VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

Ramil Mustafayev

# AN EARLY WARNING AND ALERT SYSTEM FOR SOFTWARE VULNERABILITY ASSESSMENT

Master Graduation Thesis

Information and information technologies security study programme, state code 6211BX014

Informatics engineering study field

Vilnius, 2020

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

APPROVED BY
Head of Department

____________________
(Signature)

____________________
(Name, Surname)

____________________
(Date)

Ramil Mustafayev

# AN EARLY WARNING AND ALERT SYSTEM FOR SOFTWARE VULNERABILITY ASSESSMENT

Master Graduation Thesis

Information and information technologies security study programme, state code 6211BX014

Informatics engineering study field

| | | | |
|---|---|---|---|
| **Supervisor** | (Title, Name, Surname) | (Signature) | (Date) |
| **Consultant** | (Title, Name, Surname) | (Signature) | (Date) |
| **Consultant** | (Title, Name, Surname) | (Signature) | (Date) |

Vilnius, 2020

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF FUNDAMENTAL SCIENCES
DEPARTMENT OF INFORMATION SYSTEMS

APPROVED BY
Head of Department

Informatics engineering study field

Information and information technologies security study programme, state code 6211BX014

______________________
(Signature)

______________________
(Name, Surname)

______________________
(Date)

**OBJECTIVES FOR MASTER THESIS**

......………..No. ................

Vilnius

For student Ramil Mustafayev
(Name, Surname)

Master Thesis title: An Early Warning and Alert System for Software Vulnerability Assessment

Approved on ..........................................., 2020 by Dean‘s decree No. ................
(day, Month) (year)

The Final work has to be completed by ................................................, 2020
(Day, Month) (Year)

THE OBJECTIVES:

This work aims to propose automated an early warning and alert system about newly published vulnerabilities related to software products.

Objectives:

1. To perform literature analysis on software vulnerability analysis/assessment;
2. To investigate the methods and techniques that are used for software vulnerability analysis in existed solutions;
3. To propose a concept and create an initial version of the system;
4. To perform tests and experiments to evaluate the proposed idea.

Consultants of the Master Thesis: …………………………………………………………………….……………………
…………………………………………………………………………………………………………………………………
(Title, Name, Surname)

Academic Supervisor ............................... .........................................................................
(Signature) (Title, Name, Surname)

Objectives accepted as a quidance for my Master Thesis

…………………………………..
(Student‘s signature)

…………………………………..
(Student‘s Name, Surname)

……………………………..……...
(Date)

| | |
|---|---|
| Vilniaus Gedimino technikos universitetas<br>Fundamentinių mokslų fakultetas<br>Informacinių sistemų katedra | ISBN ISSN<br>Kopijos Nr. ………<br>Data ….-….-…. |

Informacijos ir informacinių technologijų saugos studijų programos magistro darbas.

Pavadinimas: Išankstinio įspėjimo sistema programinės įrangos pažeidžiamumų įvertinimui

Autorius **Ramil Mustafayev** Akademinis vadovas **Vitalijus Gurčinas**

**Baigiamojo darbo kalba**

[ ] Lietuvių

[✓ Užsienio (Anglų)]

## Anotacija

Baigiamajame magistro darbe buvo pasiūlyta išankstinio perspėjimo sistema (EWAS) programinės įrangos pažeidžiamumų įvertinimui ir sukurtas sistemos prototipas. Pirmame darbo etape buvo atliktas teorinis tyrimas: buvo surinkta ir išanalizuota aktuali informacija, išsiaiškintos sistemos kūrimo galimybės ir apribojimai, taip pat parinkti tinkamiausi algoritmai EWAS sistemai.

Remiantis išanalizuota informacija, buvo parengtas karkasas, kurį sudaro šie elementai: duomenų surinkimo, duomenų apdorojimo, pažeidžiamumų nustatymo, išankstinio perspėjimo ir rezultatų pateikimo.

Praktinei daliai atlikti buvo pasitelkta prototipų kūrimo metodika, lyginamasis ir eksperimentinis tyrimai. Projekto rezultatai buvo įvertinti naudojant „OWASP Benchmark“ sistemą ir išbandyti atsižvelgiant į funkcinius reikalavimus. Darbo metu gauti rezultatai yra apibendrinti, pateikti komentarai ir įžvalgos, taip pat nurodyti sistemos apribojimai.

Bendras prototipo įvertinimo rezultatas, nustatant pažeidžiamumus, buvo 92 procentai. Sistema aptiko 2270 pažeidžiamumų, o 196 iš jų nurodė klaidingai, t.y. įdentifikavo, kad pažeidžiamumai yra, nors jų nebuvo. Be to, sistema sugebėjo nustatyti ir pateikti 15 išankstinių įspėjimų, susijusių su naujai paskelbtais pažeidžiamumais aptiktais komerciniuose produktuose.

Darbą sudaro: įvadas, susijusių darbų analizė, siūlomas sprendimas, eksperimentai ir pirminės sistemos versijos įvertinimas, išvados, naudotos literatūros sąrašas.

Dokumento apimtis - 75 psl., be priedų, 37 iliustracijos, 10 lentelių, 39 bibliografiniai šaltiniai.

**Raktažodžiai:** pažeidžiamumai, pažeidžiamumų vertinimas, pažeidžiamumų vertinimo sistemos, išankstinio perspėjimo, pažeidžiamumų nustatymas, įrankiai, kibernetinis saugumas

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Information Systems

ISBN ISSN
Copies No. ………
Date ….-….-….

Information and information technologies security study programme master thesis.

Title: An Early Warning and Alert System for Software Vulnerability Assessment

Author **Ramil Mustafayev** Academic supervisor **Vitalijus Gurčinas**

**Thesis language**

|  |  |
|---|---|
|  | Lithuanian |
| x | Foreign (English) |

**Annotation**

In the final master's thesis, an early warning and alert system (EWAS) for software vulnerability assessment was proposed and prototype of the system was created. In particular, a theoretical study was carried out: relevant information was collected, analyzed and research opportunities were identified, as well as the most suitable algorithms for EWAS were chosen.

Based on the analyzed information, a framework was prepared, which consists of phases of the data collection and parsing of data, vulnerability identification and an early warning selection, populating data for web-based console.

For the practical part, prototyping methodology, comparative and experimental study were chosen. Project results were evaluated using the OWASP benchmark scoring system and tested against functional requirements. The obtained results are summarized, comments, insights and possible limitations are provided.

The overall, benchmark score of the prototype in identification of vulnerabilities was 92%, and the system detected 2270 vulnerabilities and 196 of them were false positives. Additionally, the system was able to identify 15 early warnings related to newly published vulnerabilities before the commercial products.

Structure: introduction, related works analysis, proposed solution, experiments and evaluation of initial system, conclusions and references.

Thesis consist of: 75 p. text without appendixes, 37 pictures, 10 tables, 39 bibliographical entries.

**Keywords:** vulnerability, assessment, management, systems, early warning, alert, vulnerability identification, tools, cybersecurity

# CONSENT TO THE USE OF PERSONAL DATA

25/05/2020
(Date)

With this consent I, Ramil Mustafayev (hereinafter referred to as the Data subject) agree to the processing of my personal data for the learning purposes of other students by Vilnius Gediminas Technical University, legal entity code 111950243, address – Saulėtekio al. 11, LT-10223 Vilnius (hereinafter referred to as the Data controller). I agree to the processing of the following personal data (mark the relevant item):

- ☐ name, surname, Bachelor’s final thesis;
- ☐ Bachelor’s final thesis without name and surname;
- ☒ name, surname, Master’s final thesis;
- ☐ Master’s final thesis without name and surname.

Personal data processed for this purpose will not be transferred by the Data controller to any third parties, students will be allowed to access the final theses in the internal information system.

The personal data of the Data subject will be used for this purpose for a maximum of 5 years.

By this consent, the Data subject confirms that he / she is aware of the following rights:

- Access to his/her data and how it is processed (right of access);
- Require the rectification or, depending on the purposes of the processing of personal data, the addition of incomplete personal data (right of rectification);
- Destroy his/her data or suspend the processing of data (excluding storage) (right to destroy and right to be "forgotten");
- Require the Data controller to restrict the processing of personal data (right to restrict);
- The right to transfer data (right to transfer);
- Do not consent to the processing of personal data when such data are processed or are intended to be processed for other purposes;
- File a complaint with the State Data Protection Inspectorate.

*The Data subject has the right to withdraw his consent at any time.*

*[Name, surname and signature of Data subject]*

Data controller details:
Vilnius Gediminas Technical University
Legal entity code: 111950243
Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania
Tel. (+370 5) 274 5030
E-mail: vgtu@vgtu.lt
VAT payer code: LT119502413
Data protection officer tel. (+370 5) 251 2191, e-mail: dap@vgtu.lt

**(the document of Declaration of Authorship in the Final Degree Project)**

**VILNIUS GEDIMINAS TECHNICAL UNIVERSITY**

Ramil Mustafayev, 20185915

(Student‘s given name, family name, certificate number)

Faculty of Fundamental Sciences

(Faculty)

Information and information technologies security study programme, ITSfmc-18

(Study programme, academic group no.)

# DECLARATION OF AUTHORSHIP IN THE FINAL DEGREE PROJECT

____________________

(Date)

I declare that my Final Degree Project entitled An Early Warning and Alert System for Software Vulnerability Assessment is entirely my own work.

The title was confirmed on ____________________ by Faculty Dean’s order

(Date)

No. ____________. I have clearly signalled the presence of quoted or paraphrased material and referenced all sources.

I have acknowledged appropriately any assistance I have received by the following professionals/advisers: ____________________________________________

____________________________________________

____________________________________________

The academic supervisor of my Final Degree Project is Vitalijus Gurčinas.

No contribution of any other person was obtained, nor did I buy my Final Degree Project.

| | |
|---|---|
| ____________________ | RAMIL MUSTAFAYEV |
| (Signature) | (Given name, family name) |

# Table of Content

1. Introduction....12
   - 1.1. Investigation Object....13
   - 1.2. The aim of work and tasks....13
   - 1.3. The novelty of the Topic....13
   - 1.4. The relevance of the topic....14
   - 1.5. Research Methodology....14
   - 1.6. Scientific Value of the Work....14
   - 1.7. Work Results....14
   - 1.8. Structure of the Work....15
2. Related works analysis....16
   - 2.1. Vulnerability Management....16
     - 2.1.1. Vulnerability Assessment....16
     - 2.1.2. Types of Vulnerability Assessments....17
     - 2.1.3. Benefits of Vulnerability Assessment....19
   - 2.2. Overview of Vulnerability Management Systems and Its Components....20
     - 2.2.1. Vulnerability Databases and Dictionaries....24
     - 2.2.2. Using CVE and CPE in a VMS....28
     - 2.2.3. Challenges and Limitations in VMS....29
   - 2.3. Importance of Early Warning Systems in IT....32
   - 2.4. Conclusions of the second chapter....33
3. Proposed solution....34
   - 3.1. Project Overview....34
   - 3.2. Objectives of the system....36
   - 3.3. Dynamic behaviour of the system....36
   - 3.4. Components and their definitions....42
   - 3.5. Design of Database for Storing CVEs and CPEs....43
   - 3.6. System requirements and specification....45
   - 3.7. Technical Challenges and Handling Limitations....49
   - 3.8. Conclusions of the third chapter....52
4. Experiments and Evaluation of the EWAS....53
   - 4.1. Implemented Environment....53
   - 4.2. Results of Data Collection from NVD Feeds and Analysis....54
   - 4.3. Identified Vulnerabilities....58
   - 4.4. Identified Early Warnings....61
   - 4.5. Comparison of results between related work and thesis....63
   - 4.6. Evaluation Approach based on OWASP benchmark....64
   - 4.7. The graphical user interface of the system (Web-based console)....66
   - 4.8. Conclusions of the fourth chapter....70
5. Conclusions....72
6. References....73

# List of Images and Figures

Figure 1. Vulnerability Scanner System Diagram ........ 21
Figure 2. CVE example from MITRE related to Google Chrome ........ 25
Figure 3. Vulnerabilities related to Google Chrome from the NVD official site ........ 26
Figure 4. Modified and Recent feeds by NVD ........ 26
Figure 5. Example of META file from NVD ........ 27
Figure 6. CPE entry for Google Chrome 80.0.3987.149 in the official CPE dictionary ........ 27
Figure 7. NVD Vulnerability Severity Ratings from the official site ........ 28
Figure 8. CVE feed in JSON format from NVD ........ 29
Figure 9. CVE example for awaiting reanalysis from the NVD ........ 30
Figure 10. CVE-2020-11470 from the official Tenable website ........ 31
Figure 11. Proposed Framework ........ 35
Figure 12. Collecting data from NVD ........ 37
Figure 13. Collecting data from assets with Agents ........ 38
Figure 14. Parsing collected NVD data on the local database ........ 39
Figure 15. Vulnerability Identification Matcher ........ 40
Figure 16. User interaction with GUI (Web-based console) ........ 41
Figure 17. Example of EER Diagram of the proposed solution ........ 44
Figure 18. Example of Query with Natural Language Mode for CVEs and Results ........ 50
Figure 19. Example of Query with Natural Language Mode for CPE identifiers and Results ........ 51
Figure 20. Example of FuzzyWuzzy usage ........ 51
Figure 21. Testing environment ........ 53
Figure 22. The Most Vulnerable Vendors ........ 55
Figure 23. The Most Vulnerable Products ........ 56
Figure 24. Gathered information by attributes ........ 56
Figure 25. CVSS Severity Distribution Over Time ........ 57
Figure 26. Published vulnerabilities by year ........ 58
Figure 27. Comparison of False and True Positive results ........ 59
Figure 28. Early Warning related to Zoom Meetings with version 4.6 ........ 62
Figure 29. CVE-2020-11500 from the Tenable ........ 62
Figure 30. CVE-2020-11500 from the NVD ........ 63
Figure 31. Comparison results between thesis results and related work ........ 64
Figure 32. The dashboard of the system ........ 66
Figure 33. Creating Alerts ........ 67
Figure 34. Alert List ........ 68
Figure 35. Example reports of the result ........ 68
Figure 36. Example of found vulnerabilities by-products ........ 69
Figure 37. Example of found vulnerabilities related to Google Chrome v78.0.3904.108 ........ 70

# List of Tables

Table 1. Vulnerability Scanners........................................................................................21
Table 2. Used Components and Their Definitions...................................................................42
Table 3. Functional Requirements .........................................................................................46
Table 4. Non-functional Requirements...................................................................................47
Table 5. Security Requirements.............................................................................................48
Table 6. List of installed software products in the lab environment.......................................54
Table 7. Last ten collected CVEs............................................................................................55
Table 8. Found vulnerabilities by system ...............................................................................58
Table 9. Example of Found vulnerabilities related to Mozilla Firefox v48.0.2......................60
Table 10. Comparison results between thesis results and related work..................................63

## Abbreviations

**VM** – Vulnerability Management

**IDS** – Intrusion Detection System

**VE** – Virtual Environments

**VMM** – Virtual Machine Manager

**IT** – Information Technology

**CVE** – Common Vulnerabilities and Exposures

**NVD** – National Vulnerability Database

**CRR** – Cyber Resilience Review

**OSVDB** – Open Source Vulnerability Database

**NIST** – National Institute of Standards and Technology

**CVSS** – Common Vulnerability Scoring System

**VAT** – Vulnerability Assessment Tool

**PC** – Personal Computer

**CPE** – Common Platform Enumeration

**EWAS** – Name of a prototype for the proposed idea

**SCAP** – Security Content Automation Protocol

**NIST** – National Institute of Standards and Technology

**CWE** – Common Weakness Enumeration

**IP** – Internet Protocol

**TCP –** Transmission Control Protocol

**ASV** – Approved Scanning Vendor

# 1. Introduction

Cyber threats are growing sharply, and several cyber incidents are taking place every second/minute; thus, hundreds of computers are compromised due to the software vulnerabilities every day, and data breaches are continuing to increase. For example, if user persuaded for opening software sent by cybercriminals with social engineering like a phishing attack, malicious codes attached to software exploit the vulnerability. As a result, the user computer is being infected with malware. According to an analysis of the National Vulnerability Database (NVD) data feeds (*NVD Data Feeds*, n.d.), there were 18,154 disclosed vulnerabilities in 2018. Still, in 2019 those numbers are dramatically increased, and 18,938 vulnerabilities were published. As seen from the statistics, weaknesses are a never-ending problem, especially in the context of software products. Although even known and discovered vulnerabilities are published, they are still a high risk to organizations. Therefore, companies are trying to choose the best security approaches to prevent and investigate feasible scenarios exploited by attacks and amend vulnerabilities before compromising. Therefore, companies are trying to select the best security approaches to prevent and investigate potential scenarios used by attacks and prevent vulnerabilities before compromising.

