacker and related attack vectors. This step leads to the decomposition of the system by investigating each asset and attack vector individually and determining the kind of attacks to which they are vulnerable. From this effort, a list of vulnerabilities is created for the system and ordered in terms of risk, potential to cause harm, or any other criteria deemed appropriate [6].

The threat modeling process can be decomposed into three high-level steps [7, 8]:

1. *Decompose the Application* [8]:

- Define and Evaluate the Assets (i.e. items or areas that the attacker could be interested in)
- Identify the trusted boundaries, both internal and external, of the system
- Identify Data Flow Diagrams, which describe the flow of data among the different components of the product under analysis (no matter if it is a software program or an entire IT infrastructure)

2. *Determine and Rank Threats* [8]: critical to the identification of threats is using a threat categorization methodology. A threat categorization such as STRIDE, useful in the identification of threats by classifying attacker goals, can be used.

3. *Determine Countermeasures and Mitigation* [8]: a vulnerability may be mitigated with the implementation of a countermeasure. Such countermeasures can be identified using threat-countermeasure mapping lists. Once a risk ranking is assigned to the threats in step 2, it is possible to sort threats from the highest to the lowest risk and prioritize mitigation efforts.

By ranking and prioritizing the most significant threats, Threat Modeling ensures that resources are allocated effectively to develop and maintain adequate defenses. Secondly, the iterative application of threat modeling ensures that newly discovered threats are promptly and appropriately mitigated, thereby enhancing the overall security of the system [9].

There are several ways in which the process of threat modeling can be defined, depending on the domain of interest and the goal to be achieved [10].

The range of threat modeling approaches is broad and varied, reflecting the complexity of the cybersecurity field. This study focuses mainly on the STRIDE methodology, but it is important to recognize the existence of other approaches. These alternatives vary in complexity, resources, and specialization, allowing organizations to select the most appropriate framework for their specific needs.

2.1.3 Vulnerability Assessment

Vulnerability Assessment is a pivotal process in cybersecurity that systematically identifies, analyzes, and prioritizes vulnerabilities within a system or network. This approach plays a crucial role in understanding potential weaknesses that could be exploited by attackers. The process involves scanning the ICT infrastructure

to uncover known vulnerabilities, such as Common Vulnerabilities and Exposures (CVEs), associated with various assets. Each asset is scrutinized to determine its susceptibility to different attack vectors.

The Vulnerability Assessment process can be delineated into several key steps:

- Asset Identification and Evaluation:
  - Define and assess the assets within the ICT infrastructure.
  - Delve into the trusted boundaries, both internal and external, to establish the system's security perimeter.
  - Identify the flow of data among different components using data flow diagrams.
- CVE Enumeration and Risk Ranking:
  - Employ a threat categorization methodology, such as the Common Weakness Enumeration (CWE) or other relevant frameworks.
  - Enumerate and rank vulnerabilities according to their potential risk and impact on the system.
- Countermeasures and Remediation:
  - Map identified vulnerabilities to suitable countermeasures.
  - Prioritize mitigation efforts by ranking and addressing the most critical vulnerabilities first.
  - By systematically conducting Vulnerability Assessment, security practitioners gain insights into the potential risks facing the ICT infrastructure. This structured approach facilitates the identification, quantification, and remediation of security risks associated with the system.

Ranking and prioritizing vulnerabilities ensure that security resources are effectively allocated to address the most critical threats. Iterative Vulnerability Assessments further enhance the security posture by promptly addressing newly discovered vulnerabilities. This proactive approach contributes to an overall improvement in the resilience of the system against potential cyber threats.

Although the work presented in this thesis primarily focuses on the automated identification of vulnerabilities through the exploitation of publicly available knowledge bases, it is important to acknowledge the diversity of approaches within the

broader domain of Vulnerability Assessment. Organizations can explore various frameworks and methodologies to tailor their approach according to specific needs and contextual nuances within their cybersecurity landscape.

2.2 Ontology

Ontology [11], a term derived from Greek words “òntos”, meaning existence or being real, and “lògos”, meaning science or study, plays a significant role in the field of cybersecurity [12]. It provides a systematic description of all the terms in a specific subject area, their characteristics or attributes, and their relationships [13].

In today’s digital age, the exponential growth of data has made cybersecurity more critical than ever before. As the number of cybersecurity threats increases, public and private entities are struggling to manage them. Unfortunately, the absence of global standards is obstructing the cooperation between organizations to address these issues. To tackle this challenge, it is necessary to create ontologies for cybersecurity matters that can offer a shared comprehension of cybersecurity domains [12].

A valuable resource for delving into the concept of ontology and its applications in computer science is presented in [14]. The paper highlights the role of ontologies in the capture of knowledge within a specific domain, elucidating the concepts within that domain and the relationships that exist between them. Various ontology languages offer distinct capabilities, with OWL (Web Ontology Language) from the World Wide Web Consortium (W3C) being a recent standard. OWL enables the unambiguous description of concepts based on set theory and logic, facilitating the construction of complex concepts from simpler ones. The logical model supports the use of a reasoner, which can verify the mutual consistency of all statements and definitions within the ontology. It also helps in recognizing the hierarchy of concepts, especially in cases where classes can have multiple parents. Additionally, the reasoner can infer additional information; for example, if two properties are inverses, the user only needs to assert one value, and the reasoner automatically infers the inverse value. This functionality improves the accuracy of maintaining hierarchies and reduces the manual effort required by users [14].

An ontology is composed by *Classes*, *Properties* and *Individuals*.

- *Individuals* represent objects in the domain of interest (i.e. an individual is an

instance of a class)

- *Properties* represent relationships between individuals or give some information about them. There are three different types of properties:

  1. *Object Properties* are relationships between two individuals, like *attends* and its inverse *isAttendedBy* in the proposed example in Figure 2.1. In the example, it is worth noting that some *object properties* have been defined by the ontology maintainer, while other ones have been inferred by the ontology reasoner;
  2. *Data Properties* are relations between an individual and a datatype, such as xsd:string or xsd:integer, giving some additional information about the individual. In Figure 2.1, *age* is a *data property* for individuals of classes *Student* and *Professor* (that could be made sub-classes of a class *Person*);
  3. *Annotation Properties* are generally used for meta-data, like a comment or a label

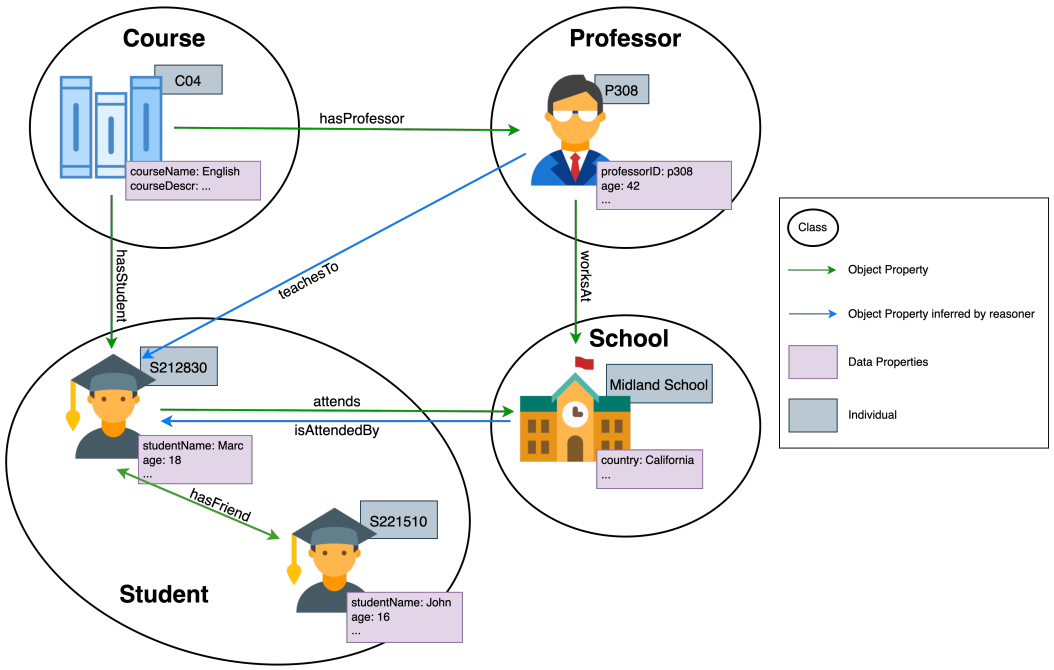


Figure 2.1: A simple example to grasp the basics of ontologies

In summary, relying on an ontology allows us to describe an ICT infrastructure in a standard, machine-readable, and exhaustive way, thus placing foundations for a

common language that would also allow the cooperation of different organizations. Additionally, as seen in the example, provided some initial data and some well-defined rules, the ontology reasoner can infer other data that we need.

2.3 MITRE Frameworks

Effective cybersecurity practices rely on comprehensive frameworks to understand and address evolving threats. In this Section, we explore the foundational frameworks curated by MITRE, a leading organization dedicated to advancing cybersecurity knowledge and capabilities.

2.3.1 CWE

MITRE is a non-profit organization that specializes in providing technical assistance, systems engineering, and research and development services to a variety of U.S. government departments. Their primary focus is to address critical challenges related to national security, healthcare, and other areas of public interest.

"CWE (Common Weakness Enumeration) is a community-developed list of common software and hardware weakness types that have security ramifications. A "weakness" is a condition in a software, firmware, hardware, or service component that, under certain circumstances, could contribute to the introduction of vulnerabilities. The CWE List and associated classification taxonomy serve as a language that can be used to identify and describe these weaknesses in terms of CWEs" [15].

The CWE (Common Weakness Enumeration) is a valuable resource for both development and security practitioner communities. It primarily aims to prevent vulnerabilities at the source by educating architects, designers, programmers, and acquirers on how to eliminate common errors before delivering products. The use of CWE helps prevent security vulnerabilities that have been a significant threat to software and hardware industries, thereby safeguarding enterprises from potential risks.

With CWE, developers and security practitioners can describe and discuss weaknesses in software and hardware in a common language, check for weaknesses in existing products, and evaluate coverage of tools targeting these weaknesses. It also provides a common baseline standard for weakness identification, mitigation,

and prevention efforts, thereby preventing software and hardware vulnerabilities before deployment [15].

The CWE List includes both software and hardware weakness types, with new content continually added to refine weakness classification trees. Since its inception in 2006, the initiative has been a community effort to develop specific and succinct definitions for each common weakness type. The CWE List is fully searchable and may be viewed or downloaded in its entirety. Additionally, the unique feature of CWE is the ability to engage with content from distinct viewpoints.

Incorporating CWE List content into research, educational materials, processes, and tools is free, per the terms of use. By leveraging the widest possible group of interests and talents, CWE ensures that each item in the list is adequately described and differentiated, making it a valuable resource for software and hardware industries.

2.3.2 CVE

The original concept for what would become the CVE knowledge base was presented as a white paper entitled "Towards a Common Enumeration of Vulnerabilities" at the 2nd Workshop on Research with Security Vulnerability Databases on January 21-22, 1999 at Purdue University in West Lafayette, Indiana, USA [16].

MITRE CVE (Common Vulnerabilities and Exposures) was established to provide a common and structured method for identifying and naming vulnerabilities in computer software and hardware, thus helping security professionals and organizations communicate about security issues effectively. It is worth mentioning that the MITRE Corporation does not enumerate the vulnerabilities and exposures all by itself, but the work is rather a collaborative effort involving multiple organizations, security experts, and the broader cybersecurity community. In fact, "in 2016 the CVE Program began actively expanding the number of organizations participating as CVE Numbering Authorities (CNAs). CNA partners are how the CVE List is built. Every CVE Record is added by a CNA. This expansion continues today with more and more organizations from around the world deciding to partner with the CVE Program to become a CNA" [16].

Every vulnerability on the CVE List is represented by a single CVE Record. These vulnerabilities are initially discovered by security researchers or ordinary users who then report them to a CVE Program participant. Upon receiving the report, the CVE Program assigns a CVE ID to the vulnerability. Once the reported

vulnerability is confirmed by identifying the essential data elements required for a CVE Record, the record is published to the CVE List. The CVE Records are made available to the public by the CVE Program's partners worldwide. An overview of the process is given in Figure 2.2.

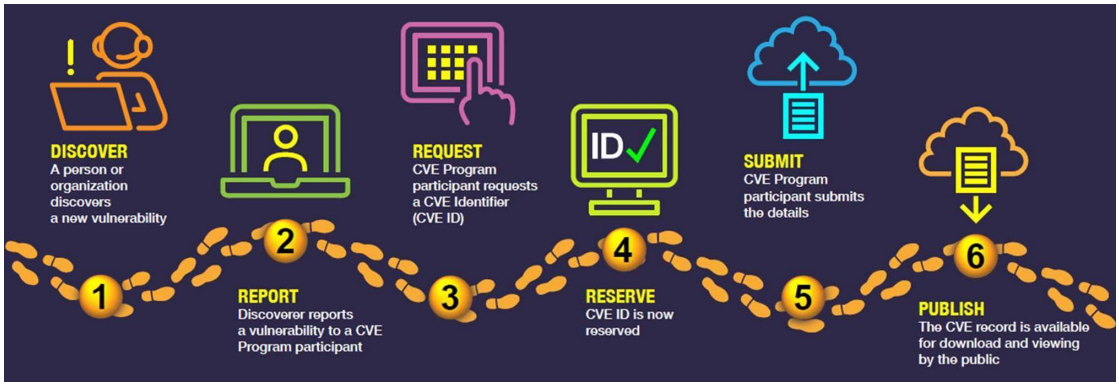


Figure 2.2: CVE Record Lifecycle [17]

Each CVE entry is a unique identifier for a specific security vulnerability or exposure. These identifiers are used to reference and discuss vulnerabilities, which makes it easier to track and manage security issues.

2.3.3 CAPEC

In the field of cybersecurity, it is becoming more and more crucial to comprehend the behavior of adversaries. To this end, there are two distinct approaches for categorizing knowledge about adversary behavior - CAPEC and ATT&CK. Although each approach is tailored to a specific set of use cases, they provide a comprehensive framework for understanding adversary behavior.

"CAPEC (Common Attack Pattern Enumeration and Classification) is an effort to provide a publicly available catalog of common attack patterns classified in an intuitive manner, along with a comprehensive schema to describe related attacks and share information about those attacks" [18].

Attack Patterns are descriptions of the common tactics and strategies used by adversaries to take advantage of known vulnerabilities in cyber-enabled capabilities. They outline the difficulties that an attacker may encounter and how they can be overcome. These patterns are based on the idea of design patterns applied

in a destructive rather than constructive way and are created from a thorough examination of actual exploit cases.

Each attack pattern captures knowledge on the design and execution of a specific attack, additionally offering guidance on how to reduce the effectiveness of the attack itself. Attack patterns are useful for developers and administrators of cyber-enabled capabilities, as they provide a better understanding of the components of an attack and how to prevent them from being successful [19].

Figure 2.3: Connection between CAPEC, CWE and CVE [20]

As shown in Figure 2.3, a CAPEC attack pattern is a method of using a CWE to carry out an attack. Therefore, most CAPEC entries include an *execution flow*, which is a set of step-by-step instructions for an attacker to follow in order to identify potential targets, test their assets and defensive capabilities, and then execute the exploit.

2.3.4 ATT&CK

"MITRE ATT&CK is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community" [21].

ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) is a knowledge that contains adversarial techniques, which classify and describe in detail the offensive actions that bad actors could perform against specific targets, such as a

particular operating system. The focus here, unlike similar works in the same field, is on how the attackers interact with the system during an operation, rather than on the tools and malware they exploit.

