**UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS**

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

**MSc «Distributed Systems, Security and Emerging Information Technologies»**

ΠΜΣ «Κατανεμημένα Συστήματα, Ασφάλεια και Αναδυόμενες Τεχνολογίες Πληροφορίας»

# MSc Thesis

Μεταπτυχιακή Διατριβή

| | |
|---|---|
| **Thesis Title:**<br><br>Τίτλος Διατριβής: | **A security control ontology to support automated risk mitigation.**<br><br>Οντολογία μέτρων ασφάλειας για την υποστήριξη αυτοματοποιημένης αντιμετώπισης κινδύνων. |
| **Student's name-surname:**<br>Ονοματεπώνυμο φοιτητή: | **Varkas Ilias**<br>Βάρκας Ηλίας |
| **Father's name:**<br>Πατρώνυμο: | **Dimitrios**<br>Δημήτριος |
| **Student's ID No:**<br>Αριθμός Μητρώου: | ΜΠΚΣΑ/19004 |
| **Supervisor:**<br>Επιβλέπων: | **Panagiotis Kotzanikolaou, Associate Professor**<br>Παναγιώτης Κοτζανικολάου, Αναπληρωτής καθηγητής |

November 2023/ Νοέμβριος 2023

**3-Member Examination Committee**

Τριμελής Εξεταστική Επιτροπή

| | | |
|---|---|---|
| **Panagiotis Kotzanikolaou**<br>**Associate Professor**<br>Παναγιώτης Κοτζανικολάου<br>Αναπληρωτής Καθηγητής | **Constantinos Patsakis**<br>**Associate Professor**<br>Κωνσταντίνος Πατσάκης<br>Αναπληρωτής Καθηγητής | **Michael Psarakis**<br>**Associate Professor**<br>Μιχαήλ Ψαράκης<br>Αναπληρωτής Καθηγητής |

# A security control ontology to support automated risk mitigation
# Οντολογία μέτρων ασφάλειας για την υποστήριξη αυτοματοποιημένης αντιμετώπισης κινδύνων

## Table of Contents

1 INTRODUCTION ..... 6
1.1 Motivation ..... 9
1.2 Contribution ..... 10
1.3 Structure of the document ..... 11
2 Related Work ..... 12
2.1 OSCTI (Open-Source Cyber Threat Intelligence) ..... 12
2.1.1 CPE ..... 12
2.1.2 CWE ..... 14
2.1.3 CVE ..... 15
2.1.4 OVAL ..... 17
2.1.5 MAEC ..... 18
2.1.6 NVD ..... 20
2.1.7 CVSS ..... 21
2.1.8 Metric Groups ..... 22
2.1.9 CWSS ..... 25
2.1.10 EPSS (Exploit Prediction Scoring System) ..... 28
2.1.11 Threat - Attack Libraries ..... 29
2.1.12 Control Libraries ..... 33
2.2 Linking Frameworks ..... 36
2.2.1 D3fend – ATT&CK Linking ..... 36
2.2.2 CAPEC – CWE Linking ..... 37
2.2.3 CVE - CWE Linking ..... 39
2.3 Related Research ..... 40
2.3.1 Threat KG ..... 40
2.3.2 NLP Based Approaches ..... 41
3 METHODOLOGY ..... 44
3.1 CVE - CWE connection ..... 44
3.2 CVE - CPE connection ..... 44
3.3 CWE - CAPEC relation ..... 45
3.4 CAPEC – ATT&CK connection ..... 45
3.5 ATT&CK - DEFEND connection ..... 46
4 IMPLEMENTATION ..... 47
5 TEST CASES AND RESULTS ..... 64
6 CONCLUSION and FUTURE WORK ..... 70
7 ACKNOWLEDGMENTS ..... 72
8 REFERENCES ..... 73

# ABSTRACT

The current state of technology has created a rapid expansion and usage of IT, OT and IoT devices in production, businesses and households. Along with these assets follows a wide range of threats that cyber security specialists will have to deal with in the coming years. There are many sources of information concerning cyber threats called Open Source Cyber Threat Intelligence (OSCTI). Such sources are MITRE's ATT&CK Framework, CWE and CAPEC, which are threat catalogues, MITRE's CPE, an asset catalogue, CVE, a vulnerability database, MITRE's Digital Artifact Ontology (DAO), which are originated from MITRE, but are currently maintained by NIST. These sources of information allow security analysts to extract data about cyber threats and known vulnerabilities that might affect their systems. A wide range of vulnerabilities and threats are enumerated in the above data sources, while a good percentage of them are semantically interconnected; these interconnections are captured to a certain degree in the OSCTI, but there is a shortage of linked vulnerabilities, threats and mitigations. In an attempt to resolve this issue, MITRE has created mitigations within the ATT&CK Framework, which apply to specific threats. An extended effort for this issue was the creation of D3fend Ontology, where specific security controls or mitigations apply to specific types of assets, derived from the Digital Artifact Ontology, which is part of the D3fend Ontology. Security controls and mitigations from the D3fend Ontology are directly connected to existing ATT&CK techniques. Therefore, a holistic approach is required in order to study all the existing interconnections within the CPE-CVE-CWE-CAPEC-ATT&CK-D3FEND-DAO spectrum and provide a statistical view. This way we may offer insights towards bridging the gap among CPE and DAO, which may yield threat profiles or known assets without known vulnerabilities. Moreover, CVE and D3FEND interconnections occur by this statistical view, which due to using multiple catalogues as stepping stones, might not be semantically correct, so it should be studied, in order to decide if it should be integrated with data derived from other interconnection methods.

# ΠΕΡΙΛΗΨΗ

Λόγω της ραγδαίας τεχνολογικής ανάπτυξης τα τελευταία χρόνια, αρκετές συσκευές τεχνολογίας έχουν εισαχθεί σε αρκετούς τομείς, όπως στην παραγωγή, στις επιχειρήσεις, τα νοικοκυριά ακόμη και στην καθημερινότητα. Μαζί με αυτές τις τεχνολογικές συσκευές, ακολουθεί ένα ευρύ φάσμα απειλών που θα πρέπει να αντιμετωπίσουν οι ειδικοί στην ασφάλεια στον κυβερνοχώρο τα επόμενα χρόνια. Ωστόσο, υπάρχουν πολλές πηγές πληροφοριών που αναφέρονται στις απειλές στον κυβερνοχώρο και ονομάζονται OSCTI (Open Source Cyber Threat Intelligence). Τέτοιες πηγές είναι το MITRE's ATT&CK Framework, το CWE και το CAPEC, που είναι κατάλογοι απειλών, το CPE του MITRE, ένας κατάλογος περιουσιακών στοιχείων, το CVE, μια βάση δεδομένων ευπάθειας, το Digital Artifact Ontology (DAO) του MITRE, που προέρχονται από το MITRE, αλλά προς το παρόν διατηρούνται από το NIST. Μέσω αυτών των πηγών, οι αναλυτές ασφαλείας μπορούν να εξάγουν δεδομένα σχετικά με απειλές στον κυβερνοχώρο καθώς και γνωστά τρωτά σημεία, που ενδέχεται να επηρεάσουν τα συστήματά τους. Στις ανωτέρω πηγές δεδομένων απαριθμείται ένα ευρύ φάσμα τρωτών σημείων και απειλών, καθώς επίσης ένα ποσοστό από αυτά είναι σημασιολογικά διασυνδεδεμένα. Οι συνδέσεις αυτές, έως έναν βαθμό, καταγράφονται στο OSCTI, αλλά υπάρχει έλλειψη συνδεδεμένων τρωτών σημείων, απειλών και άμυνας. Σε μια προσπάθεια επίλυσης αυτού του ζητήματος, έχουν εισαχθεί μέτρα και τεχνικές μετριασμού των επιθέσεων εντός του ATT&CK, τα οποία ισχύουν για συγκεκριμένες απειλές. Αυτό είχε ως αποτέλεσμα την δημιουργία της βάσης D3fend, καθώς οι έλεγχοι ασφαλείας και τα εκάστοτε μέτρα της οντολογίας D3fend συνδέονται άμεσα με τις υπάρχουσες τεχνικές επίθεσης της ATT&CK. Επομένως, μια ολοκληρωτική προσέγγιση είναι αναγκαία, προκειμένου να εξεταστούν οι διασυνδέσεις μέσα στο φάσμα που συμπεριλαμβάνει καταλόγους καταγραφής λογισμικών και υλικών - εξαρτημάτων, καταλόγους αδυναμιών, ευπάθειας, τεχνικών επίθεσης και άμυνας ( CPE-CVE-CWE-CAPEC-ATT&CK-D3FEND-DAO ) και να δοθεί μια στατιστική προσέγγιση. Με αυτόν τον τρόπο μπορούμε να προσφέρουμε πληροφορίες για τη γεφύρωση του χάσματος μεταξύ καταγεγραμένων υλικών και λογισμικών (CPE) και οποιασδήποτε ανεπιθύμητης αλλαγής που εισάγεται σε μια ψηφιακή διαδικασία (DAO), μπορεί να αποφέρουν προφίλ απειλών ή γνωστά στοιχεία χωρίς γνωστά τρωτά σημεία. Επιπλέον, οι διασυνδέσεις CVE και D3FEND προκύπτουν από αυτήν τη στατιστική προσέγγιση, η οποία λόγω της χρήσης πολλαπλών καταλόγων, ενδέχεται να μην είναι σημασιολογικά σωστή, επομένως θα πρέπει να μελετηθεί, προκειμένου να αποφασιστεί εάν θα πρέπει να ενσωματωθεί με δεδομένα που προέρχονται από άλλες μεθόδους διασύνδεσης.

# 1 INTRODUCTION

During the last decade, there has been a rise in technological devices, altering our everyday lives, the way people work and communicate, the cities we live in as well as the functionality of the factories. Herein, the current state of Cyber-Physical Systems is rapidly advancing and increasing integrations into various sectors, from manufacturing and transportation to healthcare and every day routine. Cyber Physical Systems have become more interconnected with the Internet of Things. As a result, more and more physical objects and systems are connected to the internet and have the ability to interact with other devices and systems. This connectivity allows real time data collection and remote control of physical processes. Therefore, introducing a wide range of devices in our standards of living has created an extending scope of very complex configurations, a fact that leads to an outstretched vulnerable surface.

Personal computers, initially bulky and limited in functionality, soon evolved into well developed, powerful machines that allowed individuals to access vast amounts of information and communicate with the world. The evolution of processors and micro processors during the years has resulted in a significant breakthrough in manufacturing high-end technological devices. This development set the stage for further innovations in the form of laptops, tablets, and eventually smart piece of equipment. With the use of the internet, a vast amount of appliances are connected to it, either for downloading data or communicating with other devices over the internet. Up until the April 2023, there were reported 5.18 billion users worldwide, which corresponds to the 64.6 % of the worldwide population (statista.com ). Furthermore, the technology development has changed the way users and machines cope with their data, using the cloud as a means of saving technology, delivering computing services of software and hardware and with the growth of the optical fibers and mobile networks, evolving from 3G to 4G and finally to 5G, these have increased the speed and the amount of the exchanged information.

Consequently, the state of the Cyber Physical Systems keeps evolving, as the field of technology keeps evolving rapidly.

- **Integration with IoT**: Cyber-Physical Systems have become more integrated with the Internet of Things (IoT). This means that physical objects and systems are increasingly connected to the internet and can interact with other devices and systems. This connectivity allows for real-time data collection and remote control of physical processes.
- **Industry 4.0**: CPS plays a pivotal role in the Industry 4.0 revolution, also known as the fourth industrial revolution. In manufacturing and industrial settings, CPS is used for smart factories, where machines, robots, and production systems are interconnected and can make decisions autonomously based on real-time data. This automation not only improves productivity but also enhances safety by assigning dangerous procedures to machines.
- **Healthcare**: In healthcare, CPS is being used for remote patient monitoring, smart medical devices, and even robotic surgery systems. These systems improve patient care by providing healthcare professionals with real-time patient data and enabling precise, minimally invasive procedures. Also, wearable health devices and telemedicine have become more prominent, especially during the COVID-19 pandemic.
- **Smart Homes**: IoT devices have turned out homes into smart living spaces. Smart thermostats, lighting systems, door locks, smart sensors and even more common appliances such as refrigerators can be managed through an automated system in order to reduce electrical energy and increasing the quality of life inside a smart home. Therefore, instead of managing several devices, home owners have the ability to operate all of them through one device; a smart phone or a tablet.
- **Smart Cities**: Many cities are incorporating CPS into their infrastructure to become "smart cities." This involves using sensors and data analytics to optimize transportation, energy

usage, waste management, and public services. The aim is to improve the quality of life for residents while reducing environmental impacts.

- **Energy Efficiency**: Energy efficiency is a major focus in CPS applications. By optimizing the operation of physical systems through data analysis and control algorithms, CPS can reduce energy consumption in various domains, from manufacturing to buildings.
- **Autonomous Vehicles**: The development of autonomous vehicles is heavily reliant on cyber-physical systems. These vehicles use a combination of sensors, algorithms, and connectivity to navigate and make decisions in real-time, which has the potential to revolutionize transportation and logistics.
- **Agriculture and Precision Farming**: In agriculture, by using IoT devices and various smart agriculture gadgets, farmers are getting help to optimize crop yields and lower the production risks, where smart sensors provide real time data on soil and weather conditions, reducing water and pesticide usage.

We have introduced a wide range of devices in our everyday lives from businesses and industry to smart cities and homes. This has created very complex configurations, a fact that leads to an extended vulnerable surface. As a consequence, the most representative smart device is the smart phone. They have been developed to pocket size computers, where apart from simple communication devices, smart phones are multi functional hubs, embedded with powerful processors, high resolution cameras, large storage capacity and a wide range of applications that they have become an indispensable tool from socializing and education to productivity and reading the news. They are an integral part for the majority of people worldwide. Furthermore, smart phones are gaining an essential role in banking. The users can have access to their bank account, pay services through their mobile phones and transfer money as they please. Furthermore, social network applications help individuals to socialize through their smartphones using chat messages, uploading images that depict themselves and their personal lives and publish their thoughts and concerns about facts and events and even succeed in finding locations around the globe using several applications that include online maps.

To succeed in the rise of online smart devices, powerful and robust servers are being used to cope with the users requests accordingly. These strong computers -servers- store personal data for each user, as they maintain this information. Web servers, file servers, DNS Servers, Mail Servers and every other role as a server provide the necessary data and execute the appropriate functions in order to make applications and services available. As regards our every day routine, personal computers, which are a simpler, downgraded form of a server, they help individuals to entertain themselves; they are used by schools and universities for educational purposes as well as for research goals, where they can be used to store personal data. During the recent years, personal computers contribute to telecommuting, where a rising number of people are able to work from their homes or their preferred working environment, making computers a vital piece of technology for living.

Apart from smart phones in our pockets and personal computers in our homes and working environment, smart application devices contribute to cities to operate fundamental city concepts, such as traffic control by managing traffic lights and city lighting by controlling street lights. This way, traffic congestions are reduced to the minimum, as smart applications alternate the traffic lights functionality, giving more passing time to the routes that have way more vehicles or pedestrians. Also, smart city lights contribute to the city's energy efficiency due to lighting optimizations, including light adjustments, where light intensity (luminance) is dimmed in places that less light is needed. Another factor is operating street lights based on light sensors and not based on scheduled hours. This means that city lights function when artificial lighting is needed.

Another approach in Cyber Physical Systems implementation is inside factories functionality and core processes, where Operational Technology systems administer a wide variety of machines, such as controlling valves, reassuring the perfect conditions during the production line and raises the worker's safety during swifts. Environmental Monitoring conditions inside a refinery can alert all working staff in case of air pollution due to an escape of gas during a process and this may protect the workers' lives

and decrease malfunctions and economical losses. Another OT attainment is the supervision of a dam, where the opening and closing valves are controlled by smart applications, keeping the water level between desired constants and prevent flooding.

All the above approaches lurk insidious consequences in cases where these systems are under successful cyber attacks. As regards smartphones, the risk of losing personal data such as personal photos, credentials about accounts that hand out more personal information and foremost the bank account credentials, where financial loss may take place. The victim may struggle to pay the financial expenditures, resulting electrical power outage, water service interruption and a lack of food, affecting the physical and the mental health. In addition, a privacy invasion may lead to the exposure of intimate or personal moments, gaining access to the user's social accounts or email accounts can post false or harmful information, leading to a decline in professional reputation, fraud or blackmailing.

Moreover, the threats that are introduced in traffic lights, may lead to extensive traffic jams that may take several hours to overcome. Another vital consequence of an attack in traffic light, considering a green light in a conjunction, may result vehicle damages and probably death of the cars' passengers or, even worse, pedestrian deaths when crossing the road. The city lighting threats that cause blinking and in combination with increased light luminance may result an epileptic seizure of the passer-by or, even worse, cause an accident due to flashing lights that interrupt bystander's clear sight. Another threat might include the rise of criminality in an area, where late night hours the street lights turn off, giving the opportunity to thieves or to these with delinquent behavior succeed in their harmful purpose.

Correspondingly, the threats that emanate from the control of the physical devices may result machine failures, equipment manipulation and malfunction, exposing dangers to the environment, to the industry and to the workers as well. In case of an attack in a refinery, if environmental monitoring sensors malfunction and there is a gas escape, several workers may face health issues, lack of breath or even casualties. The refinery will stop functioning, causing large economic issues and environmental damage, if the gas leakage is not dealt with success on time. By the same token, the pipes that control the flow of the water within a dam may cause the flooding of a whole region if they intentionally allow larger amounts of water to flow.

Although, the rise of technological devices has been alternating the way we interact with the world, enhancing our everyday lives, reshaping our homes and the cities we live in and optimizing industrial procedures, bringing immense benefits, there are various ethical and societal concerns. As we continue to integrate technology into our societies, there is an inevitable balance to achieve between harnessing its potential and mitigating its negative consequences. These interconnected systems present challenges related to security, privacy, data breach and environmental issues, which Open Source Cyber Threat Intelligence offers solutions to deal with the situation. OSCTI includes concise cyber security information through publicly accessible sources on the internet. This type of data is gathered from open source accessible libraries, frameworks and datasets. While OSCTI provides valuable insights, not all information found is accurate or reliable; therefore, quality measures should take place to avoid false assumptions. When they are concentrated, data are analyzed to identify patterns and possible threats. New malware signs, hacking techniques and vulnerabilities are detected by monitoring online sources. After that, information is shared through public reports, forums and specialized platforms. Therefore, OSCTI helps organizations to identify surfacing threats and attack vectors and it can be used as an early warning system, where organizations may take proactive measures against potential cyber damage.

The structure of this paper is as follows; in chapter 2 the vulnerability frameworks are explained in depth, such as CPE, CWE, CVE, OVAL, MAEC, NVD and threat – attack frameworks are described such as (CAPEC , ATT&CK ) and security control frameworks (STRIDE, D3FEND) are depicted with more details. In the above chapter, frameworks/libraries are illustrated in detail as regards what is the purpose for each one, what is the data structure and why it is created. In addition, in chapter 2 and 3 connections between these frameworks are presented and in chapter 6 there is the thesis induction, as well as ways to extent existing relations.
Therefore the main focus of the dissertation is:

- Review existing security frameworks that relate to weaknesses, vulnerabilities and threats that system entities may encounter

- Review existing security control frameworks/databases (Mitre D3fend, Stride)
- Review existing connections between threats, vulnerabilities and control frameworks
- Propose solutions to create new or extend the existing relationships between control, vulnerability and threat libraries.

## 1.1 MOTIVATION

A new cyber security method would be to effectively connect exposed vulnerabilities with defense techniques, as a way to mitigate potential attack efforts. Furthermore, the linkage between the security control frameworks, the vulnerability, the threat and the attack frameworks/datasets provide beneficial information about the automatic production of the defensive actions in order to mitigate threats efficiently.

Each one of these framework categories (vulnerability, threat and control) has a distinct role in risk assessment procedure.

Vulnerabilities can be used by adversaries in order to implement not allowed actions that affect the integrity, the availability and the confidentiality of a system, leading to an exposed state of an attack. These vulnerabilities include security gaps that extend to a faulty operation of the operating system/application/entity. There are lists that enumerate vulnerabilities like Common Vulnerabilities and Exposures (CVE), in which all the publicly found vulnerabilities are listed there. The study and understanding these vulnerabilities may create possible attack models and mitigate attacks and intrusions; indicative use is within firewall rules. Furthermore, an additional tool of listed vulnerabilities is Open Vulnerability and Assessment Language (OVAL), which includes additional information of CVE, such as possible files that encapsulate suspicious code, the application's version that is most likely to be infected as well as the system's characteristics that show its state.

In addition, weaknesses are listed in catalogues as well, such as Common Weakness Enumeration (CWE). Security flaws lead to weaknesses and attack methodologies rely on weaknesses that applications / operating systems have. These vulnerabilities are inextricably linked to forms and attack techniques that attackers carry out. It is essential to figure out the ways the adversary may use to presume upon the security gaps. For this reason, there are recorded attack techniques that take advantage of the above weaknesses, such as Common Attack Pattern Enumeration and Classification list (CAPEC), which describes the steps of an attack, the understanding of the exploitation of a weakness by the attackers in order to enhance the entity's defenses and eliminating the attack's effectiveness.

Furthermore, through these attack patterns, the appropriate attack techniques are found, consequently attack and defend relations contribute to defensive techniques.

Vulnerability catalogues and attack pattern dictionaries don't always have an immediate connection, as vulnerabilities and attack methods are often independent. Despite the existence of similarity algorithms like TF-IDF, USE and SBERT that suggest a connection between them, accurate linking proves challenging and costly. The TF-IDF algorithm estimates the importance of words based on their frequency in the input, providing a relevance score. Additionally, USE generates dimensional vectors for various NLP implementations, using either the transformer encoder or Deep Average Network (DAN). SBERT, an enhancement of BERT, employs Siamese and triplet networks for accurate sentence vectors.

Connecting CVEs to CAPEC proves to be challenging due to the high abstraction in CVE entries and the complex linkage among CWE, CVE and CAPEC. Abstract descriptions in weaknesses contribute to low accuracy rates, making it difficult to link the correct CWE entity with the CAPEC entity. Another crucial connection explored is between CAPEC attack patterns and the attack techniques inside the ATT&CK framework. While only a small portion of CAPEC entries are currently linked to ATT&CK, leveraging Natural Language Processing algorithms may enhance the connection by analyzing descriptions and text data.

Therefore, there is no immediate connection between the vulnerabilities a system has and the appropriate defending techniques that are mandatory to protect the vulnerable system before exploitation. As a result, the main focus of this thesis is to begin with vulnerabilities and connect them with defend techniques through several dictionaries and frameworks. Currently, there is no connection between vulnerabilities and defend methods. The control framework's role in risk assessment is to provide information as regards the countermeasures to mitigate attack techniques, as well as defensive techniques, populating the circumstances that this solution would take effect.

Trying to connect vulnerability dictionaries with attack methods and attack methods with attack techniques so that in the end, attack techniques relate with defensive techniques, leading to an indirect linking between vulnerabilities and defense.
The state of connection among these dictionaries/frameworks is presented using the pandas library, showing the current condition and through which future approaches may occur.