There are several ways to ensure, test and verify the security level of organizations. One of the most generic and effective ways to implement security measures is a vulnerability assessment. The process covers identification, analysis and ranking weaknesses in a computer/network environment. Additionally, it provides IT personnel and management team with adequate knowledge about existing threats in the environment. During the vulnerability, assessment organizations deploy Vulnerability Management Systems (VMSs) in their computer or network environment. Among the critical functionalities of VMS is scanning installed vulnerable software products inside of IT assets. For accomplishing these operations, vulnerability scanners collect asset data from the computer system and correlate them with information obtained from repositories such as private vendors or NVD. However, scientific sources demonstrate that due to shortcomings of NVD, using it as a core source might give false-negative results (Sanguino & Uetz, 2017). Also, reacting to vulnerabilities by the organizations on time is one of the critical aspects in terms of security (Ruohonen, 2019). For instance, the presence of newly released vulnerabilities without identifiers or detection plugins on the VMS complicates realistic vulnerability research, brings out false-negative results and distracts organizations from the initial objective. One of the ways to eliminate limitations of VSMs is choosing an appropriate concept based on an early warning, and the alert system is considered.

## 1.1. Investigation Object

The Investigation Object is a software vulnerability analysis/assessment.

## 1.2. The aim of work and tasks

This work aims to propose automated an early warning and alert system about newly published vulnerabilities related to software products.

1. To perform literature analysis on software vulnerability analysis/assessment;
2. To investigate the methods and techniques that are used for software vulnerability analysis in existed solutions;
3. To propose a concept and create an initial version of the system;
4. To perform tests and experiments to evaluate the proposed idea.

## 1.3. The novelty of the Topic

Implementation of vulnerability analysis and assessment is mainly focused on applying third party solutions, which synchronizes CPE identifiers with National Vulnerability Database information regularly. If new, missing or corrected CPE identifiers exist, vendors are modifying their content manually by human interaction accordingly (*Requirements and Recommendations for CVE Compatibility*, 2017). Being dependent on third party solutions and human control changes the vulnerability assessment process to disadvantage and makes organisations less resistance to new disclosures.

Responding to vulnerabilities by the organisations on time is one of the crucial aspects in terms of security. (Ruohonen, 2019). For instance, the existence of newly published vulnerabilities without CPE identifiers or plugins for detection makes practical vulnerability analysis complicated, turns results to false-negatives and distracts organisations from the initial aim.  In this case, organisations should wait until the vulnerability is analyzed and identifiers assigned by the NVD or third-party solutions. The proposed concept is a solution for handling the cons of vulnerability assessment mentioned above. It will help individuals or organisations to respond fast to active threats and create compensation measures on time, especially in terms of software products.

## 1.4. The relevance of the topic

There are still some open issues regarding vulnerability analysis and methods. Getting late notifications related to the newest vulnerabilities are considered a significant threat to the companies. In the worst-case scenario – the cyber-attack can occur on the same day when the weakness is discovered in software or mentioned on the vulnerability databases. At the moment, evaluation of methods shows that the vulnerability analysis tools lack synchronization of CPE identifiers with NVD which prone to losing time and getting false negatives because of not published identifiers and plugins on the vulnerability databases. Besides, for assigning CPEs, human interaction is needed during the analysis of vulnerabilities (Sanguino & Uetz, 2017). Thereby, automated tools are led to making errors due to NVD shortcomings such as CPE synchronizations, missing known affected software configurations, typographical errors and assignments of identifiers manually.

## 1.5. Research Methodology

For the analytical part of the work, literature review, analysis of scientific articles and library research methodology were used. For the practical part, prototyping methodology and comparative and experimental study were chosen. Project results were evaluated using the OWASP benchmark scoring system and tested against functional requirements.

## 1.6. Scientific Value of the Work

The proposed concept for an early warning-system ensures to get updates and alerts regarding newly published vulnerabilities, while mainly focuses on the creation of an automated tool with decreasing human interaction.

This content can be used for further studies related to vulnerability assessment on software products or as a new solution for existing vulnerability assessment tools.

## 1.7. Work Results

Analysis of related literature and existing vulnerability management platforms shows that almost all current methodologies and concepts lack the adaptability when require to warn individuals or companies regarding newly published vulnerabilities.

## 1.8. Structure of the Work

The first section contains the introduction.

The second section describes the analyzed literature.

The third section describes the proposed solution for EWAS.

The fourth section examines the experiments and evaluates the prototype of the system.

The fifth is conclusions of all chapters.

# 2. Related works analysis

This chapter includes the associated works done before. In more details, this section provides background on the early warning systems in IT, vulnerability, assessment, requirements and its benefits. Also, describes vulnerability management systems, vulnerability databases and dictionaries which can be used as a data source in VMS. Moreover, stresses problems and potential risks in the vulnerability management program.

## 2.1. Vulnerability Management

Vulnerability management is an integral and essential part of an excellent successful program. Virtually, industry standards and regulatory organizations are frequently citing a well-functioning vulnerability management framework, including testing and remediation, as a vital requirement and compulsory for security compliance. Vulnerability is characterized as a computer or network environment-related weakness which may be compromised by an intruder or threat (NIST, 2012). As a practical matter, every technological deficiency cannot be eliminated from an environment. There are several reasons for this. Some of the vulnerabilities remain latent until they will be revealed and publicly available; These types of vulnerabilities are referred to a zero-days till the public disclosure. Other vulnerabilities can stay due to difficulties related to patching devices, including those which support legacy applications, or which are controlled by third parties. Still, other vulnerabilities might be cost-prohibitive to address for different reasons. That means that every single system has at any given time would have several latent vulnerabilities, and it will create a risk to organizations. In the field of information security, the fundamental risk factors can be calculated according to the following mathematical formula:

$$Level\ of\ Risk = \frac{(Threats \times Vulnerabilities)}{Countermeasures} \times Impact \text{ (Mladenović, 2017)}$$

"*Impact*" in this formula expresses the outcome of the loss to the owner of the asset. The value of "*Threat x Vulnerability*" depicts the likelihood that the unwanted event occurs, and "*Countermeasures*" means an action, tool, process, or method that reduces a threat, weakness, or attack by removing or blocking it, by reducing the damage it can cause, or by detecting and disclosing it so that remedial step can be implemented.

### 2.1.1. Vulnerability Assessment

A vulnerability assessment is a method of finding and evaluating vulnerabilities that may exist within the organization (Mladenović, 2017). Assessment of vulnerability is a critical component of network protection and risk management process. TCP/ IP networks have overgrown over the last decades. Computer weaknesses and malicious exploitation are increased along with the appearance of this development. Updates to the operating system, vulnerability fixes, malware repositories and security releases are becoming a vital tool for any experienced network admins or security team. It is the implementation of patches and the use of information acquired from these tools that make the difference between a secure network system and a network used for malicious hacker attacks as a backdoor playground. As a system baseline analysis, routine vulnerability assessments need to be undertaken and tailored to the company's needs to maintain a comparatively secure network system.

Vulnerability assessment today relies on manual intervention by vulnerability researchers, vulnerability database owners, and systems/network administrators (Waltermire & Fitzgerald-McKay, 2018). Vulnerability assessment is usually carried out using network-based or host-based approaches via automated tools to perform discovery, analysis and reporting. Manual techniques may also be used to detect vulnerabilities related to technological, physical and governance-based weaknesses. Assessments of vulnerabilities are not exploitative by its nature (in contrast with ethical hacking or penetration tests). Practitioners (or the tools and methods they employ) do not usually exploit vulnerabilities which they discovered during a vulnerability assessment. Alternatively, a vulnerability assessment follows slightly different purposes: it helps an organization to focus on surveillance and identification of vulnerabilities in its environment. There is usually no need to exploit a potential vulnerability to recognize its presence and implement a patch.

### 2.1.2. Types of Vulnerability Assessments

NIST Special Publication 800-115, “Technical Guide to Information Security Testing and Assessment” (Scarfone et al., 2008) is a framework for cybersecurity testing and evaluation techniques. The standard addresses the following four practices related to vulnerability assessment:

1. Network-based scans;
2. Host-based scans;
3. Wireless scans;
4. Application scans;

During the vulnerability assessment targets do not need to know about tests and stealth techniques which allows the tester to avoid detection. Both internal and external systems can be scanned during the assessment. Practitioners may determine techniques for assessment such as authenticated testing in which validated credentials are given to the scanning tool to analyze the device or application at a deeper level. Technical vulnerability assessments are usually automated, but practitioners should be involved during the process for planning, implementing and evaluating outcomes.

**Network-based scans** - covers service and host discovery with an enumeration of vulnerabilities. The identification element of network-based scans enables the assessors to identify and define the type and possible attack vectors for each network device. In order to identify the type, the scanning tool is starting probe a host. It examines its functioning and replies with creating a "fingerprint" which includes device information and enables the tool to discover the characteristics of the machine. In this way, tools might enumerate running services, scan for a range of listening TCP ports, examine system banners or deploy any number of other techniques to determine the type and version of the host or device. Modern scanning tools can accurately discover the computer or network system applications of the target for all but the stealthiest of appliances, and from those features conduct targeted tests to find vulnerabilities. For instance, if a tool has detected that a target has a Web Server (IIS), it could run a set of tests based on known vulnerabilities and misconfigurations of that operating system and application.

**Host-Based Scans** - Network-based scans often skip vulnerabilities that can only be exploited by a user logging into the device (i.e. local exploits) because they can only have the functionality or be set up to check for "remotely exploitable" vulnerabilities (i.e. vulnerabilities that can be accessed from anywhere else on the network). In comparison, host-based scans are performed from the target machine or are managed remotely with authenticated access to the target nodes. These scans can provide more accuracy in the configuration settings and patch information of a device while covering ports and services that are also observable for network-based scans. Host-based scans are more detailed than network-based scans; however, the comprehensiveness usually raises their overhead and makes them more challenging to configure and run. Numerous network-based scanning tools also have an authenticated scanning method, which is likely to deploy the SCAP and support with the National Checklist Program (NCP) operated by NIST (Quinn et al., 2018).

**Wireless Network Scans** - Scanning of Wi-Fi network of organizations mainly focuses on the pattern of attack in a wireless network environment. One aspect of wireless network testing is validating that an enterprise’s systems are appropriately configured and secured. Another aim of the wireless testing is to detect rogue access points which mimic legitimate wireless networks where users can be tricked into joining the intruders' network. Enterprises should consider that internal systems can be connected to the guest wireless networks, and an intruder can target systems joined to a guest network and jump from there into internal networks. Although guest networks can be isolated, vulnerable devices connected to those networks can be an attack vector for hackers.

**Application Scans** - usually concentrate on websites to detect and enumerate flaws and misconfigurations in the software. Assessors often use manual checks or exploit kits during penetration testing; however, techniques like software-centric Dynamic Application Security Testing (DAST) help to detect web-specific vulnerabilities, such as cross-site scripting (XSS), SQL injection, inadequate input validation, and sensitive data exposure. Also, network vulnerability scanning tools have security checks for web applications, but they have fewer features that concentrate on checking weaknesses in web applications than DAST instruments. However, application security testing can be a severe problem since during testing, scanning software can alter databases or remove content, so companies should either limit testing on non-production environments or be careful about scanning production environments.

### 2.1.3. Benefits of Vulnerability Assessment

Regular vulnerability assessment (VA) is a way to secure IT assets, preserve an awareness of the sensitivities in an environment and react quickly to reduce potential threats. Organizations can be supplied with a comprehensive vulnerability assessment program on the knowledge, awareness, and risk background necessary to comprehend threats to their environment and react accordingly. Benefits of the vulnerability assessment include:

- Security benefits;
- Compliance benefits;

**Security Benefits** - Vulnerability assessments are contained initially on the suggested measures of industry systems and best practice guidance, and as the vulnerability assessment, security advantages are plenty. For example, the Center for Internet Security® (CIS), as one of the first five CIS, lists regular vulnerability assessment and remediation controls, diminishing the majority of an enterprise's security risk. Similarly, the determination and documentation of asset vulnerabilities are required by the NIST Cyber-security Framework in its Identify category of controls. In addition to target remediation plans, vulnerability assessment outcomes can be operated to point out systematic issues, such as missing patches in patch management or life cycle of asset management.

**Compliance Requirements** – these requirements are comprised of two categories: compulsory and noncompulsory. Compulsory drivers are government or industry mandated requirements that an enterprise must follow as a law or regulation. The General Data Protection Regulation (GDPR) and the recent Cybersecurity Requirements for Financial Services Companies regulation from the New York State Department of Financial Services are implied as common examples of compulsory compliance objectives. The industry-mandated compulsory requirement example is PCI DSS. PCI DSS is not a government-mandated; all credit card merchants and providers are contractually compelled to conform the PCI DSS. Though noncompulsory drivers are not mandated; but, for some enterprises, expectations from customers, clients or business partners may follow standards, such as ISO/IEC 27001:2013 Information technology – Security techniques – Information security management systems – Requirements, NIST SP 800-53 Security and Privacy Controls for Federal Information Systems and Organizations.

## 2.2. Overview of Vulnerability Management Systems and Its Components

A vulnerability management system can ease the discovery, test and remediation of problems, and in this way, help enterprises become aware of the importance of vulnerability assessment. The Vulnerability Management field centres upon the process that organizations determine, test, and control vulnerabilities in a crucial IT or operational environment. One of the critical components in organizing for and identifying the proper implementation for controls and risk management is vulnerability management. It is rational to say that the centre of cyber resilience is vulnerability management. During the vulnerability management process, organizations may often identify vulnerabilities that lead to improving requirements and criteria for tests. The organization improves, applies and evolves the controls that reduce the impact of a threat during the control management process (Carnegie Mellon University for the operation of the Software Engineering & Institute, 2016).