The *techniques* are organized into a set of *tactics* in order to provide them with a context. Each technique includes information that is relevant to both a red team or a penetration tester to understand the nature of how a technique works and also to a defender to understand the context surrounding events or artifacts generated by a technique in use [22].

Tactics represent the "why" of an ATT&CK technique, while *techniques* represent "how". The tactic is the tactical objective of the adversary to perform an action. Tactics serve as useful contextual categories for individual techniques and cover standard, higher-level notation for things adversaries do during an operation, such as persist, discover information, move laterally, execute files, and exfiltrate data. However, techniques may also represent "what" an adversary gains by performing an action. This is a useful distinction for the Discovery tactic as the techniques highlight what type of information an adversary is after with a particular action. There may be many ways or techniques to achieve tactical objectives, so there are multiple techniques in each tactic category [22].

2.4 NIST Frameworks

In this Section, our focus turns to NIST (National Institute of Standards and Technology), a pivotal entity in shaping cybersecurity standards. We will explore two vital components—NVD (National Vulnerability Database) and CPE (Common Platform Enumeration). Understanding these tools is fundamental to our goal of automating vulnerability assessment and risk evaluation, forming the backbone of the tool designed for analyzing vulnerabilities in the system under scrutiny.

2.4.1 NVD

"*The NVD is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references,*

security-related software flaws, misconfigurations, product names, and impact metrics" [23].

NIST's U.S. National Vulnerability Database (NVD), a "comprehensive cybersecurity vulnerability database that integrates all publicly available U.S. Government vulnerability resources and provides references to industry resources" [16] is synchronized with, and based upon, the CVE List.

The NVD (National Vulnerability Database) serves as a central hub for cybersecurity information. Its primary purpose is to provide a structured and standardized repository of known software and hardware vulnerabilities. It can be considered as an encyclopedia of security weaknesses. Each vulnerability listed in the NVD is assigned a unique identifier known as a CVE (Common Vulnerabilities and Exposures) number, making it easy to reference and share information about specific vulnerabilities. CVEs are essential in the cybersecurity community because they provide a common language for discussing and prioritizing vulnerabilities. Security professionals, organizations, and software vendors use CVEs to identify, track, and remediate vulnerabilities in their systems.

Furthermore, NVD employs CPE (Common Platform Enumerations) as part of its structured data. CPEs are a standardized naming scheme that is used to identify and categorize hardware and software attributes. These attributes can include details such as the manufacturer, product name, and product version. CPEs play a vital role in associating specific system configurations with vulnerabilities listed in the NVD. This integration simplifies the process of understanding which systems are affected by a given vulnerability. Security professionals and organizations use CPEs to match vulnerabilities with their systems, allowing for a more precise and targeted threat assessment. Ultimately, NVD, along with CVEs and CPEs, is an invaluable resource that allows cybersecurity stakeholders to stay informed about threats and take proactive measures to secure their systems.

2.4.2 CPE

CPE (Common Platform Enumerations) is a standardized method for identifying and naming software applications, operating systems, and hardware devices in a structured and machine-readable format [24]. Developed by the National Institute of Standards and Technology (NIST), CPE is a part of the larger Common Vulnerabilities and Exposures (CVE) system, aiming to provide a universal language for uniquely identifying and describing information technology systems.

CPE employs a hierarchical structure composed of components, making it possible to represent the attributes of a software application or a device systematically. The structure consists of a vendor, product, version, update, and edition, allowing a precise identification of a specific IT entity. For instance, a CPE entry for a particular version of a web browser might look like this:

```
cpe:2.3:a:microsoft:internet_explorer:11.0:*:*:*:*:*:*:*
```

In this example, "microsoft" is the vendor, "internet_explorer" is the product, and "11.0" is the version. The other fields can be used for more granular specifications.

CPE plays a crucial role in NVD by providing a common syntax to uniquely identify and categorize vulnerabilities. Each vulnerability entry in the NVD is associated with specific CPE identifiers, allowing users to quickly understand which software, operating systems, or devices are affected by a particular security issue. This integration facilitates efficient vulnerability management, allowing organizations to assess their exposure to potential threats based on the technology components they use.

Security analysts and researchers take advantage of the structured information provided by CPE to link vulnerabilities reported in the NVD to specific IT assets. This connection improves the accuracy and effectiveness of vulnerability assessments and allows for streamlined communication about security issues within the cybersecurity community. In general, the use of CPE within the NVD framework contributes to a standardized and interoperable approach to vulnerability identification and management.

2.5 CVSS

The CVSS (Common Vulnerability Scoring System) is an open framework owned and managed by FIRST.Org, Inc. (FIRST), a US-based non-profit organization dedicated to supporting computer security incident response teams worldwide [25]. FIRST's mission is to facilitate collaboration and information sharing to enhance global cybersecurity.

CVSS offers a standardized methodology for assessing and communicating software vulnerabilities. This process entails three primary categories of metrics: Base, Temporal, and Environmental. Each category provides unique perspectives into

the characteristics of the vulnerability. Using these metrics, organizations can more effectively evaluate and compare vulnerabilities, allowing them to prioritize and address the most critical issues.

- **Base Metrics**: These metrics represent the intrinsic qualities of a vulnerability that remain constant over time and across various user environments. They include factors such as the attack vector, attack complexity, and the impact on confidentiality, integrity, and availability.
- **Temporal Metrics**: Reflecting the characteristics of a vulnerability that can change over time, temporal metrics consider factors such as the exploitability of the vulnerability and the availability of remediation.
- **Environmental Metrics**: This group represents the characteristics of a vulnerability that are specific to a user's environment. It allows organizations to customize the CVSS score based on factors like the sensitivity of the affected system and the importance of the vulnerable asset.

The Base metrics generate a numerical score ranging from 0 to 10, indicating the severity of the vulnerability. This score can be further adjusted by incorporating the values from Temporal and Environmental metrics. Additionally, a CVSS score is represented as a vector string, providing a compressed textual representation of the underlying values used to derive the score.

This standardized approach allows security professionals to comprehensively evaluate vulnerabilities, prioritize remediation efforts, and communicate risk consistently. The use of CVSS scores in the National Vulnerability Database (NVD) facilitates a common language for the cybersecurity community, contributing to more effective collaboration and response to emerging threats.

Chapter 3

State of the Art

This chapter delves into the academic landscape of cybersecurity, exploring fundamental concepts and evaluating automated approaches to risk assessment, vulnerability assessment, and threat modeling. In addition, it investigates the use of ontologies in enhancing these cybersecurity processes. This chapter provides a comprehensive overview of relevant research, examining the strengths, limitations, and potential applications of various automated tools and ontology-based solutions.

3.1 Vulnerability Assessment

Vulnerability assessment is a critical component of cybersecurity, helping organizations identify and prioritize security weaknesses in their IT infrastructure. Traditionally, vulnerability assessment has been a manual process, relying on security professionals to scan systems and applications for known vulnerabilities. However, the increasing complexity of IT systems and the growing volume of vulnerabilities have made manual vulnerability assessment increasingly impractical. As a result, there has been a growing interest in automating vulnerability assessment.

Automated vulnerability assessment offers several advantages to organizations, such as minimizing the workload of security experts and improving the accuracy of vulnerability detection. However, there are also certain obstacles associated with this process, such as the generation of false positives or the incorrect prioritization of vulnerabilities.

Mozzaquatro et al. [26] introduced an ontology-based cybersecurity framework for the Internet of Things (IoT), leveraging the IoTSec ontology to enhance system design and operational security. During the design phase, the framework helps identify appropriate technologies based on specific requirements, such as data encryption or communication protocols. It also generates code snippets that developers can easily integrate into their software and protocols. In the operational phase, the framework facilitates security analysis by incorporating alerts generated from monitoring tools. These alerts are mapped to vulnerabilities and threats within the IoTSec ontology, enabling the identification of effective remediation actions. The effectiveness of this mapping is crucial for generating actionable insights from the collected alerts.

Syed introduced, in a similar recent work [27], the Cybersecurity Vulnerability Ontology (CVO), a formal knowledge representation model for vulnerability management. The CVO aims to organize vulnerabilities, including those from unstructured sources such as social media, enabling analysts to effectively assess their severity. By incorporating vulnerability concepts from NIST, CERT/CC, and CVSS, the CVO provides a comprehensive representation of the vulnerability management domain. Additionally, the CVO maps vulnerability concepts from Twitter, a popular microblogging platform, to the CVO's ontology. Leveraging the CVO, Syed developed the Cyber Intelligence Alert (CIA) system, an ontology-based alert system that generates cybersecurity alerts based on predefined rules. The CVO serves as the common vocabulary for the vulnerability management domain, while the CIA system generates alerts about emerging vulnerabilities, exploits, patches, and other relevant security threats. The CIA system gathers vulnerability data from various sources, including CVE, NVD, Twitter, and vendor websites. The CVO integrates and reconciles this diverse data, ensuring consistency and accuracy. Furthermore, the CIA system enhances the CVO by introducing new concepts necessary for generating cyber alerts, leading to the creation of the Cyber Intelligence Ontology (CIO). The CIO functions as a knowledge base for alert generation, enabling the CIA system to deliver timely and actionable security insights.

The work of Khazai et al. [28] introduced VuWiki, an ontology-based semantic wiki, to systematically represent vulnerabilities. The paper explores a broader approach to vulnerabilities and assessment, extending beyond the realm of cybersecurity. VuWiki was designed to facilitate the review and representation of vulnerability assessment outcomes, serving as a reference system for process description. The ontology, implemented in a semantic wiki, enables classification and annotation of vulnerability assessments. Although not intended to provide a comprehensive model of vulnerabilities, VuWiki offers a reference system for security analysts and a structured, easily accessible knowledge base for information extraction. The

ontology and semantic wiki were created to establish a solution-oriented framework, employing semantic rules to facilitate comparisons between various vulnerability assessments and applicable concepts and methodologies. In essence, the ontology's design aims to address and represent four fundamental questions: (1) vulnerability to what?, (2) vulnerability to whom?, (3) what framework was used for assessment?, and (4) what methodological approach was used for assessment?

3.2 Threat Modeling

Threat modeling is a crucial cybersecurity practice that helps identify and analyze potential threats and vulnerabilities in software systems or architectures. Traditionally, threat modeling has been a manual process, requiring security experts to analyze system diagrams and documentation to identify potential weaknesses. However, as the complexity of software systems has grown, manual threat modeling has become increasingly time-consuming and challenging. This has led to a growing demand for automated threat modeling solutions.

One widely adopted framework for threat modeling is STRIDE, an acronym that encompasses six key security categories:

- Spoofing: impersonation or misrepresentation of the identity of an entity or communication;
- Tampering: unauthorized modification of data or code;
- Repudiation: denial of actions or responsibilities;
- Information Disclosure: unauthorized access to sensitive or confidential information;
- Denial of Service: disruption or prevention of legitimate access to resources or services;
- Elevation of Privilege: unauthorized gain of access to higher levels of privilege or control.

The STRIDE framework serves as a valuable tool for organizing and analyzing potential threats, enabling security analysts to systematically assess the security posture of systems and architectures.

Leveraging the STRIDE framework, Casola et al. [29] and Flå et al. [30] propose automated threat modeling approaches for edge computing systems and smart grids, respectively. Both studies highlight the effectiveness of automated threat modeling in identifying potential security vulnerabilities and enhancing the overall security posture of complex systems.

In a study aimed at automating threat modeling for edge computing scenarios, Casola et al. [29] proposed a system that extends the capabilities of the Microsoft Threat Modeling Tool. The authors developed a library of elements specifically tailored to describe the IoT/Edge computing realm, encompassing various components characterized by distinct aspects such as location within the infrastructure, type of data processed, and so on. This information serves as a crucial input for the subsequent threat modeling phase. The system modeling approach proposed by Casola et al. revolves around three primary asset categories: physical/virtual processing nodes, software components/modules, and communication channels. Additionally, the data is categorized into user-related data, environmental data, and service data. For threat identification, the authors constructed a comprehensive threat library based primarily on scientific articles. In particular, during the threat modeling phase, the authors prioritized minimization of false negatives by excluding threats that are not applicable to specific system elements, ensuring a focus on significant results.

Flå et al. [30] introduced a smart grid threat modeling template for the Microsoft Threat Modeling Tool (TMT), which includes stencils and potential threats categorized by STRIDE. To minimize false negatives, threats were generalized, leading to fewer configurable properties in child stencils. Process stencils represent functional behavior, focusing on communication interfaces rather than traditional running code. The template includes a database as the sole data store and a single trust boundary type. The paper highlights the applicability of threat modeling and TMT to smart grids, providing a starting point for further adaptation and expansion.

In contrast to the template-based approach proposed by Flå et al., Välja et al. (2021) employed an ontology framework that offers a more versatile and adaptable method for automated threat modeling.

Välja et al. [31] proposed an ontology framework for automated threat modeling, addressing domain knowledge gaps and granularity mismatches. The ontology standardizes information from various sources, improving semantic accuracy and granularity match for threat models. The authors identified four categories of data elements and defined key concepts for each, ensuring complete information collection. The evaluation showed that the ontology framework improved the

accuracy of threat modeling and reduced manual effort. The study highlighted the importance of ontology development and maintenance, suggesting the use of machine learning for automated knowledge acquisition. The work of Välja et al. demonstrates the potential of ontologies in enhancing automated threat modeling and cyber risk assessment.

3.3 Risk Assessment

Risk assessment is vital for cybersecurity, helping organizations identify and address potential threats to their data and systems. Manual risk assessment is becoming less effective due to complex IT environments and increasing cyber threats, and automated risk assessment tools are emerging to address these challenges. These tools streamline risk assessment and provide organizations with timely insights, allowing security teams to focus on strategic decision-making for effective cybersecurity protection.

Tantawy et al. [32] proposed a model-based risk assessment approach for cyber-physical systems (CPS), facilitating the analysis and design of cybersecurity solutions for a given CPS. Their approach integrates physical system modeling, data flow modeling, and attack trees to provide a unified framework for designing safe and secure CPS. The risk assessment process begins with the creation of a comprehensive physical system model that encompasses all system components and their interactions, including sensors, controllers, supporting networks, and protocols. Next, a data flow model is developed to capture the information exchange between system components for monitoring and controlling the physical process. These models serve as inputs for threat modeling using attack trees, which are conceptual diagrams that depict potential threats and attacks that could exploit those threats. This analysis helps identify potential hazardous states of the system and informs the design of countermeasures and mitigation strategies. The authors applied their proposed model-based approach to a CPS testbed that monitors and controls an exothermic Continuous Stirred Tank Reactor (CSTR) simulated in real-time. They successfully identified potential cyber threats and assessed the associated risks, demonstrating the effectiveness of their approach for CPS security risk assessment.

Casola et al. [33] proposed a methodology to support the threat modeling and risk assessment of IoT systems through an almost completely automated process. This methodology leverages a comprehensive modeling approach and a knowledge base of security information to streamline the security analysis process for IoT

systems. The methodology encompasses three main phases: system modeling, threat modeling, risk analysis, and security control identification. In the system modeling phase, a high-level architectural model of the IoT system is created, adhering to up-to-date standard specifications. Using a security knowledge base (threat catalog), the threat modeling phase automatically identifies applicable threats based on the system model. Finally, the risk analysis and security control identification phase employs the OWASP Risk Rating Methodology to assess the risk associated with each identified threat and selects appropriate countermeasures to mitigate potential security vulnerabilities. This methodology offers several benefits, including reduced analyst burden, automated threat identification, and risk-based countermeasure selection. It provides a valuable tool for IoT security professionals, allowing them to effectively identify, analyze, and mitigate security risks in a streamlined and efficient manner.