## 1.2 CONTRIBUTION

Due to the fact that there is not always an immediate and clear connection between vulnerability catalogues and attack pattern dictionaries, because vulnerabilities and attack methods are independent. There are intermediate steps to take, in order to be able to link vulnerability and attack catalogues and that is through other dictionaries, such as weakness enumeration databases. Furthermore, this relation is being done mostly manually, meaning that a person tries to find the correlation that he is searching for.

Therefore, the aim of this thesis is to present the empty gaps between vulnerability repositories and defense databases. This is a crucial step to extract information about defensive methods through vulnerability data before a vulnerable system becomes exploited making its vulnerability a weakness. As a result, in order to present and analyze the corresponding catalogues / databases as regards their structure, their use and the information they provide such as:

- Platform enumeration (CPE): This catalogue consists of informative data about the entities that constitute each desired system. This piece of information contributes to provide information about the operating systems that are being used, the software applications and the hardware installation of the system.
- Weakness Enumeration (CWE): This dictionary includes valuable data about the flaws and the security gaps that can be identified inside a software application, a hardware execution or a network. For that reason, using this type of data can eliminate the security risk.
- Vulnerabilities and Exposures platform (CVE): This platform contains publicly divulged vulnerabilities, referring to these assets that make an entity of a system vulnerable. In addition, these vulnerability records assure that experts exchange information about the same issue.
- Vulnerability Database (NVD): This framework provides information about vulnerability assessment and control, security risk measurement, as well as security imperfections and data about systems and platforms.
- Attack pattern enumeration (CAPEC): Attack pattern catalogues include attack methods and descriptions about the execution flow of an attack. It enumerates and categorizes the Attack Patterns in order to be recognized and understood.
- Attack actions and Techniques (ATT&CK): This type of frameworks provides knowledge about the deployment of threat models and attack techniques, used attack methods and the impact that they have in the world. These techniques refer to several operating systems, hardware, network implementations and mobile devices.

- Categorized Defensive Techniques (D3fend): Defend framework categorizes mitigations and counter measures in contemplation of eliminating successful attacks, taking advantage of existing weaknesses of a system entity or a script flaw.

The use of data frames to analyze and extract the correlation among the informative catalogues, beginning with the platform enumeration catalogues and continue with vulnerabilities, weaknesses, attack and defend techniques accordingly.

## 1.3 STRUCTURE OF THE DOCUMENT

To begin with, in chapter 2 an examination of related work upon OSCTI is presented including a variety of publicly available sources, frameworks and tools, where each one of them encapsulates different information, such as platform enumeration, weaknesses, vulnerabilities, exploitations and mitigations. Furthermore, in this chapter there is an encapsulation of the existing connection between specific frameworks as well as related approaches that tried to minimize the gap between the connections. In section 3, an in depth relation between the examined sources is exhibited and in which ways these sources are interconnected. In addition, in section 4 there is a presentation of how to link the mentioned sources in order to end up with the final interconnected dataset and in section 5 the results from section 4 are inspected and various conclusions are depicted. The section 6 is a summary of the presented datasets, sources and the conclusions that arise.

# 2 Related Work

## 2.1 OSCTI (OPEN-SOURCE CYBER THREAT INTELLIGENCE)

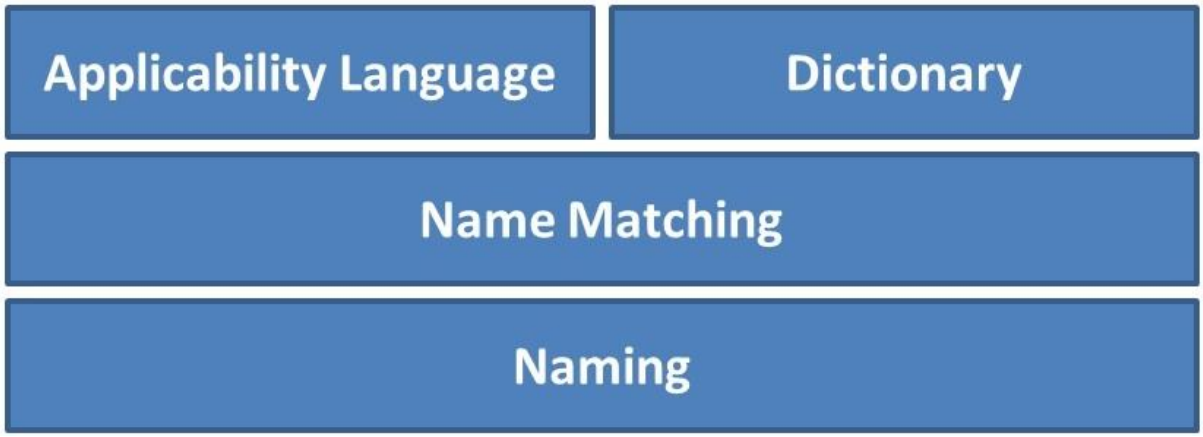
### 2.1.1 CPE

CPE stands for Common Platform Enumeration and is a procedure for describing and identifying applications, classes of applications, operating systems and hardware devices among several computing assets [1].It does not identify separate installations of applications or packages, rather than a generalized version of them.

It has a strict, structured format providing information about tests as regards operating systems, software applications and hardware units in a standardized format that is suitable for comparison. Using that information, risks and vulnerabilities can be identified. Therefore, CPE has a vital role in security assessment taking automated decisions if possible.

As regards CPE, it consists of four separate stipulations where the base specification is Naming and above this are Name Matching, Dictionary and Applicability Language. More specifically:

- *Naming*: Expounds the conventionalized methods for naming product classes using WFN (Well-Formed CPE Name). These Naming specifications include proceedings in order to link WFNs with machine readable encodings and vice versa.
- *Name Matching*: Includes a method for comparing CPE names as a set of values. As a result, the comparison can determine if the two CPE names are equal, if they have no relation or if the one is a subset of the other, making more complex comparisons achievable such as finding if a specific operating system version implements a sum of particular applications.
- *Dictionary*: Is a repository of CPE Names and data related to the names. In addition, a CPE name in the Dictionary refers to a generalized version of the named item and not to a particular installation of the product.
- *Applicability Language*: is built on top of the rest specifications to provide the appropriate functionality to develop complex logical expressions of the CPE Names that describe IT platforms. That kind of utterance is used to mark policies and documents about the platforms to which the documents refer to.




**1 CPE Structure [3]**

Furthermore, CPE provides a plan of actions for comparing entities and a guideline for machine readable encoding platform names, software and packages. The collection of CPEs is included in the CPE Dictionary.

## CPE Naming and Use Case

The implementation of labels within a sum of computing assets, which represent every discrete part of that computing asset, using the CPE naming technique, may be beneficial, for every tool that shares information about the individual software product on that end system [2]. Furthermore, these tools can exchange information, among them, for each entity individually.
Therefore, each piece of information may refer to asset management, vulnerability management or configuration assessments using the CPE names for this purpose.

Furthermore, Common Platform Enumeration catalogue conforms to a set of rules so that every cpe entry has a specific way of presenting information. That set of rules is defined as CPE Naming Specifications, currently is version 2.3 and defines a structural representation of vendors, items, operating systems and some other data that abide to each cpe record, making the use of this information easier for information technology tools that take automated decisions.
The WFN method is called Well Formed Name of CPE and is backwards compatible with older versions of cpe (cpe 2.2) and consists of a list of attribute-value pairs that there is no designated order for each pair. Well Formed cpe names begin with the word wfn and include a sum of data that can be used only once per entry, such as

- Part :this attribute can take of these three values
    - a:refers to applications
    - o:refers to operating systems
    - h:refers to hardware apparatus
- vendor: identifies the manufacturer of the product
- product: refers to the title/name of the product
- version: it characterizes a specific version of the product
- update: specifies a distinct release of the product (update, service pack)
- Edition: this characteristic is no longer in use, so it contains values to be compatible with older cpe versions, otherwise its value is ANY.
- language: it defines the language the user interface supports
- sw_edition: characterization about aiming a specific group of users
- target_sw: identifies the requirements of the operating system,in which the application is executed
- target_hw: this attribute describes the architecture of the operating system, in which the product is executed
- other: in this section there is information in a more generalized manner that relates to the product and do not conform with the rest attributes.

There are two special values that characteristics may have, defined as logical values such as:

- ANY, if there are no limitations about the value that can be assigned to
- NA, meaning not applicable and this is used when there is no significant importance or legality for that attribute.

Also, inside the attribute's value there is no whitespace used, but only underscore to separate words, par example product="internet_explorer".
*wfn example:*

wfn:[part="a",vendor="hp",product="insight_diagnostics",version="7\.4\.0\.1570",update=NA,edition=ANY,language=ANY,sw_edition="online",target_sw="win2003",target_hw="x64",other=ANY]

Functions can be used in well formed names in order to designate binding and unbinding proceedings likely:

- new(): there are no parameters passed in this function. It returns an empty WFN with no result pairs
- get(): two arguments are passed in this function. The first one is the well formed cpe name and the second parameter is the desired attribute value of that specific cpe to return. In case there is no specified value for the second parameter, the function returns ANY
- set(): this function takes three parameters and is used to alternate the attribute's value inside the wfn attribute - value pair. The first argument is a wfn entry, the second is the attribute for that wfn and the third one is a value. As a result, these function returns a new instance of the wfn, modifying or deleting attribute values.

To continue, function examples are represented below, to figure out their purpose.

- get(wfn,a):
    - get(wfn:[part="a",vendor="adobe",product="flash_player"],vendor) returns adobe
    - get(wfn:[part="a",vendor="adobe",product="flash_player"],version) returns ANY

- set(wfn,a,v):
    - set(wfn:[vendor="broadcom"],update,ANY) returns the well formed cpe name with attribute-value pairs vendor="broadcom",update=ANY
    - set(wfn:[vendor="dell",vendor,nil]) returns an empty wfn, wfn[]

### 2.1.2 CWE

The acronym CWE stands for Common Weakness Enumeration and contains a list that refers to software and hardware security flaws [4]. Also, the CWE list has a set of rules, so to be constructed and written, so that finding a specific weakness may be easier. These set of rules compose a descriptive language that can be used and maintained. Furthermore, CWE encourages developers and security specialists to inspect weaknesses in existing software and hardware implementations, estimating the efficiency of tools that aim these weaknesses, create a common set of standards to identify weaknesses as well as eliminating the appearance of weaknesses in software and hardware during the development. This CWE inventory is ordered by the significance of each security gap and its intended is to eliminate the faults, the bug errors and every kind of possible security vulnerability that originates from that type of flaws, during the development of the software or the hardware. Moreover, there are related attack patterns to some of the weakness entries. The CWE list is developed by the general community.

#### CWE List

Common Weakness Enumeration List consists of weaknesses in Software and Hardware development. There are smaller lists that include only the software and the hardware weaknesses accordingly, as well as a research list that defines each weakness in a theoretical manner. Furthermore, there are entities that categorize each entry from an abstract way to a more specific one, such as:

- *Pillar*: Is the most generalized type of weakness and represents common behavior for all class/base/variant weaknesses connected to it.

- *Category*: is a group of CWE entries that share the same characteristics or attributes.
  *Class*: is a weakness that is independent of any specific programming language or technology.
- *Base*: is a more specified weakness that provided information helps in detection and prevention.
- *Variant*: refers to a distinct weakness that is connected to a specific product, language or technology.

The CWE List consists of several group views in accordance to its content such as:

- *Software List (CWE-699)*: Demonstrates weakness approaches that exist in all stages of software development, in order to figure out flaws. Each CWE entry belongs to a CWE Category; it has an identification number (ID), a comprehensive description, the phase and the programming language in which the weakness is spotted, the consequences, the possibility to exploit the weakness (low –medium –high), attainable mitigations, examples that demonstrate the weakness using a programming language, a table of other CWE categories that include that specific CWE entry and some external references to research /white papers.
- *Hardware List (CWE-1194)*: Depicts weakness approaches in hardware design, including categories that hardware designers are well acquainted with. Each category comprises a group of entries that have the same characteristics. In addition, each entry in the Hardware List consists of description (summarized and extensive), the programming languages, the Operating Systems and the architecture techniques used to occur this weakness , an expressive example, mitigations, detection methods and related attack techniques as well as external references .
- *Research Concepts List (CWE-1000)*: Illustrates the theoretical divergence of a weakness, including their reliance, arranged in a generalized way of their behaviors. This way is considered to include every weakness within a CWE. In Research Concept List, CWEs are grouped by a more abstract type of weakness, called Pillars.

#### CWE Example

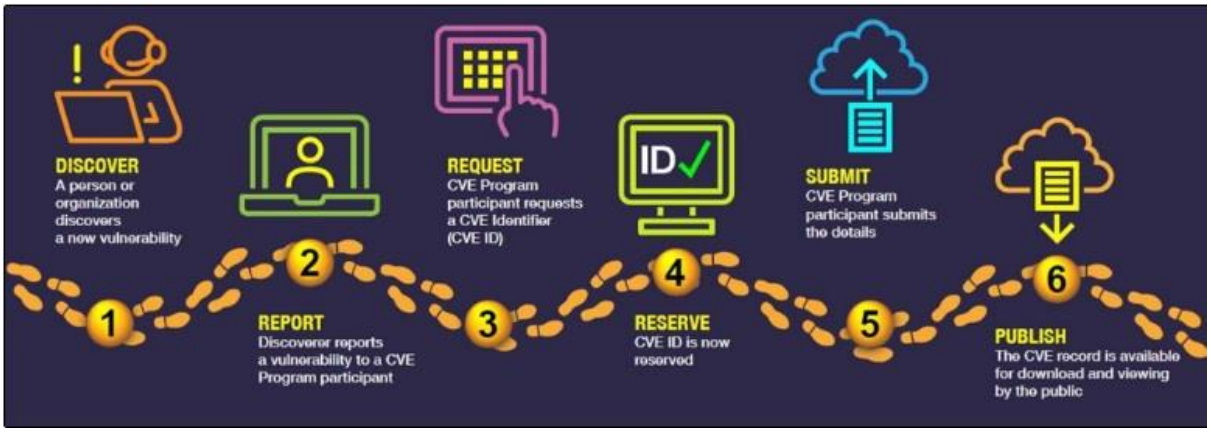
During the development of a software application, a script or a function can be threatening for the operating system it is going to be executed. That leads to the CWE-676 which defines the weakness of the use of a potentially dangerous function during the execution. The description of that CWE aims to the vulnerability issues that may occur, if a portion of a code is used inappropriately. Furthermore, there are potentially points in which this weakness may occur, such as in architecture and design phase or during the implementation of the code, as well as the common programmatic languages used, for example C or C++. Subsequently, there are examples with script code in one of the common programmatic languages, in order to figure out this weakness clearly.
There are references to vulnerabilities (CVEs) where this specific weakness links to, including the cve identification number and a brief description for each vulnerability, a mitigative solution where specific functions are not allowed to be used and / or the use of analysis tools and compilers to detect restricted functions or libraries. Additionally, detection techniques are mentioned summarily, showing how much effective each technique is, and related CWE categories that reference this weakness.

### 2.1.3 CVE

The acronym CVE stands for Common Vulnerabilities and Exposures and is a list of publicly disclosed cyber security vulnerabilities that can lead to negative consequences of lacking integrity, confidentiality

and availability [5] [6]. Therefore, CVE is a standard, which is followed by all the security vendors to list and define openly revealed vulnerabilities. Each record in the CVE list has its own identifier (id) which consists of the word CVE, the year that it was published and a number that a CVE Numbering Authority (CNA) assigned to it (CVE - Year - Number) , so that each identifier references a specific vulnerability. The term vulnerability refers to a flaw, which can be exploited to succeed unauthorized access to a system or network and the term exposure represents a mistake in software development that allows adversaries to gain no permitted access to a system or network. Also, there is a description that defines the vulnerability problem and there are three phases for each CVE record:

- *Reserved*: It is the primary state for a CVE entry, where the correlated CVE ID is being examined by a CNA , which is a CVE Numbering Authority.
- *Published*: After examining a CVE, the data are published by a CNA related to the appropriate CVE ID, including a description of the vulnerability and at least one reference publicly accessible.
- *Rejected*: Refers to an invalid CVE entry, which remains in the CVE List, in order to be recognized by the CVE users.




**2 CVE Record Lifecycle [7]**

Therefore, when a new vulnerability is discovered, a descriptive report is sent to a CVE Program participant, in order to assign an identification number to that entry. Then, the CVE ID is in reserved state meaning that the vulnerability is being analyzed by a CVE Numbering Authority (CNA), before a CVE participant concentrates all the information and the vulnerability is publicly accessible.

The Common Vulnerabilities and Exposures List aids rapid data connection regarding vulnerabilities among several information sources that comply with CVE. Therefore, CVE Records are used in cyber security products and implementations, such as firewalls, intrusion detection systems and security consultants.

## CVE List

The Common Vulnerabilities and Exposures List is a directory with all the Vulnerability records that are discovered and reported to the CVE Program. Although CVE List includes reported vulnerabilities, it contains some descriptive information including a unique identification number, which has the cve word followed by the year this vulnerability was assigned along with another integer separated with dash, a brief description of the cause of the vulnerability, a section of references including external URL's, the assigning CVE Numbering Authority, the date that this vulnerability was assigned and its current phase. As a result, each CVE Record is connected to the National Vulnerability Database

(NVD) where additional information about the severity, the confrontation and the software versions this vulnerability refers to, are displayed.

### 2.1.4 OVAL

Open Vulnerability and Assessment Language (OVAL) depicts and connects all the publicly available security content with the available security tools and services [8] [9] [10]. Also, it is written in Extensible Markup Language (XML) in order to standardize the three main assessment extensions (schemas) such as:

- Configuration information of testing systems: which is an OVAL System Characteristics extension to represent the state of the machine
- Analyzing the vulnerability of the specified entity: Which is an OVAL Definition schema for representing a specific machine state
- Reporting the assessment results: in the OVAL Results extension in order to report the results of the appraisal.

In addition to the XML documents mentioned above, there are several repositories where these OVAL XML files exist, such as the OVAL Repository supported by MITRE foundation. The MITRE's OVAL Repository is the main locus of the OVAL Community where each software vulnerability, configuration issue or patch is ascertained to exist on a system.

#### OVAL Definitions

The fundamental step to determine the vulnerability of a system is through the OVAL Definitions, due to the fact that they include definition meta data such as the status of definition, the references on which the definition is based on, like the appropriate CVE name, a summary of the specified security issue, as well as the contributors that developed the definition [9]. Furthermore, additional information is comprised stating the operating system of the entity, the suspicious file containing the vulnerability, the application version and whether the application is being executed or not.

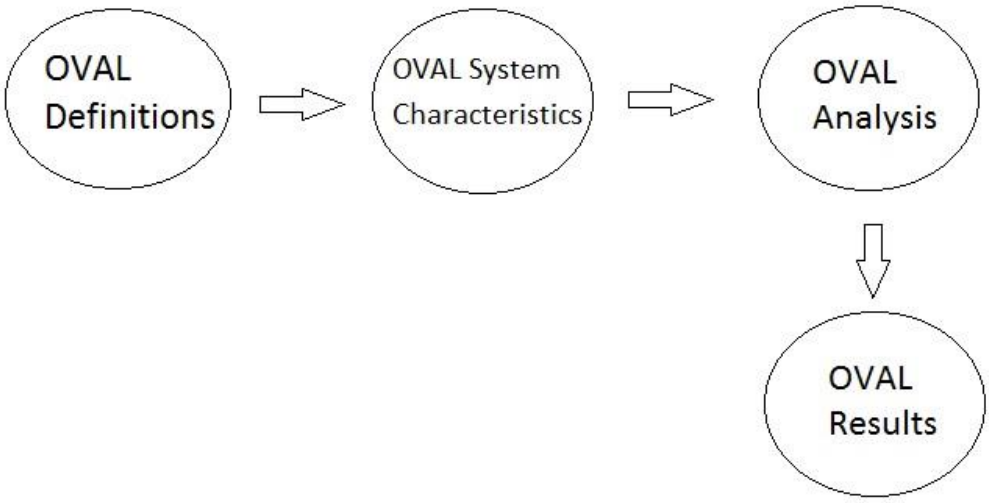
Therefore, OVAL Definitions can be processed by machines, determining the vulnerability of the entity. There are four categories of OVAL Definitions:

- *OVAL Vulnerability Definitions*: figure out the vulnerable aspects on a system.
- *OVAL Compliance Definitions*: A set of checks that define the security policy compliance.
- *OVAL Inventory Definitions*: A set of tests that define the installation of software on a machine.
- *OVAL Patch Definitions*: Compatibility patch definitions for a specific machine.

#### Use Case

The three fundamental components of OVAL Language cooperate with each other throughout a standard vulnerability assessment process, using the OVAL Interpreter for evaluating the OVAL Definitions.

To begin with, specific definitions are generated through configuration policy documents so as to create OVAL Definitions. These OVAL Definitions are constructed in ways that indicate the needed information that has to be gathered from the system, creating the OVAL System Characteristics. Next step is the analysis process. The OVAL Definitions and the OVAL System Characteristics are compared to figure out the vulnerability state of the system examined. Furthermore, the OVAL Results of the analysis are configured as an OVAL Results document.




**3 How OVAL works**

### 2.1.5 MAEC

The acronym MAEC stands for Malware Attribute Enumeration and Characterization is used to eradicate the ambivalence in malware descriptions in order to minimize the reproduction of malware analysis endeavors [11][12]. It is established through defense community and it is a structured language for distributing accurate information as regards malware specifications and eliminating the dependency related to malware signatures. As a result, malware research is improved and duplication of malware analysis is decreased, making the deployment of the countermeasures a faster process and maximizing the effectiveness of the previously observed malware occurrences.
Furthermore, it allows external references to attack techniques (ATT&CK Framework) for the specification of the used technique when implementing Malware Behavior, as well as using elements of the STIX specification language; as a consequence, detailed information about malware analysis and malware background, such as files and networks, are provided.

#### MAEC Language

The Malware Attribute Enumeration and Characterization Language includes standards and methods to share information about malicious software such as malware actions, malware instances, malware families. The simplification of complex components and the deprecation of any factors that are not in use, simplifies the complexity in analyzing malware behavior. A JSON format is being used to clarify the information for each element and for better integration with different types of applications.