Primarily, organizations are deploying the vulnerability scanners or devices which automate security inspection and can operate an essential role in cybersecurity by scanning computer/network system and web applications for several security risks during Vulnerability Management processes. An example of the common architecture of a vulnerability scanner can be seen in Figure 1.

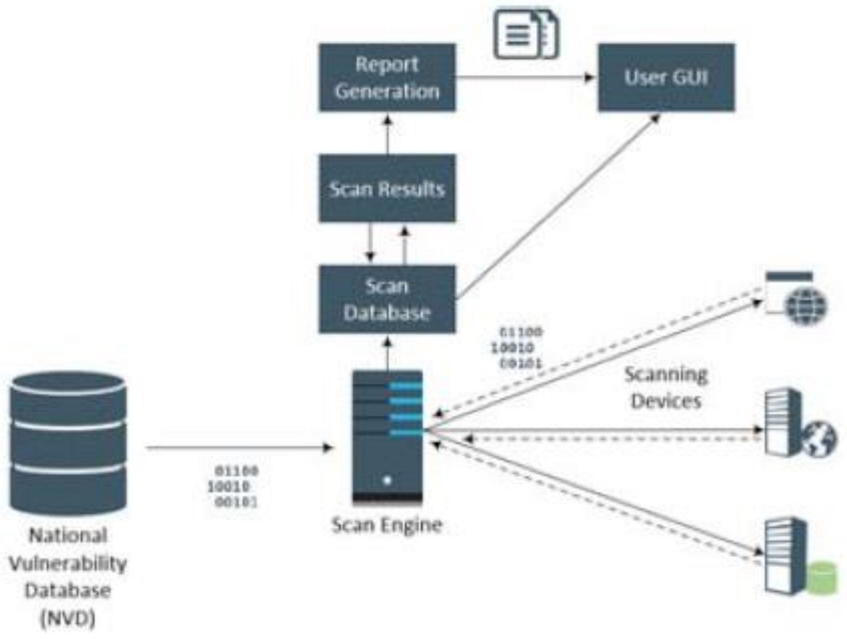


**Figure 1.** Vulnerability Scanner System Diagram

Source: (Harrell et al., 2018)

In other words, Vulnerability scanner is a software that used to scan the network, to report identified vulnerabilities, and supply instructions for remediation of found weaknesses. Some of the popular Vulnerability Scanners are shown in Table 1.

**Table 1.** Vulnerability Scanners

| OpenVAS |
|---|
| OpenVAS is one of the open-source tools that offer VA tools for both identifying vulnerabilities and managing vulnerability.<br>• Supports various operating systems;<br>• Continuously updated with the Network Vulnerability Tests;<br>• It is a complete VA tool detecting concerns linked to security in the servers and other devices of the network;<br>• Its services are cost-free and are also licensed under the GNU General Public License (GPL). |
| Nessus Professional |
| Nessus is a branded and **patented** tool of **Tenable Network Security which one of the great** and powerful vulnerability scanners.<br>• It can prevent computer and network systems from being exploited by intruders hackers by assessing vulnerabilities at the earliest stage;<br>• Can identify the weaknesses that enable remote attackers to steal system's sensitive data; |

- It supports a wide family of operating systems, hardware, software products and database applications and many other devices across cloud, virtual and physical systems;
- Millions of users around the world have installed it and used it for vulnerability identification and management.

**Microsoft Baseline Security Analyzer (MBSA)**

MBSA is a free tool offered by Microsoft that best suited to protect a Windows machine based on Microsoft's requirements or guidelines.

- It allows protection mechanisms to be improved by probing a collection of hosts for any errors in configuration, missed updates and any security patches;
- It looks at solely for service applications and security updates that leaving out the Primary and Optional upgrades;
- Medium-sized and small-sized companies are using it to monitor their network security;
- MBSA will be providing a few schemes or recommendations relevant to vulnerability fixing after testing a device.

**Comodo HackerProof**

One of the groundbreaking vulnerability scanning and trust-building tools that allow visitors to solve their security concerns. Here are a few essential advantages that can be got from HackerProof:

- Decreasing cart abandonment;
- Periodic vulnerability scanning;
- PCI scanning;
- Drive-by attack blocking;

**Nexpose Community**

It is being developed by Rapid7, is an open-source tool used to scan the vulnerabilities and perform a wide variety of network probes.

- Can be integrated with different frameworks such as Metasploit;
- It takes into consideration different vulnerabilities such as the malware kits, and it addresses the problem based on its priority;
- It is also capable of detecting and testing new devices automatically, and of analyzing weaknesses as they join the network;
- It tracks the real-time disclosure of vulnerabilities, familiarizing itself with the latest new data breaches;

**Nikto**

It is a widely used and open-source Web scanner which determines the possible problems and weaknesses.

- Nikto often used to verify if the application versions are obsolete, and also to search for any particular issue affecting the server's work;
- It is used on web servers to perform a variety of tests to scan various target machines, such as a few dangerous files or applications;
- It is for scanning the various protocols, such as HTTPS or HTTP, which provides a way for searching multiple ports of a target server.

**Tripwire IP360**

Tripwire IP360 is considered one of the leading vulnerability evaluation tools employed by numerous individuals and companies to handle their security risks.

- It uses a broad network view to detect all the bugs, settings, programs and network hosts;
- It utilizes open standards to help incorporate risk reduction and vulnerability into various business processes.

**Wireshark**

It is one of the widely used network protocol analyzers, which is considered to be the most powerful tool for security practitioners.

- It is employed to look at the networks at a deeper level across various sources such as government departments, businesses and educational institutions;
- It captures the traffic in real-time and conducts offline analysis;
- It is a cross-platform solution.

**Aircrack**

Aircrack or Aircrack-NG is a range of instruments used to determine the reliability of the WiFi network.

- Aircrack tools are also used when auditing networks;
- It supports multiple operating systems;
- It concentrates on various Wi-Fi security areas such as monitoring packets and data, checking drivers and cards, replaying attacks and cracking;
- With Aircrack, the lost keys can be recovered by capturing packets.

| Nmap |
|---|
| Nmap is an open-source network scanner that used to send packets and examine replies to identify hosts and services on a computer network. Nmap offers a range of tools to check computer networks, including host discovery and OS and device detection. Some of the Nmap features are as follows: • Host discovery – Detecting hosts on a network environment; • Port scanning – Enumeration of the open ports on target machines; • Version detection – Cross-examination of network infrastructure on remote computers in order to estimate the name and version number of the program; • OS detection – Identification of OS and hardware characteristics devices on the network; • Scriptable interaction with the target – which uses Nmap Scripting Engine (NSE) and Lua programming language. |

Source: Comodo Internet Security (last accessed on 01st of June 2019)

These vulnerability management systems can also create a priority list of the vulnerabilities we should fix, and they also define the vulnerabilities and include measures to remedy them (*Top 10 Vulnerability Assessment Scanning Tools*, 2018).

### 2.2.1. Vulnerability Databases and Dictionaries

A vulnerability database is a collection of weaknesses designed to collect, store, and share knowledge regarding found and published vulnerabilities targeting information/computer systems. Shared information is covering discovered vulnerability, the possible effect on the affected systems, and any solutions or updates to take control of the issue. The vulnerability database is also the outcome of an attempt to collect information on all documented security vulnerabilities in software or hardware products (Granova & Slaviero, 2017). In this section, some of the most reputable vulnerability databases and dictionaries will be analyzed, which includes:

- Common Vulnerabilities and Exposures (CVE);
- National Vulnerability Database (NVD);
- Common Platform Enumeration (CPE);
- Common Vulnerability Scoring System (CVSS).

**Common Vulnerabilities and Exposures** – CVE is an open-source security project maintained by MITRE Corporation and funded by the US Division of Homeland Security. The CVE dictionary employs SCAP for gathering data related to security weaknesses and exposures, classifying by different identifiers and assigning unique IDs to them. Once vulnerability data stored, MITRE shares each vulnerability with a unique ID. After several days of publication of vulnerabilities in the MITRE database, NVD publishes the CVEs with applicable security analysis. The objective of MITRE is defining CVE list as a dictionary (see Figure 2) or glossary rather than a database which contains vulnerabilities and exposures. Moreover, assures availability of this repository publicly for serving as an industry baseline for communicating and dialoguing around given exposures and security weaknesses. In other words, CVE dedicated providing a link to vulnerability database platforms and other repositories in order to enable synchronization of security services and tools. (*About CVE*, 2019).

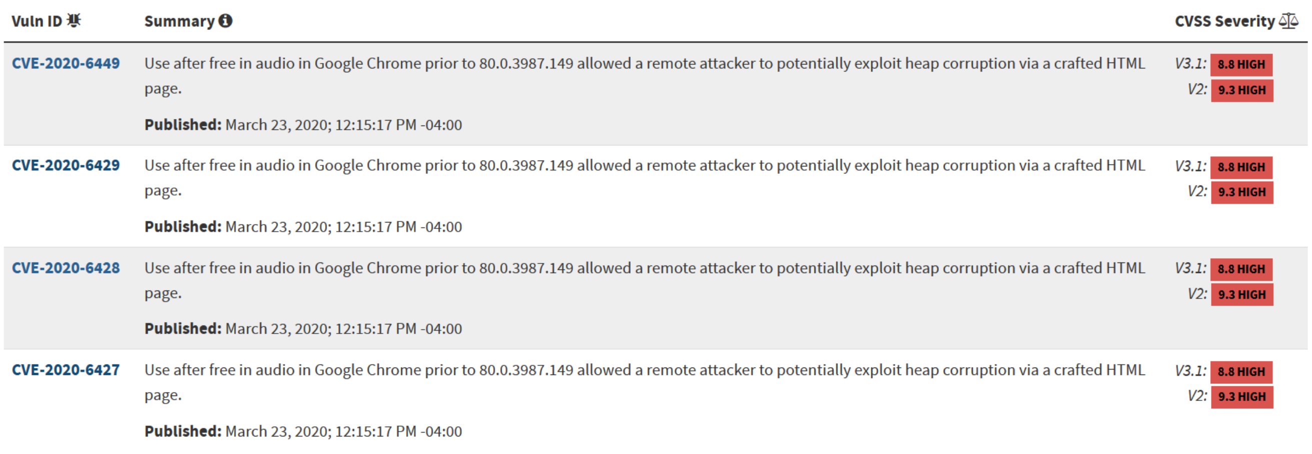
```
<item name="CVE-2020-6449" seq="2020-6449" type="CAN">
    <status>Candidate</status>
    <phase date="20200108">Assigned</phase>
    <desc>Use after free in audio in Google Chrome prior to 80.0.3987.149 allowed a remote
attacker to potentially exploit heap corruption via a crafted HTML page.   </desc>
    <refs>
        <ref    source="DEBIAN"    url="https://www.debian.org/security/2020/dsa-4645">DSA-
4645</ref>
        <ref   source="FEDORA"   url="https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/2DDNOAGIX5D77TTHT6YPMVJ5WTXTCQEI/">FEDORA-2020-
17149a4f3d</ref>
        <ref   source="FEDORA"   url="https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/6IOHSO6BUKC6I66J5PZOMAGFVJ66ZS57/">FEDORA-2020-
39e0b8bd14</ref>
        <ref   source="FEDORA"   url="https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/JWANFIR3PYAL5RJQ4AO3ZS2DYMSF2ZGZ/">FEDORA-2020-
7fd051b378</ref>
        <ref  source="GENTOO"  url="https://security.gentoo.org/glsa/202003-53">GLSA-202003-
53</ref>
        <ref    source="MISC"    url="https://chromereleases.googleblog.com/2020/03/stable-
channel-update-for-desktop_18.html">https://chromereleases.googleblog.com/2020/03/stable-
channel-update-for-desktop_18.html</ref>
        <ref source="MISC" url="https://crbug.com/1059686">https://crbug.com/1059686</ref>
        <ref  source="SUSE"  url="http://lists.opensuse.org/opensuse-security-announce/2020-
03/msg00028.html">openSUSE-SU-2020:0365</ref>
        <ref  source="SUSE"  url="http://lists.opensuse.org/opensuse-security-announce/2020-
03/msg00037.html">openSUSE-SU-2020:0389</ref>
    </refs>
      <votes>
      </votes>
     <comments>
      </comments>
</item>
```

**Figure 2.** CVE example from MITRE related to Google Chrome

Source: CVE MITRE (last accessed on 09th of April 2020)

**National Vulnerability Database** – The NVD is the repository and archive of standards-based vulnerability management data defined using the SCAP by the U.S. Government. In general, the repository facilitates the automation in vulnerability management and enforcement in information security. Repositories contain security checklist, references, flaws in software products, misconfigurations, vendor and product names, as well as impact metrics. Additionally, the NVD includes the summary and references from the CVE but provides additional resources regarding published and known vulnerabilities. In the extra details, NVD shares CVSS as risk scoring (see Figure

3), categorization using CWE specification, and details about at-risk information systems applying the CPE. (*NVD | General Information*, 2019).

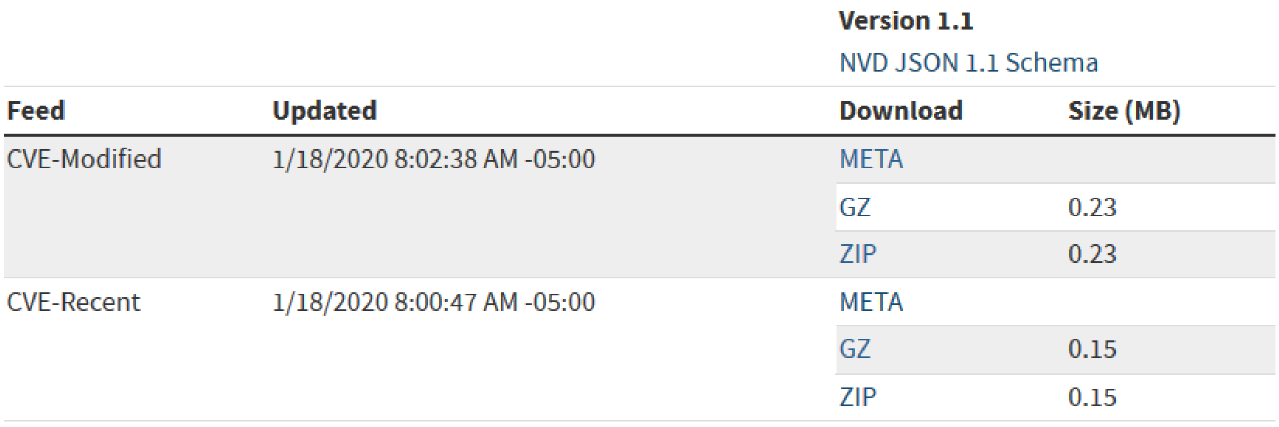


| Vuln ID | Summary | CVSS Severity |
|---|---|---|
| CVE-2020-6449 | Use after free in audio in Google Chrome prior to 80.0.3987.149 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.<br>**Published:** March 23, 2020; 12:15:17 PM -04:00 | *V3.1:* 8.8 HIGH<br>*V2:* 9.3 HIGH |
| CVE-2020-6429 | Use after free in audio in Google Chrome prior to 80.0.3987.149 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.<br>**Published:** March 23, 2020; 12:15:17 PM -04:00 | *V3.1:* 8.8 HIGH<br>*V2:* 9.3 HIGH |
| CVE-2020-6428 | Use after free in audio in Google Chrome prior to 80.0.3987.149 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.<br>**Published:** March 23, 2020; 12:15:17 PM -04:00 | *V3.1:* 8.8 HIGH<br>*V2:* 9.3 HIGH |
| CVE-2020-6427 | Use after free in audio in Google Chrome prior to 80.0.3987.149 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.<br>**Published:** March 23, 2020; 12:15:17 PM -04:00 | *V3.1:* 8.8 HIGH<br>*V2:* 9.3 HIGH |

**Figure 3.** Vulnerabilities related to Google Chrome from the NVD official site

Source: NVD official site (last accessed on 09th of April 2020)

The project also allows users to keep themselves up to date with the NVD data feeds. So that, NVD offers feeds called “CVE-Recent”, “CVE-Modified” (see Figure 4) and “META” files which are updated approximately every two hours. In more detail, the "CVE-Recent" feeds contains recently published vulnerabilities, moreover modified feeds provide changed or modified vulnerabilities which announced over eight days.