Chapter 4

Proposed Solution

In this chapter, we will discuss the proposed solution to develop a tool that can help security teams identify potential vulnerabilities and threats to a system through a more efficient and methodical approach. We will dive into the architecture of the tool, including the development of the template ontology and the modeling tool. Additionally, we will look at the use of Protégé in modeling the ontology and the key parts of the ontology that are relevant to the process.

4.1 Architecture

As mentioned in Chapter 1, the main objective of this thesis is to develop a tool that will aid the security team in identifying and assessing potential vulnerabilities and threats against the system being analyzed, in a more efficient, faster, and methodical manner.

As depicted in Figure 4.1, the initial stage of the process involves the development of the *Template Ontology*, which serves as a framework for the subsequent threat modeling stage. This ontology contains the vocabulary and inference rules that will be utilized during the threat modeling process. It is a crucial component of the process as it provides a standardized approach to defining and describing the relevant concepts and relationships pertaining to the specific system being modeled. The *Template Ontology* was developed by De Rosa et al. [34], and is taken for granted in this thesis.

Assuming that the *Template Ontology* is provided, the security team or the infrastructure architect can import it into the *Modeling Tool* and populate it by adding all the necessary information required to describe the ICT system that is the subject of the security assessment. This information could include, but is not limited to, the specific assets that make up the system, their respective types, such as *hardware*, *network* or *operating system*, their data flows, or security mechanisms that could have been put in place to protect them.

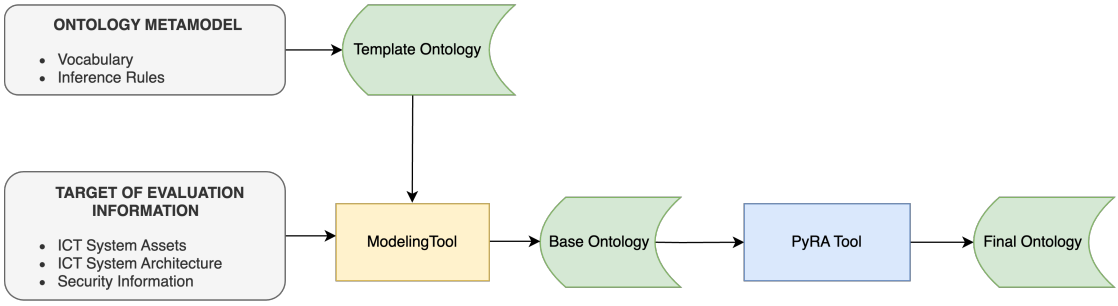


Figure 4.1: Solution architecture overview

As a result, we obtain the *Base Ontology*. This is the input for the tool developed in this thesis work, PyRA.

The tool analyzes the Base Ontology and gathers proper security data from publicly available knowledge bases to identify known vulnerabilities and adversary Tactics and Techniques. It infers possible threats to the system's assets and computes a risk score for each threat. This helps the security team prioritize the most critical threats and develop effective strategies to mitigate them, enhance the organization's overall security posture, and significantly reduce the workload of the security team.

In order to provide a better understanding of the process explained in this paper, we will present two key parts of the ontology that are relevant to this work: the *ICT sub-ontology* and the *Vulnerability sub-ontology*. These are the primary areas with which the developed tool interacts to perform its tasks. We will discuss these two sub-ontologies in order, starting with the *ICT sub-ontology* in Section 4.3 and then moving on to the *Vulnerability sub-ontology* in Section 4.4.

4.2 Modeling Tool

As stated in the previous section, a Modeling Tool is needed to model both the *Template Ontology*, which is done by the *Metamodel Maintainer*, and the *ICT Ontology*, modeled by the Infrastructure Architect. Although it would be beneficial to have a tool that provides a unified approach to complete each step of the process, integrating this functionality would require substantial development efforts that are beyond the scope of the objectives of this thesis.

For this reason, the *Template Ontology* and the *ICT Ontology* of the use case, on which PyRA has been validated, have been generated relying on *Protégé*.

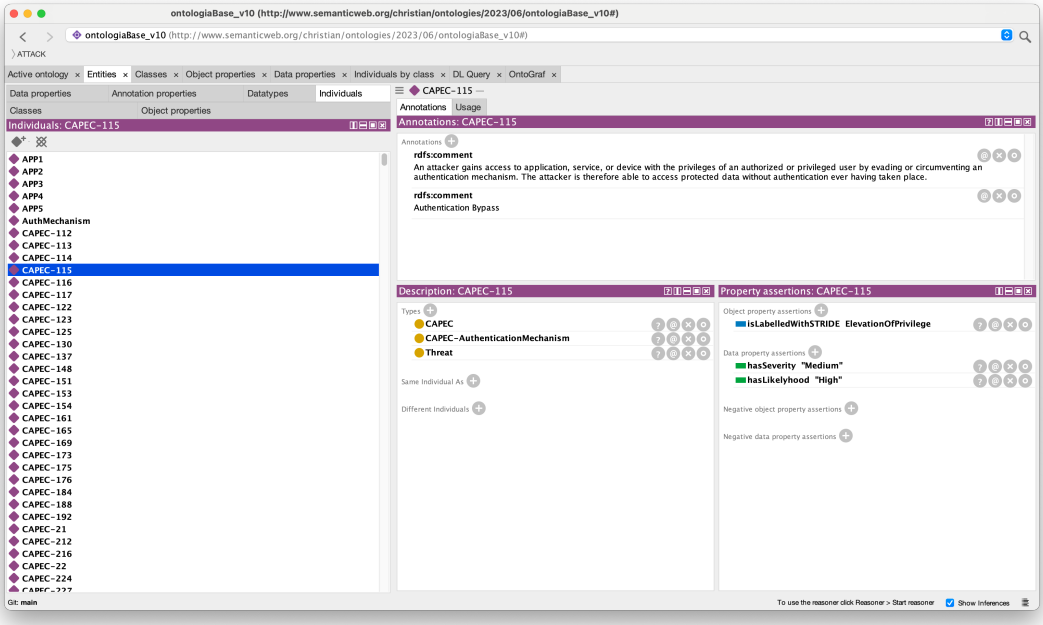


Figure 4.2: Protégé Desktop application

Protégé is a free and open-source feature-rich ontology editing software that was developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine [35]. It is available in both a web-based and a desktop application, the latter visible in Figure 4.2, and fully supports the latest OWL 2 Web Ontology Language.

4.3 ICT sub-ontology

In this section, we will introduce the ICT sub-ontology and focus on modeling the ICT infrastructure according to the proposed solution. The primary emphasis is on assets that are valuable elements and, at the same time, potential sources of vulnerability within the organization. This model, taken as-is from the works of [34] and [11], encompasses all devices and software in the ICT setup that could be targeted for attacks.

The structure of this sub-ontology, depicted in Figure 4.3, has been designed with great emphasis on the necessity to properly organize and represent assets. In the remainder of this section, we break down each class, providing brief descriptions and the reasons behind their insertion in the ontology. This paves the way for a clearer understanding of the mechanisms that allow the automation of risk assessment.

For the sake of readability, we will often use the term *"relationship"* to refer to what is technically called an *"object property"* in ontologies. *Object properties* assert a relationship between two individuals, as explained in Chapter 2.

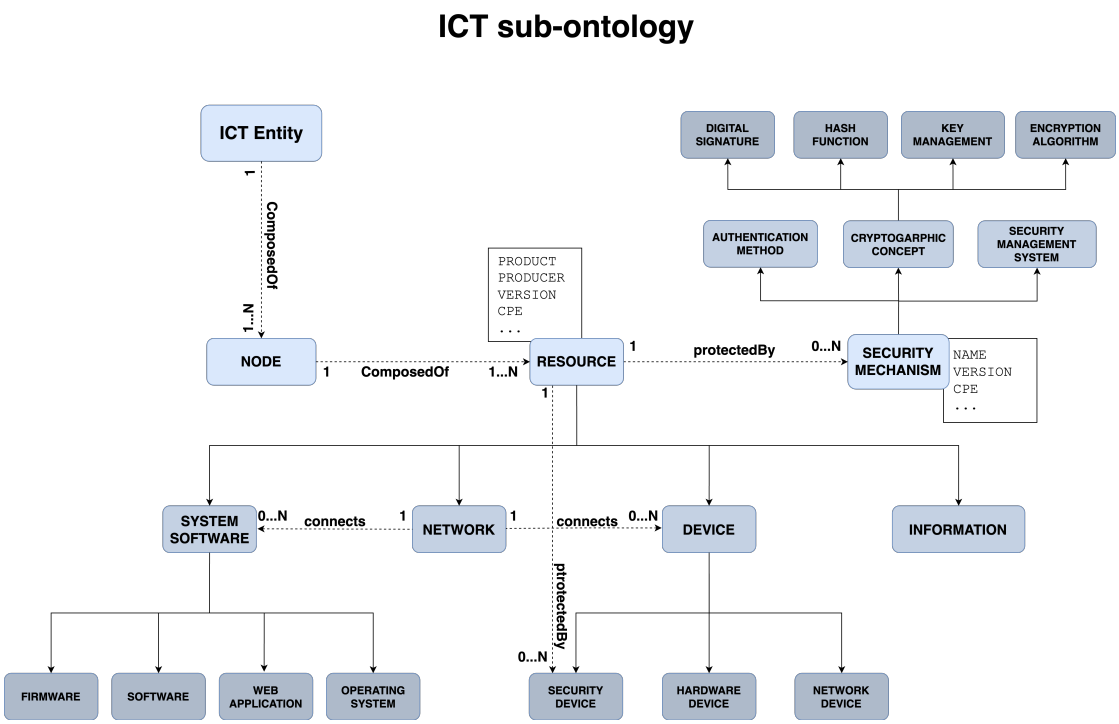


Figure 4.3: Diagram of the ICT sub-ontology [11].

The starting point of our analysis is the *ICTEntity* class, symbolizing the central focus of our investigation, the ICT infrastructure. This class establishes a pivotal relationship with the *Node* class through the *composedOf* relationship. The prescribed cardinality for this association dictates that upon introducing an ITCEntity individual, at least one corresponding Node individual must accompany it. Always looking at the cardinality of that relationship, it is clear that to enrich the modeling approach, an ITCEntity individual is intentionally designed to encompass more than one Node individual, providing a more nuanced representation of the intricate infrastructure.

The Node class assumes a crucial role in modeling individual nodes within the infrastructure. In alignment with the TOGAF ArchiMate [36] definition, a node embodies a computational or physical resource engaged in hosting, manipulating, or interacting with other resources. Essentially, the Node class functions as a consolidator of resources, representing a fusion of technology components, ranging from hardware to software, that collectively deliver specific services within the infrastructure. An individual of the Node class could thus be a physical entity, such as a data center, as well as a digital service.

The Node class plays a crucial role in organizing infrastructure resources across diverse segments, although it does not directly contribute to vulnerability management, threat modeling, or risk assessment. It provides a well-structured framework for the resources and establishes a link with the Resource class through the composedOf relationship. This relationship ensures that every Node individual is consistently associated with at least one Resource individual.

Furthermore, a Node is connected to itself through the composedOf relationship, allowing it to be composed of other nodes within the infrastructure. This relationship has no limit, so a node can be composed of as many nodes as necessary to reflect the dynamic nature of real ICT infrastructures. This feature enables the infrastructure representation to be more scalable and adaptable to changes.

The Node class, therefore, serves as a critical component in the modeling of real ICT infrastructures.

According to the definition in TOGAF ArchiMate [36], which states that a resource denotes an asset owned or controlled by an individual or organization, the Resource class serves as a modeling framework for the assets of the infrastructure under analysis. The class is internally characterized by useful data properties, such as *Product*, *Producer*, *Version*, and *CPE*, that allow for a better description of the actual resource individual, ideally allowing the tool to uniquely identify them.

The PyRA tool, as will be better explained in Chapter 5, heavily relies on those properties to properly identify the resource, so that it can retrieve the correct data from the public knowledge bases provided by MITRE and NIST.

In order to enhance the risk assessment process, the Resource class is associated with the Security Device and Security Mechanism classes through the protectedBy relationship. The cardinality of this relationship conveys the idea that a resource may or may not be safeguarded by one or multiple security mechanisms or devices.

Furthermore, the Resource class has four main subclasses: Device, Information, Network, and System Software. These subclasses specialize in the various types of resources that the infrastructure may contain, thus providing more formal modeling, avoiding possible misunderstandings in the modeling phase, and allowing specific threats to be better mapped to assets in order to prevent the inclusion of threats that are not meaningful within the analyzed context.

The Device class allows the representation of physical assets, and is subclassed to further refine their classification. The class is subdivided into three distinct subclasses to categorize specific types of devices:

- Network Device: representing devices with networking capabilities (e.g. router, Access
- Security Device: representing hardware devices designed to deliver security-related functionalities (e.g. HSM, firewall, IPS, and so on)
- Hardware Device: representing hardware devices that do not belong to the other two classes (e.g. laptops, server, and so on)

The Information class in the ontology functions as a comprehensive label for diverse data types, encompassing both general and sensitive information requiring the assurance of Confidentiality, Integrity, and Availability. This classification extends to data in various states, whether stored or in transit, providing a lucid representation of information within the ontology framework.

The Network class is a crucial tool for modeling and analyzing any network within the ICT infrastructure. It provides a comprehensive understanding of how different resources are interconnected and how data flows within the system. In fact, as shown in Figure 4.3, the class connects other subclasses of the Resource class through the *connect* relationship.

The System Software class, according to the definition given in TOGAF ArchiMate [36], is used to model any kind of software that is present within the system. To give a higher granularity in the modeling of the system, the class has four subclasses:

- Firmware: software offering low-level control for the specific hardware of a device
- Web Application: web applications
- Operating System: operating systems
- Software: any software not belonging to the other three subclasses

The SecurityMechanism class allows modeling security solutions applied to one or more assets within the infrastructure under analysis. Again, to allow modeling to be as precise as possible, the class is further divided into four subclasses. Authentication Method, representing any security mechanism method providing support for authentication, Cryptographic Concept, representing cryptographic protocols and algorithms, Security Management System, representing solutions such as SOAR Systems.

In conclusion, the ICT sub-ontology provides a comprehensive and well-structured framework for modeling ICT infrastructure assets. The ontology encompasses all relevant elements of the infrastructure, including devices, software, information, and networks. The use of subclasses allows for a more granular representation of assets, enabling the identification of specific vulnerabilities and threats. The relationships between classes enable the automation of risk assessment tasks, making it easier to identify and mitigate security risks.

4.4 Vulnerability sub-ontology

This section provides an overview of the *Vulnerability sub-ontology*. It includes a description of each class, its *data properties* and *object properties*, and an explanation of why they were included and their importance.

We present here the part of the ontology that is designed to model the vulnerabilities that affect the security mechanisms and resources that make up the ICT infrastructure. This part of the ontology comes from the work of *Maunero* [11], but has been slightly revised.