#### Language Data Types

Apart from the common data types that are used in several programming languages, such as Boolean, Integer, float, list etc, there are some types that top level objects use, so as to better describe the information after a malware analysis is executed. These additional data types are:

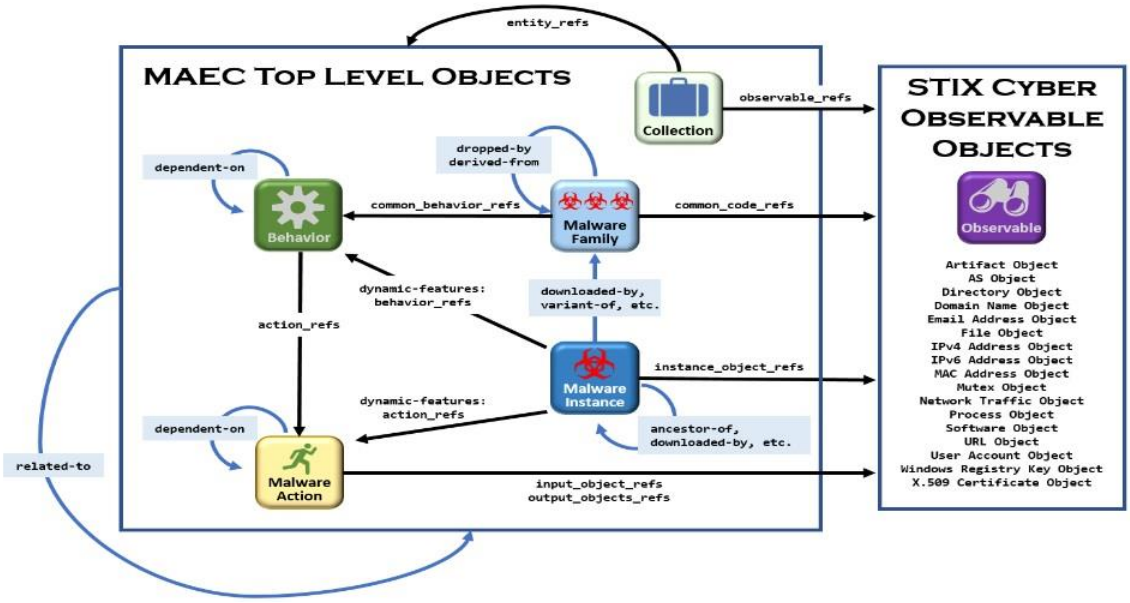
- *Api Call*: The representation of malware actions that define the address of the call in hexadecimal order, the return values of the call, the parameters that the function uses and the function name.

- *Analysis Metadata*: A sum of data that characterize output analysis on a malware instance including if the analysis is automated, the start and finish time, the type of the analysis, comments of the user performing the analysis and the tools that this implementation uses, references to other data or reports, as well as a decision if the binary was found to be malicious or not.
- *Binary Obfuscation*: Represents explanatory data about methods the binary may be obfuscated with namely the used method to obfuscate the file, the encryption algorithm, the entry point address and the name of the packer.
- *Capability*: Related information about the persistence and the tolerance in analysis this binary may have containing a name for the capability which indicates if the malware is able to bypass or debilitate access control mechanisms, if it is designed to obfuscate debuggers or if it has the ability to avert is execution inside an emulator. Furthermore, there are external references to attack tactics that associate with this capability, attributes and ID Behaviors that connect to this capability.
- *Dynamic Features*: Apprehends the Behavior IDs and the Action IDs of the malware instance that are detected through several methods such as static analysis or reverse engineering, identifies and records network traffic executed by a process
- *Field Data*: A set of information as regards the time the malware was first observed, the vectors that distributed it and the date and time it was last captured.
- *Malware Development Environment*: Contains information about the development of the malware instance, namely the tools that are used and the debugging files that associate with.
- *Name*: Contains the name of the malware, the external references and the accuracy value of the assigned name.
- *Relationship Distance*: Depicts the difference (distance) between the source reference and the target reference of the malware figuring out the differences of the two sources.

### MAEC Entities

The Malware Attribute Enumeration and Characterization entities define a total of Top Level Objects that have several properties and existing relations among them. For instance:

- Behavior: Specifies a particular objective behind a code snippet, as implemented by a malware.
- Collection: Comprises a sum of MAEC objects that relate to each other.
- Malware Action: Contains information about the formation of a specific file on the hard disk and/ or the launching a port and are collected by dynamic analysis tools.
- Malware Family: Consists of a sum of malware occurrences that are related by similar derivations and origins.
- Malware Instance: is a part of a malware family that has its distinctive type, id and references to the Cyber Observable Objects that define the binary code.




**4 MAEC Top Level Objects [12]**

### 2.1.6 NVD

NVD is an abbreviation of National Vulnerability Database and it provides information about software security gaps, product compositions and implications assessment [13]. It is a collection of vulnerability related metadata to use a feed of information that comes from cve. Therefore, National Vulnerability Database hands out a searchable interface to apprise about the type and the severity of the vulnerability. When a new CVE is created, NVD analyzes every vulnerability that is published by CVE and identifies what weaknesses the given vulnerability is exploiting and the number of vulnerability characteristics, using the CVSS (Common Vulnerability Scoring System). If additional information is provided later in time, the CVSS score may change, helping in the prioritization of vulnerabilities and risk management. In addition, NVD associates what products are vulnerable through the investigation of the information provided. This information empowers automation of vulnerability management and security appraisal. Therefore, through National Vulnerability Database other frameworks are connected such as Common Platform Enumeration (CPE), Common Vulnerabilities and Exposures (CVE) meaning that CPE and CVE frameworks can be searched through NVD. Furthermore, the National Vulnerability Database uses the Security Content Automation Protocol (SCAP), which is a set of specifications that assist in security automation, providing information about the vulnerable versions of software.

#### NVD Example

Through the provided searching interface, the user is able to search for vulnerability information about the desired software or operating system. For a generalized search, there are several CVE's in the result table. The results begin with the most recent published CVE entry and continue to older ones. The preview of each result contains the vulnerability identification number along with a CVSS score, if it is available and a short description. After selecting the desired result, a CVE ID is provided on top and below that a description of the asset / function that causes the vulnerability. Also, the affected versions of the software are mentioned and a severity score provides the importance of the vulnerability. Furthermore, the severity score is calculated based on the Common Vulnerabilities Scoring System (CVSS), in which it provides the Base Score along with the Vector. The displayed information about the vector shows more specific information relative to the distinct factors of the Base group. To continue, there is an informational table where each row refers to Solutions and Tools that correlate

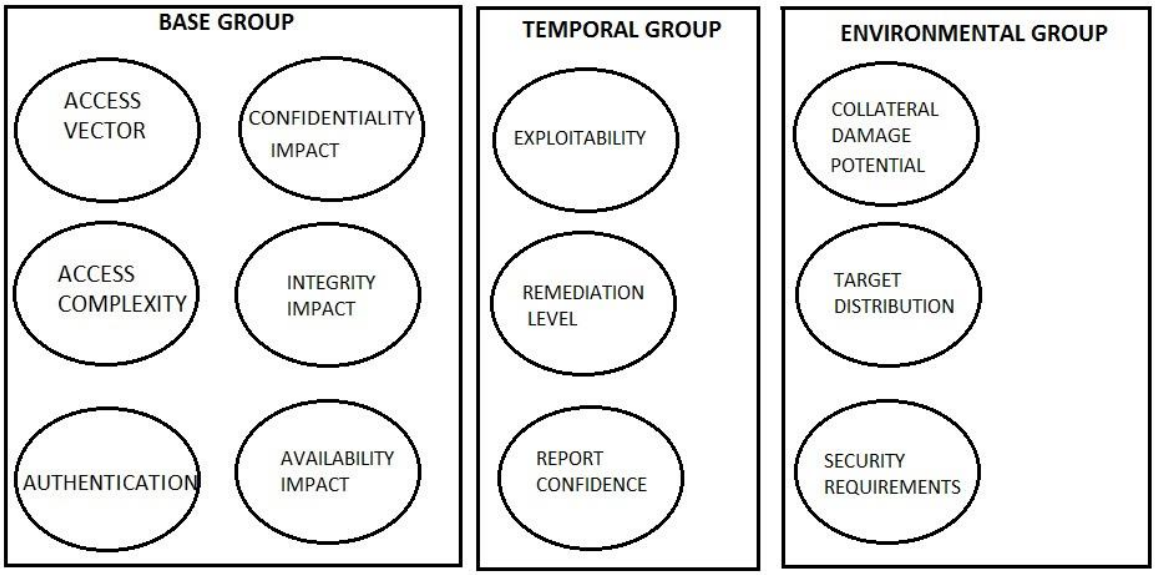
with the vulnerability, in the form of hyperlinks. In addition, the last two pieces of information refer to the weakness enumerations and platform enumerations accordingly.
More specific, the weakness table includes the weaknesses that are related to the searched vulnerability. These weaknesses consist of the Common Weakness Enumeration identification number (CWE-ID), the name of the weakness and the source that found that weakness. As regards the platform enumeration table, it displays the platforms/applications/operating systems that are affected by this vulnerability, including the appropriate version. That information follows the CPE specifications, where additional data can be extracted using the well-formed name (WFN), such as if it is application/operating system, the vendor, the product and the version of the vulnerable product.

### 2.1.7 CVSS

The Common Vulnerability Scoring System (CVSS) is a public and open source framework that depicts the attributes and the significance of the vulnerabilities [14] [15]. It is constituted by three main groups: Base (includes the inherent attributes of vulnerability), Temporal (contains the variable attributes of vulnerability) and Environmental (the exclusive feature the vulnerability has over a specific operating system/environment). Each one of these groups has a score number that ranges from zero to ten, a Vector and a brief description of the attributes used to calculate the final score. As a result, CVSS contributes to the creation of a vulnerability policy, figure out the specific attributes that make each vulnerability important to cope with, over other vulnerabilities with less score. CVSS is divided in three main categories:

- *Base*: the essential aspects of a vulnerability that do not change over time, including Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrity Impact, Availability Impact
- *Temporal*: Temporary characteristic parameters that alternate over time, such as Exploitability, Remediation Level, Report Confidence
- *Environmental*: includes all the parameters that are exclusive to specific parameters, such as Collateral Damage Potential, Target Distribution, Confidentiality Requirement, Integrity Requirement, Availability Requirement

These three categories help in the definition of the primary characteristics of each vulnerability, providing a better understanding of the vulnerability. This approach contributes to accurately mitigate the underlying risks of the vulnerability.




**5 CVSS Metric Groups**

For each of the three unique categories, apart from the score (0-10), an additional text exists that includes the values for the categories, in order to explain the scoring number for the appropriate category. As regards the temporal score, temporal metrics combine the base score to calculate the temporal score. Similarly, to calculate the environmental score, the temporal score along with the environmental metrics are combined.

### 2.1.8 Metric Groups

#### Base Metric Group

As regards Base metrics that consist of Access Vector, Access Complexity and Authentication that apprehend the way of the vulnerability is approached and if additional parameters are obligatory to exploit it [16].

- Access Vector (AV) :represents the way the vulnerability can be exploited, having the following values:
  - Local (L): the adversary has local access, meaning either access to the local account or physical access to the vulnerable system.
  - Adjacent Network (A): the attacker has access to the local network, either from a local IP address, Bluetooth access or wifi.
  - Network(N):The vulnerable entity can be exploited through the general network, meaning that no local access is required.
- Access Complexity (AC): represents how complex the attack needs to be in order to exploit the vulnerability. It has the following values:
  - High (H): particular accessibility conditions are made by the attacker
  - Medium (M): The accessibility conditions are not highly specialized.

    - Low (L): There are no particular access conditions and the vulnerable entity requires access to several systems and/or users. That means that the exploitation is quite complex.
- Authentication (Au): shows how many times the attack actor should authenticate in order to successfully take advantage of the vulnerability. As a consequence, for few authentication steps, the vulnerability score arises. The values that the Authentication metric can have are three:
    - Multiple (M): The authentication process consists of more than 2 times.
    - Single (S): Only one authentication is required to exploit the vulnerability
    - None (N): No indispensable authentication is needed to misuse the vulnerability.
- Confidentiality Impact (C): computes the confidentiality consequence when the vulnerability is manipulated and it may have the following values:
    - None (N): Does not affect the system's confidentiality
    - Partial (P): The adversary has limited control over files or obtained information.
    - Complete (C): The adversary has full control over the obtained information, such as memory, files etc.
- Integrity Impact (I): shows the stability of the system when a vulnerability is exploited. There are three values for this metric listed below:
    - None (N): The system's integrity is not affected.
    - Partial (P): The attacker has a limited access control over system files. Therefore, the system's alternation is partial.
    - Complete (C): The adversary may alternate any file or information on the affected system.
- Availability Impact (A): depicts the accessibility on system's resources, where they can be used during the attack, affecting the availability of the entity. Availability Impact metric has the following values:
    - None (N): The system's availability is not affected.
    - Partial (P): A portion of the system's resources are affected.
    - Complete (C): All the system's resources are affected by the attacker, making the exploited system unavailable.

### Temporal Metric Group

Temporal metrics mention these parameters that can change over time. This category consists of the following factors [16]:

- Exploitability (E): The existence of public techniques or code that can be used to exploit the vulnerability, resulting ease exploitation. Also, for an easily exploited vulnerability, the higher the score.
    - Unproven (U): Non existence of exploitation code or technique
    - Proof-of-Concept (POC): A technique or a code exists to exploit the vulnerability, but it cannot be implemented on most systems or it needs several alterations by an experienced adversary.
    - Functional (F): A script code is functional and it can successfully take advantage of the vulnerability where it exists.

    - High (H): There is extant code that exploits the vulnerability, even if it is being delivered to the vulnerable entity.
    - Not Defined (ND): This value suggests avoiding this particular metric.
- Remediation Level (RL): includes all the actions to reverse or stop the exploitation. The possible values for this parameter are:
    - Official Fix (OF): The publisher of the vulnerable system/application has an official solution to fix this vulnerability either by a patch or by an upgrade.
    - Temporary Fix (TF): The official fix offers interim remediation.
    - Workaround (W): There is an unofficial solution available.
    - Unavailable: No existing solution available.
    - Not Defined: It is a value that does not affect the score, meaning that this metric should be avoided.
- Report Confidence (RC): includes the technical details and the existence rate of the corresponding vulnerabilities, including the following values:
    - Unconfirmed (UC): There is no official source for the reports, as well as there is no validation for the reports.
    - Uncorroborated (UR): There is no official source and there is conflict about the technical details.
    - Confirmed (C): The official source has accepted the vulnerability of its product.
    - Not Defined (ND): This value does not affect the score. It means to ignore this metric.

### Environmental Metric Group

As regards environmental metrics group, there are specific characteristic attributes of a vulnerability that may be affiliated with a more specified computer infrastructure, networking or other devices. This group has the following attributes [16]:

- *Collateral Damage Potential* (CDP): This indicates the amount of loss, either economical or productive, the device's damage and the potential false functioning of equipment. For higher score, the bigger the damage. Potential values are:
    - None (N): No prospective loss of physical damage, equipment or productivity loss through the exploitation of the vulnerability.
    - Low (L): A resultant of possible loss of revenue or productivity or physical damage, after taking advantage of the vulnerability.
    - Low- Medium (LM): A slight bigger damage/loss of revenue or productivity.
    - Medium-High: A great loss of earnings or productivity as an aftermath of the vulnerability's exploitation.
    - High (H): The impact of the vulnerabilities exploitation is high and causes calamitous damage and loss.
    - Not Defined (ND): This is not affecting the score. It skips the metric.
- *Target Distribution* (TD): This attribute emphasizes in the percentage of vulnerable system entities and may have the following values:
    - None (N): None of the systems entities may be affected by the exploitation.
    - Low (L): A very low percentage of the entirely environment is at risk (1%-25%)
    - Medium (M): The risk of damaging systems entities are between 26%-75%.

    - High (H): The most of the system's environment is at risk (76%-100%)
    - Not Defined (ND): It is a value to skip the metric.
- *Security Requirements* (CR, IR, AR): These parameters give the CVSS user the opportunity to customize the final score, depending on the entity that is going to be affected. As a consequence, the asset that has greater meaning of confidentiality, it can be assigned to it a greater value than the availability and integrity. Therefore, the environmental score is altered depending on the weight of the corresponding value. On the other hand, the base values do not change. The value for each one of the three are:
    - Low (L)
    - Medium (M)
    - High (H)
    - Not Defined (ND)

Each metric group has a vector. The vector is an abbreviation of all the parameters mentioned above, where the name is followed by a colon and a slash to separate the each metric, for example: AV:L/AC:M/Au:N/C:N/I:P/A:C.

### 2.1.9 CWSS

The Common Weakness Scoring System contributes a process of software's weakness prioritization [17]. CWSS represents the risks of software in a more accurate approach, taking into consideration the specific implementation this software has. Furthermore, Common Weakness Scoring System contributes to the measurable weaknesses that exist inside an application's code and therefore a prioritization of each weakness contributes to figure out which are the important weakness types to solve first. The operating mode of CWSS is divided into three groups:

- *Base Finding*: refers to the ingrained danger of the weakness, the accuracy rate of discovering the weakness
- *Attack Surface*: This category presents the obstacles that an adversary should overcome to take advantage of the weakness
- *Environmental*: this group cites the parameters of a weakness that are distinguishing for a specified system / environment.

Furthermore, each one of the three groups mentioned above, include a component that a value is assigned to it. Through these values, the score for each group is calculated. As regards Base Finding, its score varies between 0 -100 and for the other two categories (Attack Surface and Environmental) their score varies between 0 and 1. In order to find the final score of CWSS, the score of each category is multiplied to produce a total between 0-100, thus BaseFindingSubscore * AttackSurfaceSubscore * EnvironmentSubscore = Total CWSS Score

To begin with the first category, Base Finding, in which there are five factors, such as:

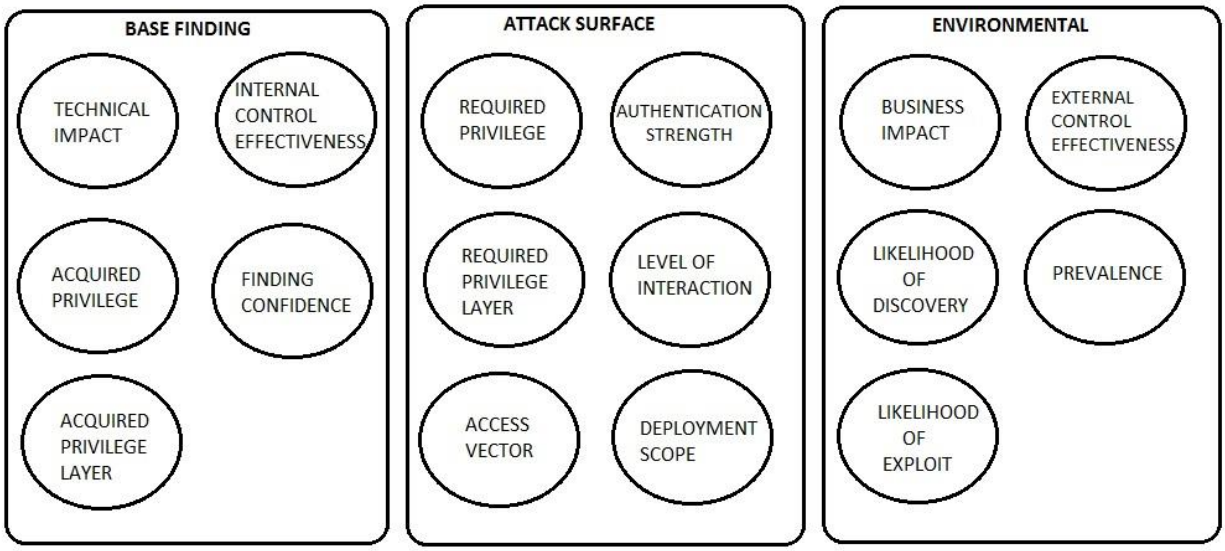
- *Technical Impact* (TI): prospective consequences that derive from a successfully exploited weakness.
- *Acquired Privilege* (AP): What privileges an adversary should have in order to take advantage of the weakness.
- *Acquire Privilege Layer* (AL): A functional layer in which the adversary gains privileges exploiting a weakness.
- *Internal Control Effectiveness* (IC): The attacker is not capable of a weakness exploitation, due to control's effectiveness
- *Finding Confidence (*FC): The certainty that an uprising matter is going to be a weakness.

The second category (Attack Surface) includes the following factors:

- *Required Privilege* (RP): The access points that the attacker should have so as to attain the weakness.
- *Required Privilege Layer* (RL): The attacker's functional layer to endeavor an attack.
- *Access Vector* (AC): The attacker's method of approach to access the weakness's content.
- *Authentication Strength* (AS): How strong the authentication process is to avoid weakness's access from adversaries.
- *Level of Interaction* (IN): What actions should be implemented by the user to permit a successful attack?
- *Deployment Scope* (SC): The amount of system entities that the weakness affects.

The third metric group is called Environmental and it includes the factors mentioned below:

- *Business Impact* (BI): The repercussion of a weakness's exploitation.
- *Likelihood of Discovery* (DI): The possibility to find the weakness by the attacker.
- *Likelihood of Exploit* (EX): The probability to accomplish a successful weakness's exploitation.
- *External Control Effectiveness* (EC): The alleviations that exist apart from the software that harden the weakness's exploitation.
- *Prevalence* (P): The weakness's frequency of occurrence.




### 6 CWSS Metric Groups

Through a CWSS vector, important information can be extracted for each factor that exists in the vector, due to the representation of the vector. More specifically, the layout of a CWSS vector has the Factor Name first followed by its value and weight. The values and the names are separated with a colon and factors separate each other with forward slash characters like:
FactorName1:Value,Weight/FactorName2:Value,Weight/FactorName3:Value,Weight etc.
In case of a vector that does not include weights for a value, then an implementation error should be raised.

## CWSS Example

The example refers to an application that it is not the main source of income and as a result the business value is medium. In addition, application is not business-critical, so the overall business impact is low and registration in the application is made through e-mail and user confirmation.
Factors from all the metric groups are presented in the form of name, weight, short description, as well as the way to calculate the final CWSS score.

Base Metric Factors:

- *Technical Impact*: High (H) 0.9 sensitive and important information may be obtained
- *Acquired Privilege*: Partially - Privileged User (P) 0.9 -> the attacker should have a user privileges, but no administrator ones.
- *Acquired Privilege Layer:* Network (N) 0.7 -> The attacker acquires privileges to access the network
- *Internal Control Effectiveness*: Moderate (M) 0.7 -> A protection process exists, but it may be bypassed.
- *Finding Confidence*: Proven Locally True (LT) 0.8 -> A particular function encapsulates a weakness, although the attacker may not reach this function.