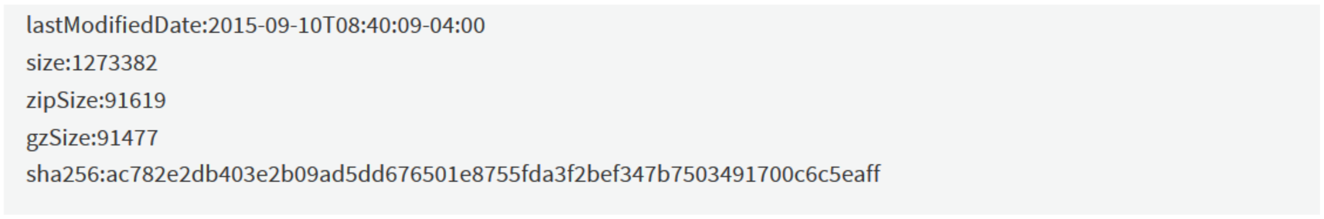


| | | Version 1.1<br>NVD JSON 1.1 Schema | |
|---|---|---|---|
| **Feed** | **Updated** | **Download** | **Size (MB)** |
| CVE-Modified | 1/18/2020 8:02:38 AM -05:00 | META | |
| | | GZ | 0.23 |
| | | ZIP | 0.23 |
| CVE-Recent | 1/18/2020 8:00:47 AM -05:00 | META | |
| | | GZ | 0.15 |
| | | ZIP | 0.15 |

**Figure 4.** Modified and Recent feeds by NVD

Source: NVD official site (last accessed on 18th of January 2020)

Furthermore, “META” files (shown in Figure 5) are used to check whether imported feeds from the NVD to the local database were updated or not. It ensures to avoid unneeded downloads of information which will result in a reasonable use of requests given by NVD.



```
lastModifiedDate:2015-09-10T08:40:09-04:00
size:1273382
zipSize:91619
gzSize:91477
sha256:ac782e2db403e2b09ad5dd676501e8755fda3f2bef347b7503491700c6c5eaff
```

**Figure 5.** Example of META file from NVD

Source: NVD official site (last accessed on 18th of January 2020)

**Common Platform Enumeration** – CPE is a way of describing an IT asset naming system, such as software, hardware products, and operating systems. CPE is a component of the SCAP standard (Cheikes & Scarfone, 2011), developed by the National NIST. Currently, CPE 2.2 and CPE 2.3 versions of CPE specification exist. Version 2.3 states a stack composed of five specifications, including the CPE naming and the CPE dictionary specification. Well-Formed CPE Name (WFN), a set of attributes define the CPE naming scheme which includes: *part*, *vendor*, *product*, *version*, *update*, *edition*, *language*, *sw_edition*, *target_sw*, *target_hw*, and other (*CPE - Common Platform Enumeration: CPE Specifications*, 2020). The CPE specification is employed for assigning identifiers to assets within a computer/network environment and supports two formats: URI (defined in version 2.2of CPE), and formatted string (defined in version 2.3 of CPE). NVD ensures the management and support of the CPE dictionary provided in XML format (see Figure 6).

```
<cpe-item name="cpe:/a:google:chrome:80.0.3987.149">
       <title xml:lang="en-US">Google Chrome 80.0.3987.149</title>
       <references>
          <reference     href="https://chromereleases.googleblog.com/2020/03/stable-channel-
update-for-desktop_18.html">Advisory</reference>
          <reference
href="https://www.google.com/chrome/browser/desktop/">Product</reference>
          <reference
href="http://src.chromium.org/viewvc/chrome/releases/">Product</reference>
       </references>
<cpe-23:cpe23-item name="cpe:2.3:a:google:chrome:80.0.3987.149:*:*:*:*:*:*:*"/>
</cpe-item>
```

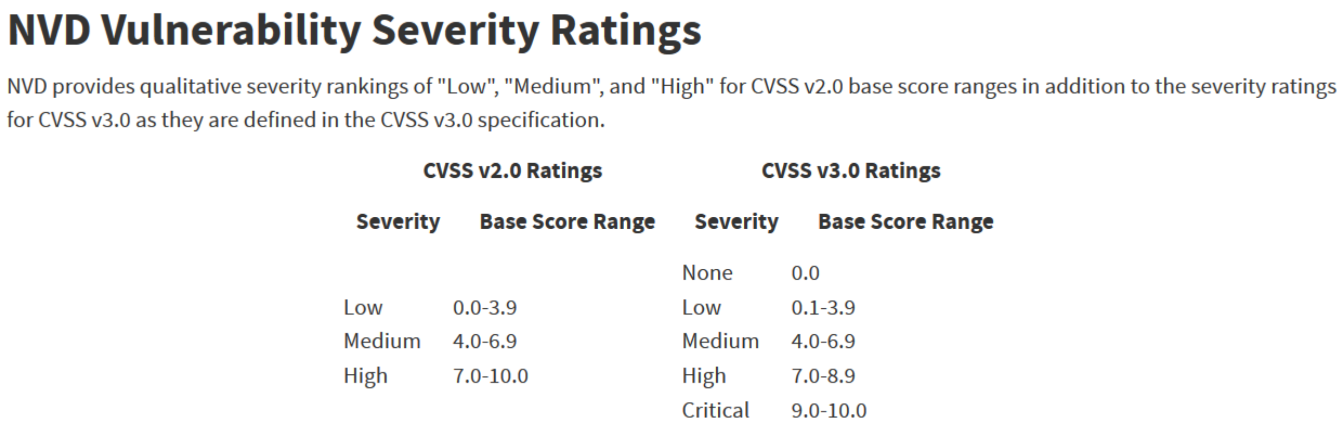
**Figure 6.** CPE entry for Google Chrome 80.0.3987.149 in the official CPE dictionary

Source: NVD official CPE dictionary (last accessed on 09th of April 2020)

Figure 6 shows a CPE entry of the official dictionary version 2.3. The listing contains the URI (for backward compatibility with CPE version 2.2) and formatted string that identifies the application Google Chrome 80.0.3987.149.

**Common Vulnerability Scoring System** – One of the open industry standards for determining vulnerabilities by severities in a computer/network environment is The Common Vulnerability Scoring

System (CVSS). According to a threat, CVSS (see Figure 7) tries to set severity scores to known vulnerabilities also provides prioritizing responses and resources.


NVD Vulnerability Severity Ratings

NVD provides qualitative severity rankings of "Low", "Medium", and "High" for CVSS v2.0 base score ranges in addition to the severity ratings for CVSS v3.0 as they are defined in the CVSS v3.0 specification.

| CVSS v2.0 Ratings | | CVSS v3.0 Ratings | |
|---|---|---|---|
| **Severity** | **Base Score Range** | **Severity** | **Base Score Range** |
| | | None | 0.0 |
| Low | 0.0-3.9 | Low | 0.1-3.9 |
| Medium | 4.0-6.9 | Medium | 4.0-6.9 |
| High | 7.0-10.0 | High | 7.0-8.9 |
| | | Critical | 9.0-10.0 |

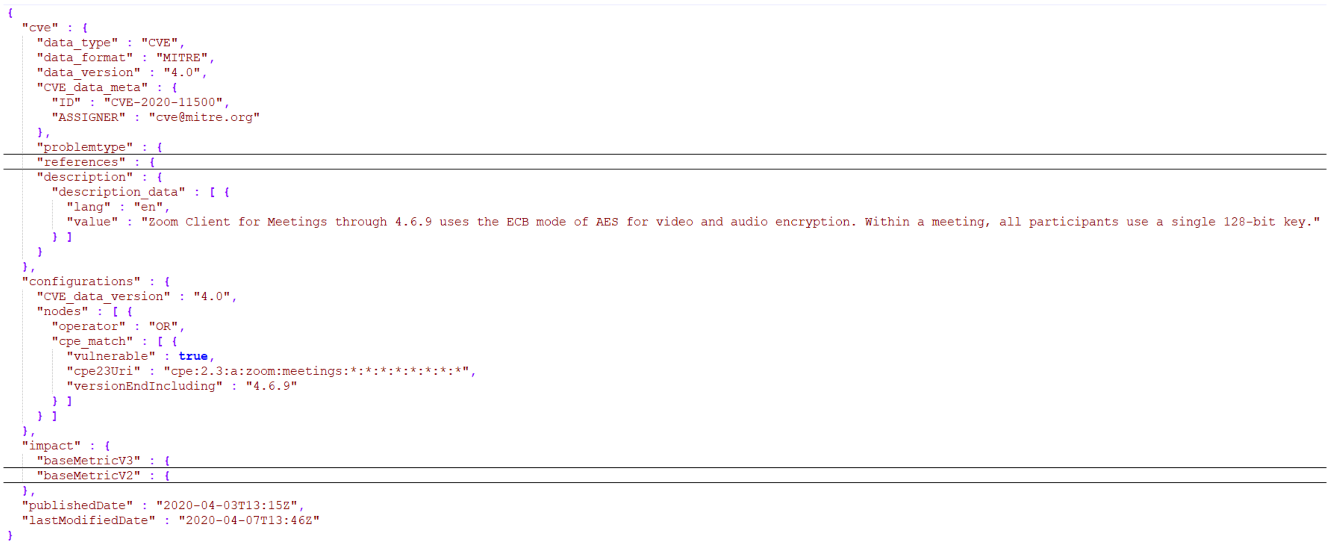
**Figure 7.** NVD Vulnerability Severity Ratings from the official site

Source: NVD official site (last accessed on 09th of April)

Given rankings are computed with a formula that determined on various metrics such as comparative ease and impact of exploit. Calculated scores vary from zero to ten, with the most severe grade is ten (Mell et al., 2006).

### 2.2.2. Using CVE and CPE in a VMS

The identification of vulnerable system, hardware or software products in an IT environment is one of the objectives of a VMS. Besides, other fields also may require a VMS, such as risk control and remedial actions. In these situations, a significant role is played by the CPE dictionary, and CVE feeds, but VMSs that employ CPE and CVE rely on the compatibility between both datasets. A VMS queries the CPE dictionary to find CPE identifiers in the environment where the VMS is running that match IT objects. Generally, the entries in the CVE feeds contain a list of vulnerable applications in CPE format (see Figure 8), it enables the VMS to detect vulnerabilities for the IT assets using their assigned CPE identifier.



```
{
  "cve" : {
    "data_type" : "CVE",
    "data_format" : "MITRE",
    "data_version" : "4.0",
    "CVE_data_meta" : {
      "ID" : "CVE-2020-11500",
      "ASSIGNER" : "cve@mitre.org"
    },
    "problemtype" : {
    "references" : {
    "description" : {
      "description_data" : [ {
        "lang" : "en",
        "value" : "Zoom Client for Meetings through 4.6.9 uses the ECB mode of AES for video and audio encryption. Within a meeting, all participants use a single 128-bit key."
      } ]
    }
  },
  "configurations" : {
    "CVE_data_version" : "4.0",
    "nodes" : [ {
      "operator" : "OR",
      "cpe_match" : [ {
        "vulnerable" : true,
        "cpe23Uri" : "cpe:2.3:a:zoom:meetings:*:*:*:*:*:*:*:*",
        "versionEndIncluding" : "4.6.9"
      } ]
    } ]
  },
  "impact" : {
    "baseMetricV3" : {
    "baseMetricV2" : {
  },
  "publishedDate" : "2020-04-03T13:15Z",
  "lastModifiedDate" : "2020-04-07T13:46Z"
}
```

**Figure 8.** CVE feed in JSON format from NVD

Source: NVD official site (last accessed on 09th of April 2020)

Currently, NVD stopped supporting XML for CVE feeds and offering a vulnerability data feed using the JSON format.

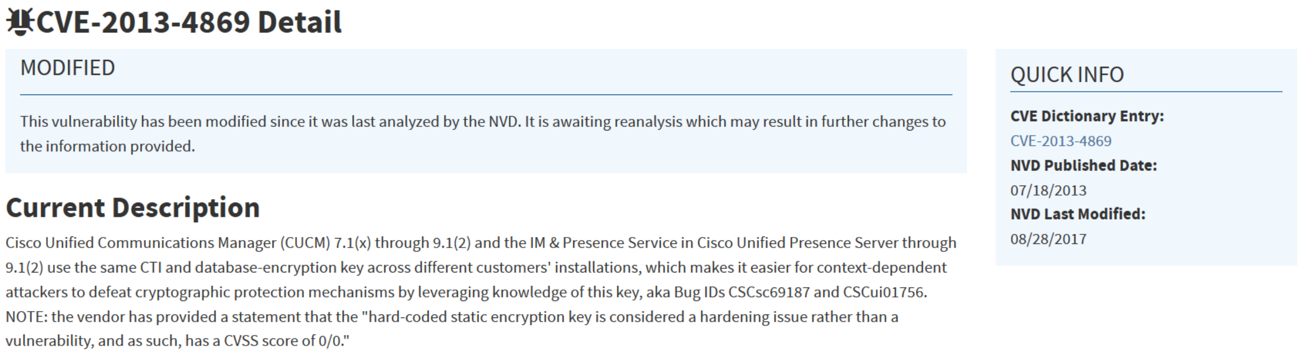
### 2.2.3. Challenges and Limitations in VMS

There are several motives for conduction vulnerability assessment. The primary purpose is to find out and determine vulnerabilities within the organization's computer/network system before the attacker exploits. Thereby, organizations, companies or individuals are referring to vulnerability management programs in order to sustain the security of valuable assets. In the market, several vendors offer vulnerability scanning tools with plenty of features. When using vulnerability assessment tools, enterprises can be sure that their computer system will be reliable and secure, since the tools scan and find out existing vulnerabilities in the environment, as well as they, make very hard for computer environment to become compromised. Despite there are many vulnerability databases which vendors use them as a reference, however, they may be late in getting warning and alerts related to new vulnerabilities and exposures due to synchronization or lack of identifiers. In this section, the vulnerability detection capabilities of VMSs and current limitations on datasets were analyzed and research opportunities identified as follow:

- CVEs without assigned CPE identifiers;
- CPE entry has been created in the official CPE dictionary;
- Deprecation process;
- Lack of synchronization with NVD;

- Newly published vulnerabilities without CPEs or plugins and undergoing analysis.