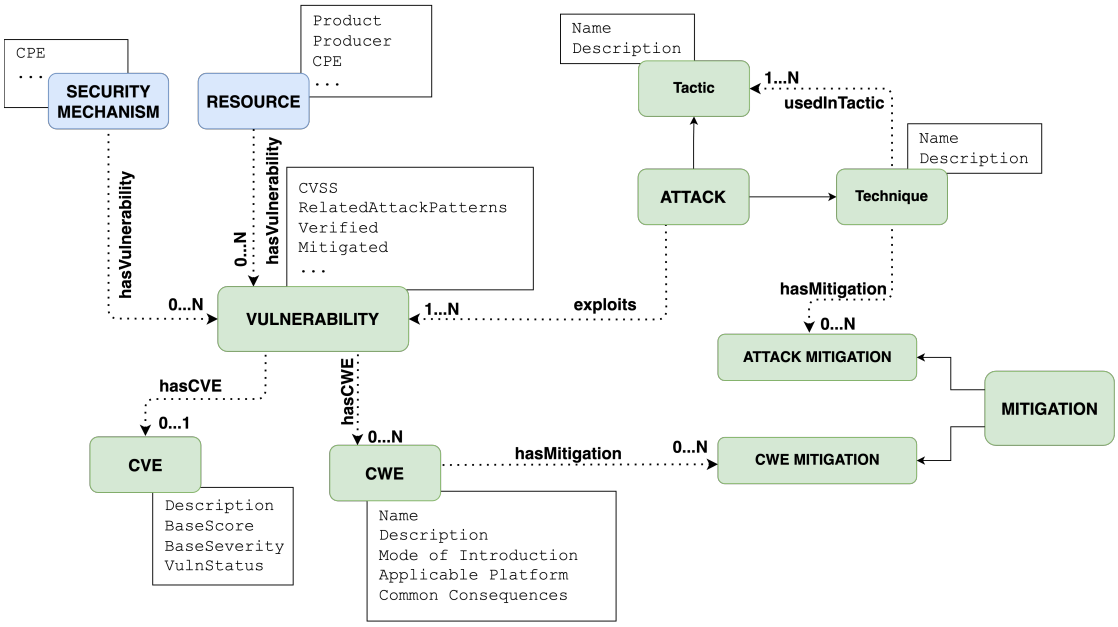


Figure 4.4: Diagram of the Vulnerability sub-ontology [11].

Figure 4.4 shows the complete diagram of the sub-ontology, with the highlighted green classes being the ones that PyRA will populate as discussed in Chapter 5.

The *Vulnerability* class is used to model the possible vulnerabilities that affect one or more resources of the system. It is internally characterized by the following data properties, which give useful information to the security team:

- *CVSS*: representing the vulnerability severity score[1], ranging from 0 to 10
- *Related Attack Patterns*: a list of CAPECs that, according to the mapping done in common knowledge bases, could exploit the vulnerability to achieve some malicious objective
- *Verified*: used to mark the vulnerability as verified by the security team
- *Mitigated*: used by the security team to mark the vulnerability as mitigated

Each individual of the *Vulnerability* class is tied to the asset it affects, being it an individual of the *SecurityMechanism* class or of the *Resource* class, and mapped

[1]Note that CVSS is not a measure of risk, but rather a qualitative measure of severity [37]

to the related CVE, CWE and ATT&CK entities. This approach facilitates a comprehensive representation of data, which subsequently enables security teams to conduct a thorough analysis.

The *hasVulnerability* object property relates the *Vulerability* class to the classes *Resource* and *Security Mechanism*. The cardinality of the relationship indicates that each instance of *Resource* or *Security Mechanism* may have several vulnerabilities or none at all. In order to understand the work presented in this thesis, it is sufficient to know the following about these two classes.

The *Resource* class is used to represent infrastructure components and is internally characterized by some *data properties*, such as *producer*, *version*, and *CPE*, which enable a more precise description of the particular resource being depicted.

The *Security Mechanism* class is employed to represent security measures that are used to protect a particular element or resource of the infrastructure. It is further characterized by additional data properties, such as *name*, *version*, and *CPE*.

Furthermore, the *Vulnerability* class is associated with the CVE, CWE, and ATT&CK classes. First, we examine the CVE class.

Looking at the cardinality of the *hasCVE* relationship, we can observe that a vulnerability can have at most one CVE. In the tool flow, which is discussed in detail in the next chapter, the process starts from a CPE to search for CVEs and then, from these, all the other related security data are retrieved. For this reason, the only scenario in which a vulnerability does not have a CVE is when it is a custom vulnerability created by the security team. This is done, for example, to keep track of vulnerabilities identified during a VAPT assessment that are missing from public knowledge bases. Additionally, the class has two *data properties*:

- *description*: a description of the vulnerability, often containing some additional useful information such as the versions of the product it affects (e.g. "When reading a file, an uninitialized value could have been used as read limit. This vulnerability affects Firefox < 113, Firefox ESR < 102.11, and Thunderbird < 102.11." - CVE-2023-32213)
- *BaseScore*: representing the vulnerability severity score
- *BaseSeverity*: a qualitative severity rating
- *VulnStatus*: the status assigned to vulnerabilities within the NVD

Now, let us focus on the *CWE* class. According to MITRE, *"a weakness is a condition in a software, firmware, hardware, or service component that, under certain circumstances, could contribute to the introduction of vulnerabilities"* [15]. This should make it pretty clear why the *CWE* class has been inserted into the ontology and why more than one CWE instance can be assigned to a single vulnerability through the *hasCWE* relationship. The class is further enriched with the following *data properties*:

- *Name*: a really explanatory name of the weakness, that is more like a short description of it (e.g. "Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')" - CWE-470)
- *Description*: a full description of the weakness
- *CommonConsequence*: describing a common possible consequence that could come from the exploitation of the given weakness; the common consequence is detailed with *scope*, which identifies the security area that is violated (e.g. *confidentiality*), and *impact*, showing the negative technical impact that arises if an adversary succeeds in exploiting the weakness; there can be more than one specified for each weakness

The class further relates, as shown in Figure 4.4, to the *CWE MITIGATION* class (subclass of *MITIGATION*), through the *hasMitigation* relationship. As can be seen from the cardinality of the relationship, a weakness can relate to multiple mitigations. The *CWE MITIGATION* class provides information about how weaknesses can be mitigated, thus helping the security team in the subsequent steps of vulnerability management.

Finally, the *ATTACK* class is related to the *Vulnerability* class with the *exploits* relationship. As indicated by the cardinality of the relationship, a vulnerability can be exploited by more than one individual of the *ATTACK* class. The class has two subclasses, *Tactic* and *Technique*, connected by the *usedInTactic* relationship. A *technique* can be relevant in the context of multiple adversarial *tactics* that cyber adversaries might employ. These classes aim to provide a comprehensive and detailed view of adversarial behavior, to better understand the diverse ways in which adversaries can exploit vulnerabilities to achieve different tactical objectives. Both subclasses have only two data properties, *Name* and *Description*, but the *Technique* subclass is also related to the *ATTACK MITIGATION* subclass. Individuals in that class represent recommended strategies and countermeasures that organizations can use to defend against or reduce the risk associated with specific adversary

techniques. Mitigations are essential to improve an organization's cybersecurity posture and reduce its exposure to potential threats.

In conclusion, the purpose of the *Vulnerability sub-ontology* is to provide a comprehensive overview of potential threats and malicious behaviors, as well as the corresponding countermeasures available to help organizations improve their cybersecurity posture. As a result, informed decisions can be made about the vulnerabilities and threats identified. This allows for a more strategic planning process, as well as the implementation of targeted measures to manage risks.

4.5 The PyRA Tool

This section provides an overview of the developed tool, PyRA, which will be analyzed in more detail in Chapter 5.

PyRA (Python Risk Assessor) is a powerful tool that analyzes the ontology describing the ICT infrastructure and identifies potential threats and vulnerabilities, thus automating a lot of the risk assessment process and helping the security team prioritize the most critical threats to develop effective strategies to mitigate them.

The tool starts by performing an analysis of the content of the ICT Ontology, which includes information about what constitutes the ICT infrastructure under analysis, specifically looking for each resource for a CPE or, if missing, the name, version, and vendor of the product. Based on this information, the tool gathers security-related data from publicly available knowledge bases that are specifically relevant to the system under analysis. The tool is able to identify known vulnerabilities and weaknesses that could potentially affect the system's resources, as well as adversary tactics and techniques that could exploit these vulnerabilities. Where available, the tool retrieves, for each individual of the classes *CWE* and *Technique*, the corresponding suggested *mitigations*, to better support the security team in the following steps of the vulnerability management process.

To further enhance its capabilities, the tool integrates an ontology reasoner that makes use of the rules defined in the first step of the process within the ontology template. This reasoner, analyzing the rules defined by the metamodel along with the information describing the ICT infrastructure, is able to infer possible threats to the various assets of the system. This step is particularly important because it allows the tool to identify potential threats that are not based on mapping single

components to the security data available from public knowledge bases, but rather on the well-defined rules that also take into account the data flows and how different components of the system interact with each other.

Finally, the developed tool computes a risk score for each threat identified by the ontology reasoner. This score helps the security team prioritize the most critical threats and develop effective strategies to mitigate them. By sorting threats according to their risk score, the team can focus on the ones that pose the highest risk and tackle them first. This approach ensures that security management is more targeted, efficient, and effective in addressing potential vulnerabilities and threats while protecting the organization's assets. As a result, the overall security posture of the organization can be greatly improved, and the risk of cyberattacks and data breaches reduced, if the security team really takes advantage of all the resulting security data.

Chapter 5

Implementation

In this chapter, we delve into the implementation process of the PyRA tool. We will discuss the key decisions made during the technology stack selection phase and how they impacted the development process. Additionally, we will explore the crucial step of retrieving data from public knowledge bases and accurately mapping them to populate the *Vulnerability* ontology. Overall, this chapter focuses on the technical details of how the tool operates, while also providing insights into the reasoning behind the implementation decisions.

5.1 Technology Stack selection

The *Technology Stack selection* is a crucial decision in the software development process, and it involves choosing the appropriate programming languages, frameworks, libraries, databases, and other technologies that best suit the requirements and goals of the project. The technology stack can significantly impact the development process, the performance of the application, and its scalability, so it is an important decision that developers and project stakeholders need to make.

This phase played a key role in clarifying the goals of the development. In fact, the objective was initially to develop a complete all-round tool, thus also integrating the ICT infrastructure modeling functions. After an initial analysis of the available cross-platform frameworks and the related available libraries that could facilitate at least the development of the interactive modeling part, we realized that such

development would require a lot of time, resources, and expertise for the UI/UX part, which is beyond the scope of this thesis. Therefore, it was decided to focus on the core part, which deals with the collection and manipulation of vulnerabilities, weaknesses, and threats data, while leaving the modeling functionalities for future developments.

In the process of selecting the technology stack, careful consideration was given to various programming languages. However, the ultimate choice was narrowed down to Java, Rust, and Python, due to the critical constraint of smoothly working with ontologies. In fact, for each of these programming languages, there is at least one robust and well-established ontology library, such as OWLAPI for Java, Horned-OWL for Rust, and Owlready2 for Python.

During the early stages of development, we conducted tests using both Rust and Python to evaluate their performance. However, after testing them on the simple use case we were working on, we found no significant differences that would justify sacrificing development efficiency by choosing Rust over Python. We also considered that our goal was to create a working prototype to demonstrate our solution rather than a production-ready software.

At the end of the technology stack selection phase, Python emerged as the final and optimal choice for the development of the tool. Its selection as the preferred programming language was driven by its versatility, ease of development, and availability of the Owlready2 ontology library, which aligned perfectly with the project's goals and constraints. This choice ensured a streamlined and efficient development process, emphasizing the project's core objectives while leaving the door open for potential future expansions into modeling functionalities.

5.2 Retrieving data

In order for the tool to be developed effectively, it was essential to devise a method for retrieving all relevant data from the public knowledge bases of NVD and MITRE. This was a crucial step, as it is necessary to first analyze all the information within these public databases and accurately map it as needed, to be able to populate the *Vulnerability ontology*.

The NIST (National Institute of Standards and Technology) official website provides the CVE data in JSON or XML format. Similarly, the official websites for CWE and

CAPEC offer XML data feeds. In the case of MITRE ATT&CK, the JSON data feeds are available for download. Furthermore, the NIST National Vulnerability Database (NVD) provides a RESTful API, and MITRE ATT&CK offers a public STIX/TAXII service for programmatic access to their data, which allows anyone to query and retrieve information through official APIs.

So, the goal at this step of the development was to download all that data, parse them to create relationships that would allow the tool to easily navigate them when needed, and finally store them along with their relations in a unique database. Figure 5.1 shows how all these public knowledge bases relate to each other, which is exactly what the tool needs to be able to populate the ontology.

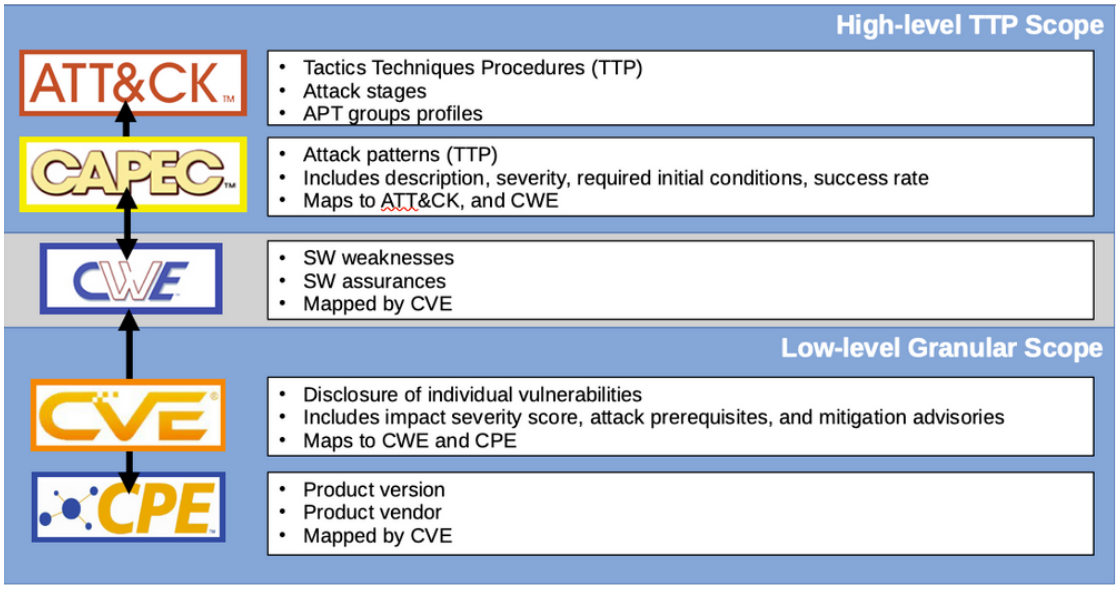


Figure 5.1: Schematic showing how CPE, CVE, CWE, CAPEC, and ATT&CK data can be mapped all together [38].

Of course, another critical constraint was the ability to keep that centralized database up-to-date, since all the aforementioned public knowledge bases are constantly updated as soon as new vulnerabilities, weaknesses, threats, attack patterns, and attack tactics and techniques are discovered.

Various possibilities have been taken into account, and at first a custom solution was explored, which worked by gathering all the data feeds, parsing them, and storing into a SQLite database the data and the relationships connecting instances of the various knowledge bases.

However, by doing more in-depth research on the use of public knowledge bases to automate cybersecurity operations, we stumbled upon a paper, "Automated Attack Tree Generation and Evaluation: Systemization of Knowledge"[39], in which the author shows the work of Dr. Erik Hemberg, *BRON*, as a good framework for interaction with data from MITRE and NIST.

5.2.1 BRON

As described by Erik Hemberg et al. [40], BRON is a relational graph that serves to structure the information from various sources into specific node types and their corresponding edges. The graph is designed to denote both unidirectional and bidirectional links, as shown in Figure 5.1, even when they are unidirectional in the source material.