*Base Finding Score:* [ (10 * Technical Impact + 5*(Acquired Privilege + Acquired Privilege Layer) + 5*Finding Confidence) * f(Technical Impact) * Internal Control Effectiveness ] * 4.0

[ ( 10*0.9 + 5*(0.9+0.7) + 5*0.8 ) * 1 * 0.7 ] * 4.0 = (9 + 8 + 4 ) * 0.7 * 4.0 = 58.8

Attack Surface Metric Factors:

- *Required Privilege*: Partially - Privileged User (P) 0.6 -> The user is validated but here are less privileges than administrator privileges.
- *Required Privilege Layer*: Enterprise Infrastructure (E) 1.0 The adversary should have privileges to access a server.
- *Access Vector*: Not Applicable (NA) 1.0
- *Authentication Strength*: Moderate (M) 0.8 -> The required authentication uses passwords and usernames (moderate strong methods).
- *Level of Interaction*: Typical/Limited (T) 0.9 -> The user must be convinced to perform a common action.
- *Deployment Scope*: Rare (R) 0.8 -> The weakness is present in few configurations

*Attack Surface Score:* [20*(Required Privilege + Required Privilege Layer + Access Vector) + 20*Deployment Scope + 15*Level of Interaction + 5*Authentication Strength] / 100

[20* (0.6 + 1.0 + 1.0) + 20 * 0.8 + 15 * 0.9 + 5 * 0.8] / 100 = [52 + 16 + 13.5 + 4] / 100 = 0.855

Environmental Metric Factors:

- *Business Impact*: Medium (M) 0.6 -> There is no considerable damage to business operations.
- *Likelihood of Discovery*: Medium (M) 0.6 -> Although the weakness can be discovered, but it needs a skillful adversary too implement an exploitation.
- *Likelihood of Exploit*: Unknown (UK) 0.5 Not enough information for the likelihood of exploitation.
- External Control Effectiveness: Moderate (M) 0.7 -> A protection flow is implemented, but it can be hardly bypassed.
- Prevalence: Common (C) 0.8 -> The weakness occurs occasionally

*Environmental Score*: [ (10*Business Impact + 3*Likelihood Of Discovery + 4*Likelihood Of Exploit + 3*Prevalence) * f(Business Impact) * External Control Effectiveness ] / 20.0

f(Business Impact) = 0 if Business Impact == 0; otherwise f(Business Impact) = 1

[(10 * 0.6 + 3 * 0.6 + 4 * 0.5 + 3 * 0.8) * 1 * 0.7] / 20 = [(6 + 1.8 + 2 + 2.4) * 1 * 0.7] / 20 = 0.427

Total CWSS Score = BaseFindingSubscore * AttackSurfaceSubscore * EnvironmentSubscore
Total CWSS Score = 58.8 * 0.855 * 0.427 = 21.467

### 2.1.10 EPSS (Exploit Prediction Scoring System)

The EPSS framework from Kenna framework assists in vulnerability management through facilitating a vulnerability management model [18] [19]. When Kenna VM starts there are many visualizations of data that represent the risk posture of the system or organization. Important metrics are shown such as a risk score, the mean time that has passed to remediate and graphics that represent the risk over time. This risk score can be compared with other organization's risk score and take informed decisions about the security programs by comparing the risk posture with that of related industry. Kenna is a vendor agnostic solution that can ingest and analyze security data already possessed, as it encapsulates several integrations about risk assessment tools.
Kenna allows grouping assets when data is imported, for example group all assets that include a CVE, all assets that have an open ticket, as well as group assets in a hierarchy that allow better asset management. For each Risk Meter there is a score that ranges between 1 and 1000.
The Risk Meter module provides information about scores assigned to the assets as well as each unique vulnerability within that asset group. Risk Scoring assists in understanding the risk posture and the actions/measures to reduce risk impact.
Kenna uses machine learning and real word data science to process and analyze data from several sources such as threat and exploit feeds. This data combined with internal security information allows Kenna to assign vulnerability scores for each entity within the organization. Furthermore, it prioritizes the vulnerabilities that should be remediated first and the specific impact each action will have on each entity's risk posture, based on the infrastructure and operations to each organization.

Common features in Kenna:

- *Asset Status:* In Kenna framework, assets include the entities and configuration files that are linked to one or more connectors. These assets include zero or more vulnerabilities that are discovered and reported on the platform. In addition, some assets are reported as active or inactive. These active entities are inspected by Kenna connector within a time threshold, and that because only active assets are taken into consideration for calculating the risk score. If that threshold time is over, then entity's status changes to inactive. Therefore, the inactive parts are not calculated to the risk score.
- *Vulnerability Status:* The vulnerabilities in Kenna framework may have one of four values, such as:
    - Open: The open vulnerabilities have an active score that affect the overall asset score.
    - Closed: The vulnerabilities with status closed are mitigated by the platform, thus they are not counted in the asset score. Also, if that vulnerability is found on a new connector, it is going to be opened, but if the administrator changes the status to close, and then Kenna does not supervise that vulnerability automatically.

  - Risk Accepted: This group of vulnerabilities is not going to be fixed and they do not affect the score of a system's entity. They exist but they are ignored.
  - False Positive: The status of false positive refers to a wrong found result that it was falsely considered as vulnerability.
- *Risk Meter/Asset Searches*: As regards the Risk Meter, it includes a group of assets that provide a range of capabilities inside the framework. The categorization of the assets may be executed using inherent functionality or implementing custom functions with query builder.
- *Asset prioritization*: For the prioritization of the assets in Kenna, there is a value assigned to each one that represents the necessity for a vulnerability to be remediated before others. This price varies between 1 and 10, where the default value is ten. In addition, asset prioritization value contributes to the asset score.
- *Scoring in Ken*na: There are three score meters in Kenna framework, that contribute to a variety of functionalities, such as:
  - Risk Score: This type of score refers to the vulnerabilities within Kenna that have a CVE-ID and are dynamically scored based on threat and exploit databases, using machine learning functionality. The score varies between 0 - 100.
  - Asset Score: This type of score represents the highest vulnerability score for the asset and it varies between 0 - 1000. Furthermore, vulnerabilities with status 'closed' or 'risk accepted' are not taken into consideration for the calculation of this score. Also, Kenna adds 200 additional points to vulnerable assets that they do not have IP address or if they exist outside of the private network.
  - Risk Meter Score: The Risk Meter score for a group of assets is the average of all non-zero assets that make up that Risk Meter.

### 2.1.11 Threat - Attack Libraries

#### CAPEC

CAPEC (Common Attack Pattern Enumerations and Classifications) is a catalogue of attack patterns that explains how to take advantage of known (public) vulnerabilities in software / applications, surpassing the obstacles that may occur [20] [21]. Therefore, CAPEC list describes the execution flow that takes place during the attack and provides instructions about how to understand and eliminate the attack's effectiveness. It enumerates and categorizes the Attack Patterns in order to be recognized and understood.

#### CAPEC List

Capec is organized in two main ways called Views:

- *Mechanisms of attack:* There are nine categories for this view such as:
  - *Engage in Deceptive Interactions:* Adversary deludes the target, convincing the target is interacting with another organization. These types of attack techniques are often recognized with the term spoofing.
  - *Abuse Existing Functionality:* The change in the functionality of an application, leading to malicious results and even affecting the proper execution of the application.
  - *Manipulate Data Structures:* The exploitation of the system data structures in order to alter the usage of the data.

  - *Manipulate System Resources:* The adversary has the advantage to violate the target's system resources, modifying software and/or hardware integrity.
  - *Inject Unexpected Items:* The submitted data to an application violates its functionality and manipulates the application's behavior, implementing unwanted steps.
  - *Employ Probabilistic Techniques:* The exploitation of infrequent security gaps.
  - *Manipulate Timing and State:* The execution of malicious code under certain application states or timing, operating actions and accessing directories that would be forbidden.
  - *Collect and Analyze Information:* This attack technique refers to the collection of all kinds of information from the targeted machine, including active querying and passive observation.
  - *Subvert Access Control:* The attacker takes advantage of the authentication and authorization weaknesses that exist in a system altering its functionality and its accessibility to data.
- *Domains of attack:* There are six categories for this view such as:
  - *Software:* This group of attacks aims the attention at software applications exploiting the weaknesses in the applications design or implementation.
  - *Hardware:* The hardware category of attack patterns takes advantage of hardware components targeting the chips, the device ports, the motherboard and all the parts of the computer / embedded system.
  - *Communications:* This category targets the communication protocols that are used in order to block or manipulate transactions.
  - *Supply Chain:* This group technique focuses on the interference of the supply chain focusing on the control of the hardware system, the software as well as the services aiming at sensitive data interception, disorganization of critical operations.
  - *Social Engineering:* These attack patterns are used against users (people) to confess sensitive data and confidential information, giving the adversary access to computer systems or facilities.
  - *Physical Security:* This attack pattern focuses on the bypass of system's physical security.

Each of these views is a depiction of attack patterns related to a specific advantageous position the attackers might have, beginning with the appropriate Category followed by Meta attack / Standard attack patterns and end with detailed attack pattern.

- *Category*: Each category refers to a group of attacks that share a common attribute.
- *Meta Attack Pattern*: Describe a generalization of a particular attack technique, aiming at the comprehension of the attack technique.
- *Standard Attack Pattern*: It is a more specific plan of attack, which is a fully functional attack pattern, providing information about how the technique accomplishes its essential target.
- *Detailed Attack Pattern*: Consists of a definitive methodology, referring to distinguish software and operating systems, composing integrated steps to accomplish a successful attack.

### ATT&CK

To begin with, ATT&CK is an acronym for Adversarial Tactics Techniques & Common Knowledge and lays out the behaviors of adversaries representing the actions and the techniques that are used to accomplish a tactical objective that are documented [22][23]. The ATT&CK Database creates a common vocabulary that analysts can use to communicate using the same terms and minimizing the friction among the vendors, consultants and customers that study them. Also, there are three technological fields that ATT&CK is separated according to the system the adversarial actor targets, such as:

- Enterprise: Represents enterprise networks and cloud systems
- Mobile: Represents mobile devices
- Industrial Control Systems: Includes the devices, the systems, the networks and the controls used in an industrial process.

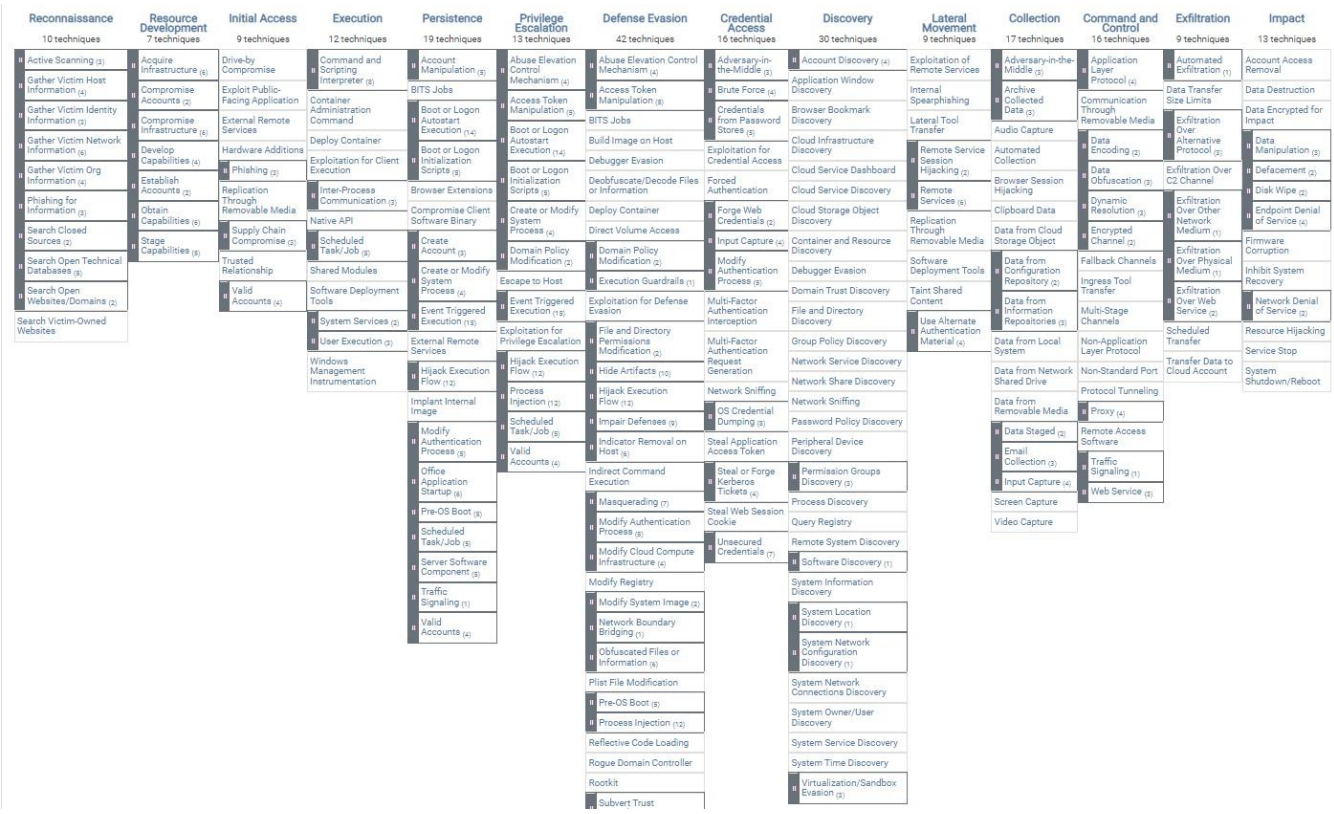
In all the above technological fields there are techniques and tactics that an attacker uses according to each domain. Some of the techniques refer to the procedures before executing an attack, such as reconnaissance and requirements gathering. Some techniques and sub techniques can apply to several systems.

There are 14 tactics that categorize each action and below them there are techniques and sub-techniques that each one of them is a subset to the high level techniques, containing a more detailed approach of the appropriate technique. Also, not all techniques comprise sub-techniques. The fourteen tactics are:

- Reconnaissance
- Resource Development
- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact

Every tactic is a separate category that includes a description representing briefly the type of techniques and sub techniques that accommodates. Also, an attack actor may execute several techniques and procedures to accomplish his/her purpose, so a deeper understanding of that behavior may be helpful for implementing better defensive actions.

Apart from the categorization of tactics and techniques, the ATT&CK framework lists all the publicly documented threat groups and intrusion sets under the Group List. The Group List contains information about the name and an identification number of each adversarial group, the techniques that are used as well as a description about the purpose and the accomplished actions. In addition, the Group List relates to the Software list, which is a List that consists of malicious software that attacking groups use. As a result, software entities combine tools that are available through operating systems as well as malwares that refer to a specific platform, such as Windows or Android.




**7 Mitre ATT&CK Matrix – Enterprise [24*Error! Reference source not found.*]**

## ATT&CK MATRIX

The ATT&CK Matrix is a depiction of all the tactics, the techniques and the sub-techniques that ATT&CK framework accommodates. Through that matrix, the relations between techniques and sub-techniques are apparent and can be visualized. Some techniques have sub – techniques, some of them don't. The name of each tactic symbolizes the meaning of the approach of the adversary, likely the Defense Evasion tactic means that the attacker wants to circumvent the target's defenses. Therefore, the techniques describe these actions that should be done to accomplish an objective and sub techniques refer to more specific descriptions of the appropriate objective.

### Structure of Techniques Sub –Techniques

When a Technique is selected, there are segments that provide information about the implementation of the technique, actions that are reported to be found under real circumstances, some are related to CAPEC entities and some information about detection and mitigation procedures. In a more detailed approach, there is:

- Name: the name of the technique
- ID: the unique identification number, starting with capital T
- Sub-Techniques: The sub techniques that a technique comprises, within a drop down menu
- Tactic: The tactic that the specified technique refers to
- Description: A brief presentation of the technique, what purpose is meant to accomplish and several variations that adversaries have used. Also, there are external references to research papers and articles that explain the provided information in a more detailed approximation.

- Platform: The platform tag is the operating system or the application that the attacker aims. A variety of techniques refer to several platforms that can be implemented.
- System Requirements: That piece of information explains the demands the adversary should have or the machine state that should be in, in order to perform the technique.
- Permissions Required: The minimum permissions the attacker should obtain alluding to the implementation of the technique
- Effective Permissions: The accessible permissions the adversary has obtained by implementing the technique.
- Data Source: A collection of information gathered by a process, including the attacker's movement to implement a technique or a sub-technique.
- Supports Remote: Refers to Execution techniques or sub-techniques that can be applied remotely.
- Defense Bypassed: Refers to Defense Evasion techniques or sub-techniques and aims at the evasion of a defensive tool (Antivirus) or a defensive process.
- CAPEC ID: This is a link to the CAPEC entry in CAPEC database.
- Version: This is the current version of the technique.
- Impact Type: Indicates attacks on integrity or availability, as regards Impact type techniques / sub-techniques.
- Contributor: An informational donor that contributes to the development of the technique / sub-technique.
- Procedure Examples: Provide information about the usage of the sub-technique that a group has already implemented and there is documented use.
- Detection: Includes the steps, processes and external references to detect malicious actions within a system.
- Mitigation: Consists of the tools and the security concepts to prevent the successful execution of a technique / sub-technique. Each Mitigation entry is related to the appropriate attack technique, proposing generalized methods. There are 43 Mitigation entities for the enterprise field, 11 Mitigation entities for the mobile systems field and 51 mitigations as regards Industrial Control Systems (ICS).

Furthermore, each sub-technique is related to only one technique to avoid complexity. Although there are situations where some techniques use more than one tactic and as a result, several sub-techniques come under a few techniques either theoretically or practically.

### 2.1.12 Control Libraries

#### Stride Model

The Stride model assists in determining a list of possible threats to the system [25]. It is an acronym for

- Spoofing: A malicious user pretends to be a different user
- Tampering: An adversary modifies the data used by the system
- Repudiation: A malicious user can change the systems state
- Information Disclosure: An adversary can extract information that is secret

- Denial of Service: The attacker can exhaust the systems resources, so that the system is no longer functioning as intended.
- Elevation of Privilege: An adversary can increase his ability to cope with the systems resources by gaining escalated privileges in the system.

## D3fend Framework

Due to the increase of the defensive technologies, a representative tool began to format through the analyzed tactics called D3fend [26] [27] [28]. D3fend depicts the defensive techniques and categorizes them in tactics. Each entry is a link to more descriptive information.D3fend is a database framework in which strategies, methods and techniques are being shared, so that new defensive techniques can be developed against specific cyber threats. It provides information about the countermeasures related to attack techniques that ATT&CK framework has. Also, there is a graphical depiction of this linking in order to clarify these countermeasure specifications. The terminology used in D3fend is selected for avoiding confusion. Therefore, each technique has a succinct entry name. The defensive expedients may counter adversary behavior directly, or refer to a more general concept, giving the opportunity to subclass this generalization with more specific definitions.
D3fend has classified the defensive techniques in five categories and each one of these includes several others in accordance to their top category, such as:

- Harden: It is implemented before the actual attack of an attacker as it strengthens several aspects of systems parts and includes techniques such as: Application Hardening, Credential Hardening, Platform Hardening and Message Hardening
- Detect: is used as a tactic to discover unauthorized access to systems and suspicious activity by the adversary and includes File Analysis, Identifier Analysis, Message Analysis, Network Traffic Analysis, Platform Monitoring, Process Analysis and User Behavior Analysis
- Isolate: is a defensive tactic that implements methods to create boundaries inside systems in order to limit the possibilities for an attacker to have access to other system's entities.
- Deceive: aims to lure the attacker to a controlled environment called as the Decoy Environment and encapsulates two techniques: Decoy Environment and Decoy Object
- Evict: includes the techniques to expel the attacker from the system or property through the techniques of Credential Eviction and Process Eviction.

The D3fend framework is able to correlate the functionality that a product is determined to have, using defensive technique allotment. As a consequence, vulnerabilities and product divergences can be detected in a more accurate way, improving the final functionality. The efficiency of defensive techniques of software can be tested through d3fend, since the defensive techniques are related to the attack techniques in the ATT & CK database framework. This type of testing can determine the effectiveness the defensive product claims to have.
The D3fend framework aims at the development assistance of mitigation techniques and the neutralization of the attacks. The acronym D3FEND stands for Detection Denial and Disruption Framework Empowering Network Defense. It is beneficial for understanding the cyber security counteracting methods. D3fend techniques derive from publicly disclosed intellectual properties from several cyber security providers. Each cyber security provider has its own implementation for a specific complication, so each issue has several implemented techniques that each one of them has its own approach and all of them refer to the same problem.


**8 D3fend Framework [27*Error! Reference source not found.*]**

These tactics include similar techniques that have high resemblance in each implementation. Furthermore, the following techniques are segregated in two levels: the base techniques and the derivative techniques. Therefore, the techniques derive only from one base technique (tactic). As regards the defend graph, more generalized techniques are presented in top levels and the more specific ones exist below them. The five tactic categories are represented in a sequenced way, meaning that the defender should first detect an adversary in order to isolate the attacker. The purpose of D3fend is to help security architects figure out the capabilities of several defensive techniques, which are shared publicly. Tactics are the vital information the defenders take against an adversarial action.

Depending on the approach, a respective sub category is created. There are several sources that defend extracts information such as:

- *Patents*: due to scientific perspective, cyber security patents are accurately presented.
- *Existing* Knowledge Bases: MITRE Cyber Analytic Repository, ATT&CK
- Other *data sources*: academic papers, technical documentations, GitHub source code.

As regards the development of a technique, there should exist a technical document or appropriate information which sufficiently details the technological approach. Otherwise, if there is no sufficient public information about that technology, it can't be included in D3fend framework.

Therefore, the Defend framework is a database model that specifies cyber-security countermeasures for specific cyber threats, populating the circumstances that this solution would take effect.

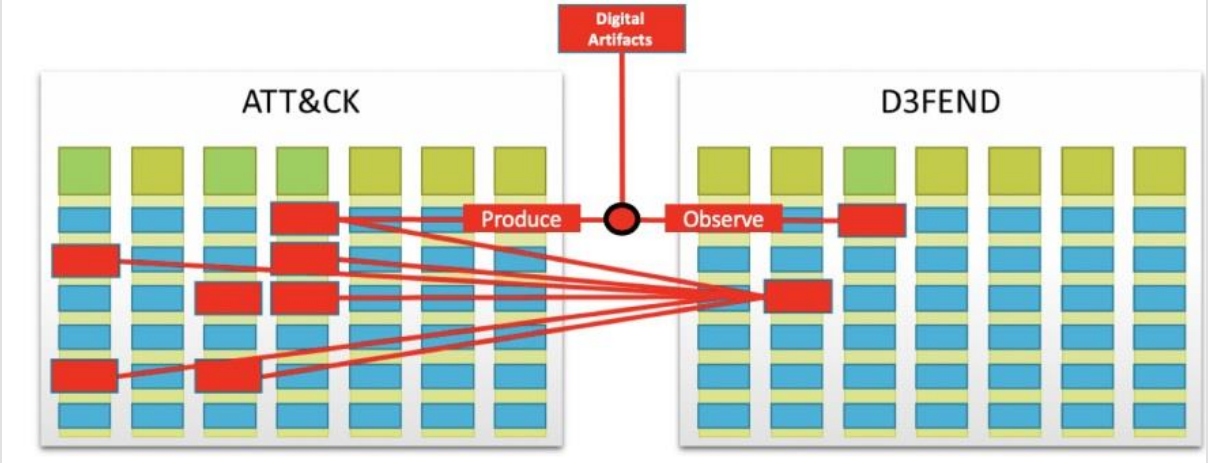
Through this information, several solutions may develop estimating vulnerabilities and weaknesses. Therefore, D3fend framework depicts the main concepts in defensive cyber security field and connects those concepts to each other.

## 2.2 LINKING FRAMEWORKS

### 2.2.1 D3fend – ATT&CK Linking

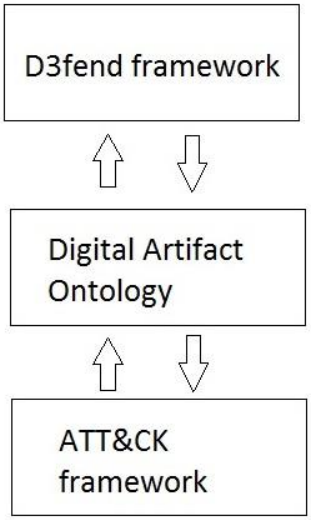
Although attack techniques were the first to develop, there were no documented defensive techniques before the D3fend implementation. Furthermore, the defensive countermeasures should be linked with the appropriate attack techniques in order to make defend more understandable.
On the other hand, a direct, hard – coded connection between countermeasures and attack techniques may lack in generalization and categorization.