One of the issues in the VMSs is CVEs without assigned CPE identifiers. The CVE feeds (as of April 5th, 2020) contain 210 CVE entries that do not have any CPE identifiers. For example, CVE-2013-4869 is a vulnerability for the products Cisco Unified Communications Manager and the IM & Presence Service in Cisco Unified Presence Server (*CVE-2013-4869 Detail*, 2013). This CVE has no CPE identifiers, even though the official CPE dictionary and CPE Match Feed has identifiers for these products, as shown in Figure 9. One of the primary reasons is that CVE is awaiting due to the possibility of reanalysis, which may result in additional adjustments to the data obtained.


CVE-2013-4869 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

Cisco Unified Communications Manager (CUCM) 7.1(x) through 9.1(2) and the IM & Presence Service in Cisco Unified Presence Server through 9.1(2) use the same CTI and database-encryption key across different customers' installations, which makes it easier for context-dependent attackers to defeat cryptographic protection mechanisms by leveraging knowledge of this key, aka Bug IDs CSCsc69187 and CSCui01756. NOTE: the vendor has provided a statement that the "hard-coded static encryption key is considered a hardening issue rather than a vulnerability, and as such, has a CVSS score of 0/0."

QUICK INFO

CVE Dictionary Entry:
CVE-2013-4869
NVD Published Date:
07/18/2013
NVD Last Modified:
08/28/2017

**Figure 9.** CVE example for awaiting reanalysis from the NVD

Source: NVD official site (last accessed on 09th of April)

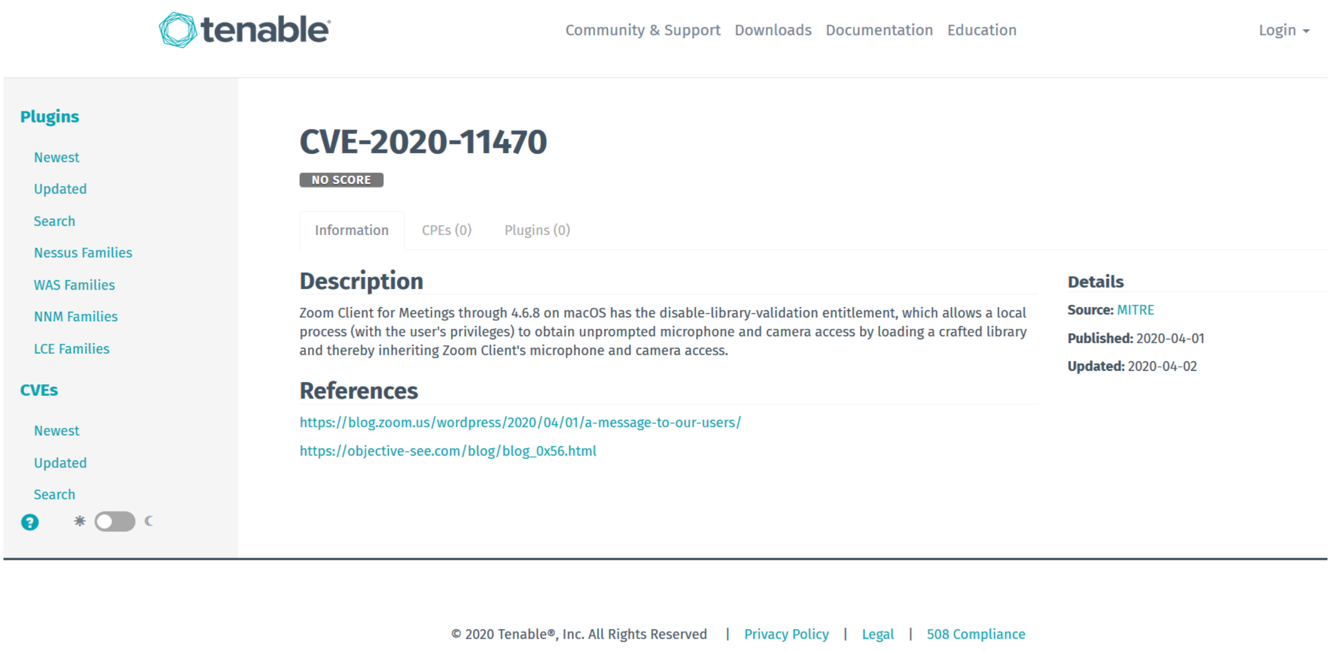
In a result, VMSs that only use CPE identifiers to find related CVE entries for software products deliver incomplete results. For example, if a vulnerability is found and a CVE entry is created for it without CPE identifiers, a VMS would not generate any alert for the IT assets affected by that vulnerability which will lead to false negatives.

Secondly, there are some products for which no CPE entry has been created in the official CPE dictionary for them (Sanguino & Uetz, 2017). In that case, full automation (i.e., without human interaction) of the process of assigning a CPE identifier to a software product can lead to false positives or false negatives. Since there are software products for which no CPE identifier exists in the CPE dictionary, a fully automated system trying to assign an existing CPE to a software product would either assign an incorrect best match or not assign a CPE at all. Consequently, either incorrect CVE entries are found for these software products (false positives), or the actual entries related to those products are not found (false negative).

Another thing is deprecation process, according to the CPE dictionary specification version 2.3 (Cheikes & Scarfone, 2011), CPE identifiers are deprecated for three reasons: identifier name correction, identifier name removal, or additional information discovery. Therefore, if a VMS assigns a CPE identifier to a software product that is later corrected, then CVEs containing the corrected CPE could not be found.

Fourthly, synchronization and being up to date with vulnerability databases. For example, there some modules such as Advanced vulnerability scanning with Nmap NSE for NMAP (network scanner) which turns it to vulnerability scanner as well. However, if to support the latest disclosed vulnerabilities, clients need to keep their local vulnerability databases up to date. If they want to update local databases, they need to go to the vulnerability databases' web sites and download files manually (scip ag, 2020). As a result of this, if users update their local databases manually, they would miss newly published vulnerabilities due to lack of automation.

Finally, responding to vulnerabilities by the organizations on time is one of the crucial aspects in terms of security (Ruohonen, 2019). The existence of newly published vulnerabilities without CPE identifiers or plugins for detection makes practical vulnerability analysis complicated, turns results to false-negatives and distracts organizations from the initial aim.  For instance, CVE-2020-11470 is a vulnerability for Zoom Client for Meetings (*CVE-2020-11470*, 2020) through version 4.6.8, despite few days passed, CPE identifier and plugins were not published for detection of weakness by the VMSs, see Figure 10.



CVE-2020-11470

Description

Zoom Client for Meetings through 4.6.8 on macOS has the disable-library-validation entitlement, which allows a local process (with the user's privileges) to obtain unprompted microphone and camera access by loading a crafted library and thereby inheriting Zoom Client's microphone and camera access.

References

https://blog.zoom.us/wordpress/2020/04/01/a-message-to-our-users/

https://objective-see.com/blog/blog_0x56.html

Details

**Source:** MITRE

**Published:** 2020-04-01

**Updated:** 2020-04-02

© 2020 Tenable®, Inc. All Rights Reserved | Privacy Policy | Legal | 508 Compliance

**Figure 10.** CVE-2020-11470 from the official Tenable website

Source: Official Tenable website (last accessed on 03rd of April 2020)

These limitations, mainly, being late for publishing plugins to detect vulnerabilities, spending hours even days for analysis and audit of vulnerability, are led VMSs to get wrong and false-negative results.

## 2.3. Importance of Early Warning Systems in IT

An Early Warning System (EWS) is a necessary technology for creating and propagation of timely and substantial warning information on prevention which allows communities and the organizations who are in risk group to be prepared and work correctly and a short time to decrease potential damage or losses. The EWS ensures the gathering, processing and presentation of data in a logical and meaningful way to aid the generation and transmission of alarm messages through warning communication to the organizations at risk (Mulero Chaves & De Cola, 2017).

In information technology, detection techniques are vital to tackle the possible dangers of malicious actions of attackers for fulfilling an image of abnormal activities on computer environments. Furthermore, the EWS is a proactive detection solution like Intrusion Detection Systems (IDS) which provides vital tools for capturing and investigating anomalies. It supplies prediction options to tackle the further steps of continuous multi-step attacks by effectively tracking its early behaviours. Besides, there are also Reactive solutions such as Intrusion Prevention Systems (IPS) which aim to prevent attacks and threats by implementing a set of pre-defined security policies. To sum up, proactive approaches such as EWS seek to analyze the current security situation and avoid the incident from before potential the attack (Ramaki & Atani, 2016). According to proactive solutions, there are two ways after a successful prediction of an attack:

- By taking proper security controls;
- By converting the incident from unplanned situations to planned.

Early warning systems have extended information security borders after IDSs and IPSs and computer systems. More specifically, the functionality of EWSs focused on detecting some anomalies from a current or normal state (Apel et al., 2010). Namely, the potentially dangerous unknown behaviours are identified by the EWS.

In this thesis, as a terminology, the early warning has two different meanings:

(i) identifying and giving warning related to newly published vulnerabilities to prevent or minimize the risk to an acceptable level and;

(ii) capability to process unknown and inadequate information.

## 2.4. Conclusions of the second chapter

In this section, the main idea of early warning systems, vulnerability management systems, the different open source vulnerability databases and vulnerability management program were analyzed, as well as highlighted potential limitations and problems in VMSs.

Despite there are many solutions, methodologies and platforms for vulnerability assessment and management, still getting right and quick information is needed. Therefore, there are still some open issues regarding in vulnerability analysis and methods. Getting late update related to the newest vulnerabilities are considered a significant threat to the companies, or in a worst-case scenario – the cyber-attack can occur on the same day a weakness is discovered in software or mentioned on the vulnerability databases. During the analysis of related works, key points were identified, which lead VMSs to error-prone such as:

(i) CVEs without assigned CPE identifiers,

(ii) CPE entry has been created in the official CPE dictionary,

(iii) Deprecation process,

(iv) Lack of synchronization with NVD,

(v) Newly published vulnerabilities without CPEs or plugins and undergoing analysis.

In conclusion, staying up to date on security trends is vital for hardening our computer/network systems, as well as knowing our environment’s vulnerabilities is a high priority before attackers are using against us. Tools that only rely on NVD lead to making errors due to shortcomings such as CPE synchronizations, missing known affected software configurations, typographical errors or assignments of identifiers manually.

# 3. Proposed solution

Although there are numerous vulnerability assessment and management distributions with countless tools and scripts, the thesis goal is to propose an early warning concept based on open-source tools and databases which can be merged to commercial products as well.

The proposed solution includes a framework which helps to understand workflow and ensures finding answers to mentioned in the thesis. By using the Framework, weaknesses of the system can be assessed and be informed about newly available vulnerabilities before the compromising. Diagrams were used to represent the Framework, data flow and dynamic behaviour of the system, as well as the flow of used algorithms.

To sum up, this section provides diagrams, system requirements and specifications, technical challenges and offered the solution for the proposed concept.

## 3.1. Project Overview

The proposed solution consists of 4 (four phases), which are shown in Figure 11. In the initial phase, also called “Collecting Data” phase, organisations ensure 2 (two) inputs – assets and vulnerability database(s). Data Collectors gather data from the vulnerability database(s) and the IT environment. Afterwards, collected data is sent to the second phase for normalization and storing in the MariaDB which called “Parsing”. During the 2nd phase, the Query Parser algorithm is managing all analysis of the dataset sent by collectors and writing to the local database. In the third stage, the most critical part of the whole process is going to be started by Matcher algorithm under the “Vulnerability Identifications” phase. Matcher algorithm gets asset and vulnerability data from the database and matches them against each other using complex calculations in order to identify vulnerabilities. After identification of vulnerabilities, the processor looks patterns of CVE data such as the published date and description of vulnerabilities and decides whether the found vulnerability is an early warning or not. Finally, the processor sends processed and merged data to the last phase called “Populating” which results are populated for the user interface.

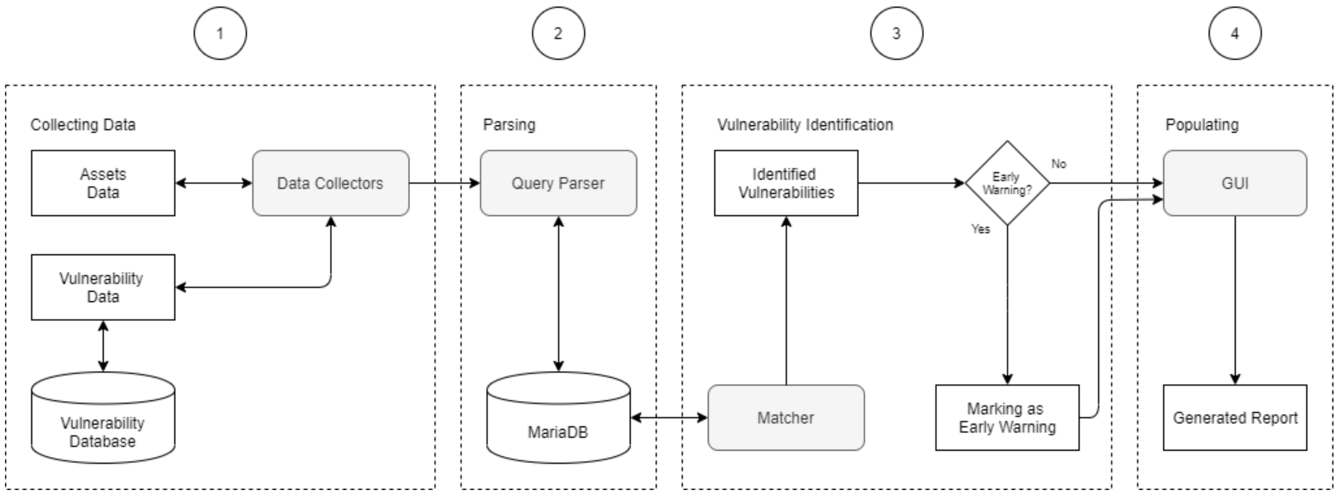


**Figure 11.** Proposed Framework

Source: by author

This research will analyze these areas and seek answers to these essential questions:

- How can we extract data from vulnerability databases/dictionaries;
- How can we obtain information from our assets (such as software products, vendors and other details);
- How can we efficiently match these two datasets;
- How to decrease human interaction;
- How to detect early warnings and give alerts;
- Finally, how can we avoid “false negatives”?

To achieve goals and to make it easy to use, a web application for a simplified version of the framework will be created. As a vulnerability resource, the NVD was chosen, which provides identifiers, dictionaries and feeds for publicly disclosed security vulnerabilities. Firstly, this choice was made for the reason that it offers a generic identifier for a specific vulnerability or exposure. Knowing this particular identifier enables us to get details about the vulnerabilities efficiently and reliably through numerous records linked to CVE. The NVD offers feeds for synchronization that scanner vendors are using, and updates datasets per 2 (two) hours. If new, missing or corrected CVE or CPE entries exist, they modify their content accordingly and publish modified and recent data feeds. Moreover, NVD includes an improved CVE content that is completely synchronized with the CVE List, so any changes to the CVE List instantly appear on NVD.