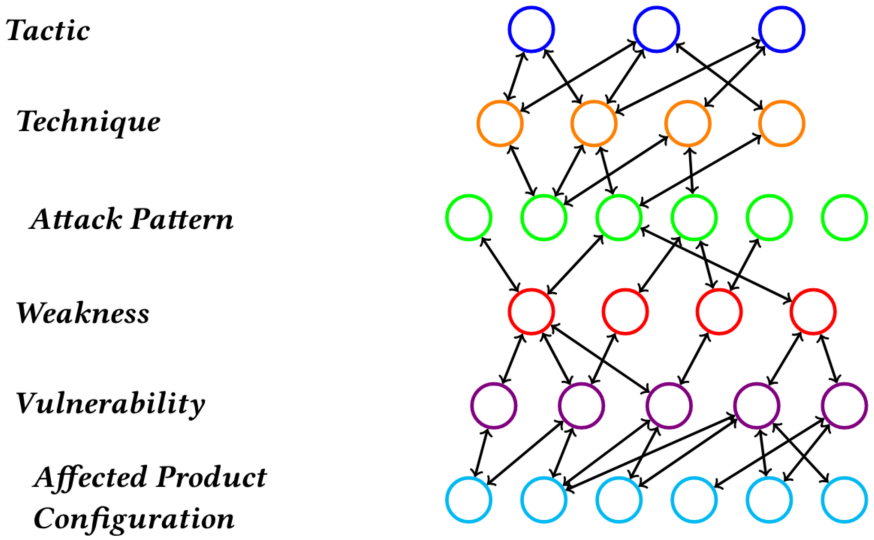


Figure 5.2: Schematic of BRON's graph, in which source entries are nodes and relational links are edges [40].

So we decided to perform some tests with it, and it turned out to be an excellent choice due to its various advantages. BRON is built on ArangoDB, an efficient database technology for storing and querying graph-based relationships, making it ideal for mapping relationships between CPE, CVE, CWE, CAPEC, and ATT&CK. Additionally, BRON is plug-and-play, can be run as an independent docker instance, is efficient, and its GitHub project is still actively maintained. This makes it a

great option for interacting with MITRE and NIST's data.

Unfortunately, nothing is perfect, and BRON is no exception. By testing it more thoroughly, it was realized that it has some problems in mapping CVEs to CPEs. In fact, when the results of the CVEs provided by BRON for the CPEs being analyzed are compared with those that can be retrieved by going to the NVD's website and searching by hand, it is often observed that the CVEs provided by BRON are a subset of those actually available. Furthermore, as shown in Figure 5.3 with the yellow question mark, in some cases BRON returns some CVEs that, according to the results that can be obtained by performing manual research on NVD's website, should not be assigned to the given CPE.

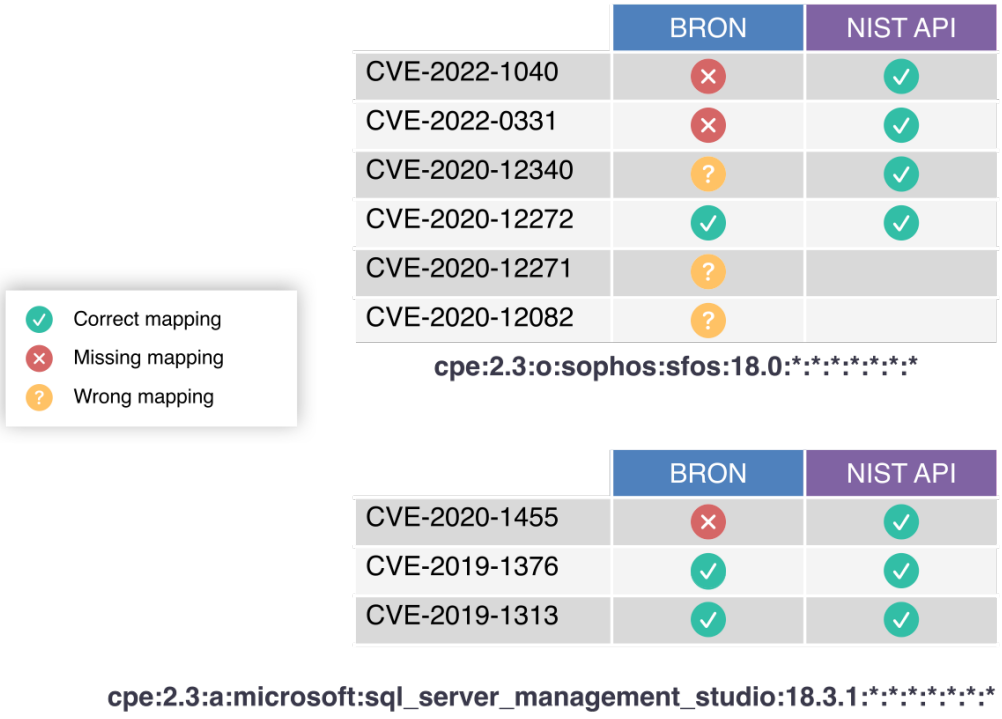


Figure 5.3: Comparing BRON and NIST API results of CVEs per CPE queries.

In Figure 5.3 we compare, for two CPEs of choice, BRON's results with those provided by NIST's NVD API because it is the solution that we chose to adopt in the final version of our tool. Although a separate column for manually searched results could have been presented, it was deemed unnecessary as the results were found, as expected, to be identical to those obtained from the NIST API.

5.2.2 NVD CVE API

The NVD CVE API is a service provided by NIST (National Institute of Standards and Technology) to access information about vulnerabilities in software systems. The API allows users to programmatically access the CVE information, making it easier to integrate vulnerability data into security tools, applications, and workflows. With the API, developers can retrieve information about specific vulnerabilities, query for vulnerabilities that meet certain criteria, and stay informed about the latest security vulnerabilities.

Therefore, to limit as much as possible the introduction of false positives in the resulting ontology, the final implementation consists of a hybrid solution. For each CPE available in the *ICT ontology*, the tool retrieves all the related CVEs using NIST's API, which, by the way, is NIST's own suggested way of working with its knowledge base as an alternative to data feeds. At this point, starting from the CVEs, all the other data are queried from BRON, which reliably maps all the remaining data sources, as far as we could observe in our tests.

5.3 Vulnerability Assessment

The final phase of development involves the intricate task of populating the ontology. As detailed earlier in this chapter, our chosen approach utilizes the Owlready2 library. Renowned for its robust capabilities, Owlready2 is a Python package that empowers users to seamlessly manipulate, load, modify, and save OWL 2.0 ontologies [41]. The library also boasts built-in support for reasoning engines, HermiT and Pellet, enhancing its utility in ontology development.

Once the requisite data has been meticulously extracted from public knowledge bases, the ontology population process unfolds effortlessly with Owlready2. The Python tool orchestrates the creation of missing classes, meticulously designing each entity, including object properties and data properties, in accordance with the architectural framework delineated in the preceding chapter. This meticulous approach ensures the semantic accuracy and completeness of the ontology structure.

With the groundwork laid, the tool proceeds to instantiate each entity, generating instances for CVEs, CWEs, Tactics, and Techniques, accompanied by their respective data properties. The richness and specificity of these data properties contribute to the depth and precision of the ontology's knowledge representation.

To show how powerful and straightforward it is to populate the ontology through Owlready2, here is a piece of code that inserts into the ontology an instance of the CWE class:

Listing 5.1: Function for adding a CWE to Ontology

```
def add_cwe_to_ontology(cwe, cwe_id):
    CweClass = onto.CWE
    new_cwe = CweClass(cwe_id)
    if "name" in cwe and cwe["name"] is not None:
        new_cwe.Name.append(cwe["name"])

    if not "metadata" in cwe or cwe["metadata"] is None:
        return

    metadata = cwe["metadata"]
    if "description" in metadata:
        description = description_from_metadata(metadata)
        new_cwe.comment.append(description)

    if "common_consequences" in metadata:
        common_consequences = metadata["common_consequences"]
        for cc in common_consequences:
            new_cwe.CommonConsequences.append(str(cc).strip() + "\n")
```

Furthermore, the tool establishes intricate connections between entities through carefully defined object properties. This interlinking of entities enhances the ontology's ability to capture nuanced relationships and dependencies within the cybersecurity domain. The resultant ontology, enriched with extensive data and intricate relationships, serves as a robust knowledge base, ready for advanced querying, reasoning, and analysis. The meticulous process facilitated by Owlready2 ensures that the ontology not only adheres to semantic standards, but also captures the intricacies of the cybersecurity landscape in a nuanced and accurate manner.

5.4 Threat Modeling

As stated in Chapter 2, threat modeling is often considered a substep of risk assessment, as it provides the input for the risk assessment process.

Although this part of the process comes almost for free from the works of De Rosa et al. [34] and Maunero [11], here we provide a brief overview of how it works.

As already seen in Chapter 4, the tool takes as input what we call the Base Ontology, which in turn is based on the Template Ontology. The latter contains a set of CAPECs, also classified according to the STRIDE threat modeling methodology, and a set of well-defined rules SWRL establishing how the CAPEC threats will be assigned to the assets of the system under evaluation. For more details on that, the reader is referred to the aforementioned works.

So, once the ontology describing the target ICT infrastructure is received as input, the tool only needs to use the ontology reasoner it integrates in order to produce a threat model.

5.5 Risk Assessment

As anticipated, the tool performs risk assessment by leveraging the ontology reasoner and carefully defined rules.

Thus, having arrived at this point in the process, the ontology contains a complete description of the resources that make up the system, as well as a list of CAPECs and rules that establish, according to precise criteria, how they should be mapped to the resources.

The tool exploits the capabilities of the reasoner, thus being able to provide a complete list of possible threats (CAPECs) associated with each resource in the described ICT infrastructure. Each CAPEC has within its data properties Severity and Likelihood, provided in the MITRE knowledge base, and the Risk score, calculated by the tool as the product of the two available values:

$$Risk = Likelihood \times Severity$$

It is crucial to provide clarifications on the evaluation of *Typical Severity* and *Likelihood of Attack* properties within the MITRE knowledge base. The qualitative values assigned to these properties span from *Very Low* to *Very High*. It is also worth noting that within the knowledge base, CAPEC entries lack explicit *Likelihood* evaluations marked as *Very Low* or *Very High*. As a result, while it appears that *Likelihood* ranges from *Low* to *High*, our approach assumes the potential for the *Likelihood* property to encompass the entire spectrum. To calculate a risk score, these qualitative evaluations must be converted to numerical values. The conversion

scales used by the tool are provided in Table 5.2 and Table 5.1.

Table 5.1: CAPEC Typical Severity Conversion Scale

Very Low	Low	Medium	High	Very High
1	2	3	4	5

Table 5.2: CAPEC Likelihood of Attack Conversion Scale

Very Low	Low	Medium	High	Very High
(1)	2	3	4	(5)

The risks calculated in the final report can be represented by keeping the resulting numerical scores, to obtain a higher level of detail, or, if preferred, by converting them again to a qualitative level according to the matrix shown in Figure 5.4.

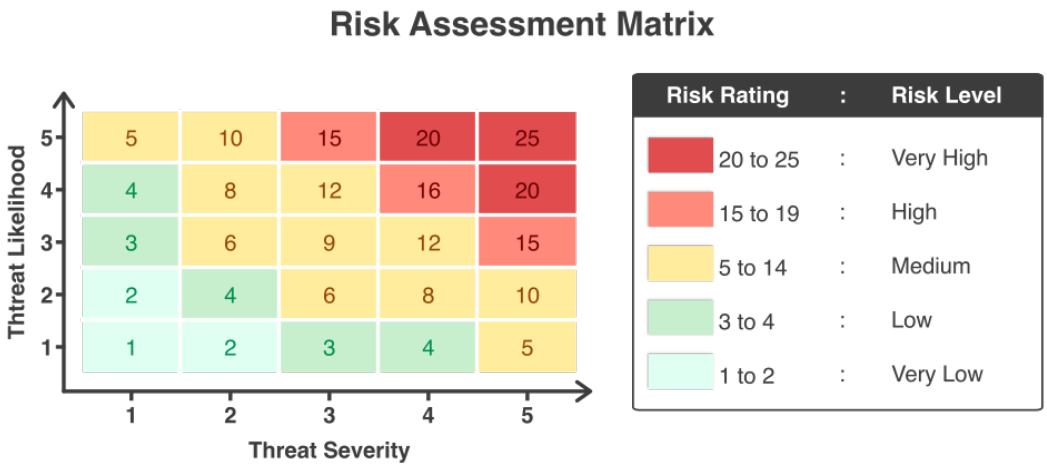
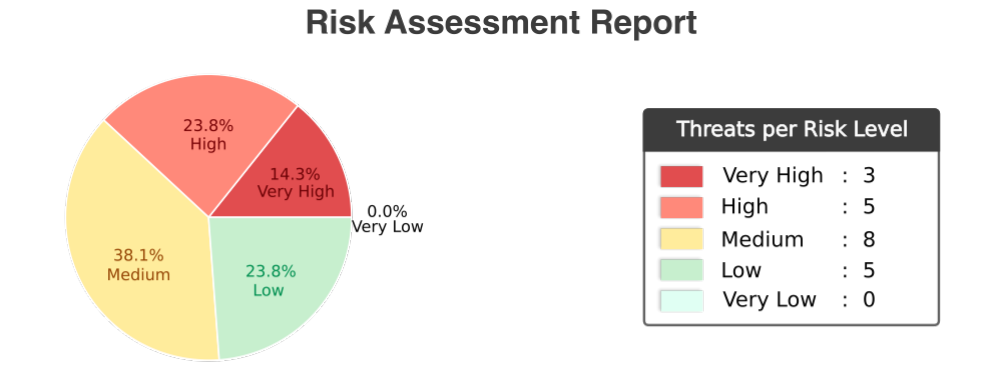


Figure 5.4: Risk matrix used to map Risk scores to Risk qualitative levels.

Furthermore, some CAPEC entries lack data for either the Likelihood of Attack or the Typical Severity, or both. In such instances, our tool automatically attributes a 'High' value to the missing property. This choice is deliberate and comes with the aim of addressing data gaps and ensuring a conservative estimation of the risk associated with each CAPEC. To enhance transparency, the instances where

default values are assumed are clearly indicated with an asterisk (*) in the report, as can be seen in Figure 5.5. Who will be in charge of analyzing the report is encouraged to review these marked values and consider the associated assumptions in their interpretation of the risk assessment results.



Resource	Resource Type	CAPEC	Likelihood	Severity	Risk Score	Risk Level
OS3	OperatingSystem	CAPEC-173	High	Very High	20	Very High
OS2	OperatingSystem	CAPEC-123	High	Very High	20	Very High
Network1	Network	CAPEC-94	High	Very High	20	Very High
PC2	HardwareDevice	CAPEC-607	High*	High*	16	High
APP5	Software	CAPEC-21	High	High	16	High
Network2	Network	CAPEC-607	High*	High*	16	High
Router1	NetworkDevice	CAPEC-458	High*	High*	16	High
APP1	Software	CAPEC-175	Medium	Very High	15	High
OS3	OperatingSystem	CAPEC-153	High*	Medium	12	Medium
Firewall1	SecurityDevice	CAPEC-440	Low	High	8	Medium
OS1	OperatingSystem	CAPEC-184	High*	Low	8	Medium
FIRM1	Firmware	CAPEC-184	High*	Low	8	Medium
Storage	HardwareDevice	CAPEC-624	Low	High	8	Medium
PC2	HardwareDevice	CAPEC-440	Low	High	8	Medium
Network1	Network	CAPEC-117	Low	Medium	6	Medium
Network2	Network	CAPEC-117	Low	Medium	6	Medium
Network2	Network	CAPEC-192	Low	Low	4	Low
Network2	Network	CAPEC-224	High	Very Low	4	Low
APP1	Software	CAPEC-188	Low	Low	4	Low
Network1	Network	CAPEC-169	High	Very Low	4	Low
PC2	HardwareDevice	CAPEC-188	Low	Low	4	Low

Figure 5.5: Example of a Risk Assessment report.