The D3fend framework does not map an offensive technique directly. Instead each technique, either defensive or offensive, interacts with a digital artifact and as a consequence, a graphical structure is produced. In addition, artifacts are categorized and each category owns a sub category, in order to include specialization for the artifacts. Artifacts are the fundamental notions that exist in computer science.



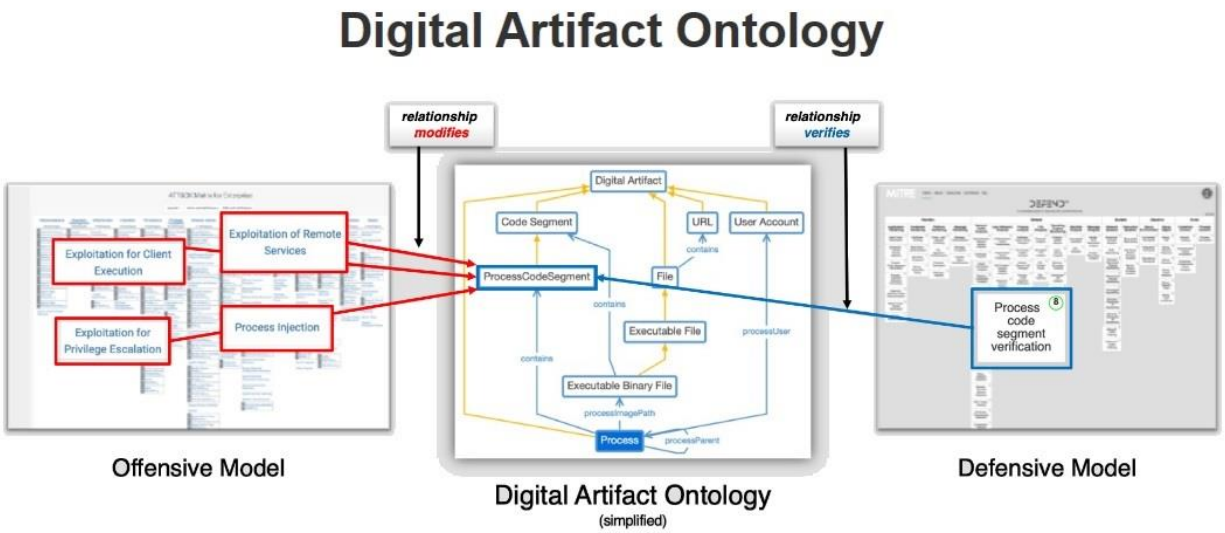

**9 Digital Artifact Ontology [26]**

From the above interconnection arises an ontology which includes all the entities that an adversary or a defender is able to interact with, representing factors of concern as regards cyber security analysis. There are currently 415 digital artifact entities in this Ontology. Each artifact consists of a short definition, a direct relation to another entity that may include as well as related techniques that allude to counter measure methods and offensive techniques. A hierarchical approach of DAO contributes to the comprehension of adjacent entries.




**10 Bidirectional connection of frameworks**

D3fend contains a sum of connections named the Digital Artifact Ontology (DAO).The Digital Artifact Ontology connects the techniques of the Defensive Model with the techniques to the Offensive Model (attack) accordingly. Therefore, each entity (artifact) interacts with one defensive technique and with more than one attack techniques. On the other hand, some artifacts do not have a specified relationship with a specific sub technique; consequently they are linked to a parent technique.

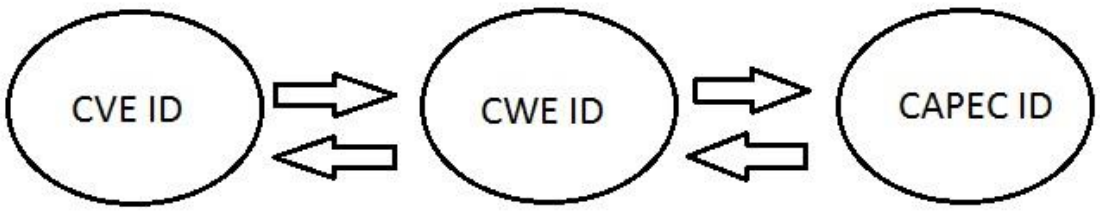



**11 Defend's DAO [26]**

### 2.2.2 CAPEC – CWE Linking

To begin with, in order to detect a malicious threat, one must first study what is the most likely case attacking a system (what are its vulnerabilities) and what are the most likely offensive techniques that could have been used. The CAPEC framework shows in detail the malicious movements, but it does not indicate the actions of the defender, which is why it is used in threat hunting, only when it is decided what information should be used.
That is, starting from the most probable software that has been infected, the according CVEs that are

connected to the CWE are checked and finally all the CAPEC attack techniques are found to take the necessary actions, such as the collection of information (log) for any suspicious actions. The CAPEC entries are used to exploit weaknesses in CWE framework and specific instances of those weaknesses are presented in CVE framework.




**12 CVE - CWE - CAPEC LINK**

As a result, the documented vulnerabilities connect to publicly known weaknesses and these exploited weaknesses are related to attack patterns through CAPEC framework. Common Vulnerabilities and Common Attack Patterns are not related explicitly. In addition, not all weakness entries abide to attack patterns, although several weakness entities refer to more than one attack method. That is, due to the diversity of the exploited weaknesses, an adversary may take advantage of the vulnerability under different circumstances, using several attack techniques and not just one. Furthermore, a single attack pattern described in CAPEC, may relate to more than one CWE entry.

## CAPEC – CWE Example

In this section, a descriptive example is used to explain the connection between the CWE and CAPEC knowledge base. To begin with, if an application executes a single factor authentication, to authenticate its users, this may lead to account compromisation, sensitive information leak leading to lack of confidentiality and integrity.

The CWE entry with identification number CWE-308 refers to the use of single factor authentication; usually the user provides a single password to authenticate. The likely of exploit is high, as there are multiple authentication methods that, if combined, still secure the application if one the provided methods fail.

The adversary has several attack techniques to exploit that weakness such as:

- CAPEC-16 *Dictionary based Password Attack*: The use of a sum of words as passwords to gain access to the desired account. Through the applications desired language, the adversary detects and uses that language. All these words are part of a formed dictionary that can be combined to gain higher possibility of exploitation.
- CAPEC-49 *Password Brute Forcing*: All the possible combinations of letters, numbers and symbols are used as possible passwords, covering the length of the specific password. The user password should be compliant to a password policy, otherwise the password brute forcing attack may be more efficient.
- CAPEC-55 *Rainbow Table Password Cracking*: This type of attack requires the hashed passwords, in order to retract the original password. It is computational expensive due to the creation of a table containing the hash chain of the original password, the extraction of the hashed passwords and the construction of a table for the numerous hash algorithms.

- CAPEC-555 *Remote Services with Stolen Credentials*: The attacker takes advantage of the remote access features of operating systems when used, such as remote desktop that allows users to log on a system remotely. The adversary extracts the required credentials and gains access through remote applications.
- CAPEC – 560 *Use of Known Domain Credentials*: The main feature of this technique is to acquire permissible credentials either by application violation through exposed configuration files that include the desired information or estimating these credentials.
- CAPEC – 565 *Password Spraying*: As regards this technique, the adversary tries a small amount of potential passwords in accordance with the target's password policy using a list of common words or easily guessed passwords. This technique is similar to capec-16 dictionary based password attack because they use a list of potentially passwords, although password spraying tries one password per user account and then moves to the next account.
- CAPEC-600 *Credential Stuffing*: Several users apply credentials on different systems and services, where this technique tries to accomplish authenticated access to supplementary services the client has. Frequent used ports are targeted, causing high possibility for the attack to happen.
- CAPEC – 644 *Use of Captured Hashes (Pass the Hash)*: Achieve a successful authentication without knowing the password, but only the hashed one. This is efficient if the service uses Lan Man or NT Lan Man session protocols. The hashed information may be obtained through spoofing, impersonating or data exfiltration.
- CAPEC – 645 *Use of Captured tickets (Pass the ticket)*: This attacking technique refers to systems that implement Kerberos as a network authentication protocol. Kerberos uses tickets to allow communication between end nodes of a network. If the adversary steals one of these keys then authentication is succeeded without the user's certificates.

Taking into consideration all the above, through a weakness several attack methods can be implemented. Securing every possible security gap that comes from a weakness, the chances of mitigating adversarial actions increases.

### 2.2.3 CVE - CWE Linking

Common Vulnerabilities connect with Common Weaknesses either if there is identification number on either entry, meaning that cve entry may have related cwe-id or cwe-id is related to a cve-id. In addition, there are vulnerabilities that refer to weaknesses through the description. In a more detailed way, the description that vulnerability has, refers to a weakness descriptively.
Furthermore, another approximation is through the National Vulnerability Database, in which the presented vulnerabilities refer to the weaknesses accordingly, using the appropriate identification numbers for each entity, if they are available. CVE and CWE are closely interconnected data sets provided by MITRE that support vulnerability management and software security analysis. The mapping between CVEs and CWEs helps establish the relationship between specific vulnerabilities and underlying weaknesses , allowing a better understanding of the root causes of security issues and enabling more effective vulnerability remediation strategies.

## 2.3 RELATED RESEARCH

### 2.3.1 Threat KG

In the field of cyber security research, understanding the intricate web of connections among platforms, vulnerabilities, attack techniques and defense strategies has emerged as an imperative problem. Prior investigations have contributed on various facets of this complex landscape, yet a comprehensive synthesis of these interconnected elements remains an elusive goal. A work in this domain is ThreatKG [29].

The ThreatKG approach is a threat knowledge graph for automated threat gathering and depiction. THREATKG operates through a structured process consisting of three primary phases: (1) the collection of OSCTI reports, (2) the extraction of threat knowledge, and (3) the construction of a threat knowledge graph. Each of these phases encompasses one or more distinct processing steps, such as parsing and extracting.
In Phase I, THREATKG actively gathers OSCTI reports from a diverse array of sources using a Crawler. During Phase II, THREATKG carries out several tasks, including grouping multi-page report files (Porter), parsing the reports (Parser), filtering out non-threat-related reports (Checker), and ultimately extracting valuable threat knowledge (Extractor). Phase III sees THREATKG constructing a comprehensive threat knowledge graph and storing it within a database. It's important to note that THREATKG operates automatically and continuously, periodically collecting new reports and incrementally extracting and integrating fresh knowledge into the knowledge graph through a process known as knowledge fusion.

To create a comprehensive representation of threats, it establishes a hierarchical threat knowledge ontology that encompasses a wide range of threat-related entities and relationships, allowing us to capture both detailed low-level threat activities and broader high-level threat contexts. This ontology is structured into three distinct layers.
The first layer, known as the report context layer within the ontology, encompasses knowledge related to individual reports. For each report, we associate it with an entity corresponding to its type. These report entities possess attributes such as title, URL, publication date, and more. The inclusion of dedicated report entities serves to assist threat analysts in establishing connections between other threat-related entities (e.g., malware, Indicators of Compromise (IOCs), Tactics, Techniques, and Procedures (TTPs)) extracted from the same report. This facilitates the creation of a comprehensive understanding of the threat landscape. Additionally, threat analysts can access the original report by following the URL attribute, enabling them to access further context. We also create entities for the specific authors and CTI (Cyber Threat Intelligence) vendors responsible for these reports, with these entities and their relationships constituting the report context layer.
The second layer, known as the threat behavior layer of the ontology, contains information regarding low-level threat behaviors. As demonstrated in previous research, IOCs and their relationships provide critical insights into the sequence of low-level actions comprising a threat. This knowledge is valuable for identifying specific system events (e.g., a process reading a file) that are integral components of an attack sequence. Such insights greatly enhance defensive measures like cyber threat hunting. For instance, in Figure 2b, two filename IOCs, "Office Monkeys (Short Flash Movie).exe" and "player.exe," are connected through a launch relation. In the threat behavior layer, we consider various types of IOCs and their associated relationships. Examples of IOC types include filename, filepath, IP address, URL, domain, registry entries, and cryptographic hashes. Building upon prior research, we also account for interaction verbs (e.g., read, write, open, send) as the relationships between these IOCs.

The threat context layer within the ontology furnishes overarching contexts for threats, complementing the detailed steps of threat behavior. These contexts are pivotal for achieving a thorough comprehension of threats and for devising effective countermeasures accordingly. In this layer, we encompass a diverse array of entities, including but not limited to:

- Malware
- Vulnerabilities (e.g., Common Vulnerabilities and Exposures (CVEs)

- Threat actors (e.g., the CozyDuke Advanced Persistent Threat (APT) actor [35])
- Tactics and techniques (e.g., spearphishing link )
- Vulnerable software products (e.g., Microsoft Word)
- Security-related tools (e.g., Mimikatz )
- Mitigations (e.g., data backup)
- Relevant entities (e.g., infected computers)

These entities may establish various types of relationships among them. Let's denote an entity placeholder of a specific type as "TYPE_ENT." Examples of entity-relation triplets within this layer include <ACTOR_ENT, use, MALWARE_ENT> and <SOFTWARE_ENT, has, VULNERABILITY_ENT>.

Entities from distinct layers can also be interconnected. For instance, entities within the threat behavior layer and the threat context layer, derived from the same report, are linked to the corresponding report entity (situated within the report context layer) through a "reported_in" relation. Entities may also possess attributes in the form of key-value pairs (e.g., malware type, software version).

Collectively, the three layers of this ontology comprehensively model threats from various dimensions and levels of granularity. Compared to other existing cyber ontologies , THREATKG's ontology offers a considerably broader spectrum of threat knowledge types, empowering threat analysts to attain a more holistic perspective on threats.

Furthermore, during text extraction, some connections among entities can be determined through the using verbs and capturing both low and high level threats. THREATKG utilizes a relation extractor based on dependency parsing to discern interaction verbs connecting two entities. Our method involves utilizing dependency parsing to scrutinize the grammatical arrangement of a sentence, thereby forming a dependency tree. Subsequently, a set of dependency grammar rules is applied to identify subject-verb-object relationships between Indicators of Compromise (IOCs) and to isolate the specific relation verb. Additionally, our approach can also capture the sequential sequence of IOC interaction steps if they are available, offering valuable insights into comprehending the threat scenario.
THREATKG employs a Deep Learning (DL)-based relation extractor that utilizes a Piecewise Convolutional Neural Networks model with an attention mechanism (PCNN-ATT) to conduct neural relation extraction (RE). The PCNN model bears resemblance to Convolutional Neural Networks (CNN), which are commonly used for image and text classification tasks. However, PCNN is specifically tailored for relation extraction. Instead of employing a single max-pooling operation to merge features, as seen in CNN, PCNN employs piecewise max-pooling. This approach divides a sentence into three segments using the two entities in question and calculates the maximum value for each segment. In contrast to CNN, PCNN is better suited for relation extraction because it captures the structural details of a sentence by considering the locations of the two entities. This is critical for identifying the essential tokens that signify the relation. Additionally, an attention layer is incorporated to enable the model to focus on the most important tokens, as not all tokens in a sentence contribute equally to relation extraction.
Following Named Entity Recognition (NER) and preceding relation extraction (RE), THREATKG conducts co reference resolution to identify all expressions (e.g., pronouns) in the text that refer to a specific entity. This process allows the RE to benefit from the information provided by the resolved entities, facilitating the connection of extracted triplets to form a comprehensive understanding of threat knowledge.

### 2.3.2 NLP Based Approaches

There is not always an immediate connection between vulnerability catalogues and attack pattern dictionaries, because vulnerabilities and attack methods are not always connected, due to the fact that they are independent. Although there are similarity algorithms (patterns) that suggest a connection between them such as term frequency–inverse document frequency (TF-IDF), universal sentence encoder (USE), and sentence BERT (SBERT) Recall and reciprocal rank evaluate the score for each one of the similarity algorithms for tracing capec-ids with cves [30]. There are several cves that cannot be

connected to capec because there is no link between cwe and capec (cve -> cwe -> capec), because vulnerabilities (cve) connect with capec through weaknesses (cwe) manually.
A list of associated CAPEC-ID candidates is generated for a given CVE-ID. Then, the linkage is determined based on the similarity between the CAPEC document and the CVE description. With the intention of tracing CVE-Ids from CAPEC-Ids there are four steps to consider:

- A sum of cve descriptions and capec-id documents are generated
- Using a similarity algorithm, an embedded document is created using three algorithms to create document embeddings: TF-IDF (term frequency–inverse document frequency), USE (universal sentence encoder), and SBERT (sentence Bert - python framework - sentence, text and image embeddings generator).These embeddings can then be compared with cosine-similarity to find sentences with a similar meaning.
- cve-id and appropriate capec-ids are used to calculate cosine similarity. As regards cosine similarity, there is a way to determine how similar two documents are, despite their size. It is helpful because older approaches to determine the similarity of two documents were based on the amount of the common used words. That hides flaws if the documents have big difference in size.
- That means the number of common words tend to increase although they include completely different topics. In mathematics: check the angle between two projected vectors; Smaller the angle, higher the similarity. In our case, vectors represent a table of similar words with their counts for cve and capec accordingly. Calculate the cosine similarity by using either from python library sklearn the function cosine_similarity() or using the mathematic formula.
- Capec references are sorted by similarity score to select a sum of capec-ids, and correlated capec-ids derive from cve-ids. Due to lack of detailed description in cves and/or the lack of connection between cve and cwe and the high abstraction in cwe terms, leads to results that cves and capec cannot be fully correlated.

There are three different approaches to connect cve with capec, as regards the documented formation:

- The input (capec) is as a whole.
- The input is a separation of every capec section and then a similarity is calculated for each section.
- Calculate the similarity for each capec section and then calculate the average similarity for all sections.

Tracing Based on term frequency–inverse document frequency (TF-IDF) estimates the value of each word in the input. In addition, the importance of the word increases according to the number of times this word appears in a given text. To calculate the TF-IDF two parameters should be multiplied together. The first parameter is the term frequency of a word. This estimates the number of times a word appears in a text and it can be computed by counting these instances. The second parameter is the inverse document frequency of the word among a set of input texts or documents. The inverse document frequency depicts how common or rare a word is in a document. As a result, the word that is commonly used and shows up in several text inputs approaches number zero (0). On the contrary, with fewer appearances in documents, it approaches number one (1). As a consequence, by multiplying these two results a score number shows how relevant the specified word is according to the text. Higher the score, the more appropriate the word is. In addition to the similarity algorithms, the universal sentence encoder exports dimensional vectors normalized by length that can be used for several NLP implementations, such as document classification, similarities etc. There are two approaches to acquire these word vectors:

- The first one uses the transformer encoder, which is more accurate but it requires more computational resources.
- The second uses Deep Averaging Network (DAN), which is less accurate but it requires less computational resources.

For these two implementations there are pre trained models available.

The third similarity algorithm is Sentence BERT (SBERT). The Sentence BERT (SBERT) is an amendment of the BERT network that uses Siamese and triplet networks to create accurate sentence vectors. The SBERT has a better performance than the BERT approach and it uses similarity measure methods such as cosine similarity or Euclidean distance. Also, SBERT attaches a pooling operation to the output of BERT to generate a fixed sized sentence embedding. Taking into consideration the high abstraction several CVE entries have, it is difficult to connect all the cve-ids to the capec ids. Due to the fact that cve-id links to cwe and one cwe often connects to several capec-ids, it is difficult and complex to link the correct cwe entity with the capec entity. Another reason for this low accuracy rate is the abstract description that several weaknesses have, therefore the underlying attack methods that occur may not be found. Furthermore, an additionally important connection is between the attack patterns of CAPEC and the attack techniques of ATT&CK framework, due to the fact that later on the defense techniques for the corresponding attacks can be traced successfully.

Currently, only a small amount of capec entries are linked to attack framework, through the Taxonomy Mappings section. For example 112 capec attack patterns out of 546 are related to ATT&CK framework. In an attempt to connect capec - attack entities using Natural Language Processing algorithms using the descriptions of each entity and any other text data from these entries in order to use Natural Language Processing methods to effectively combine and connect attack patterns (capec) with att&ck techniques (MITRE ATT&CK framework) in order to find the appropriate defending solutions according to existing vulnerabilities.

Therefore, using the existing connections between attack patterns and attack techniques as pre-trained dataset before using Natural Processing Language implementations, can lead to a more effective approach to link CAPEC and MITRE ATT&CK. To conclude, there is no direct connection from vulnerabilities to defend methods. The time consuming and error prone existing approach deriving from several in between datasets and frameworks makes defend techniques not possible through vulnerabilities. As a result, analyzing all the intermediate steps (catalogues/dictionaries/frameworks) and using Natural Language Processing algorithms, possible defend methods may be proposed to mitigate cyber attacks.

# 3 METHODOLOGY

To begin with the methodology of this process, there are several procedures, steps and informational data that should be found and processed. The objective of this process is to collect information and datasets in various formats, in order to end up with a group of information – dataset- that consists of a combination of the gathered information. Therefore, a relation between the Common Platform Enumeration, that includes software and hardware assets, and the vulnerabilities and weaknesses for each asset accordingly, has a meaningful purpose in order to connect all these three categories with attack techniques, tactics and attack patterns that take advantage of one or more asset's weaknesses. As a consequence from the above relation pattern, defend techniques and tactics may be suggested for related attack techniques, which are all associated with the platform enumeration approach. Therefore, presenting known defend techniques for a specific software or platform, based on their weaknesses, contributes to prevent an attack from that entity up to some extent. The relationship between CVE and CWE enables the analysis of patterns and trends in software vulnerabilities and weaknesses. By examing the CWEs associated with a set of CVEs, security researchers and organizations can identify common weaknesses that contribute to multiple vulnerabilities. This information is invaluable for developing effective mitigation strategies, improving software development practices and prioritizing security efforts.

The procedure of relating platforms (software - hardware) with their perspective defend techniques, presupposes a sum of steps that need to be done. Collecting the appropriate information through academic papers, knowledge databases, repositories and research frameworks, hands out a better understanding of the current state of the available data. Analyzing each data set and figuring out their structure, relations arise that some of the information sets include pair wise similarities. At the beginning of this approach, an analysis of the already known relations from certain frameworks and then researching new ones to the rest of the datasets. Due to the fact that the datasets have different format files, that is why a convertion to a common format file is needed, such as the comma separated value files (csv) for each relation separately as far as the full association.

## 3.1 CVE - CWE CONNECTION

An expected correlation is being found between the vulnerabilities and the weaknesses of a platform, by using the data available through MITRE framework. CVE (Common Vulnerabilities and Exposures) and CWE (Common Weakness Enumeration) are two essential data sets maintained by the MITRE Corporation to support vulnerability management and security analysis in the field of cyber security. While they serve different purposes, the two data sets are closely related and complement each other in providing valuable information about vulnerabilities and weaknesses in software systems. CVE and CWE are connected through their mappings, where each CVE entry can be associated with one or more CWE entries. This mapping helps establish the relationship between specific vulnerabilities and the underlying weaknesses or flaws in the software that lead to those vulnerabilities.