The whole project is going to be coded with PHP, which is a fast-growing programming language. As well is considered one of the easiest programming languages so any user with a little programming background can modify the code for his own needs (Powers, 2019).

In order to search and match vulnerabilities in the local database, Natural Language Full-Text Searches (*MySQL 8.0 Reference Manual*, 2020) and Fuzzy String Matching algorithm (*Fuzzy string matching for PHP*, 2020) will be used.

## 3.2. Objectives of the system

The objectives are mainly focused on ensuring all crucial functions for automation of the process. The solution will provide the following functionalities:

- Collecting Data:
    - Extracting and collecting data from NVD (if new vulnerabilities are published, the program will automatically update its database);
    - Collecting data from assets (vendor, product name and version);
- Parsing and storing collected data on the local database;
- Vulnerability identification:
    - Identification of vulnerabilities for giving alerts (it will match assets data against vulnerability identifiers);
    - Identification of early warnings for avoiding false negatives (it will match assets data against vulnerability summaries which do not have identifiers);
- Populating results on the Web-based console (GUI) for giving reports to users.

## 3.3. Dynamic behaviour of the system

This section is about the process of proposed framework related to each phase for understanding the whole working system, as well as roughly explains written algorithms. Stages of the framework will be depicted by activity diagrams which as follows:

1. Collecting data from NVD with NVD Data Collector;
2. Collecting data from assets with Agents;
3. Parsing collected NVD data on the local database with Query Parser;
4. Vulnerability Identification with Matcher;
5. User interaction with GUI (Web-based console).

**NVD Data Collector –** is one of the crucial parts of the system which is responsible for collecting data from the NVD. It was purely written in PHP and used all capabilities of its to aggregate data in a sufficient way. As seen from Figure 12, NVD Data Collector requests feed on the NVD. The NVD offers

META files which contain the hash of feed files and are updated approximately every two hours (*How to keep up-to-date with the NVD data*, n.d.).

The system checks and compares shared hashes with existed hashes which are stored in its database. If new hash exists, the system downloads archive files and extracts feed file from them or waits 2 (two) hours for rechecking the updates on the NVD. Finally, the system puts feed files (in JSON format) to the Queue Pool, which will be processed in the next phase.

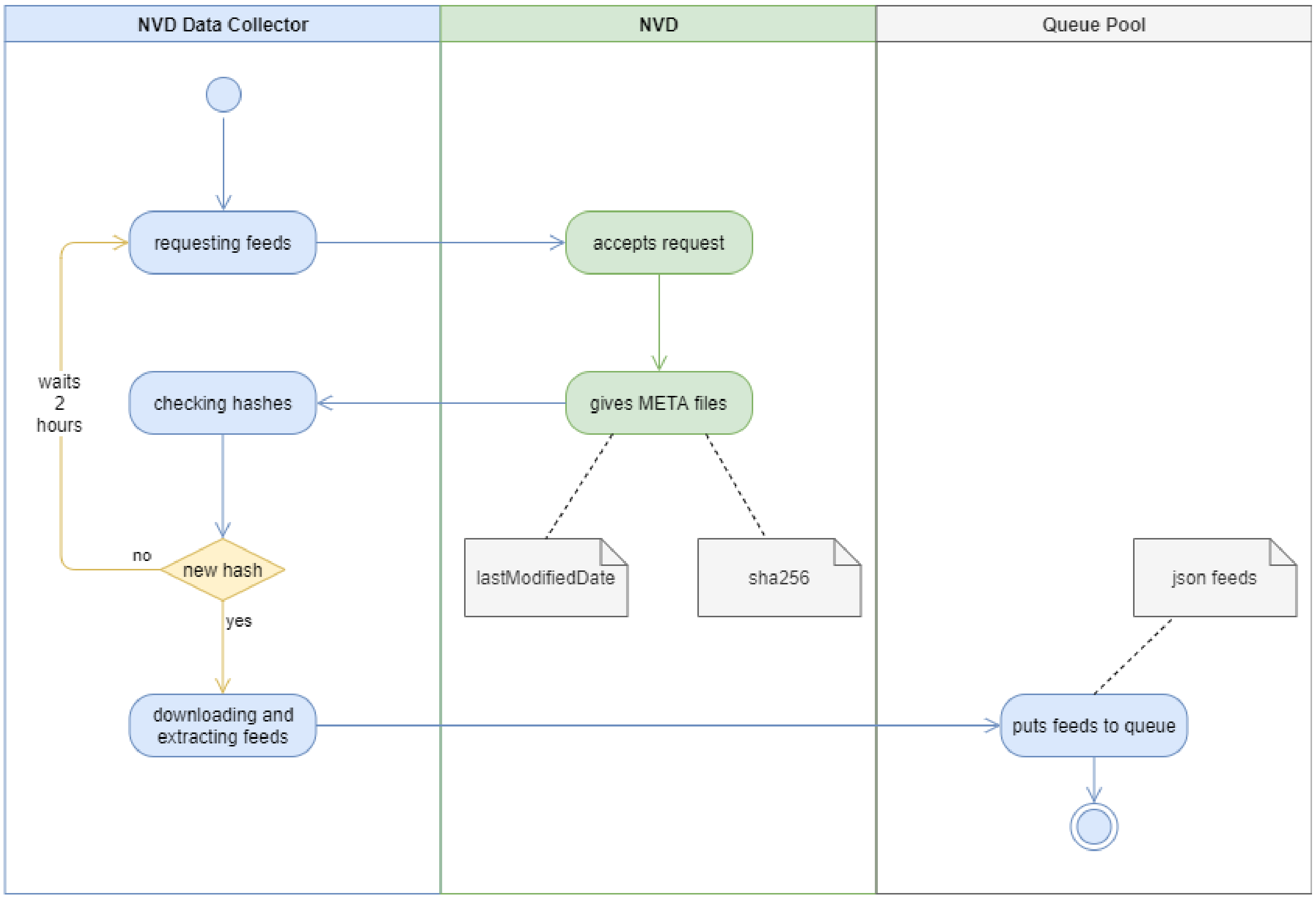


**Figure 12.** Collecting data from NVD

Source: by author

**Agents** – they are responsible for collecting data from assets or installed software products. As seen in Figure 13, agents are installed on the operating systems in that case on the Windows OS. First of all, agents requests to Windows Management Instrumentation (WMI) using WMI tasks in order to obtain information such as which software is installed by the Microsoft Windows Installer (MSI) and software versions (*WMI Tasks*, n.d.). Secondly, checks (every 5 min.) availability of newly installed software on the Windows and obtains data for storing in the SQLite local database. The reason for storing data in the local database is decreasing the massive requests to the server. Finally, the agent converts data to JSON string for sending to the RESTful API of the system.

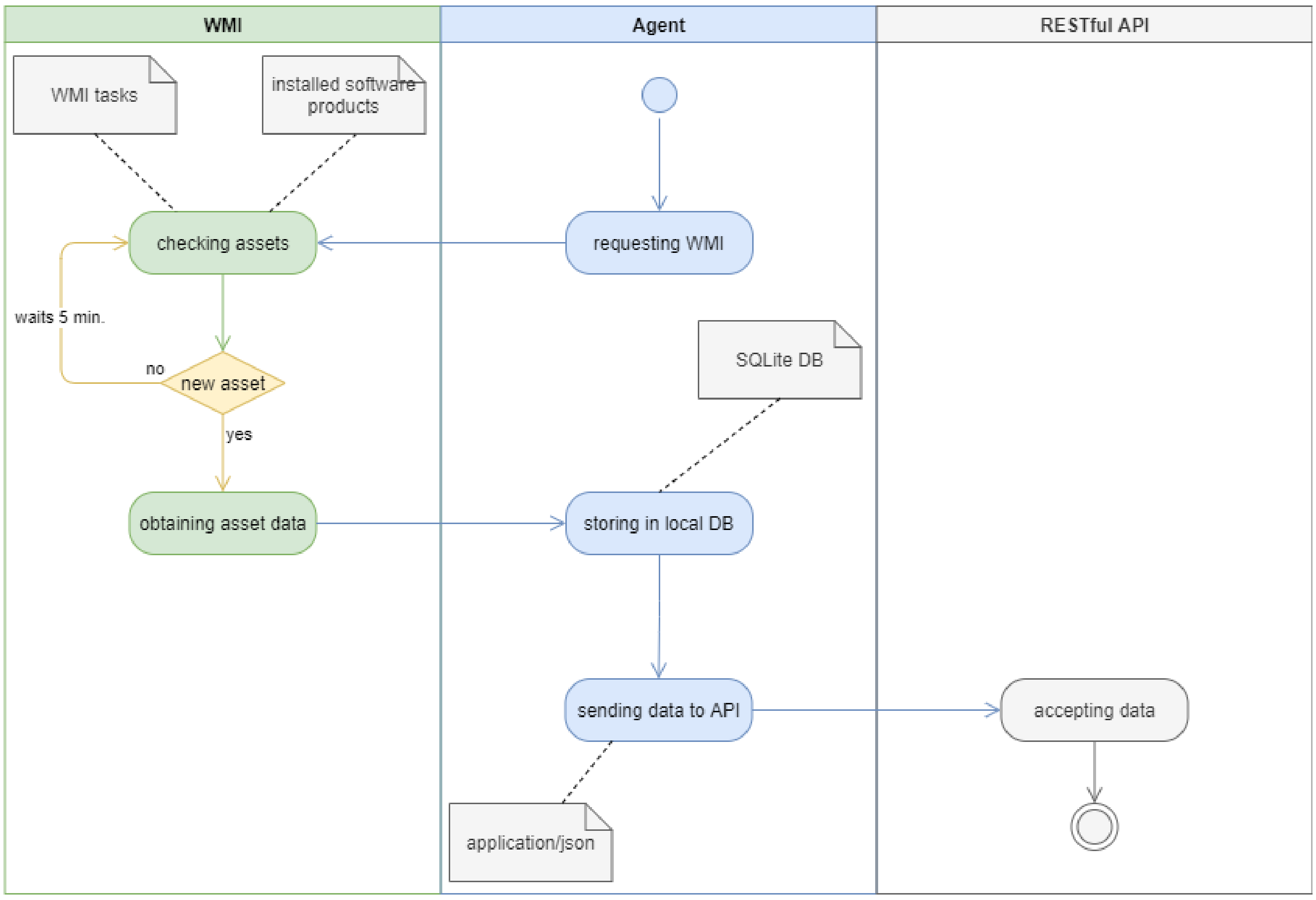


**Figure 13.** Collecting data from assets with Agents

Source: by author

**Query Parser** – this functionality ensures parsing, normalizing and storing data on the local database, which comes from the Data Collectors. In Figure 14, will be demonstrated parsing the data which comes from the NVD Data Collector. As mentioned in the first phase, extracted data from the NVD was put to the Queue Pool for the parsing process. The Query Parser in the initial stage, checks the feeds in the queue, if they are available, starts to parse them, otherwise periodically checks the queue. Feeds are in the JSON format contains essential information related to vulnerabilities such as published dates, CPE identifiers, summaries (CVEs), CVE ids, severities and impact scores. The necessary extracting procedure is related to CVEs and CPEs; that is why an only small part of the process was presented in the diagram. During the parsing feeds, the system extracts CVE summaries (additional mentioned data as well) and identifiers (if they exist in CVEs). Then, checks those data in order to define whether entries were inserted before or not. According to a made decision, the system is doing storing or updating actions on the database.

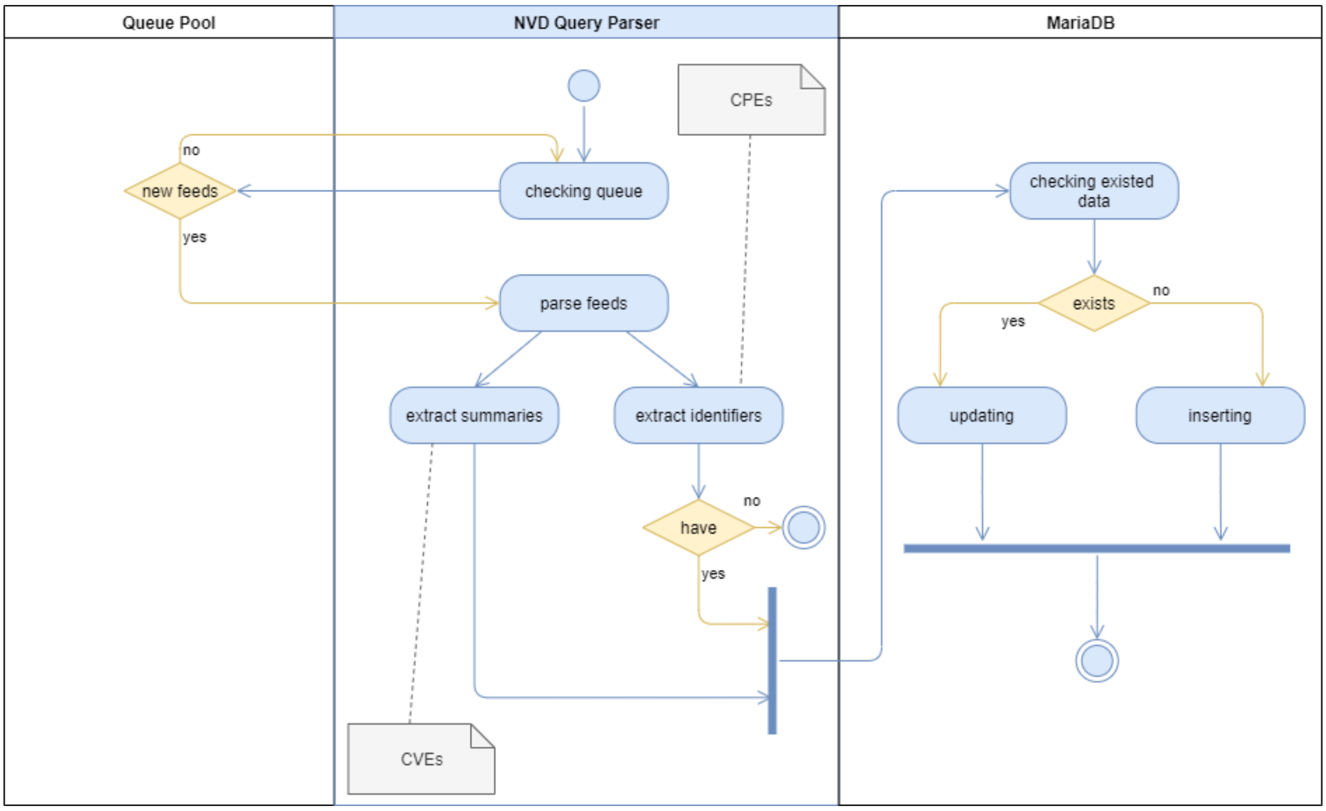


**Figure 14.** Parsing collected NVD data on the local database

Source: by author

**Vulnerability Identification** – is a crucial part of the system, which through detecting vulnerabilities gives alerts, as well as finds early warnings earlier than other existing solutions in the market. In terms of thesis, an alert is a vulnerability detection and warning service, offered by the early warning and alert engine of the system. If given alerts were met with predefined patterns and related to newly published vulnerabilities, they are converted to early warnings by the engine. Beside the process flow, this diagram also explains how the system's algorithm works. In Figure 15, the example of Vulnerability Identification Matcher algorithm and process flow chart is shown.