Figure 5.5 provides an example of a pie chart and a report table, showcasing the tool's ability to present the results in an easily understandable manner. At the end of the Risk Assessment process, the tool is capable of generating a comprehensive report in multiple formats, including *CSV*, *xlsx* (Excel), and *HTML*. In addition, it can create a clear visual representation of the distribution of risk levels through pie charts, as well as through bar charts, as shown in Figure 5.6.

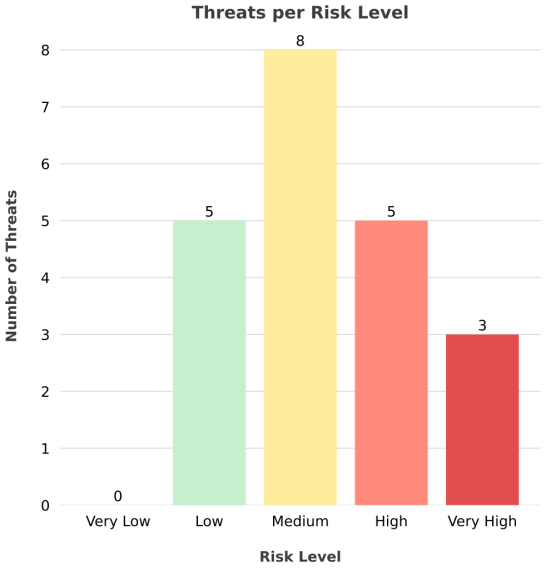


Figure 5.6: Example of a Risk Assessment bar chart summary.

Chapter 6

Results

In this crucial chapter, we dive into the application and validation of our Python tool in a real-world setting. First, we present the chosen use case scenario that mirrors some of the complexities of contemporary ICT infrastructures. The chapter presents the dynamics of the ICT environment and analyzes the tool's efficacy, focusing on the results obtained. We delve into the outcomes of the Risk Assessment, providing a detailed analysis that sheds light on the effectiveness of our approach.

6.1 A Representative ICT Infrastructure

To validate the efficacy and practical applicability of PyRA, rigorous tests were carried out in a real-world scenario. The chosen use case for this testing encompasses a representative Information and Communication Technology (ICT) infrastructure. This carefully selected ICT environment serves as a realistic testing ground, allowing the tool to be evaluated under conditions that mirror real-world cybersecurity contexts.

As a first step, we had to model the target ICT infrastructure, but, as seen in Chapter 4, this step requires the Template ontology, containing the metamodel as well as the catalog of threats (CAPEC) and the inference rules allowing to map them to the resources of the system. In this work, we did not model the template ontology, so we took the one defined in the works of De Rosa et al. [34] and Maunero [11], and from there we manually modeled our target ICT infrastructure,

thus obtaining the Base ontology to give as input to the PyRA tool.

The target ICT infrastructure, depicted in Figure 6.1, is a slightly simplified version of the one used in the works of De Rosa et al. [34] and Maunero [11]. We chose to use a simpler version of the architecture not because the tool could not analyze it effectively, but because we wanted to limit the size of the results we would present. In fact, as we discuss in Section 6.5, the tool generates a significantly higher number of results as the size of the infrastructure increases. This is especially true for the vulnerability management part, particularly if the selection of CPE that identifies a particular resource is not carefully considered.

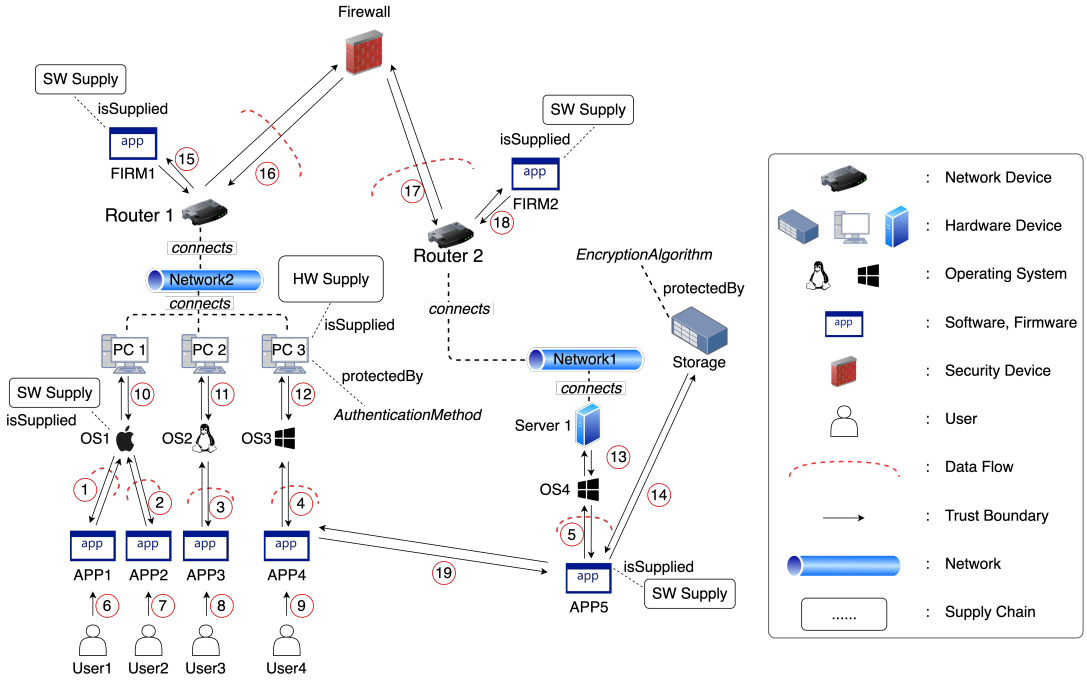


Figure 6.1: A diagram of the ICT Infrastructure employed to evaluate the tool.

The second step of the evaluation consisted of loading the modeled Base ontology into the PyRA tool and performing the risk assessment. In Table 6.1, we present, before moving on to the results obtained, a complete list of the data flows present in the target infrastructure to make it clear how they are described in the ICT ontology and, thus, in the final reports. Any other asset shown in Figure 6.1 is easy enough to identify within the final reports without further explanation.

Furthermore, to make it easier for the reader to understand the results of the

Data Flow	Source	Destination	Bidirectional	Crosses
DF1	APP1	OS1	Yes	TB1
DF2	APP2	OS1	Yes	TB2
DF3	APP3	OS2	Yes	TB3
DF4	APP4	OS3	Yes	TB4
DF5	APP5	OS4	Yes	TB5
DF6	User1	APP1	No	
DF7	User2	APP2	No	
DF8	User3	APP3	No	
DF9	User4	APP4	No	
DF10	OS1	PC1	Yes	
DF11	OS2	PC2	Yes	
DF12	OS3	PC3	Yes	
DF13	OS4	Server1	Yes	
DF14	APP5	Storage	Yes	
DF15	Router1	FIRM1	Yes	
DF16	Router1	Firewall1	Yes	TB16
DF17	Firewall1	Router2	Yes	TB17
DF18	Router2	FIRM2	Yes	
DF19	APP4	APP5	Yes	

Table 6.1: A complete table of the data flows in the target ICT infrastructure.

Vulnerability Assessment phase and to allow independent verification of the results, Figure 6.2 provides a comprehensive table that includes all the resources of the system examined. This table shows, where provided, the CPE, Vendor, Product, and Version of each resource. The Table reports the resources exactly as they are in the ontology used to test the tool. The choice to leave some of the information incomplete is deliberate, taking into consideration the possibility that, in a real case, it may not be possible to derive all of it and that the CPE database, although

vast, does not include all the products on the market.

Resource	CPE	Vendor	Product	Version
APP1	cpe:2.3:a:apple:numbers:*:*:*:*:*:mac_os_x:*:*	MISSING	MISSING	MISSING
APP2	cpe:2.3:a:apple:safari:14.0:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
APP3	cpe:2.3:a:mozilla:thunderbird:101.0:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
APP4	cpe:2.3:a:microsoft:sql_server_management_studio:*:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
APP5	cpe:2.3:a:microsoft:sql_server:2022:*:*:*:*:*:x64:*	MISSING	MISSING	MISSING
FIRM1	cpe:2.3:o:tp-link:tl-er7520g_firmware:-:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
FIRM2	cpe:2.3:o:dlink:dsr-250n_firmware:3.17:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
Firewall1	cpe:2.3:o:sophos:sfos:18.0:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
Network1	MISSING	MISSING	MISSING	MISSING
Network2	MISSING	MISSING	MISSING	MISSING
OS1	cpe:2.3:o:apple:mac_os_x:10.14:*:*:*:*:*:*:*	MISSING	MISSING	MISSING
OS2	cpe:2.3:o:canonical:ubuntu_linux:22.04:*:*:*:lts:*:*:*	MISSING	MISSING	MISSING
OS3	cpe:2.3:o:microsoft:windows_11_22h2:*:*:*:*:*:*:x64:*	MISSING	MISSING	MISSING
OS4	cpe:2.3:o:microsoft:windows_server_2022:10.0.20348.1547:*:*:*:azure:*:x64:*	MISSING	MISSING	MISSING
PC1	MISSING	MISSING	MISSING	MISSING
PC2	MISSING	dell	alienware_13_r2	MISSING
PC3	MISSING	MISSING	MISSING	MISSING
Router1	MISSING	MISSING	MISSING	MISSING
Router2	MISSING	MISSING	MISSING	MISSING
Server1	MISSING	dell	poweredge_t440	MISSING
Storage	MISSING	MISSING	MISSING	MISSING

Figure 6.2: Table showing, for each resource, the information available for the Vulnerability Assessment phase.

6.2 Threat Modeling

In this section, we provide an overview of the identified threats to the system. Table 6.2 presents the results of the risk assessment's threat modeling subprocess. For each asset with at least one threat identified by the PyRA tool, the Table reports the total number of threats to that asset, a complete list of the categories to which the identified threats belong, and the list of their STRIDE labels.

Table 6.2 presents a comprehensive summary of the identified threats to the system. Each target is associated with a count of threats, classified based on threat categories, and further classified using the STRIDE model. Threat categories provide insights

into the nature of potential risks, while STRIDE classification aids in understanding the specific types of threats, such as Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. This detailed analysis facilitates a strategic and targeted threat mitigation approach.

Target	**Threats Count**	**Threat Categories**
APP1	25	SupplyChain, AuthenticationMechanism, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput
APP2	25	SupplyChain, AuthenticationMechanism, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput
APP3	25	SupplyChain, AuthenticationMechanism, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput
APP4	27	ClientServerInteraction, SupplyChain, AuthenticationMechanism, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput
APP5	32	ClientServerInteraction, SupplyChain, AuthenticationMechanism, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput, SoftwareSupply
DF1	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
OS1	27	SupplyChain, AuthenticationMechanism, HardwareInput, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput, SoftwareSupply
DF10	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel

Continued on next page

Table 6.2: Table with summarized results of the Threat Modeling phase.

Target	Threats Count	Threat Categories
PC1	22	SupplyChain, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
DF11	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
PC2	22	SupplyChain, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
OS2	21	SupplyChain, AuthenticationMechanism, HardwareInput, PrivilegeOrPermissionAbuse, SoftwareInput
DF12	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
PC3	25	SupplyChain, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
OS3	21	SupplyChain, AuthenticationMechanism, HardwareInput, PrivilegeOrPermissionAbuse, SoftwareInput
DF13	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
Server1	22	SupplyChain, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
OS4	21	SupplyChain, AuthenticationMechanism, HardwareInput, PrivilegeOrPermissionAbuse, SoftwareInput
DF14	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel

Continued on next page

Table 6.2: Table with summarized results of the Threat Modeling phase.

Target	Threats Count	Threat Categories
Storage	22	SupplyChain, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
DF15	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
FIRM1	19	SupplyChain, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput, SoftwareSupply
Router1	27	SupplyChain, AuthenticationMechanism, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
DF16	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
Firewall1	27	SupplyChain, AuthenticationMechanism, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
DF17	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
Router2	27	SupplyChain, AuthenticationMechanism, HardwareInput, HardwareSupply, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse, SoftwareInput
DF18	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel

Continued on next page

Table 6.2: Table with summarized results of the Threat Modeling phase.

Target	Threats Count	Threat Categories
FIRM2	19	SupplyChain, HardwareInput, Hardware, Software, CommunicationChannel, SoftwareInput, SoftwareSupply
DF19	30	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
DF2	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
DF3	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
DF4	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
DF5	36	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel, PrivilegeOrPermissionAbuse
DF6	15	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
DF7	15	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
DF8	15	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
DF9	15	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
EncAlgo	1	Crypto
Network1	15	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel

Continued on next page

Table 6.2: Table with summarized results of the Threat Modeling phase.

Target	Threats Count	Threat Categories
Network2	15	SupplyChain, NetworkCommunication, Hardware, Software, CommunicationChannel
Total	1039	

Table 6.2: Table with summarized results of the Threat Modeling phase.

6.3 Vulnerability Assessment

In the following section, we present a summarized overview of the results obtained from the vulnerability assessment carried out on the ICT infrastructure.

Before proceeding, it is necessary to lay down a premise. For the readers interested in conducting their own personal verification of the results, it is crucial to understand that we have deliberately chosen to only consider CVEs that have been reported from CVE-2021 onward. By focusing solely on relatively recent CVEs, our analysis is better suited to provide accurate and relevant results.

Resources	Analyzed Resources	CVEs	CWEs	ATT&CKs
21	14	974	118	214

Table 6.3: Table summarizing how the Vulnerability sub-ontology has been populated.

Now we can start by looking at Table 6.3, which we could consider as a preliminary report, since it does not give real insights about the vulnerability assessment. The Table shows that the PyRA tool was able to analyze only 14, out of the 21 instances of the Resource class defined in the ontology, due to the missing information needed to accurately identify them, as anticipated in Section 6.1. Furthermore, it shows the total number of CVE (974), CWE (118), and ATT&CK (214) entries that were collected from public knowledge bases to populate the Vulnerability sub-ontology. However, it should be clear that some of them are related to more than one resource, which explains the numbers shown in Table 6.4. Therefore, Table 6.3 is more about how the Vulnerability ontology was populated rather than about the outcome of

the vulnerability assessment phase. Finally, given its purpose, the table does not contain a CAPECs column, since in the Vulnerability sub-ontology CAPECs are populated as a data property of the Vulnerability class (a unique string of CAPEC IDs separated by commas), as seen in Chapter 4, rather than as individuals of the CAPEC class.

Resource	**CVEs**	**CWEs**	**CAPECs**	**ATT&CKs**
APP2	110	18	60	40
APP3	123	33	143	100
APP5	17	3	12	3
Firewall1	2	1	10	11
OS1	163	36	180	126
OS2	19	20	111	73
OS3	499	46	254	163
OS4	38	9	75	37
PC2	1	1	11	21
Server1	2	2	10	1
Total	974	169	866	575

Table 6.4: Vulnerability Assessment summary.

Table 6.4, on the contrary, gives the first good overview of the outcome of the vulnerability assessment phase. Of course, the resources without the information necessary for this phase (see Figure 6.2), and thus without any results to show here, were omitted from the table. However, the table contains only 10 resources, even though Table 6.3 shows that 14 resources have been analyzed by the tool. This is because we also omitted resources that, despite the fact that they have been analyzed, did not produce any results, which could be either because the tool could not find any CVE related to them, or because the only CVEs it could find were not recent enough to satisfy the limit we imposed, as anticipated in the premises of the current Section.