## 3.2 CVE - CPE CONNECTION

Each vulnerability in the CVE database is assigned a unique identifier, called a CVE ID, which is commonly referred to as a "CVE number." These CVE IDs are widely used by security researchers, vendors, and organizations to reference and discuss specific vulnerabilities. By assigning a CVE ID to vulnerability, MITRE ensures that it has a consistent and standardized reference across the cyber security community, enabling effective communication and collaboration between different stakeholders. On the other hand, CPE is used to uniquely identify hardware and software products and their respective versions. It provides a standardized format to describe the characteristics and configurations of these products. CPE identifiers consist of various fields that specify the vendor, product name, version, and other relevant attributes. These identifiers are crucial for accurately tracking and managing vulnerabilities across different systems and software components. CPEs are used in vulnerability assessment tools, security information and event management (SIEM) systems, and other

cyber security applications to match vulnerabilities with affected products. The relationship between CVE and CPE is established through the association of CVE IDs with specific CPE identifiers. When vulnerability is discovered and assigned a CVE ID, it is then linked to the relevant CPE(s) representing the affected software or hardware products and versions. As a result, CVE and CPE are complementary standards in the cyber security domain. CVE provides a unique identifier for vulnerabilities, while CPE offers a structured naming scheme for products and versions. The association of CVE IDs with CPE identifiers enables effective vulnerability tracking, assessment, and remediation across diverse software and hardware systems. Together, CVE and CPE contribute to the overall goal of improving cyber security by promoting standardized vulnerability management practices.

## 3.3 CWE - CAPEC RELATION

Attack Pattern Mapping: CAPEC attack patterns often reference or exploit specific weaknesses identified in CWE. CAPEC provides cross-references to related CWE weaknesses, helping developers understand the underlying vulnerabilities that attackers may target.
Weakness Mitigation: CWE weaknesses provide guidance on how to identify and mitigate vulnerabilities during software development.
Developers can leverage CWE to address weaknesses that are referenced in CAPEC attack patterns, ensuring their software is resilient against known attack techniques. Also, Security Training and Awareness: CAPEC and CWE are valuable resources for security training and awareness programs. They provide a structured and standardized vocabulary for discussing software vulnerabilities and attack techniques, enabling organizations to educate their developers and security teams effectively. As a consequence, CAPEC and CWE are complementary frameworks that provide valuable insights into the attack patterns employed by adversaries and the weaknesses that make software susceptible to exploitation. By leveraging both frameworks, organizations can better understand, assess and mitigate vulnerabilities in their software systems.

## 3.4 CAPEC – ATT&CK CONNECTION

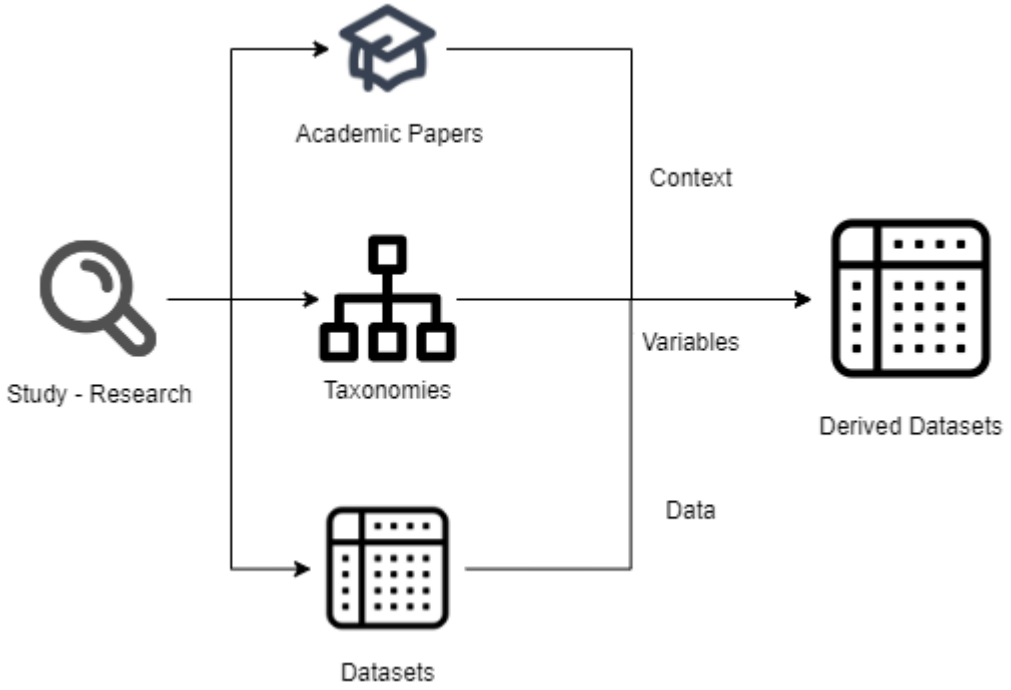
CAPEC is a valuable resource for understanding and analyzing attack scenarios. It helps security professionals, developers, and researchers to identify potential threats, assess vulnerabilities, and develop appropriate mitigation strategies. It is commonly used for threat modeling, security assessments, penetration testing, and security training. On the other hand, ATT&CK focuses on the "what" and "why" of attacks, providing insights into the tools, techniques, and procedures used by adversaries. It covers a broad range of attack techniques and tactics, allowing organizations to map their defensive strategies and identify gaps in their security controls. ATT&CK also includes information on how attacks can progress through various stages, such as initial access, persistence, privilege escalation, and exfiltration.

Attack Pattern Mapping: Both CAPEC and MITRE ATT&CK provide structured taxonomies for categorizing attacks. While CAPEC focuses more on the specific methods and techniques used by attackers, MITRE ATT&CK offers a broader view, encompassing the entire lifecycle of an attack. In some cases, certain attack patterns described in CAPEC may align with techniques documented in MITRE ATT&CK, allowing for cross-referencing and analysis. Threat Intelligence: Organizations can use both CAPEC and MITRE ATT&CK as sources of threat intelligence. CAPEC provides detailed insights into specific attack patterns and their associated weaknesses, while MITRE ATT&CK provides a broader understanding of adversary behaviors and techniques. By considering information from both frameworks, organizations can enhance their threat intelligence capabilities and strengthen their defenses against known attack patterns and tactics.In summary, CAPEC and MITRE ATT&CK are distinct frameworks that provide valuable insights into cyberattack techniques and behaviors. While they serve different purposes, there can be connections and overlap between specific attack patterns and techniques described in both frameworks.

## 3.5 ATT&CK - DEFEND CONNECTION

The MITRE ATT&CK framework is primarily used by organizations to understand and analyze the techniques employed by adversaries during cyber attacks. It helps in threat intelligence, identifying security gaps, improving defenses, and assessing the effectiveness of security controls. Organizations can use the framework to map their existing security measures to specific techniques to identify any weaknesses or areas where they need to improve their defenses. MITRE DEFEND: The MITRE DEFEND framework complements the MITRE ATT&CK framework by focusing on defensive techniques and countermeasures. It provides guidance and best practices for organizations to improve their security posture and effectively defend against the techniques documented in the MITRE ATT&CK framework. The DEFEND framework encompasses various aspects of defense, including prevention, detection, response, and remediation. MITRE DEFEND offers specific guidance and recommendations for implementing security controls and practices to mitigate the risk posed by the techniques outlined in MITRE ATT&CK. It helps organizations in developing robust defensive strategies, improving incident response capabilities, and strengthening their overall security posture. The DEFEND framework is designed to be used in conjunction with the ATT&CK framework to ensure a comprehensive approach to cyber security defense. As a consequence, the MITRE ATT&CK framework provides knowledge about adversary tactics and techniques, while the MITRE DEFEND framework provides guidance on defensive techniques and countermeasures to mitigate the risk associated with those techniques. Together, these frameworks help organizations understand the threat landscape, identify vulnerabilities, and implement effective security measures.

As a subsequently result, all the above mentioned data sets and knowledge bases can be related and form a whole new dataset that includes the hardware and software entities, the group of weaknesses and the vulnerabilities each one has, the appropriate attack tactics and techniques and, last but not least, the defend techniques and precautions for these systems.




**13Thesis Architecture DIagram**

# 4 IMPLEMENTATION

The implementation of creating a data set, that includes several separated frameworks, involves the process of data cleansing and processing. The two main frameworks that are used are MITRE and National Vulnerability Database. MITRE's contributions to cybersecurity include the creation and maintenance of important resources and frameworks, such as: Common Platform Enumeration (CPE), Common Weakness Enumeration (CWE), Common Vulnerabilities and Exposures (CVE), Common Attack Pattern Enumeration and Classification (CAPEC), ATTACK knowledge base and DEFEND framework. Furthermore, the NVD provides detailed information about vulnerabilities, including their descriptions, impact ratings, and references to mitigation strategies. It includes vulnerabilities in various software products, operating systems, libraries, and other components. The database also assigns unique identifiers to each vulnerability, known as CVE IDs, which are maintained in collaboration with MITRE.

Using the Jupyter Lab as an Integrated Development Environment, this encapsulates the suitable libraries to cope with this type of information. The first attempt is to download the vulnerabilities dataset from the National Vulnerabilities Database, throughout the years 2002 until current year (2023). This is the start, as this set of information maps to the weaknesses of the corresponding year. In addition, the pandas library is used to parse all these json files. Under the CVE_Items tag, there is the information that should be extracted, such as the CVE-ID (inside the tag CVE_data_meta) and then the weakness identification number cwe-id, which is the value inside the problemtype_data tag. If the cwe-id is empty or if it is not including any cwe-id, therefore the relation does not exist. All the cve-id and cwe-id are saved in a pandas dataframe and then write the results of the cve cwe connection in a csv file.

| | CVEID | CWEID |
|---|---|---|
| **0** | CVE-1999-0001 | CWE-20 |
| **1** | CVE-1999-0002 | CWE-119 |
| **6** | CVE-1999-0007 | CWE-327 |
| **26** | CVE-1999-0027 | CWE-119 |
| **178** | CVE-1999-0179 | CWE-17 |
| **...** | ... | ... |
| **215461** | CVE-2023-32569 | CWE-89 |
| **215462** | CVE-2023-32570 | CWE-362 |
| **215463** | CVE-2023-32573 | CWE-369 |
| **215469** | CVE-2023-32955 | CWE-78 |
| **215470** | CVE-2023-32956 | CWE-78 |

151313 rows × 2 columns

**14 Vulnerabilities and Weaknesses Dataframe sample**

Due to the fact that NVD vulnerabilities are in separated files, a study for each year is going to be displayed, in order to examine the relations among the years 2002 until 2023.

- *Year 2002:* There are 6.769 entries between cve and cwe, without excluding the rows that have no cwe id for a cve. If the rows without actual mapping are ignored, then the new relation group has 400 rows.
  This means that there are 6.369 vulnerabilities that do not correlate with any weakness.
- *Year 2003*: There are 1.553 total cve ids that only the 298 rows have cwe ids accordingly.
- *Year 2004*: In this year there are 2.707 rows in the cve-cwe list, although the 193 of them have a full connection. This means that 2.514 entities have no cwe id in that year.
- *Year 2005: There* are 4.765 cve entries in the year 2005, which they have a relation to cwe entrities the 393 of them. As a result, there are 4.372 rows without mapping.
- *Year 2006*: In the year 2006, there are 7.142 vulnerability entries, but the 993 of them have an actual connection with cwe, as a consequence, 6.149 rows have no relations.
- *Year 2007*: In the year 2007 there are 6.579 cve entries. However, the 2.677 have a cwe relation, meaning that 3.902 have no cwe information.
- *Year 2008*: There are 7.174 cve rows in this year and the 6.403 entities of them relate to cwe. This means that for the current year 771 vulnerabilities are not connected with weaknesses.
- *Year 2009*: The vulnerability rows that are recorded are 5.029 and the relations exist on 4.192 of them. This means that 837 cve entries have no cwes.
- *Year 2010*: There are 5.199 cve records and the 4.024 of them have cwe connection. As a result, 1.175 rows have no cve - cwe data mapping.
- *Year 2011*: The data frame of this year shows that 4.834 rows for cve entries, the 3.832 have a reference to cwe entities, showing that 1.002 do not include a conjunction to cwe.
- *Year 2012:* After parsing this vulnerability file, there are 5.853 cve rows in total and the amount of reference to cwe entities is 4.341 rows, meaning that 1.512 cve rows do not correlate with cwe instances.
- *Year 2013*: In this year there are 6.700 cve rows, but the 5.031 of them have cve –cwe relation. Therefore, for the year 2013 there are 1.669 entries without relation.
- *Year 2014*: With the help of pandas library, a dataframe with 8.967 rows depicts the group of cves for 2014. To continue, 7.315 rows have a successful relation with cwe, therefore 1.652 entries have no association with weaknesses.
- *Year 2015*: As far as the number of cve rows is concerned, there are 8.701 in total, but the 6.809 have a connection with cwe. The rest 1.892 do not have a relation with cwe.
- *Year 2016*: The year 2016 there were 10.538 cve entries in total, 8.200 of these had a mapping towards cwe information. Overall, 2.338 rows had no cwe connection.
- *Year 2017*: There are 16.889 cve entities for the year 2017. The 12.707 rows are related to cwe information; as a result 4.182 rows do not have a relation.
- *Year 2018:* This year the sum of the cve entries is 16.965 rows and the 13.723 do not match any cwe id. That explains that 3.242 cve rows do not relate with any of cwe entities.
- *Year 2019:* In this year the documented cve entries are 16.892 and the 12.775 have a cwe connection. As a consequence, the 4.117 rows are not related with cwe.
- *Year 2020*: There are 20.125 rows for vulnerabilities of the year 2020, where the 14.666 entries have a cwe connection. As a result, 5.459 rows are not cwe related.
- *Year 2021*: There are 21.678 vulnerability entries, although the 17.306 rows have a cwe connection. The rest 4.372 rows do not match any cwe connection.

- *Year 2022*: There are 23.438 total cve rows in 2022 and the 19.594 have a conjunction with cwe entities. The remaining 3.844 cve ids do not correlate with weakness identification numbers.
- *Year 2023:* In this year, there are 7.005 cve entries and there are 5.441 rows that relate to weaknesses, figuring out that 1.564 cves do not have a connection with cwes.

Taking everything into consideration, the total amount of rows is 151.313 vulnerability entities that all of them have a relation to weaknesses. On the other hand, the whole amount of cve – cwe rows is 215.470 but only the 151.313 have a connection, as mentioned above.

| cve - cwe | | | |
|---|---|---|---|
| Year | all relations | without empty relations | no relations |
| 2002 | 6.769 | 400 | 6.369 |
| 2003 | 1.553 | 298 | 1.255 |
| 2004 | 2.707 | 193 | 2.514 |
| 2005 | 4.765 | 393 | 4.372 |
| 2006 | 7.142 | 993 | 6.149 |
| 2007 | 6.579 | 2.677 | 3.902 |
| 2008 | 7.174 | 6.403 | 771 |
| 2009 | 5.029 | 4.192 | 837 |
| 2010 | 5.199 | 4.024 | 1.175 |
| 2011 | 4.834 | 3.832 | 1.002 |
| 2012 | 5.853 | 4.341 | 1.512 |
| 2013 | 6.700 | 5.031 | 1.669 |
| 2014 | 8.967 | 7.315 | 1.652 |
| 2015 | 8.701 | 6.809 | 1.892 |
| 2016 | 10.538 | 8.200 | 2.338 |
| 2017 | 16.889 | 12.707 | 4.182 |
| 2018 | 16.965 | 13.723 | 3.242 |
| 2019 | 16.892 | 12.775 | 4.117 |
| 2020 | 20.125 | 14.666 | 5.459 |
| 2021 | 21.678 | 17.306 | 4.372 |
| 2022 | 23.438 | 19.594 | 3.844 |
| 2023 | 7.005 | 5.441 | 1.564 |
| **sum** | **215.502** | **151313** | **64.189** |

**15 CVE - CWE relations summary**

To continue, after downloading the vulnerability files from the NVD, these files may be used to connect specific platforms, operating systems with vulnerabilities. As a result, when parsing these types of files, common platform enumeration data can be found in the cpe_match tag under the cpe23Uri tag info. Therefore, related results among the available years 2002 until 2023 are:

- *Year 2002:* As regards the connection between platforms and vulnerabilities in 2002, all rows are 24.563 and the rows with existing relation are 24.297.
- *Year 2003:* In this year, there are 9.438 cpe-cve relations and the 9.353 have no empty relations. Therefore, 85 rows have no connection.
- *Year 2004*: There are 23.196 rows in the dataset, but the 23.117 have both columns filled. As a result, the empty connections are 79.

- *Year 2005*: There are 33.726 rows with cpe-cve connections, but the 33.571 are related to each other, meaning that 155 are the empty relations.
- *Year 2006*: In this year, there are 39.045 rows, but the 38.847 have the actual relation, meaning that the empty rows are 198.
- *Year 2007*: There are 32.346 rows and the 31.892 rows are related. As a consequence, there are 454 non related rows.
- *Year 2008*: In this year 56.311 rows are included between cpe-cve and the 55.577 rows contain relations, meaning that 734 rows are not related.
- *Year 2009*: There are 82.071 rows in this dataset and the 81.226 of them have cpe-cwe connections, making the empty rows 845.
- *Year 2010*: In this year, there are 62.414 overall rows and the ones that have a connection are 61.240. Therefore, 1.174 rows have no relation.
- *Year 2011:* There are 77.487 rows and the 76.586 of them have the cpe-cwe connection, making the empty rows 901.
- *Year 2012:* The overall rows in this year's dataset are 90.659 and the ones that have a relation are 89.455. As a result, the empty rows are 1.204.
- *Year 2013*: There are 98.116 rows and the 96.790 contain the relation, meaning that 1.326 are empty connections.
- *Year 2014:* Overall rows in this dataset are 54.774 and the ones that have a relation are 53.568 rows. As a result, 1.206 entities have empty relations.
- *Year 2015:* There are 39.408 rows and the 37.656 of them have a relation. As a consequence, there are 1.392 empty connections.
- *Year 2016*: In this year, the amount of rows in this cpe-cve dataset is 58.641 and the 56.121 of them are related, meaning that the empty connections are 2.520.
- *Year 2017*: In this year, the amount of rows is 76.290 and the 71.545 have the actual connection. As a result, the empty rows are 4.745.
- *Year 2018*: Overall rows in this dataset are 40.672 and the 36.500 of them include a connection, meaning that the empty rows are 4.172.
- *Year 2019*: In this year's data, there are 48.270 rows and the 43.455 of them have a connection, meaning that empty relations are 4.815.
- *Year 2020*: In the year 2020, the whole amount of rows is 88.125 and the 83.303 of them include a valid connection. Therefore, the empty relations are 4.822.
- *Year 2021*: There are 85.455 rows in this dataset and the 80.664 of them include a relation. As a result, 4.791 vector rows are empty.
- *Year 2022*: As regards the number of rows in this dataset, there are 79.281 rows and 74.112 are fully related, making the number of empty connections 5.169.
- *Year 2023*: This is the final year of finding the overall rows and the actual related entities in this dataset, introducing 23.278 rows, where the 21.611 are fully related, making the empty rows 1.667.

Taking everything into consideration, the overall number of rows is 1.223.566 and the actual related rows are 1.180.486, making the empty rows 43.080 for cpe-cve connection. These results are extracted by merging together all the datasets of each year, using the pandas library and the merge function it includes.

| cve - cpe | | | |
|---|---|---|---|
| Year | all relations | without empty relations | no relations |
| 2002 | 24.563 | 24.297 | 266 |
| 2003 | 9.438 | 9.353 | 85 |
| 2004 | 23.196 | 23.117 | 79 |
| 2005 | 33.726 | 33.571 | 155 |
| 2006 | 39.045 | 38.847 | 198 |
| 2007 | 32.346 | 31.892 | 454 |
| 2008 | 56.311 | 55.577 | 734 |
| 2009 | 82.071 | 81.226 | 845 |
| 2010 | 62.414 | 61.240 | 1.174 |
| 2011 | 77.487 | 76.586 | 901 |
| 2012 | 90.659 | 89.455 | 1.204 |
| 2013 | 98.116 | 96.790 | 1.326 |
| 2014 | 54.774 | 53.568 | 1.206 |
| 2015 | 39.408 | 37.656 | 1.392 |
| 2016 | 58.641 | 56.121 | 2.520 |
| 2017 | 76.290 | 71.545 | 4.745 |
| 2018 | 40.672 | 36.500 | 4.172 |
| 2019 | 48.270 | 43.455 | 4.815 |
| 2020 | 88.125 | 83.303 | 4.822 |
| 2021 | 85.455 | 80.664 | 4.791 |
| 2022 | 79.281 | 74.112 | 5.169 |
| 2023 | 23.278 | 21.611 | 1.667 |
| **sum** | **1223566** | **1180486** | **42720** |

**16 CVE - CPE relation summary**

| | CVEID | cpe |
|---|---|---|
| 1 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.5:*:*:*:*:*:*:* |
| 2 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.2:*:*:*:*:*:*:* |
| 3 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.7:*:*:*:*:*:*:* |
| 4 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.3:*:*:*:*:*:*:* |
| 5 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.0.5:*:*:*:*:*:*:* |
| 6 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.1.5.1:*:*:*:*:*:*:* |
| 7 | CVE-1999-0001 | cpe:2.3:o:bsdi:bsd_os:3.1:*:*:*:*:*:*:* |
| 8 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.8:*:*:*:*:*:*:* |
| 9 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.0:*:*:*:*:*:*:* |
| 10 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.6.1:*:*:*:*:*:*:* |
| 11 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2:*:*:*:*:*:*:* |
| 12 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:3.0:*:*:*:*:*:*:* |
| 13 | CVE-1999-0001 | cpe:2.3:o:openbsd:openbsd:2.4:*:*:*:*:*:*:* |
| 14 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.1:*:*:*:*:*:*:* |
| 15 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.4:*:*:*:*:*:*:* |
| 16 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.6:*:*:*:*:*:*:* |
| 17 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.6:*:*:*:*:*:*:* |
| 18 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.7.1:*:*:*:*:*:*:* |
| 19 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.0.1:*:*:*:*:*:*:* |
| 20 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.2:*:*:*:*:*:*:* |
| 21 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.5:*:*:*:*:*:*:* |
| 22 | CVE-1999-0001 | cpe:2.3:o:openbsd:openbsd:2.3:*:*:*:*:*:*:* |
| 23 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.0:*:*:*:*:*:*:* |

**17 cve-cpe dataframe sample**

Also, pandas dataframe is used for saving and comparing each row of the json files, and then a matching table is saved as a csv file with the cve-id and cpe header tags.
From these two steps, we can derive information as regards the cve – cwe connection and cve – cpe connection. Therefore, we are going to connect these two data files using the mutual variable, cve. A link between cve – cwe and cve – cpe allows us to come up with the cpe – cve – cwe dataset. This means that we have information for the enumerated software - hardware components, which vulnerabilities each one can have and the weaknesses that make this system vulnerable.