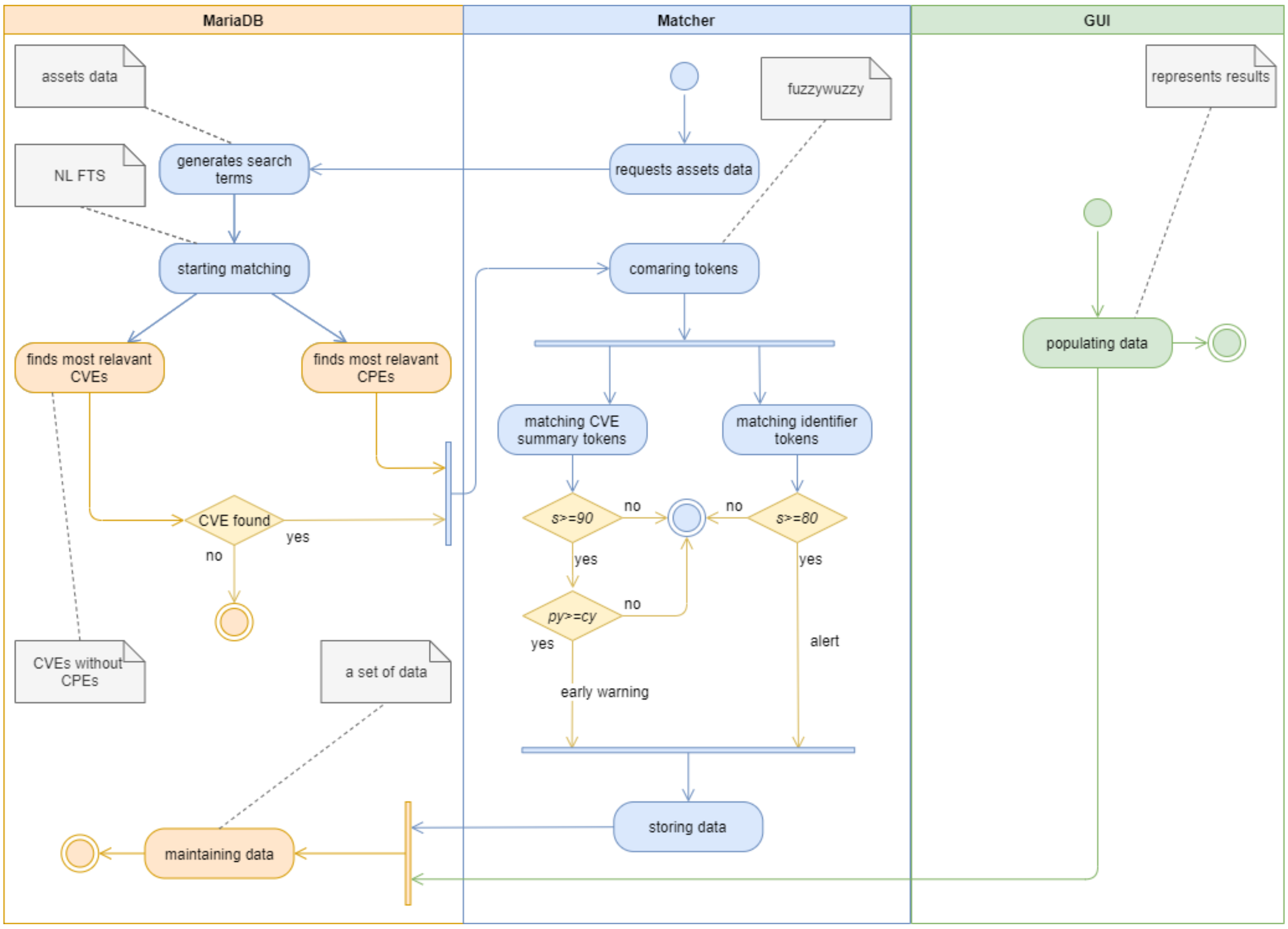


**Figure 15.** Vulnerability Identification Matcher

Source: by author

This algorithm is responsible for accomplishing those below-mentioned tasks:

- Requests and gets assets' data from the local storage;
- Tokenizes data and generates search terms;
- Accomplishes matching process:
  - Using Natural Language Full-Text searches (NL FTS) for getting the most relevant results by scoring them;
  - Finding the most relevant CVEs without CPE identifiers and sorting them by scores;
  - Finding the most relevant CPE identifiers and sorting them by ratings;
- Compares aggregated data:

Tokenized CVE summaries which do not have CPE identifiers with generated search terms using FuzzyWuzzy string matching algorithm and passing the over 90 percent similar results to the next step;

    - CPE identifiers with generated search terms using FuzzyWuzzy string matching algorithm, marking them as an alert, and forwarding over 80 per cent similar results to the storing process;
- Identifying early warnings by looking pattern of received data such as published year (greater than current year) for deciding whether a vulnerability is an early warning or not;
- Storing the collected data and sending to the maintenance process by the MariaDB because those data will be populated by the Web-based console (GUI).

**User interaction with GUI (Web-based console)** – is the final process of the whole working system. In this stage, user logins to the system which can configure the system and checks the reports. In order to check the reports, the user firstly, requests reports through Web-based console. The console by the request of the user automatically populates data from the database such as extracted vulnerabilities from the NVD, assets data and essential results of vulnerability identification process which contains alerts and early warnings. Then, it generates a report which user will review reports and analyze the results.

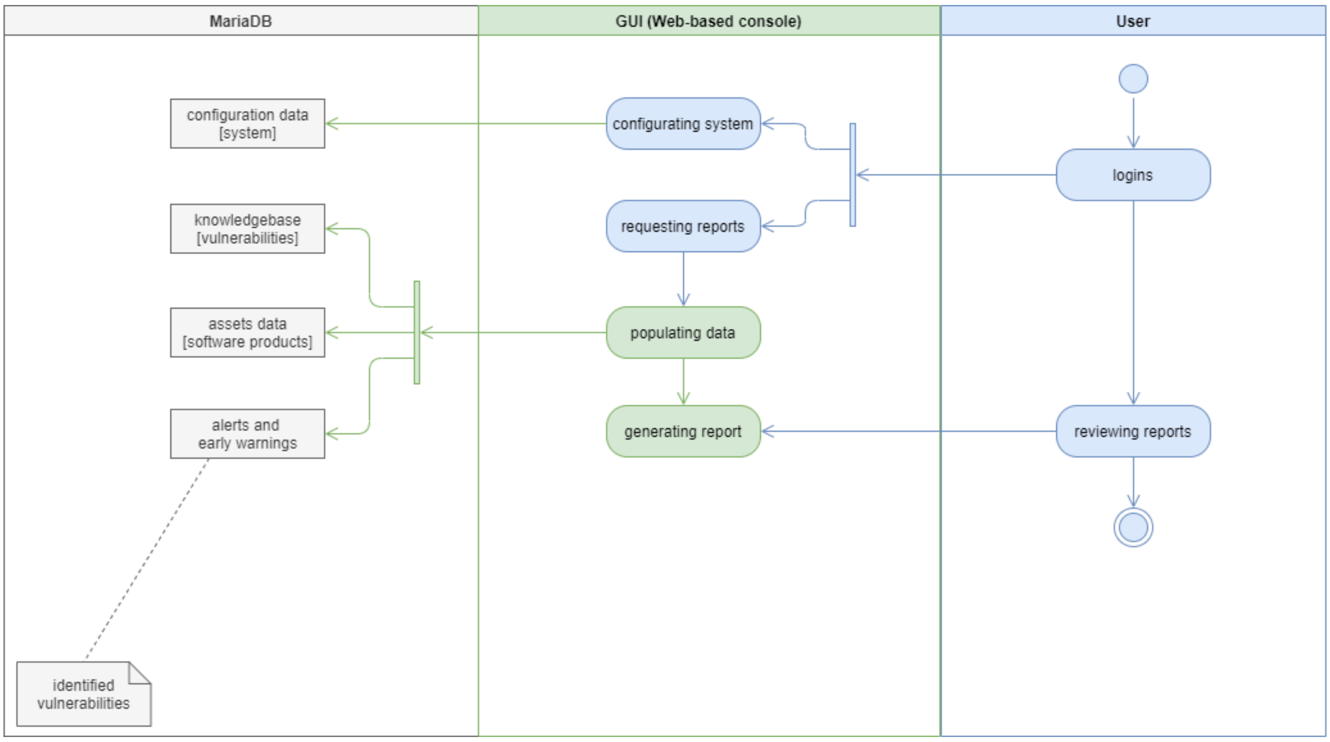

MariaDB
GUI (Web-based console)
User
configuration data [system]
knowledgebase [vulnerabilities]
assets data [software products]
alerts and early warnings
identified vulnerabilities
configurating system
requesting reports
populating data
generating report
logins
reviewing reports

**Figure 16.** User interaction with GUI (Web-based console)

Source: by author

## 3.4. Components and their definitions

In this section, important components of the project are presented, and definitions of each component are listed (Table 2).

**Table 2.** Used Components and Their Definitions

| *CID* | *Component* | *Type/Category* | *Definitions* |
|---|---|---|---|
| *C01* | Data Feeds<br>• CVE;<br>• CPE. | *Dataset* | CVE is a list of entries that contains an identification number, a summary, and at least one known reference related to publicly disclosed vulnerabilities in IT security. CVE records are used in various worldwide information security products and services, including NVD;<br><br>CPE is a standardized structure or naming scheme for applications and products. Based on Uniform Resource Identifiers (URI) generic syntax, CPE holds a structured name format which is a method for probing names against a system, and a description format for binding text and testing to a name. |
| *C02* | MariaDB | *Software* | MariaDB is a free and open-source fork of the MySQL which built under the GNU General Public License. |
| *C03* | PHP | *Programming Language* | PHP is a commonly used open-source, general-purpose scripting language that can be embedded in HTML and is especially suited for web development. |
| *C04* | Natural Language Full-Text Search | *Function* | This method performs a natural language search for a keyword against a text array and calculates similarities between the search |

| | | | |
|---|---|---|---|
| | | | string and the target in that row in the columns. |
| *C05* | XAMPP | *Software* | XAMPP is a free solution kit built by Apache Friends, contains Apache HTTP Server, MariaDB, PHP and Perl programming language script interpreters. |
| *C06* | Web-based console | *Web Application* | The console is a web-based application that provides the interface, alert and early warning views, reports, vulnerabilities, asset information, and administrative functions. |
| *C07* | Vulnerable Machine | *Virtual Machine* | It is configured environment with known vulnerable applications in order to experiment and evaluation of the proposed solution. |
| *C08* | Fuzzywuzzy | *Library* | Fuzzywuzzy is an open-source library that uses the Levenshtein Distance algorithm to measure and calculate differences between sequences and patterns in the given text. Networks, including host discovery and service and operating system detection. |
| *C09* | Agents | *Application* | Agents are developed and written in Python for gathering asset/software products data from the host machines. |
| *C10* | Data Collectors | | API-Based Applications which used to receive/collect data from the given sources. |

Source: by author

## 3.5. Design of Database for Storing CVEs and CPEs

In this section, database design will be given for storing the CVE and CPE identifiers in the MariaDB by demonstrating the EER Diagram (see Figure 17). Entity Relationship Diagram is a sort of architectural design for use in database design which is also understood as ERD or ER model. An ERD contains various symbols and connectors which show two essential details. On the other hand, Enhanced entity-relationship diagrams (EER) are essentially an extended model of ER diagrams. EER models or diagrams are useful tools to build a high-level model of databases. With their heightened capabilities, database models can be more accurately planned by more explicitly delving into the properties and constraints (Database Design and Modeling, 2020).

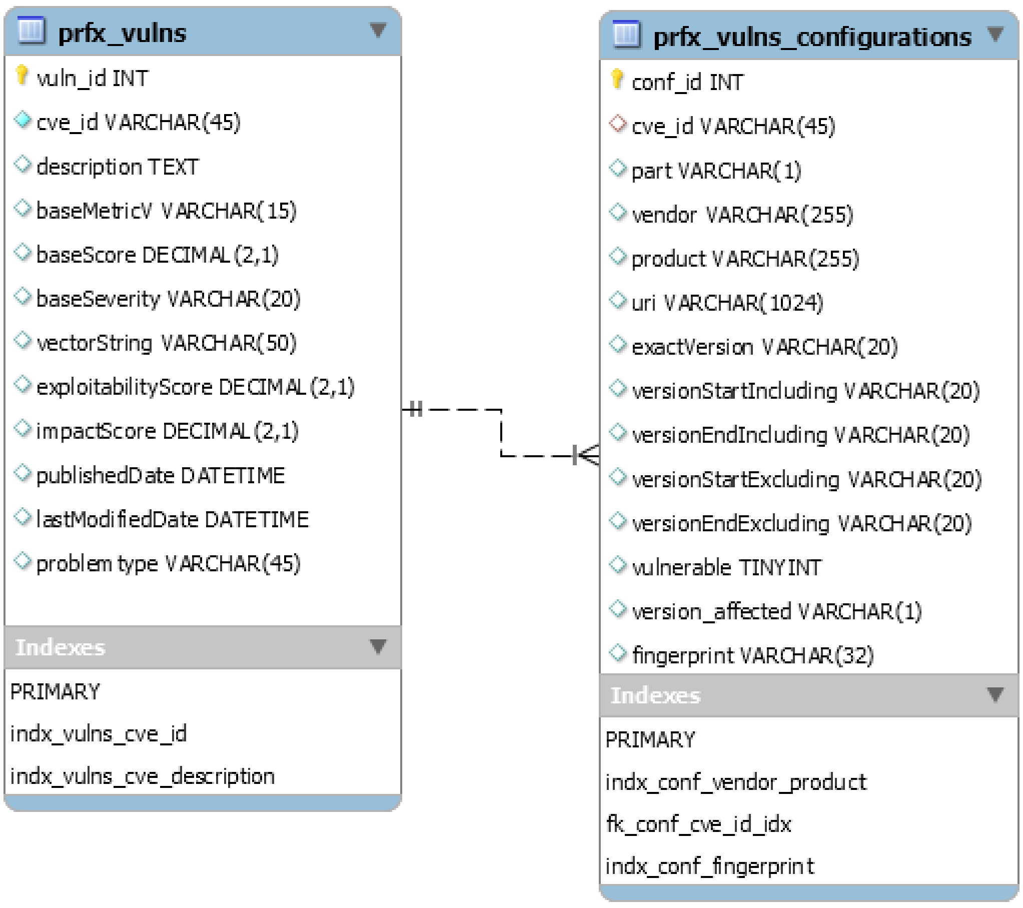


**Figure 17.** Example of EER Diagram of the proposed solution

Source: by author

As mentioned in the previous chapters, the NVD offers and publishes JSON feeds in order to keep up to date approved scanning vendors. In that case, scanning tools need to extract stored information from the Semi-structured data. For storing the data in the MariaDB, two essential tables were created. The “prfx_” patterns in the table names were created for security reasons which help to prevent attackers from guessing the table names through the database. It can be a mix of random keywords and numbers in the production environment. Those two tables include:

- **prfx_vulns (T1)** – for storing the CVE information such as ID, description, severities, published date, etc.;

- **prfx_vulns_configurations (T2)** – is considered for keeping the CPE identifiers and creating the fingerprints which will be used during the matching process by the EWAS engine.