The quantitative insights provided by this summary table offer a preliminary understanding of the security posture of individual resources and the overall system. It should be noted that the automated generation of these results by our tool

significantly expedites the vulnerability assessment process. Manual enumeration of CVEs, CWEs, CAPECs, and ATT&CKs for each resource would have been a time-intensive endeavor. The efficiency gains realized through automation not only save time, but also enhance the scalability and reliability of the assessment.

To dig deeper into the outcomes of the VA we present, in the remainder of this section, a series of detailed results. Specifically, we present a short list of vulnerabilities for each resource that produced valuable results, each spotlighting three vulnerabilities ordered by their CVSS scores. The decision to highlight only a small subset of vulnerabilities streamlines the presentation, providing a concise but insightful overview of the main security concerns for each resource. This approach ensures that the reader can efficiently grasp the key vulnerabilities that affect each component. The tables offer a pragmatic balance between granularity and readability.

Vulnerabilities for **APP2**

1. **CVE**: CVE-2021-30953
    - **CVSS**: 8.8
    - **CWEs**: N/A
    - **CAPECs**: N/A
    - **ATT&CKs**: N/A
2. **CVE**: CVE-2023-38599
    - **CVSS**: 6.5
    - **CWEs**: N/A
    - **CAPECs**: N/A
    - **ATT&CKs**: N/A
3. **CVE**: CVE-2021-23841
    - **CVSS**: 5.9
    - **CWEs**: N/A
    - **CAPECs**: N/A
    - **ATT&CKs**: N/A

Vulnerabilities for **APP3**

1. **CVE**: CVE-2022-2505
   - **CVSS**: 8.8
   - **CWEs**: CWE-787
   - **CAPECs**: CAPEC-59, CAPEC-95, CAPEC-62, CAPEC-33, CAPEC-278, CAPEC-96, CAPEC-201, CAPEC-155
   - **ATT&CKs**: N/A
2. **CVE**: CVE-2022-2226
   - **CVSS**: 6.5
   - **CWEs**: CWE-294
   - **CAPECs**: CAPEC-102, CAPEC-645, CAPEC-701, CAPEC-652, CAPEC-60, CAPEC-644, CAPEC-94, CAPEC-145, CAPEC-555, CAPEC-509, CAPEC-19, CAPEC-561
   - **ATT&CKs**: T1021, T1021.002, T1027.009, T1087.003, T1114.002, T1129, T1132.001, T1133, T1134.001, T1546.004, T1546.016, T1550.002, T1550.003, T1550.004, T1557, T1558, T1558.003, T1583.001, T1596
3. **CVE**: CVE-2022-36314
   - **CVSS**: 5.5
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A

Vulnerabilities for **APP5**

1. **CVE**: CVE-2023-21705
   - **CVSS**: 8.8
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A
2. **CVE**: CVE-2023-21528
   - **CVSS**: 7.8

- **CWEs**: CWE-924
- **CAPECs**: CAPEC-130, CAPEC-25, CAPEC-85, CAPEC-129, CAPEC-128, CAPEC-490
- **ATT&CKs**: T1498.002, T1499.003, T1499.004

3. **CVE**: CVE-2023-21704
   - **CVSS**: 7.8
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A

Vulnerabilities for **Firewall1**

1. **CVE**: CVE-2022-1040
   - **CVSS**: 9.8
   - **CWEs**: CWE-305
   - **CAPECs**: CAPEC-114, CAPEC-135, CAPEC-651, CAPEC-670, CAPEC-173, CAPEC-134, CAPEC-217, CAPEC-612, CAPEC-75, CAPEC-40
   - **ATT&CKs**: T1048, T1111, T1127, T1134.005, T1195.001, T1525, T1548, T1550, T1552.002, T1558.003, T1564.005
2. **CVE**: CVE-2022-0331
   - **CVSS**: 5.3
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A

Vulnerabilities for **OS1**

1. **CVE**: CVE-2021-1736
   - **CVSS**: 7.8
   - **CWEs**: CWE-125, CWE-1266
   - **CAPECs**: CAPEC-560, CAPEC-546, CAPEC-680, CAPEC-540, CAPEC-675, CAPEC-150, CAPEC-37, CAPEC-545

- **ATT&CKs**: T1003, T1005, T1027.003, T1027.011, T1052, T1078, T1119, T1195.001, T1213, T1499.003, T1505.002, T1530, T1546.009, T1547.004, T1547.015, T1552.004, T1555, T1555.001, T1602

2. **CVE**: CVE-2021-1737
   - **CVSS**: 7.8
   - **CWEs**: CWE-787, CWE-807
   - **CAPECs**: CAPEC-59, CAPEC-95, CAPEC-62, CAPEC-33, CAPEC-278, CAPEC-96, CAPEC-201, CAPEC-155
   - **ATT&CKs**: N/A
3. **CVE**: CVE-2021-1738
   - **CVSS**: 7.8
   - **CWEs**: CWE-787, CWE-807
   - **CAPECs**: CAPEC-59, CAPEC-95, CAPEC-62, CAPEC-33, CAPEC-278, CAPEC-96, CAPEC-201, CAPEC-155
   - **ATT&CKs**: N/A

Vulnerabilities for **OS2**

1. **CVE**: CVE-2022-0492
   - **CVSS**: 7.8
   - **CWEs**: CWE-437
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A
2. **CVE**: CVE-2022-1055
   - **CVSS**: 7.8
   - **CWEs**: CWE-416
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A
3. **CVE**: CVE-2022-29581
   - **CVSS**: 7.8
   - **CWEs**: N/A

- **CAPECs**: N/A
- **ATT&CKs**: N/A

Vulnerabilities for **OS3**

1. **CVE**: CVE-2022-41094
   - **CVSS**: 7.8
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A
2. **CVE**: CVE-2022-44682
   - **CVSS**: 6.8
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A
3. **CVE**: CVE-2023-21525
   - **CVSS**: 5.3
   - **CWEs**: CWE-61
   - **CAPECs**: CAPEC-53, CAPEC-27, CAPEC-35, CAPEC-96, CAPEC-150
   - **ATT&CKs**: T1003, T1027.006, T1027.008, T1027.009, T1027.011, T1119, T1213, T1530, T1547.015, T1555, T1564.009, T1566.002, T1578.002, T1583.001, T1602

Vulnerabilities for **OS4**

1. **CVE**: CVE-2021-26441
   - **CVSS**: 7.8
   - **CWEs**: CWE-269, CWE-287
   - **CAPECs**: CAPEC-140, CAPEC-115, CAPEC-22, CAPEC-57, CAPEC-633, CAPEC-593, CAPEC-114, CAPEC-233, CAPEC-76, CAPEC-194, CAPEC-94, CAPEC-58, CAPEC-151, CAPEC-650, CAPEC-122, CAPEC-268

- **ATT&CKs**: T1040, T1070, T1087.004, T1134, T1185, T1505.003, T1548, T1550, T1550.001, T1557, T1558.003, T1562.002, T1562.003, T1562.008, T1563, T1578.004, T1583

2. **CVE**: CVE-2021-36953
   - **CVSS**: 7.5
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A
3. **CVE**: CVE-2021-26414
   - **CVSS**: 4.8
   - **CWEs**: N/A
   - **CAPECs**: N/A
   - **ATT&CKs**: N/A

Vulnerabilities for **PC2**

1. **CVE**: CVE-2022-24410
   - **CVSS**: 4.2
   - **CWEs**: CWE-312
   - **CAPECs**: CAPEC-78, CAPEC-114, CAPEC-701, CAPEC-664, CAPEC-123, CAPEC-534, CAPEC-577, CAPEC-672, CAPEC-582, CAPEC-37, CAPEC-665
   - **ATT&CKs**: T1005, T1021.001, T1021.003, T1027.011, T1033, T1123, T1134.004, T1135, T1195.003, T1211, T1542.002, T1546.009, T1547.004, T1548, T1552.003, T1552.004, T1552.005, T1556, T1558.003, T1559.002, T1560.001

Vulnerabilities for **Server1**

1. **CVE**: CVE-2023-25537
   - **CVSS**: 7.8
   - **CWEs**: CWE-787

- **CAPECs**: CAPEC-59, CAPEC-95, CAPEC-62, CAPEC-33, CAPEC-278, CAPEC-96, CAPEC-201, CAPEC-155
- **ATT&CKs**: N/A

2. **CVE**: CVE-2021-21557
   - **CVSS**: 6.7
   - **CWEs**: CWE-125
   - **CAPECs**: CAPEC-680, CAPEC-540
   - **ATT&CKs**: T1195.001

6.4 Risk Assessment

In this section, we present the results of the risk assessment phase. As explained in Chapter 5, in this phase PyRA takes the threats found in the Threat Modeling phase and, for each of them, computes the risk score based on their likelihood and severity values.

Table 6.5 provides a summarized overview of the risk posture of the assessed ICT infrastructure by listing the number of threats for each target and their respective risk levels. The details of the risk computation have previously been discussed in Chapter 5.

Target	Very Low	Low	Medium	High	Very High
APP1	0	1	16	6	2
APP2	0	1	16	6	2
APP3	0	1	16	6	2
APP4	0	1	16	8	2
APP5	0	1	19	10	2
DF1-A	0	3	8	6	1

Continued on next page

Table 6.5: Summary of risks by resource and risk level.

Target	Very Low	Low	Medium	High	Very High
DF1-B	0	3	8	6	1
DF10-A	0	3	7	4	1
DF10-B	0	3	7	4	1
DF11-A	0	3	7	4	1
DF11-B	0	3	7	4	1
DF12-A	0	3	7	4	1
DF12-B	0	3	7	4	1
DF13-A	0	3	7	4	1
DF13-B	0	3	7	4	1
DF14-A	0	3	7	4	1
DF14-B	0	3	7	4	1
DF15-A	0	3	7	4	1
DF15-B	0	3	7	4	1
DF16-A	0	3	8	6	1
DF16-B	0	3	8	6	1
DF17-A	0	3	8	6	1
DF17-B	0	3	8	6	1
DF18-A	0	3	7	4	1
DF18-B	0	3	7	4	1
DF19-A	0	3	7	4	1
DF19-B	0	3	7	4	1
DF2-A	0	3	8	6	1
DF2-B	0	3	8	6	1

Continued on next page

Table 6.5: Summary of risks by resource and risk level.

Target	Very Low	Low	Medium	High	Very High
DF3-A	0	3	8	6	1
DF3-B	0	3	8	6	1
DF4-A	0	3	8	6	1
DF4-B	0	3	8	6	1
DF5-A	0	3	8	6	1
DF5-B	0	3	8	6	1
DF6-A	0	3	7	4	1
DF7-A	0	3	7	4	1
DF8-A	0	3	7	4	1
DF9-A	0	3	7	4	1
EncAlgo	0	0	0	1	0
FIRM1	0	0	13	4	2
FIRM2	0	0	13	4	2
Firewall1	0	1	17	7	2
Network1	0	3	7	4	1
Network2	0	3	7	4	1
OS1	0	0	17	8	2
OS2	0	0	13	6	2
OS3	0	0	13	6	2
OS4	0	0	13	6	2
PC1	0	1	14	5	2
PC2	0	1	14	5	2
PC3	0	1	15	7	2

Continued on next page

Table 6.5: Summary of risks by resource and risk level.

Target	Very Low	Low	Medium	High	Very High
Router1	0	1	17	7	2
Router2	0	1	17	7	2
Server1	0	1	14	5	2
Storage	0	1	14	5	2
Total	0	121	553	291	74

Table 6.5: Summary of risks by resource and risk level.

Consistency in the results obtained from similar target types, like Data Flows (DFxx), for example, supports the notion that semantic reasoning enables automatic inference of threats based on specific rules and relationships within the ontology. The fact that similar resources produce similar or identical results is in line with the expectation that threats are computed by inference using SWRL rules. This implies that the automated approach effectively captures and applies similar threat scenarios to related resources.

Table 6.6 presents an extract of the detailed Risk Assessment report, which contains only a few of the threats provided by the PyRA tool. However, they are ranked by risk level as they would be in the complete report. This ranking allows one to identify the most important threat in the infrastructure under analysis, to efficiently plan and execute risk management activities.

Table 6.6: Extract of the detailed Risk Assessment report.

Resource	Type	Threat	Risk	Risk Level
APP1	Software	CAPEC-123	20	Very High
APP1	Software	CAPEC-173	20	Very High
APP2	Software	CAPEC-123	20	Very High
DF1-A	DataFlow	CAPEC-94	20	Very High

Continued on next page

Resource	Type	Threat	Risk	Risk Level
OS1	OperatingSystem	CAPEC-123	20	Very High
OS1	OperatingSystem	CAPEC-173	20	Very High
DF1-B	DataFlow	CAPEC-94	20	Very High
DF10-A	DataFlow	CAPEC-94	20	Very High
PC1	HardwareDevice	CAPEC-233	16	High
PC1	HardwareDevice	CAPEC-240	16	High
PC2	HardwareDevice	CAPEC-233	16	High
OS2	OperatingSystem	CAPEC-21	16	High
FIRM1	Firmware	CAPEC-240	16	High
FIRM1	Firmware	CAPEC-116	12	Medium
Router1	NetworkDevice	CAPEC-114	12	Medium
Router1	NetworkDevice	CAPEC-115	12	Medium
Firewall1	SecurityDevice	CAPEC-114	12	Medium
Firewall1	SecurityDevice	CAPEC-115	12	Medium
Router2	NetworkDevice	CAPEC-114	12	Medium
Network1	Network	CAPEC-169	4	Low
Network1	Network	CAPEC-192	4	Low
Network2	Network	CAPEC-169	4	Low

Table 6.6: Extract of the detailed Risk Assessment report.

6.5 Considerations

In the course of this work, several considerations arise with respect to the effectiveness and limitations of the proposed solution. A primary constraint lies in the reliance on open source security knowledge bases that, while valuable, are not consistently exhaustive. For example, as demonstrated in Chapter 5, the PyRA tool

addresses the absence of Likelihood or Severity values in a CAPEC by assigning a default value of *High*. However, this reliance on default values can introduce inaccuracies in the final risk assessment.

The process of identifying vulnerabilities is intricately tied to the choices made in populating the ICT ontology. The precision and specificity of the data inserted play a pivotal role in determining the outcome of the Vulnerability Identification phase. As an example, choosing one of the following two CPEs to identify on Operating System makes a huge difference in the number of vulnerabilities identified by the tool:

1. cpe:2.3:o:microsoft:windows_server_2022:-:*:*:*:azure:*:x64:*
2. cpe:2.3:o:microsoft:windows_server_2022:10.0.20348.1547:*:*:*:azure:*:x64:*

The first CPE provides a more generic representation of the Windows Server 2022 operating system without specifying the version. In contrast, the second example offers a more detailed entry, including the specific version (10.0.20348.1547). The choice between these two representations has a profound impact on the results of the Vulnerability Identification phase. Opting for the more generic representation yields a significantly larger set of results, such as 1099 vulnerabilities. On the other hand, the more specific representation narrows down the results to 38 vulnerabilities. A more specific representation not only aids in identifying relevant vulnerabilities but also reduces the likelihood of false positives. The security team's workload is consequently streamlined, focusing on actionable and accurate results rather than sifting through an overwhelming number of potential vulnerabilities. This emphasizes the need for meticulous attention to detail when crafting entries within the ICT ontology for optimal outcomes in the Vulnerability Identification process.

Furthermore, it is often observed that the link between vulnerabilities (CVE) and their corresponding weaknesses (CWE) and threats (CAPEC) or attack tactics and techniques (ATT&CK) is not readily available. This lack of linkage can be seen in the detailed reports provided in Section 6.3. As a result, this limitation creates a hurdle in using the available information to its full potential, leading to incomplete risk assessments.