The next association that follows is the merge among the datasets of cpe, cve and cwe. The amalgamation of the above data has a result of 1.223.566 rows, where the 855.996 include existing consolidations, meaning that 367.570 rows have an empty relation.

```
output1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1223566 entries, 0 to 1223565
Data columns (total 3 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   CVEID   1223566 non-null  object
 1   cpe     1223566 non-null  object
 2   CWEID   1223566 non-null  object
dtypes: object(3)
```

**18 Cpe-Cve-Cwe all relations info**

```
output1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 855996 entries, 0 to 855995
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   CVEID   855996 non-null  object
 1   cpe     855996 non-null  object
 2   CWEID   855996 non-null  object
dtypes: object(3)
```

**19 Cpe-Cve-Cwe without empty relations info**

| | CVEID | cpe | CWEID |
|---|---|---|---|
| 1 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.5:*:*:*:*:*:*:* | CWE-20 |
| 2 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.2:*:*:*:*:*:*:* | CWE-20 |
| 3 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.7:*:*:*:*:*:*:* | CWE-20 |
| 4 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.3:*:*:*:*:*:*:* | CWE-20 |
| 5 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.0.5:*:*:*:*:*:*:* | CWE-20 |
| 6 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.1.5.1:*:*:*:*:*:*:* | CWE-20 |
| 7 | CVE-1999-0001 | cpe:2.3:o:bsdi:bsd_os:3.1:*:*:*:*:*:*:* | CWE-20 |
| 8 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.8:*:*:*:*:*:*:* | CWE-20 |
| 9 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.0:*:*:*:*:*:*:* | CWE-20 |
| 10 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.6.1:*:*:*:*:*:*:* | CWE-20 |
| 11 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2:*:*:*:*:*:*:* | CWE-20 |
| 12 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:3.0:*:*:*:*:*:*:* | CWE-20 |
| 13 | CVE-1999-0001 | cpe:2.3:o:openbsd:openbsd:2.4:*:*:*:*:*:*:* | CWE-20 |
| 14 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.1:*:*:*:*:*:*:* | CWE-20 |
| 15 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.4:*:*:*:*:*:*:* | CWE-20 |
| 16 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.2.6:*:*:*:*:*:*:* | CWE-20 |
| 17 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.6:*:*:*:*:*:*:* | CWE-20 |
| 18 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.7.1:*:*:*:*:*:*:* | CWE-20 |
| 19 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.0.1:*:*:*:*:*:*:* | CWE-20 |
| 20 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:1.2:*:*:*:*:*:*:* | CWE-20 |
| 21 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.1.5:*:*:*:*:*:*:* | CWE-20 |
| 22 | CVE-1999-0001 | cpe:2.3:o:openbsd:openbsd:2.3:*:*:*:*:*:*:* | CWE-20 |
| 23 | CVE-1999-0001 | cpe:2.3:o:freebsd:freebsd:2.0:*:*:*:*:*:*:* | CWE-20 |

**20 cpe-cve-cwe dataframe sample**

Furthermore, when studying the cwe dataset, which can be found under the mitre organization (cwe.mitre.org), relations arise. That is because a mapping exists between the weaknesses and the attack patterns (cwe - capec). Exploring the data in cwe and exploring the data in capec, we should find the connection with the most available relations cwe - capec, meaning that both of these datasets map to each other.

Now, that the cpe-cve-cwe relation dataset is generated, the consolidation of attack patterns (capec) and weaknesses (cwe) is going to be implemented. This extraction can be made through capec dataset and through cwe dataset. As a result, the two datasets are examined and the similarities between the two are going to be used.

Firstly, the capec dataset is examined, extracting 1.379 rows, where the 1.214 of them include an amalgamation. Therefore, 165 entries are not connected.

```
capec_cwe_all_relations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1379 entries, 0 to 1378
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CWEID     1379 non-null   object
 1   capec-id  1379 non-null   object
dtypes: object(2)
```

**21 capec-cwe all relations**

```
capec_cwe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1214 entries, 0 to 1213
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CWEID     1214 non-null   object
 1   capec-id  1214 non-null   object
dtypes: object(2)
```

**22 capec-cwe without empty relations**

To continue, the cwe dataset is examined and 12.840 rows are extracted, where the 1.212 rows include a relation. As a result, the empty rows are 11.628.

```
cwe_capec.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12840 entries, 0 to 12839
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CWEID     12840 non-null  object
 1   capec-id  12840 non-null  object
dtypes: object(2)
```

**23 cwe-capec all relations**

```
cwe_capec.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1212 entries, 0 to 1211
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CWEID     1212 non-null   object
 1   capec-id  1212 non-null   object
dtypes: object(2)
```

**24 cwe-capec without empty relations**

Furthermore, by extracting the similarities between the above datasets accordingly, there are 3.985 capec-cwe rows and the 1.212 of them include a connection, making the empty rows 2.773.

```
unique_results.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3985 entries, 0 to 1378
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   CWEID     3985 non-null   object
 1   capec-id  3985 non-null   object
dtypes: object(2)
```

**25 Similarities cwe-capec & capec-cwe all relations**

```
result.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1212 entries, 0 to 1211
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   CWEID     1212 non-null   int64 
 1   capec-id  1212 non-null   int64 
dtypes: int64(2)
```

**26 Similarities cwe-capec & capec-cwe without empty relations**

| | CWEID | capec-id |
|---|---|---|
| 1 | CWE-1007 | 632 |
| 2 | CWE-1021 | 103 |
| 3 | CWE-1021 | 181 |
| 4 | CWE-1021 | 222 |
| 5 | CWE-1021 | 504 |
| 6 | CWE-1021 | 506 |
| 7 | CWE-1021 | 587 |
| 8 | CWE-1021 | 654 |
| 9 | CWE-1037 | 663 |
| 10 | CWE-112 | 230 |
| 11 | CWE-112 | 231 |
| 12 | CWE-113 | 105 |
| 13 | CWE-113 | 31 |
| 14 | CWE-113 | 34 |
| 15 | CWE-113 | 85 |
| 16 | CWE-114 | 108 |
| 17 | CWE-114 | 640 |
| 18 | CWE-116 | 104 |
| 19 | CWE-116 | 73 |
| 20 | CWE-116 | 81 |
| 21 | CWE-116 | 85 |
| 22 | CWE-117 | 268 |
| 23 | CWE-117 | 81 |

**27 cwe-capec dataframe sample**

Therefore, until now there are two datasets: cpe-cve-cwe and cwe-capec; so the connection between these two is expected, in order to create the cpe-cve-cwe-capec dataset. Again, there are going to be two approaches, in which the first has all the related entities and the second has the entities without

empty relations. As expected, the approach that does not include empty entities is going to have less data.

After the data association, the new dataset that emerges has 24.111.960 total rows, where the 11.371.142 of them includes the actual related information, making the empty association values 12.740.818. As a general conclusion, more than the half entities do not have a related value, showing the gap that exists among the data.

```
output2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24111960 entries, 0 to 24111959
Data columns (total 4 columns):
 #   Column    Dtype
---  ------    -----
 0   CWEID     object
 1   capec-id  object
 2   CVEID     object
 3   cpe       object
dtypes: object(4)
```

**28 cpe-cve-cwe-capec all relations**

```
cpe_cve_cwe_capec_without_empty_rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11371142 entries, 0 to 11371141
Data columns (total 4 columns):
 #   Column    Dtype
---  ------    -----
 0   CWEID     object
 1   capec-id  int64
 2   CVEID     object
 3   cpe       object
dtypes: int64(1), object(3)
```

**29 cpe-cve-cwe-capec without empty relations**

The above pictures demonstrate the cpe-cve-cwe-capec data frame connections, using the pandas library and the functions that this library includes. In this current step, there is an association between the documented software – hardware systems, how their weaknesses are being exploited and which are the documented attack techniques that the adversaries use.

```
CWEID,capec-id,CVEID,cpe
CWE-1021,103,CVE-2005-2407,cpe:2.3:a:kayako:liveresponse:2.0:*:*:*:*:*:*:*
CWE-1021,103,CVE-2008-2716,cpe:2.3:a:xigla:absolute_form_processor_xe:4.0:*:*:*:*:*:*:*
CWE-1021,103,CVE-2011-1244,cpe:2.3:a:microsoft:open_xml_file_format_converter:*:*:mac:*:*:*:*:*
CWE-1021,103,CVE-2011-1244,cpe:2.3:a:microsoft:office:2008:*:mac:*:*:*:*:*
CWE-1021,103,CVE-2011-1244,cpe:2.3:a:microsoft:office:2004:*:mac:*:*:*:*:*
CWE-1021,103,CVE-2011-1244,cpe:2.3:a:microsoft:excel:2002:sp3:*:*:*:*:*:*
CWE-1021,103,CVE-2011-1244,cpe:2.3:a:microsoft:office:2011:*:mac:*:*:*:*:*
CWE-1021,103,CVE-2013-2675,cpe:2.3:h:cooperindustries:smp_4_gateway_\(data_concentrator\):-:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-2675,cpe:2.3:h:cooperindustries:smp_4\/dp_gateway_\(data_concentrator\):-:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-2675,cpe:2.3:h:cooperindustries:smp_16_gateway_\(data_concentrator\):-:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-2682,No_cpe_match
CWE-1021,103,CVE-2013-5594,cpe:2.3:a:oracle:jdk:1.6.0:update65:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5594,cpe:2.3:a:oracle:jre:1.6.0:update65:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.4:beta_1:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.3:beta_2:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.4:rc_1:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:*:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.4:beta_3:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.3:beta_3:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.3:rc_1:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.4:beta_2:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:openinfosecfoundation:suricata:1.4:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.3.1:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.3.2:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.3.3:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.3.4:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.3.5:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.3.6:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.4.1:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.4.2:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.4.3:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-5614,cpe:2.3:a:suricata-ids:suricata:1.4.4:*:*:*:*:*:*:*
CWE-1021,103,CVE-2013-6772,cpe:2.3:a:hancom:hancom_office_2010_se:8.5.8:*:*:*:*:*:*:*
CWE-1021,103,CVE-2014-1480,cpe:2.3:a:mozilla:netscape_portable_runtime:4.2:*:*:*:*:*:*:*
```

**30 cpe-cve-cwe-capec dataframe sample**

Further on our implementation, the data file that is created includes operating systems, platforms, vulnerabilities, weaknesses and attack patterns. To conclude this vendor data, attack techniques and defend tactics should be covered.

Consequently, a relation should be considered to link cpe-cve-cwe-capec list with attack and defend framework accordingly. To make this happen, a check among the available data should be done in accordance with mitre attack, questioning which of the previous data files (cpe,cve,cwe,capec) may relate its data with attack.

As it turns out, capec knowledge base is associated with the attack technique data, as it encapsulates related attack ids. The number of connections is 349 rows that associate capec-ids with attack ids and the new csv data table consists of capec-ids and attack-ids (capec-attack).

| | capec-id | attack_id |
|---|---|---|
| 1 | 1 | 1574.010 |
| 2 | 100 | 07 |
| 3 | 101 | 36 |
| 4 | 105 | 24 |
| 5 | 11 | 1036.006 |
| 6 | 112 | 1110 |
| 7 | 112 | 11 |
| 8 | 114 | 1548 |
| 9 | 115 | 1548 |
| 10 | 122 | 1548 |
| 11 | 125 | 1498.001 |
| 12 | 125 | 1499 |
| 13 | 125 | 10 |
| 14 | 126 | 33 |
| 15 | 127 | 1083 |
| 16 | 13 | 1562.003 |
| 17 | 13 | 1574.006 |
| 18 | 13 | 1574.007 |
| 19 | 130 | 1499.003 |
| 20 | 130 | 10 |
| 21 | 131 | 1499 |
| 22 | 131 | 10 |
| 23 | 132 | 1547.009 |

**31 capec-attack dataframe sample**

When excluding the rows that do not have a relation, the number of rows is 310 and by extracting the empty relations from the dataset, 39 empty associations arise.

```
# Dataframe info
capec_attack.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349 entries, 0 to 348
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   capec-id   349 non-null    object
 1   attack_id  349 non-null    object
dtypes: object(2)
```

**32 capec-attack all relations info**

```
capec_attack__without_empty_rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 310 entries, 0 to 348
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   capec-id   310 non-null    object
 1   attack_id  310 non-null    object
dtypes: object(2)
```

**33 capec-attack without empty relations info**

Using the same point of view, merge the capec-attack csv file with the cpe-cve-cwe-capec csv file to add the attack techniques to the previous data file. As a result, the newly created data list consists of cpe, cve, cwe, capec and attack, including 10.126.642 entries, where the 4.003.963 entries do not include the empty affiliations.

```
cpe_through_attack_all_rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10126642 entries, 0 to 10126641
Data columns (total 5 columns):
 #   Column     Dtype 
---  ------     ----- 
 0   CWEID      object
 1   capec-id   object
 2   CVEID      object
 3   cpe        object
 4   attack_id  object
dtypes: object(5)
```

**34 cpe-cve-cwe-capec-attack all relations info**

```
cpe_through_attack_without_empty_rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4003963 entries, 0 to 4003962
Data columns (total 5 columns):
 #   Column     Dtype  
---  ------     -----  
 0   CWEID      object 
 1   capec-id   int64  
 2   CVEID      object 
 3   cpe        object 
 4   attack_id  float64
dtypes: float64(1), int64(1), object(3)
```

**35 cpe-cve-cwe-capec-attack without empty relations info**

Correspondingly, the calculation of the non related vectors leads to a data hole with 6.122.679 rows. What this means, is that from the all the documented platforms, more than the half of the platforms do not connect with attack methods. A percentage of approximately 40% of them has a relation with the corresponding attack techniques.

Moreover, the last connection that has to be made is the amalgamation between the attack and defend knowledge bases. With the help of MITRE corporation, mappings between attack and defend entities are recorded and are available through the D3fend project.

In order to find the mappings between the defend framework and all the rest, we should consider that defend framework is a complementary framework to Attack, that focuses on defensive strategies and respond to attack techniques from Attack framework. Therefore, the defend mappings file, includes

information related to attack techniques, so a new data list may be created with attack-ids and defend techniques (attack-defend).

After obtaining the dataset and accessing it, there are several columns that depict information, various mappings, relationships, labels and URIs that associate defensive and offensive techniques, tactics, artifacts and their relationships within the Defend Ontology. Analyzing this piece of information, the columns are:

- query_def_tech_label: The label or name of the defensive technique being queried or searched
- top_def_tech_label: The label or name of the top level defensive technique associated with the query technique
- def_tactic_label: The label or name of the defensive tactic associated with the query technique.
- def_tactic_rel_label: The label or name describing the relationship between the query technique and the defensive tactic.
- def_tech_label: The label or name of the defensive technique.
- def_artifact_rel_label: The label or name describing the relationship between the defensive technique and a defensive artifact.
- def_artifact_label: The label or name of the defensive artifact associated with the defensive technique.
- off_artifact_label: The label or name of the offensive artifact associated with the defensive technique.
- off_artifact_rel_label: The label or name describing the relationship between the offensive artifact and the defensive technique.
- off_tech_label: The label or name of the offensive technique associated with the defensive technique.
- off_tech_parent_label: The label or name of the parent offensive technique of the offensive technique.
- off_tech_parent_is_toplevel: Indicates whether the parent offensive technique is a top-level technique.
- off_tactic_rel_label: The label or name describing the relationship between the offensive tactic and the defensive technique.
- off_tactic_label: The label or name of the offensive tactic associated with the defensive technique.
- def_tactic: The URI or link to the defensive tactic in the d3fend ontology.
- def_tactic_rel: The URI or link to the relationship between the query technique and the defensive tactic in the d3fend ontology.
- tactic_def_tech: The URI or link to the relationship between the defensive tactic and the defensive technique in the d3fend ontology.
- def_tech: The URI or link to the defensive technique in the d3fend ontology.
- def_artifact_rel: The URI or link to the relationship between the defensive technique and the defensive artifact in the d3fend ontology.
- def_artifact: The URI or link to the defensive artifact in the d3fend ontology.
- off_artifact: The URI or link to the offensive artifact in the d3fend ontology.

- off_artifact_rel: The URI or link to the relationship between the offensive artifact and the defensive technique in the d3fend ontology.
- off_tech: The URI or link to the offensive technique in the d3fend ontology.
- off_tech_parent: The URI or link to the parent offensive technique in the d3fend ontology.
- off_tactic_rel: The URI or link to the relationship between the offensive tactic and the defensive technique in the d3fend ontology.
- off_tactic: The URI or link to the offensive tactic in the d3fend ontology.

Due to the fact that not all these columns are needed, specific ones are selected such as: query_def_tech_label, def_artifact_label, off_artifact_label, off_artifact_rel_label, off_tech_label, off_tech. Therefore, the data that off_tech includes is the link to attack id that is being used and the rest columns refer to the defend approach. Part of the link pattern is deleted and the attack id interconnection emerges.

| | query_def_tech_label | def_artifact_label | off_artifact_label | off_artifact_rel_label | off_tech_label | attack_id |
|---|---|---|---|---|---|---|
| 1 | Access Modeling | User Account | User Account | uses | Valid Accounts | 1078 |
| 2 | Access Modeling | Access Control Confi… | Access Control Confi… | modifies | File and Directory Per… | 1222 |
| 3 | Access Modeling | Access Control Confi… | Group Policy | modifies | Group Policy Modifica… | 1484 |
| 4 | Access Modeling | Access Control Confi… | Group Policy | reads | Group Policy Discovery | 1615 |
| 5 | Access Modeling | User Account | User Account | modifies | Account Access Rem… | 1531 |
| 6 | Access Modeling | User Account | User Account | uses | Valid Accounts | 1078 |
| 7 | Access Modeling | User Account | User Account | creates | Create Account | 1136 |
| 8 | Access Modeling | User Account | User Account | modifies | Account Manipulation | 1098 |
| 9 | Access Modeling | User Account | User Account | uses | Valid Accounts | 1078 |
| 10 | Access Modeling | User Account | User Account | creates | Create Account | 1136 |
| 11 | Access Modeling | Access Control Confi… | Group Policy | modifies | Group Policy Modifica… | 1484 |
| 12 | Access Modeling | User Account | User Account | uses | Valid Accounts | 1078 |
| 13 | Access Modeling | Access Control Confi… | Group Policy | accesses | Group Policy Prefere… | 1552.006 |
| 14 | Access Modeling | User Account | Local User Account | uses | Local Accounts | 1078.003 |
| 15 | Access Modeling | User Account | Default User Account | uses | Default Accounts | 1078.001 |
| 16 | Access Modeling | User Account | Default User Account | uses | Default Accounts | 1078.001 |
| 17 | Access Modeling | User Account | Domain User Account | uses | Domain Accounts | 1078.002 |
| 18 | Access Modeling | User Account | Default User Account | uses | Default Accounts | 1078.001 |
| 19 | Access Modeling | User Account | Cloud User Account | uses | Cloud Accounts | 1078.004 |
| 20 | Access Modeling | User Account | Default User Account | uses | Default Accounts | 1078.001 |
| 21 | Access Modeling | User Account | Local User Account | uses | Local Accounts | 1078.003 |
| 22 | Access Modeling | User Account | Local User Account | uses | Local Accounts | 1078.003 |
| 23 | Access Modeling | User Account | Domain User Account | uses | Domain Accounts | 1078.002 |

**36 defend-attack dataframe sample**

The affiliation between the attack and defend frameworks leads to 8.006 entries.

```
d3fend_attack_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8006 entries, 0 to 8005
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   query_def_tech_label    8006 non-null   object
 1   def_artifact_label      8006 non-null   object
 2   off_artifact_label      8006 non-null   object
 3   off_artifact_rel_label  8006 non-null   object
 4   off_tech_label          8006 non-null   object
 5   off_tech                8006 non-null   object
dtypes: object(6)
```

**37 defend-attack all relations info**

In addition, the last procedure is to connect the attack – defend relations with the datasets that include cpe-cve-cwe-capec-attack all relations and cpe-cve-cwe-capec-attack without empty connections, so that the final datasets are created.
The common key to this connection is the attack id, which is a common attribute between the datasets. The results are the following: 85.951.297 rows including missing relations and 57.070.042 rows without missing vectors. The outcome is 28.881.255 rows that have no connection among the datasets. Consequently, empty relations refer to a lack of or missing connections between two entities or concepts. In the context of MITRE's Common Platform Enumeration (CPE) and MITRE's Defend framework, it's important to note that these are separate initiatives with distinct purposes.

```
cpe_through_defend_all_rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 85951278 entries, 0 to 85951277
Data columns (total 10 columns):
 #   Column                Dtype
---  ------                -----
 0   CWEID                 object
 1   capec-id              int64
 2   CVEID                 object
 3   cpe                   object
 4   attack_id             object
 5   query_def_tech_label  object
 6   def_artifact_label    object
 7   off_artifact_label    object
 8   off_artifact_rel_label  object
 9   off_tech_label        object
dtypes: int64(1), object(9)
```

**38 cpe-cve-cwe-capec-attack-defend all relations info**

```
cpe_through_defend_without_empty_rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57070043 entries, 0 to 57070042
Data columns (total 10 columns):
 #   Column                 Dtype
---  ------                 -----
 0   CWEID                  object
 1   capec-id               int64
 2   CVEID                  object
 3   cpe                    object
 4   attack_id              float64
 5   query_def_tech_label   object
 6   def_artifact_label     object
 7   off_artifact_label     object
 8   off_artifact_rel_label object
 9   off_tech_label         object
dtypes: float64(1), int64(1), object(8)
```

**39 cpe-cve-cwe-capec-attack-defend without empty relations info**

| cpe-cve-cwe-capec-attack-defend info | | |
|---|---|---|
| **cpe through defend overall rows** | **cpe through defend without empty relations** | **cpe through defend no relation rows** |
| **85.951.297** | **57.070.042** | **28.881.255** |

**40 CPE-CVE-CWE-CAPEC-ATT&CK-DEFEND Info**

```
CWEID,capec-id,CVEID,cpe,attack_id,query_def_tech_label,def_artifact_label,off_artifact_label,off_artifact_rel_label,off_tech_label
CWE-1021,504,CVE-2005-2407,cpe:2.3:a:kayako:liveresponse:2.0:*:*:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2005-2407,cpe:2.3:a:kayako:liveresponse:2.0:*:*:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masquerade Task or Servic
CWE-1021,504,CVE-2005-2407,cpe:2.3:a:kayako:liveresponse:2.0:*:*:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2005-2407,cpe:2.3:a:kayako:liveresponse:2.0:*:*:*:*:*:*:*,1036.004,Platform Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2005-2407,cpe:2.3:a:kayako:liveresponse:2.0:*:*:*:*:*:*:*,1036.004,Scheduled Job Analysis,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2008-2716,cpe:2.3:a:xigla:absolute_form_processor_xe:4.0:*:*:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2008-2716,cpe:2.3:a:xigla:absolute_form_processor_xe:4.0:*:*:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masquerade T
CWE-1021,504,CVE-2008-2716,cpe:2.3:a:xigla:absolute_form_processor_xe:4.0:*:*:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or
CWE-1021,504,CVE-2008-2716,cpe:2.3:a:xigla:absolute_form_processor_xe:4.0:*:*:*:*:*:*:*,1036.004,Platform Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2008-2716,cpe:2.3:a:xigla:absolute_form_processor_xe:4.0:*:*:*:*:*:*:*,1036.004,Scheduled Job Analysis,Task Schedule,Task Schedule,modifies,Masquerade Task or Servi
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:open_xml_file_format_converter:*:*:mac:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or S
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:open_xml_file_format_converter:*:*:mac:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masq
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:open_xml_file_format_converter:*:*:mac:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:open_xml_file_format_converter:*:*:mac:*:*:*:*:*,1036.004,Platform Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:open_xml_file_format_converter:*:*:mac:*:*:*:*:*,1036.004,Scheduled Job Analysis,Task Schedule,Task Schedule,modifies,Masquerade Task
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2008:*:mac:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2008:*:mac:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masquerade Task or Servic
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2008:*:mac:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2008:*:mac:*:*:*:*:*,1036.004,Platform Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2008:*:mac:*:*:*:*:*,1036.004,Scheduled Job Analysis,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2004:*:mac:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2004:*:mac:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masquerade Task or Servic
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2004:*:mac:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2004:*:mac:*:*:*:*:*,1036.004,Platform Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2004:*:mac:*:*:*:*:*,1036.004,Scheduled Job Analysis,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:excel:2002:sp3:*:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:excel:2002:sp3:*:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:excel:2002:sp3:*:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:excel:2002:sp3:*:*:*:*:*:*,1036.004,Platform Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:excel:2002:sp3:*:*:*:*:*:*,1036.004,Scheduled Job Analysis,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2011:*:mac:*:*:*:*:*,1036.004,Asset Inventory,Digital Artifact,Task Schedule,modifies,Masquerade Task or Service
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2011:*:mac:*:*:*:*:*,1036.004,Asset Vulnerability Enumeration,Digital Artifact,Task Schedule,modifies,Masquerade Task or Servic
CWE-1021,504,CVE-2011-1244,cpe:2.3:a:microsoft:office:2011:*:mac:*:*:*:*:*,1036.004,Operating System Monitoring,Task Schedule,Task Schedule,modifies,Masquerade Task or Service
```

**41 Sample of the final dataset**

# 5 TEST CASES AND RESULTS

Now that the final dataset is generated, the need to inspect the final results and to extract a few metrics that will help in understanding the data better emerges. The dataset that includes the results without the empty relations is examined and the most common weakness for all the enumerated platforms is weakness with id 200 (CWE-200) that refers to the exposure of sensitive information to an Unauthorized Actor.

```
#find the most common weakness in the full dataframe
full_dataframe['CWEID'].mode()[0]
```

```
'CWE-200'
```

**42 Most common weakness in platform enumeration**

This weakness is also known as Information Exposure and it refers to cases where sensitive information is unintentionally revealed to unauthorized individuals or entities, possibly leading to security breaches or privacy violations. The impact of cwe-200 can range from minor to severe, depending on the nature and the sensitivity of the exposed information. These consequences may include unauthorized access to systems, identity theft, financial losses or reputational damage.