To overcome this challenge, traditional methods require extensive manual effort to analyze and establish links between these databases. However, there are recent approaches that involve the use of machine learning, as demonstrated by Kanakogi et al. [42], aimed at bridging the gaps between CVE and CAPEC, linking vulnerabilities with the corresponding attack patterns.

Another concern is that the solution proposed in this work identifies only known vulnerabilities in infrastructure components using external knowledge from vulnerability databases. It does not identify unknown vulnerabilities or those associated with internally developed components. To address this limitation, the use of the ontology for data representation allows users to enrich knowledge base information with results from manual Vulnerability Assessment and Penetration Testing (VAPT), ensuring interoperability with the rest of the information in the ontology.

Although automating threat modeling and risk assessment is a positive achievement, relying on publicly available knowledge for threat evaluation may not provide precise results. The generic knowledge provided by CAPEC may not sufficiently contextualize the threat assessments. It is essential to evaluate the results in the specific context of the assessment activity. Public databases inherently lack detailed contextualization, which may lead to over or underestimated risk levels concerning the case under analysis. So, to improve contextualization, it is necessary to incorporate additional data into the proposed model.

Despite these challenges and limitations, the proposed solution serves as a valuable tool to collect and organize security information related to Vulnerability Assessment, Penetration Testing, and Risk Management activities. By combining this information in an automated manner, a comprehensive view of the risk on the analyzed infrastructure is obtained, facilitating the planning of an effective risk management process.

Chapter 7

Conclusions

Throughout this thesis work we developed PyRA, an ontology-based tool developed in Python aimed at automating the Risk Assessment of ICT infrastructures. The tool uses an ontology-driven approach to simplify threat identification and vulnerability management.

PyRA efficiently processes an ontology that describes the ICT infrastructure, including assets, data flows, security mechanisms, and CPEs of resources. The ontology also incorporates CAPECs for threat modeling and risk assessment, supported by SWRL rules for systematic mapping. With the help of the ontology reasoner, the tool can produce a threat model in seconds. Once threats have been identified, it automates risk assessment by computing the risk for each threat.

The tool also makes it easier to identify vulnerabilities by retrieving CVEs based on the CPEs provided in the ICT ontology. It adds to the ontology additional knowledge from the CVE, CWE, and ATT&CK knowledge bases, such as vulnerabilities and weaknesses affecting the resources in the system, as well as attack tactics and techniques that could exploit these vulnerabilities. When available, it also collects the suggested mitigations from CWE and ATT&CK, thus streamlining the next steps in the vulnerability and risk management process.

By automating all these processes, PyRA brings great benefits, reducing the time and cost of improving the security posture of the ICT infrastructure under analysis. The tool also reduces the human error factor as it facilitates complicated and time-consuming procedures. Finally, it simplifies the execution of consistent assessments over time.

Although PyRA has proven to be effective, it has some limitations that have been discussed throughout the thesis, and analyzed in Section 6.5. Some of these limitations are beyond our control, such as the inconsistencies in public knowledge bases or the missing links between their data. However, there is still a lot of room for improvement for the limitations that we can address.

To further enhance user experience and functionality, future improvements to the PyRA tool could focus on developing a user interface (UI). This would also allow us to implement the ability to directly model the ICT infrastructure within the tool itself, rather than forcing the user to rely on external tools such as Protégé. A User Interface would also allow the users to better navigate the results of the assessments, as well as all the rest of the ontology's content. For example, the user could directly click on an ATT&CK Technique listed in the vulnerability report to take a look at the proposed mitigations.

To address the lack of contextual data, the tool could benefit from the introduction of guided procedures based on well-known frameworks. The introduction of guided procedures would assist users in accurately populating the ICT ontology, ensuring a more precise representation of the infrastructure, which would lead to more reliable results.

In conclusion, the PyRA tool represents a significant step forward in automating vulnerability assessment and risk management for ICT infrastructures. With ongoing development, incorporating suggested improvements, the tool has the potential to further enhance its capabilities and contribute even more to efficient and effective security practices.

Bibliography

[1] *What Is Cyber Risk Management? | IBM*. URL: https://www.ibm.com/topics/cyber-risk-management (visited on 10/10/2023) (cit. on pp. 1, 2).

[2] Kevin Stine, Stephen Quinn, Greg Witte, and R. K. Gardner. *Integrating Cybersecurity and Enterprise Risk Management (ERM)*. National Institute of Standards and Technology, Oct. 13, 2020. DOI: 10.6028/NIST.IR.8286. URL: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8286.pdf (visited on 10/18/2023) (cit. on p. 4).

[3] Joint Task Force Transformation Initiative. *Managing Information Security Risk :: Organization, Mission, and Information System View*. NIST SP 800-39. Gaithersburg, MD: National Institute of Standards and Technology, 2011, NIST SP 800–39. DOI: 10.6028/NIST.SP.800-39. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-39.pdf (visited on 10/08/2023) (cit. on pp. 5, 6).

[4] Joint Task Force Transformation Initiative. *Guide for Conducting Risk Assessments*. NIST SP 800-30r1. Gaithersburg, MD: National Institute of Standards and Technology, 2012, NIST SP 800–30r1. DOI: 10.6028/NIST.SP.800-30r1. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf (visited on 10/08/2023) (cit. on p. 5).

[5] National Institute of Standards and Technology. *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1*. NIST CSWP 04162018. Gaithersburg, MD: National Institute of Standards and Technology, Apr. 16, 2018, NIST CSWP 04162018. DOI: 10.6028/NIST.CSWP.04162018. URL: http://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf (visited on 10/18/2023) (cit. on p. 6).

[6] *IEEE/ISO International Standard-Health Informatics–Device Interoperability–Part 40101: Foundational–Cybersecurity–Processes for Vulnerability Assessment*. IEEE. DOI: 10.1109/IEEESTD.2022.9738537. URL: https://

ieeexplore.ieee.org/document/9738537/ (visited on 10/18/2023) (cit. on p. 6).

[7] *Threat Modeling - OWASP Cheat Sheet Series*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html (visited on 10/18/2023) (cit. on p. 6).

[8] *Threat Modeling Process | OWASP Foundation*. URL: https://owasp.org/www-community/Threat_Modeling_Process (visited on 10/18/2023) (cit. on pp. 6, 7).

[9] Cynthia Gonzalez. *What Is Threat Modeling? Key Steps and Techniques*. Exabeam. June 7, 2023. URL: https://www.exabeam.com/information-security/threat-modeling/ (visited on 10/19/2023) (cit. on p. 7).

[10] Wenjun Xiong and Robert Lagerström. «Threat Modeling – A Systematic Literature Review». In: *Computers & Security* 84 (July 2019), pp. 53–69. ISSN: 01674048. DOI: 10.1016/j.cose.2019.03.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404818307478 (visited on 02/25/2023) (cit. on p. 7).

[11] Nicolò Maunero. «Methodologies and tools to support Vulnerability Assessment and Cyber Risk Analysis activities within the Italian Cybersecurity Perimeter». In: (2023) (cit. on pp. 9, 28, 31, 32, 43, 48, 49).

[12] Dietmar P. F. Möller. «Cybersecurity Ontology». In: *Cybersecurity in Digital Transformation: Scope and Applications*. Cham: Springer International Publishing, 2020, pp. 99–109. ISBN: 978-3-030-60570-4. DOI: 10.1007/978-3-030-60570-4_7. URL: https://doi.org/10.1007/978-3-030-60570-4_7 (cit. on p. 9).

[13] *Ontologies and Their Use in Cybersecurity*. June 28, 2021. URL: https://www.kaspersky.com/blog/cybersecurity-ontology/40404/ (visited on 10/19/2023) (cit. on p. 9).

[14] Michael Debellis. «A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins». In: (2021) (cit. on p. 9).

[15] *CWE - About - CWE Overview*. URL: https://cwe.mitre.org/about/ (visited on 10/23/2023) (cit. on pp. 11, 12, 34).

[16] *History | CVE*. URL: https://www.cve.org/About/History (visited on 10/23/2023) (cit. on pp. 12, 16).

[17] *Process | CVE*. URL: https://www.cve.org/About/Process (visited on 10/23/2023) (cit. on p. 13).

[18] *Related Projects | MITRE ATT&CK®*. URL: https://attack.mitre.org/resources/related-projects/ (visited on 10/23/2023) (cit. on p. 13).

[19] *CAPEC - About CAPEC*. URL: https://capec.mitre.org/about/index.html (visited on 10/23/2023) (cit. on p. 14).

[20] *CAPEC - New to CAPEC?* URL: https://capec.mitre.org/about/new_to_capec.html (visited on 10/23/2023) (cit. on p. 14).

[21] *MITRE ATT&CK®*. URL: https://attack.mitre.org/ (visited on 10/23/2023) (cit. on p. 14).

[22] Blake Strom. *ATT&CK 101*. MITRE ATT&CK®. June 24, 2020. URL: https://medium.com/mitre-attack/att-ck-101-17074d3bc62 (visited on 10/23/2023) (cit. on p. 15).

[23] *NVD - Home*. URL: https://nvd.nist.gov/ (visited on 10/20/2023) (cit. on p. 16).

[24] *NVD - CPE*. URL: https://nvd.nist.gov/products/cpe (visited on 11/27/2023) (cit. on p. 16).

[25] *CVSS v4.0 Specification Document*. FIRST — Forum of Incident Response and Security Teams. URL: https://www.first.org/cvss/v4.0/specification-document (visited on 11/27/2023) (cit. on p. 17).

[26] Bruno Augusti Mozzaquatro, Carlos Agostinho, Diogo Goncalves, João Martins, and Ricardo Jardim-Goncalves. «An Ontology-Based Cybersecurity Framework for the Internet of Things». In: *Sensors* 18.9 (9 Sept. 2018), p. 3053. ISSN: 1424-8220. DOI: 10.3390/s18093053. URL: https://www.mdpi.com/1424-8220/18/9/3053 (visited on 11/28/2023) (cit. on p. 20).

[27] Romilla Syed. «Cybersecurity Vulnerability Management: A Conceptual Ontology and Cyber Intelligence Alert System». In: *Information & Management* 57.6 (Sept. 1, 2020), p. 103334. ISSN: 0378-7206. DOI: 10.1016/j.im.2020.103334. URL: https://www.sciencedirect.com/science/article/pii/S0378720620302718 (visited on 11/28/2023) (cit. on p. 20).

[28] Bijan Khazai, Tina Kunz-Plapp, Christian Büscher, and Antje Wegner. «VuWiki: An Ontology-Based Semantic Wiki for Vulnerability Assessments». In: *International Journal of Disaster Risk Science* 5.1 (Mar. 2014), pp. 55–73. ISSN: 2095-0055, 2192-6395. DOI: 10.1007/s13753-014-0010-9. URL: http://link.springer.com/10.1007/s13753-014-0010-9 (visited on 11/28/2023) (cit. on p. 20).

[29] Valentina Casola, Alessandra De Benedictis, Carlo Mazzocca, and Rebecca Montanari. «Toward Automated Threat Modeling of Edge Computing Systems». In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. 2021 IEEE International Conference on Cyber Security and Resilience (CSR). Rhodes, Greece: IEEE, July 26, 2021, pp. 135–140. ISBN: 978-1-66540-285-9. DOI: 10.1109/CSR51186.2021.9527937. URL: https:

//ieeexplore.ieee.org/document/9527937/ (visited on 11/28/2023) (cit. on p. 22).

[30] Lars Halvdan Fla, Ravishankar Borgaonkar, Inger Anne Tondel, and Martin Gilje Jaatun. «Tool-Assisted Threat Modeling for Smart Grid Cyber Security». In: *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. 2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA). Dublin, Ireland: IEEE, June 14, 2021, pp. 1–8. ISBN: 978-1-66542-529-2. DOI: 10.1109/CyberSA52016.2021.9478258. URL: https://ieeexplore.ieee.org/document/9478258/ (visited on 11/28/2023) (cit. on p. 22).

[31] Margus Välja, Fredrik Heiding, Ulrik Franke, and Robert Lagerström. «Automating Threat Modeling Using an Ontology Framework: Validated with Data from Critical Infrastructures». In: *Cybersecurity* 3.1 (Dec. 2020), p. 19. ISSN: 2523-3246. DOI: 10.1186/s42400-020-00060-8. URL: https://cybersecurity.springeropen.com/articles/10.1186/s42400-020-00060-8 (visited on 11/28/2023) (cit. on p. 22).

[32] Ashraf Tantawy, Sherif Abdelwahed, Abdelkarim Erradi, and Khaled Shaban. «Model-Based Risk Assessment for Cyber Physical Systems Security». In: *Computers & Security* 96 (Sept. 2020), p. 101864. ISSN: 01674048. DOI: 10.1016/j.cose.2020.101864. URL: https://linkinghub.elsevier.com/retrieve/pii/S016740482030136X (visited on 11/28/2023) (cit. on p. 23).

[33] Valentina Casola, Alessandra De Benedictis, Massimiliano Rak, and Umberto Villano. «Toward the Automation of Threat Modeling and Risk Assessment in IoT Systems». In: *Internet of Things* 7 (Sept. 2019), p. 100056. ISSN: 25426605. DOI: 10.1016/j.iot.2019.100056. URL: https://linkinghub.elsevier.com/retrieve/pii/S2542660519300290 (visited on 03/08/2023) (cit. on p. 23).

[34] Fabio De Rosa, Nicolo Maunero, Paolo Prinetto, Federico Talentino, and Martina Trussoni. «ThreMA: Ontology-Based Automated Threat Modeling for ICT Infrastructures». In: *IEEE Access* 10 (2022), pp. 116514–116526. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3219063. URL: https://ieeexplore.ieee.org/document/9936611/ (visited on 02/27/2023) (cit. on pp. 25, 28, 43, 48, 49).

[35] *Protégé*. URL: https://protege.stanford.edu/ (visited on 10/28/2023) (cit. on p. 27).

[36] *ArchiMate® 3.2 Specification. Standard C226* (cit. on pp. 29, 31).

[37] *NVD - Vulnerability Metrics*. URL: https://nvd.nist.gov/vuln-metrics/cvss (visited on 11/21/2023) (cit. on p. 32).

[38] *Understand Common Attack Patterns in Cybersecurity*. Fncyber. URL: https://www.fncyber.com/web-of-trust-article/understand-common-attack-patterns (visited on 10/23/2023) (cit. on p. 39).

[39] Sam Nguyen. «Automated Attack Tree Generation and Evaluation: Systemization of Knowledge». In: () (cit. on p. 40).

[40] Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. *Linking Threat Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations for Cyber Hunting*. Feb. 10, 2021. arXiv: 2010.00533 [cs]. URL: http://arxiv.org/abs/2010.00533 (visited on 11/11/2023). preprint (cit. on p. 40).

[41] *Welcome to Owlready2's Documentation! — Owlready2 0.44 Documentation*. URL: https://owlready2.readthedocs.io/en/v0.44/index.html (visited on 11/15/2023) (cit. on p. 42).

[42] Kenta Kanakogi, Hironori Washizaki, Yoshiaki Fukazawa, Shinpei Ogata, Takao Okubo, Takehisa Kato, Hideyuki Kanuka, Atsuo Hazeyama, and Nobukazu Yoshioka. «Tracing CVE Vulnerability Information to CAPEC Attack Patterns Using Natural Language Processing Techniques». In: *Information* 12.8 (8 Aug. 2021), p. 298. ISSN: 2078-2489. DOI: 10.3390/info12080298. URL: https://www.mdpi.com/2078-2489/12/8/298 (visited on 11/27/2023) (cit. on p. 69).