Furthermore, the top three weaknesses are:

- cwe-200: 30.201.875 occurrences (Information Exposure)
- cwe-20: 16.908.024 occurrences (Improper Input Validation)
- cwe-287: 3.105.040 occurrences (Improper Authentication)

```
# find the top 3 most common weaknesses in the full_dataframe
full_dataframe['CWEID'].value_counts().nlargest(3)
```

```
CWE-200    30201875
CWE-20     16908024
CWE-287     3105040
Name: CWEID, dtype: int64
```

**43 Top 3 Weaknesses in final dataset**

To continue, let's explain the rest two weaknesses. The weakness cwe-20 refers to the failure to properly validate input data before it is used by a software application. Improper input validation may lead to a variety of security vulnerabilities, such as injection attacks, denial of service, remote code execution.

As regards the weakness cwe-287, it refers to a weakness in software security known as "Improper Authentication". This occurs when a system fails to properly authenticate or validate the identity of a user / entity before granting access or performing an action. The consequences of cwe-287 may include unauthorized access to sensitive data, privilege escalation and unauthorized system alternations.

As a consequence, cwe-200, cwe-20 and cwe-287 while they may have different specific descriptions, they share similarities in terms of the impact and the potential consequences they might have on software security, including:

- Impact on Information Exposure: All three weaknesses involve the potential exposure of sensitive information. CWE-200 specifically focuses on information exposure as a broad category, while the CWE-20 and the CWE-287 represent specific subcategories within information exposure.
- Unauthorized Access to Information: Information Exposure (cwe-200), Improper Input Validation (cwe-20) and Improper Authentication (cwe-287) can all lead to unauthorized

access to sensitive data. CWE-200 focuses on various ways sensitive information can be unintentionally exposed, while CWE-20 and CWE-287 address vulnerabilities related to input validation and authentication, respectively, which can lead to obtain unauthorized information.

- Privacy and Security Risks: These weaknesses share the potential to introduce privacy and security risks. CWE-200 can lead to privacy breaches or unauthorized access. CWE-20 can result in data integrity issues, denial of service or remote code execution, where the CWE-287 may allow attackers to bypass authentication mechanisms, leading to unauthorized access and data exposure.

While the specific contexts and technical details of these weaknesses may differ, they all revolve around the information exposure and the impact it may have on software security and privacy.

Another result of the final dataset is to find the software system that has the most security controls. This software is the cpanel with 160.006 rows of vulnerabilities and therefore security measures, followed by Microsoft windows server 2012 R2 and linux kernel.

```
full_dataframe['cpe'].value_counts().nlargest(20)
```

```
cpe:2.3:a:cpanel:cpanel:*:*:*:*:*:*:*:*                    160006
cpe:2.3:o:microsoft:windows_server_2012:r2:*:*:*:*:*:*:*    147716
cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:*               132729
cpe:2.3:o:apple:iphone_os:*:*:*:*:*:*:*:*                  131952
cpe:2.3:o:microsoft:windows_server_2012:-:*:*:*:*:*:*:*    128789
cpe:2.3:o:microsoft:windows_10:1607:*:*:*:*:*:*:*          126930
cpe:2.3:o:microsoft:windows_10:-:*:*:*:*:*:*:*             124698
cpe:2.3:o:apple:mac_os_x:*:*:*:*:*:*:*:*                   120937
cpe:2.3:a:gitlab:gitlab:*:*:*:*:enterprise:*:*:*           109886
cpe:2.3:a:gitlab:gitlab:*:*:*:*:community:*:*:*            106952
cpe:2.3:o:microsoft:windows_7:-:sp1:*:*:*:*:*:*            103350
cpe:2.3:o:google:android:*:*:*:*:*:*:*:*                    96544
cpe:2.3:a:google:chrome:*:*:*:*:*:*:*:*                     94777
cpe:2.3:o:google:android:6.0:*:*:*:*:*:*:*                  92657
cpe:2.3:o:microsoft:windows_server_2008:-:sp2:*:*:*:*:*:*   92234
cpe:2.3:o:microsoft:windows_10:1703:*:*:*:*:*:*:*           91412
cpe:2.3:o:microsoft:windows_10:1511:*:*:*:*:*:*:*           91383
cpe:2.3:o:microsoft:windows_rt_8.1:-:*:*:*:*:*:*:*          90472
cpe:2.3:o:microsoft:windows_server_2016:-:*:*:*:*:*:*:*     85912
cpe:2.3:o:google:android:7.0:*:*:*:*:*:*:*                  83203
Name: cpe, dtype: int64
```

#### 44 Top 20 cpe through defend entities

Furthermore, the most common attack identification number is 1574.007 with 8.978.412 occurrences. This attack contains the Path Interception by PATH Environment Variable under the category Hijack Execution Flow.
Path Interception is a tactic used by adversaries to manipulate or redirect the search path (also known as the PATH environment variable) used by an operating system to locate executables or libraries. By manipulating the search path, attackers can trick the system into executing malicious code or hijack legitimate system binaries with malicious ones. The PATH environment variable is a system variable that contains a list of directories in which the operating system looks for executable files when a command is issued. The directories are typically separated by a colon (":") on Unix-based systems or a semicolon (";") on Windows systems.

```
full_dataframe['attack_id'].value_counts().nlargest(1)
```

```
1574.007    8978412
Name: attack_id, dtype: int64
```

**45 Most common attack id**

Attackers attempt to modify the PATH variable to include a directory under their control or modify the order of directories so that their malicious files take precedence over legitimate ones. When a user or system process invokes a command, the operating system searches for the executable in the directories specified in the PATH variable. If the attacker's directory is prioritized or the attacker has replaced a legitimate binary, the malicious code will be executed instead of the intended command. Path Interception attacks can have severe consequences, allowing adversaries to execute arbitrary code, gain unauthorized access, escalate privileges, or perform other malicious activities on a compromised system.

In order to examine the results that emerge from platform enumeration and attack techniques, a classification between the cpes and the attack ids is performed.

```
unique_couples_count = full_dataframe.groupby(['cpe', 'attack_id']).size().reset_index(name='Counter')
```

```
unique_couples_count.head()
```

| | cpe | attack_id | Counter |
|---|---|---|---|
| **0** | cpe:2.3:a:.matteoiammarrone:iamma_simple_galle... | 1036.001 | 19 |
| **1** | cpe:2.3:a:.matteoiammarrone:iamma_simple_galle... | 1539.000 | 12 |
| **2** | cpe:2.3:a:.matteoiammarrone:iamma_simple_galle... | 1562.003 | 29 |
| **3** | cpe:2.3:a:.matteoiammarrone:iamma_simple_galle... | 1574.006 | 51 |
| **4** | cpe:2.3:a:.matteoiammarrone:iamma_simple_galle... | 1574.007 | 57 |

**46 grouping cpe attack id**

A new column Counter is added, which represents the number of occurrences of each row in the data frame. The data frame includes 815.539 rows. For example, the attack with id 1036.001 is found 19 times inside the data frame. This column is helpful, in case of splitting the data frame into smaller ones, so to study more specific cpe categories such as applications, operating systems and hardware.

```
unique_couples_count.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 815539 entries, 0 to 815538
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   cpe         815539 non-null  object
 1   attack_id   815539 non-null  float64
 2   Counter     815539 non-null  int64
 3   attributes  815539 non-null  object
 4   count       815539 non-null  int64
dtypes: float64(1), int64(2), object(2)
memory usage: 31.1+ MB
```

**47 cpe - attack id relations dataframe**

To continue, the cpe – attack id data frame incorporates all the cpe categories, therefore a division into smaller data frames based on the platform type is needed.

The first split dataset contains all the cpe applications, reaching 642.063 rows and the most common attack in applications enumeration is T1574.007 (PATH Interception by PATH Environment Variable) counting 24.681 occurrences.

```
# cpe application with attack_id dataframe
```

```
cpe_a_df = pd.DataFrame()
cpe_a_df = unique_couples_count.loc[unique_couples_count['cpe'].str.contains("cpe:2.3:a")]
```

```
cpe_a_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 642063 entries, 0 to 642062
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   cpe         642063 non-null  object
 1   attack_id   642063 non-null  float64
 2   Counter     642063 non-null  int64
 3   attributes  642063 non-null  object
 4   count       642063 non-null  int64
dtypes: float64(1), int64(2), object(2)
memory usage: 29.4+ MB
```

**48 cpe applications - attack ids**

```
cpe_a_df.loc[cpe_a_df['Counter'] == cpe_a_df['Counter'].max()]
```

| | cpe | attack_id | Counter | attributes | count |
|---|---|---|---|---|---|
| **108767** | cpe:2.3:a:cpanel:cpanel:*:*:*:*:*:*:*:* | 1574.007 | 24681 | [cpe, 2.3, a, cpanel, cpanel, *, *, *, *, *, *... | 13 |

**49 Most common cpe platform - attack relation**

As a result, the most common attack technique is PATH Interception by PATH Environment Variable (T1574.007) on the cPanel application in all versions.

The second split includes the operating systems and the attack techniques. As a consequence, a new data frame with 129.544 rows occurs and the most common operating system is linux kernel with the attack technique T1574.007.

```
cpe_o_df = pd.DataFrame()
cpe_o_df = unique_couples_count.loc[unique_couples_count['cpe'].str.contains("cpe:2.3:o")]
```

```
cpe_o_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129544 entries, 685995 to 815538
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   cpe         129544 non-null  object 
 1   attack_id   129544 non-null  float64
 2   Counter     129544 non-null  int64  
 3   attributes  129544 non-null  object 
 4   count       129544 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 5.9+ MB
```

**50 cpe os - attack id connection info**

```
cpe_o_df.loc[cpe_o_df['Counter'] == cpe_o_df['Counter'].max()]
```

| | cpe | attack_id | Counter | attributes | count |
|---|---|---|---|---|---|
| **767909** | cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* | 1574.007 | 19209 | [cpe, 2.3, o, linux, linux_kernel, *, *, *, *,... | 13 |

**51 most common cpe(os) with attack id**

The last split refers to the hardware systems of the enumerated platforms and the newly created data frame consists of 43.932 rows, where the most common hardware occurrence is cisco.

```
# cpe hardware with attack_id dataframe
```

```
cpe_h_df = pd.DataFrame()
cpe_h_df = unique_couples_count.loc[unique_couples_count['cpe'].str.contains("cpe:2.3:h")]
```

```
cpe_h_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 43932 entries, 642063 to 685994
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   cpe         43932 non-null  object 
 1   attack_id   43932 non-null  float64
 2   Counter     43932 non-null  int64  
 3   attributes  43932 non-null  object 
 4   count       43932 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 2.0+ MB
```

**52 cpe hardware - attack id relation dataframe**

The usage of the above datasets may be helpful in several use case such as finding the most common attack for a particular operating system or an application. In this approach, we will find the most

common attack techniques for Microsoft operation system and the most common attack techniques for an application such as Apache.

As regards the Microsoft, there are 5.890 entries and the most common of Microsoft operating system is Microsoft windows server 2012 R2.

```
# create cpe_attack with microsoft OSs
```

```
cpe_microsoft_df = pd.DataFrame()
cpe_microsoft_df = unique_couples_count.loc[unique_couples_count['cpe'].str.contains("cpe:2.3:o:microsoft")]
```

```
cpe_microsoft_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5890 entries, 793129 to 799018
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   cpe         5890 non-null   object
 1   attack_id   5890 non-null   float64
 2   Counter     5890 non-null   int64 
 3   attributes  5890 non-null   object
 4   count       5890 non-null   int64 
dtypes: float64(1), int64(2), object(2)
memory usage: 276.1+ KB
```

**53 cpe microsoft info**

On the other hand, apache application consists of 19.514 entries and the most common apache functionality that is affected is traffic server functions.

```
cpe_apache_df = pd.DataFrame()
cpe_apache_df = unique_couples_count.loc[unique_couples_count['cpe'].str.contains("cpe:2.3:a:apache")]
```

```
cpe_apache_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19514 entries, 11431 to 30944
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   cpe         19514 non-null  object
 1   attack_id   19514 non-null  float64
 2   Counter     19514 non-null  int64 
 3   attributes  19514 non-null  object
 4   count       19514 non-null  int64 
dtypes: float64(1), int64(2), object(2)
memory usage: 914.7+ KB
```

**54 CPE apache info**

# 6 CONCLUSION and FUTURE WORK

CPE is a standardized method for describing and identifying platforms and software systems using a structured naming scheme. CPE is primarily focused on classifying and identifying the components of a system.

On the other hand, MITRE's Defend framework is a knowledge base that focuses on providing defensive techniques, tactics, and countermeasures to protect against cyber threats and attacks. It offers a comprehensive set of information and guidance for defending systems, networks, and data.

Since CPE primarily deals with identification and classification of software components, and Defend focuses on defensive techniques, the two frameworks do not have direct relations in terms of their objectives or specific mappings. However, it's possible that there could be some indirect relationships or connections between specific software components identified by CPE and the defensive techniques described in the Defend framework. These connections are established separately in the previous steps based on specific use cases or mapping efforts.

Relating datasets among various MITRE initiatives, such as CPE, CVE, CWE, CAPEC, ATT&CK, and DEFEND, serves several important purposes in the field of cyber security. These relationships help provide a comprehensive understanding of vulnerabilities, threats, and countermeasures, and enable stakeholders to effectively manage and mitigate risks. Here are some key reasons for relating these datasets:

- *Improved Vulnerability Management*: By establishing connections between CPE, CVE (Common Vulnerabilities and Exposures), CWE (Common Weakness Enumeration), and other frameworks, it becomes easier to track and manage vulnerabilities. CVE provides a standardized identifier for known vulnerabilities, CWE identifies common weaknesses that lead to vulnerabilities, and CPE helps identify the specific components and versions affected by vulnerabilities. Relating these datasets allows security professionals to understand the impact of vulnerabilities and prioritize their remediation efforts.
- *Threat Intelligence and Analysis*: Relating datasets such as CVE, CAPEC (Common Attack Pattern Enumeration and Classification), ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge), and DEFEND facilitates comprehensive threat intelligence. CVE provides information about specific vulnerabilities, CAPEC describes common attack patterns, ATT&CK outlines adversary techniques and tactics, and DEFEND provides defensive techniques and countermeasures. Linking these datasets helps analysts understand how vulnerabilities are exploited, the tactics employed by attackers, and the corresponding defensive measures.
- *Mitigation and Countermeasure Development*: By connecting datasets like CAPEC, ATT&CK, DEFEND, and others, organizations can develop effective mitigation strategies and countermeasures. CAPEC provides insights into known attack patterns, ATT&CK offers a comprehensive framework of adversary tactics and techniques, and DEFEND provides defensive techniques. Relating these datasets helps identify effective defensive strategies and develop proactive security measures to counter known attack patterns.
- *Cross-Referencing and Knowledge Sharing*: Establishing relationships between datasets encourages cross-referencing and knowledge sharing among different cybersecurity initiatives. It allows researchers, analysts, and practitioners to leverage the collective knowledge across multiple frameworks and benefit from a more holistic understanding of vulnerabilities, threats, and defense strategies.

- *Contextualization and Risk Assessment*: Relating datasets provides context and enhances risk assessment. Understanding how vulnerabilities, weaknesses, attack patterns, and defense techniques relate to each other allows for a more accurate assessment of potential risks and their impact on an organization's systems and assets. It enables security teams to prioritize their efforts and allocate resources effectively.

To conclude, relating datasets among CPE, CVE, CWE, CAPEC, ATT&CK and DEFEND improves vulnerability management, enhances threat intelligence, facilitates the development of effective countermeasures, fosters knowledge sharing, and enables better risk assessment. It supports a more comprehensive and proactive approach to cyber security.

# 7 ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Professor Mr. Kotzanikolaou P. for his invaluable guidance, unwavering support, and dedication throughout the duration of this project. His profound knowledge, insightful suggestions, and constructive feedback were instrumental in shaping the direction and enhancing the quality of our work. The wisdom and expertise he shared have been truly inspiring, and I am grateful for the opportunity to learn from him.

I would also like to extend my heartfelt appreciation to Mr. Gregoriades C. , the professor's assistant, for his significant contribution to this project. His tireless efforts, attention to detail, and willingness to assist have been instrumental in ensuring the smooth progress and successful completion of my work. Their commitment and expertise were invaluable in navigating the challenges we encountered, and I am immensely grateful for their assistance.

I would like to acknowledge the entire academic community at University of Piraeus, whose vibrant and intellectually stimulating environment has nurtured my academic growth. The opportunity to engage with fellow students, exchange ideas, and receive valuable feedback has been enriching and has significantly contributed to the development of this project.

Furthermore, I extend my appreciation to my family and friends for their unwavering support, encouragement, and understanding throughout this project. Their belief in my abilities and their constant motivation has been a source of strength, and I am truly grateful for their presence in my life.

Without the collective efforts, guidance, and support of all these individuals, this project would not have been possible. I am truly honored and humbled to have had the privilege of working with such remarkable people, and I am grateful for their contributions to the success of this endeavor.

# 8 REFERENCES


1. M. C. CPE, "*About CPE*," [Online]. Available: https://cpe.mitre.org/about/.
2. B. A. Cheikes, D. Waltermire and K. Scarfone, "*Common Platform Enumeration: Naming Specification Version 2.3*," National Institute of Standards and Technology.
3. nist, "*cpe specifications*," [Online]. Available: https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe.
4. "*CWE*," [Online]. Available: https://cwe.mitre.org/about/index.html.
5. M. CVE, "*cve*," [Online]. Available: https://cve.mitre.org/.
6. cve-redhat, "*what is cve*," [Online]. Available: https://www.redhat.com/en/topics/security/what-is-cve.
7. "*CVE Record Life*," [Online]. Available: https://www.cve.org/About/Process#CVERecordLifecycle.
8. oval, "*oval mitre*," [Online]. Available: https://oval.mitre.org/.
9. "*oval documents*," [Online]. Available: https://oval.mitre.org/about/documents.html.
10. Oval_Community, "*Open Vulnerability and Assessment Language*" Available: https://makingsecuritymeasurable.mitre.org/docs/oval-intro-handout.pdf
11. "*About Maec*," [Online]. Available: https://maecproject.github.io/about-maec/.
12. "*maec overview*," [Online]. Available: https://maecproject.github.io/documentation/overview/.
13. NVD, "*National Vulnerability Database*," [Online]. Available: https://nvd.nist.gov/.
14. "*cvss vulnerability metrics*," [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss.
15. "*cvss specification document*," [Online]. Available: https://www.first.org/cvss/specification-document.
16. P. Mell, K. Scarfone and S. Romanosky, "*cvss v2 guide*," [Online]. Available: https://www.first.org/cvss/v2/guide.
17. "*cwss*," September 2014. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
18. "*Kenna Getting Started Guide*," February 2022. [Online]. Available: https://help.kennasecurity.com/hc/en-us/article_attachments/4466606464404.
19. "*Kenna Security Implementation Success Guide*," [Online]. Available: https://docplayer.net/129526198-Kennasecurity-com-kenna-security-implementation-success-guide-get-up-running-in-under-an-hour.html.
20. "*new to capec*," [Online]. Available: https://capec.mitre.org/about/new_to_capec.html.
21. "*CAPEC Introductory Brochure*," 2013. [Online]. Available: https://makingsecuritymeasurable.mitre.org/docs/capec-intro-handout.pdf.
22. B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington and C. B.Thomas, "*MITRE ATT&CK: Design and Philosophy*," MITRE Corp, 2020.
23. K. Nickels, "*Getting Started with ATT&CK: Threat Intelligence*," 2019.
24. "enterprise matrices," [Online]. Available: https://attack.mitre.org/matrices/enterprise/.
25. A. Hewko, "*STRIDE Threat Modeling*," 2021. [Online]. Available: https://www.softwaresecured.com/stride-threat-modeling/.
26. "*DAO*," [Online]. Available: https://d3fend.mitre.org/dao/.
27. "*D3FEND Matrix*," [Online]. Available: https://d3fend.mitre.org/.
28. P. E. Kaloroumakis and M. J. Smith, "*Toward a Knowledge Graph of Cyber Security Countermeasures*," MITRE Corp.

29. P. Gao, X. Liu, E. Choid, S. Ma, X. Yang, Z. Ji, Z. Zhang, D. Song and V. Tech, "*THREATKG: A Threat Knowledge Graph for Automated Open-Source Cyber Threat Intelligence Gathering and Management*," 2022.
30. K. Kanakogi, H. Washizaki, Y. Fukazawa, S. Ogata, T. Okubo, T. Kato, H. Kanuka, A. Hazeyama and N. Yoshioka, "*Comparative Evaluation of NLP-Based Approaches for Linking CAPEC Attack Patterns from CVE Vulnerability Information*," 2022.