T1 can have multiple CPE entries or not. However, without CVEs, CPE identifiers cannot exist. T2 linked to T1 (cve_id) with a foreign key called "FK cve_id". In the T1, two indexes were created for improving the speed of data retrieval operations. They are as follows:

- indx_vulns_cve_id – as a standard index;
- indx_vulns_cve_description – as a full-text which is a particular type of index that provides index access against a character or binary column data for full-text queries.

Additionally, in T2, three indexes were created, which are below:

- indx_conf_vendor_product – it includes two columns such as vendor and product names, as well as was indexed as a full-text for Natural Language Full-Text Searches;
- fk_conf_cve_id_idx – indexed as a standard way for speeding up;
- indx_conf_fingerprint – indexed as a usual way for speeding up.

Those created tables are only for model and design, which can be improved by any of the other scientists or researchers in order to get more accurate results.

## 3.6. System requirements and specification

Any system's performance depends on meeting requirements listed under two corresponding categories. First, the functional requirements (Table 3) are the operations of the system from the user's viewpoint, which define the internal and external interactions with the system under the predefined state. Second, the non-functional requirements (NFRs) are primarily the constraints of the system which impose specific requirements and qualities on the system to be built (Table 4). Acceptance of testing of the program must also be focused on both the functional and non-functional system specifications (Kassem Saleh, 2009). In this section, the main parts of the specification and types of requirements for the proposed solution will be presented (Table 5).

**Functional Requirements** – it covers 6 (six) essential functional requirements (see Table 3) that describes what the software system should do in order to fit criteria.

**Table 3.** Functional Requirements

| **Requirement ID** | **Description** | **Must/Want** | **Fit criteria** |
|---|---|---|---|
| **FR01** | The system should extract data from vulnerability databases/resources such as NVD or CVE MITRE. | *Must* | The system should be run automatically and give functionality (web-based console) to users which they can be configured system. |
| **FR02** | Agents should gather data from assets such as which software products were installed on the environment. | *Must* | The system should give functionality which accepts data from the agents and make them able to send data to the system. |
| **FR03** | The system should be able to parse and dispatch queries to local vulnerability database and get accurate results. | *Want* | The requested data should match with vulnerability descriptions and identifiers which stored on local vulnerability database. |
| **FR04** | The system should identify vulnerabilities and ensure to avoid false negatives. | *Must* | The System should achieve this goal through detecting vulnerabilities, giving alerts and early warnings using complex algorithms. |
| **FR05** | The system should produce a report regarding the result of the identification process. | *Must* | The system should ensure reviewing and downloading of reports by users. |
| **FR06** | The system should offer API for integrated services in order to interact with Web-based Console and system itself. | *Must* | The system should give functionality which accepts data from the agents, collectors and integrated services and make them able to send data to the system. |

Source: by author

**Non-functional Requirements -** Through non-functional requirements,  is going to be listed as a performance characteristic of the system, while placing constraints on how the system will do so and fit the security requirements. First of all, in  Table 4, performance-based non-functional requirements will be presented.

**Table 4.** Non-functional Requirements

| Requirement ID | Description | Must/Want | Fit criteria |
|---|---|---|---|
| **NR01** | The application shall be easy to use by users, including administrators and managers. | *Must* | It defines how difficult it will be for a user to learn and operate the system. |
| **NR02** | The application shall allow several requests to be made at the same time without downgrading performance. | *Want* | The system should provide the ability to deliver service at specified levels for a stated period. |
| **NR03** | Access permissions for the system information may only be changed by the system's administrator. | *Must* | The system should ensure that the software is protected from unauthorized access to the system and its stored data. |
| **NR04** | The database update process must roll back all related updates when any update fails. | *Want* | The system should ensure work without failure for a given period. |
| **NR05** | The system should have internet access and access to vulnerability resources. | *Must* | The system should able to connect to external resources in order to update its database. |
| **NR06** | Monitoring, crawling, and analysis process should not influence production environment and its availability. | *Want* | The system should gather information from assets once, then should use them for vulnerability identification. |

Source: by author

Addition to requirements mentioned above, one of the important aspect is security requirements (see Table 5) which ensure that the software is protected from unauthorized access/changes to the system and its stored data. It considers different levels of authorization and authentication across different users' roles. For instance, *data privacy* is a security characteristic that describes who can create, see, copy, change, or delete information. Security also includes protection against virus/malware attacks and exploits.

**Table 5.** Security Requirements

| Requirement ID | Possible Vulnerabilities | Severity | Minimal Security Solutions |
|---|---|---|---|
| **SFR01** | Transmitted data and its privacy between communicating applications and users can be compromised. | *Critical* | For information flow, encryption methods shall be used. For instance, Transport Layer Security (TLS) is a cryptographic protocol intended to provide connections protection across a network. For authentication, we should avoid vulnerable protocols which can be compromised. |
| **SFR02** | Unvalidated inputs can be compromised by illegitimate users. | *Critical* | We should validate data preventing SQL injection attacks, using encryption methods for preventing session/cookie hijacking attacks, and preventing from buffer overflow. |
| **SFR03** | Customer 's sensitive data is kept in the local database, due to data leakage, those founded vulnerabilities of their environment/computer system can be compromised intentionally/accidentally by users/attackers which this | *Critical* | For preventing the system from these types of threats, database encryption should be used, as well as Data Leakage Prevention Systems can be used. In addition, the monitoring system should be implemented for tracking traffics and malicious behaviours or anomalies in the system. |

|  |  |  |  |
|---|---|---|---|
|  | information does not belong to them. |  |  |
| **SFR04** | If a system is compromised, illegitimate users can escalate their privileges. | *High* | In order to strengthen the system, the provider must implement Access Control List or Management for preventing privilege escalation and obtaining customer's data using an illegitimate way. |
| **SFR05** | Receiving malicious codes from predefined vulnerability database or resources. So, during retrieving data from resources, malicious codes can be a cause of compromising system accidentally. | *Medium* | Because of that, data validation is important by using white-listing or black-listing, as well as filtering input data using drivers such as PDO. |

Source: by author

## 3.7. Technical Challenges and Handling Limitations

One of the limitations of the proposed concept is unwanted results during the searching and matching vulnerabilities on the database. Due to using traditional algorithms and the nature of search engines, we are always getting the undesired search results. Another limitation that must be considered is CVEs without CPE identifiers. In order to handle such or mentioned limitations in previous chapters, sophisticated algorithms and functions will be proposed in this section. Used ranking methods for identified vulnerabilities and found results are shown below:

- Natural Language Full-Text Search for:
    - CVEs without CPE identifiers;
    - Incorrectly assigned CPE identifiers such as typographical errors;
    - Decreasing undesired search results;
- Fuzzy String Matching with Levenshtein Distance algorithm for double scoring.

On the one hand, the MySQL full-text search provides a simple way to implement various advanced search techniques such as natural language search, Boolean text search and query expansion. This method performs a natural language search for a keyword against a text array and calculates similarities between the search string and the target in that row in the columns. Example of query for getting the most relevant results for CVEs without CPE identifiers is shown in Figure 18. As a keyword for search "zoom meetings" was used. The query looks CVE records in the `*prfx_vulns*` table which do not have CPE identifiers in the `*prfx_vulns_configurations*` table where keyword matches the description of CVE by Natural Language Mode and put the score to results.

```
SET @keyword = 'zoom meetings';

SELECT t1.`cve_id` as cve_id, t1.`description`, t2.`cve_id` as cpe, t1.`publishedDate`,
MATCH(t1.description)
AGAINST
(@keyword IN NATURAL LANGUAGE MODE) AS score
FROM prfx_vulns as t1
NATURAL LEFT JOIN prfx_vulns_configurations as t2
where MATCH(description) AGAINST (@keyword IN NATURAL LANGUAGE MODE) and t2.`cve_id` IS
NULL group by t1.`cve_id` order by score desc limit 0,10;
```

| cve_id | description | cpe | publishedDate | score |
|---|---|---|---|---|
| CVE-2020-11876 | airhost.exe in Zoom Client for Meetings 4.6.11 uses the SHA-256 hash of 0123425234234fsdfsdr3242 for initialization of an OpenSSL EVP AES-256 CBC context. | NULL | 04/17/2020 16:15 | 21.898325 |
| CVE-2020-11877 | airhost.exe in Zoom Client for Meetings 4.6.11 uses 3423423432325249 as the Initialization Vector (IV) for AES-256 CBC encryption. | NULL | 04/17/2020 16:15 | 21.898325 |

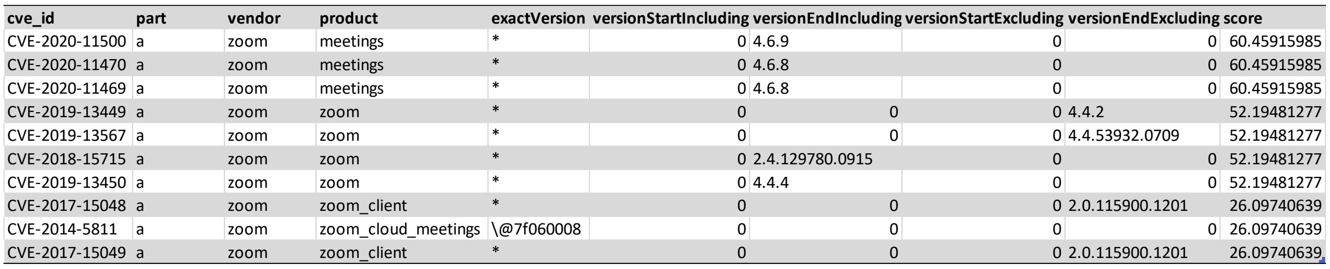
**Figure 18.** Example of Query with Natural Language Mode for CVEs and Results

Source: by author

The score is a positive floating-point number; that is, a similarity measure between the search string and the text. When the score is zero, it refers that there is no similarity. MySQL computes the score or relevance based on diverse factors including the number of words in the document, the number of unique words in the document, the total number of words in the collection, and the number of documents (rows) that comprise a particular word. In the same way, for looking at the CPE identifiers `*prfx_vulns_configurations*` in the table, the same feature was used. It is shown in Figure 19.

```
SET @keyword = 'zoom meetings';

SELECT  `cve_id`, `part`,  `vendor`, `product` , IFNULL(`exactVersion`, 0) as exactVersion,
IFNULL(`versionStartIncluding`, 0) as versionStartIncluding, IFNULL(`versionEndIncluding`,
0) as versionEndIncluding,
IFNULL(`versionStartExcluding`, 0) as versionStartExcluding,
IFNULL(`versionEndExcluding`, 0) as versionEndExcluding,
MATCH(vendor, product)
AGAINST
(@keyword IN NATURAL LANGUAGE MODE) AS score
FROM prfx_vulns_configurations
       where MATCH(vendor, product) AGAINST (@keyword IN NATURAL LANGUAGE MODE) and `part` =
'a' group by cve_id order by score desc limit 0,10;
```



| cve_id | part | vendor | product | exactVersion | versionStartIncluding | versionEndIncluding | versionStartExcluding | versionEndExcluding | score |
|---|---|---|---|---|---|---|---|---|---|
| CVE-2020-11500 | a | zoom | meetings | * | 0 | 4.6.9 | 0 | 0 | 60.45915985 |
| CVE-2020-11470 | a | zoom | meetings | * | 0 | 4.6.8 | 0 | 0 | 60.45915985 |
| CVE-2020-11469 | a | zoom | meetings | * | 0 | 4.6.8 | 0 | 0 | 60.45915985 |
| CVE-2019-13449 | a | zoom | zoom | * | 0 | 0 | 0 | 4.4.2 | 52.19481277 |
| CVE-2019-13567 | a | zoom | zoom | * | 0 | 0 | 0 | 4.4.53932.0709 | 52.19481277 |
| CVE-2018-15715 | a | zoom | zoom | * | 0 | 2.4.129780.0915 | 0 | 0 | 52.19481277 |
| CVE-2019-13450 | a | zoom | zoom | * | 0 | 4.4.4 | 0 | 0 | 52.19481277 |
| CVE-2017-15048 | a | zoom | zoom_client | * | 0 | 0 | 0 | 2.0.115900.1201 | 26.09740639 |
| CVE-2014-5811 | a | zoom | zoom_cloud_meetings | \@7f060008 | 0 | 0 | 0 | 0 | 26.09740639 |
| CVE-2017-15049 | a | zoom | zoom_client | * | 0 | 0 | 0 | 2.0.115900.1201 | 26.09740639 |

**Figure 19.** Example of Query with Natural Language Mode for CPE identifiers and Results

Source: by author

In the first stage, a feature mentioned above of MySQL helps the system to decrease undesired results by scoring them and avoiding returning similarities which equals to 0 (zero).

On the other hand, in order to reduce this unwanted result and finding the most similar records, still is a problem. As seen from Figure 19, search terms were "zoom meetings", but query returned "vendor" and "product" columns which contain only "zoom" keyword. In that case, we do not need to get these results, although they have pretty good scores. For solving this problem, one of the ways is using Fuzzy String-Matching algorithm which helps to compare and measure similarities between the word tokens. In PHP, there is an opportunity to use FuzzyWuzzy library which utilizes Levenshtein distance algorithm for the calculating similarities and scoring the results. In Figure 20 is shown example of extracting the most relevant result by using a specific scorer. As seen from the output, the algorithm obtained 100% similar search term from the choices.

```
use FuzzyWuzzy\Fuzz;
use FuzzyWuzzy\Process;
use FuzzyWuzzy\Collection;

$fuzz = new Fuzz();
$process = new Process($fuzz);

>>> $choices = ['zoom meetings', 'zoom zoom', 'zoom_client', 'zoom_cloud_meetings']
>>> $process->extractOne('zoom meetings', $choices, null, [$fuzz, 'tokenSetRatio'])

=> [
     "zoom meetings",
     100,
   ]
```

**Figure 20.** Example of FuzzyWuzzy usage

Source: by author

This is the reason why the "Levenshtein distance" algorithm is selected. This algorithm helps to measure the difference between two sequences (Haldar & Mukhopadhyay, 2011).

## 3.8. Conclusions of the third chapter

Firstly, the requirements for objectives were mainly based on ensuring all important functions for automation of the process. The system will ensure the requirements for objectives as follows:

- Extraction of data from the NVD will be done using PHP libraries and written codes from scratch;
- The system will use Agents for gathering information about installed software products in the operating systems;
- Limitation of the system during the matching two datasets will be solved using Natural Language Full-Text Searches and Fuzzy string-matching with Levenshtein Distance or Edit Distance algorithm;

Secondly, the system will give functionalities or reporting which users can review identified vulnerabilities during the analysis of results and early warnings related to newly published vulnerabilities. Due to the nature of the program, “false negatives” will be avoided and given accurate results.

Finally, other functionalities were programmed, such as creating and configuring alerts regarding identified vulnerabilities and generating the report. The proposed solution shall give more accurate results during which searching and analysing process rather than functionalities that traditional vulnerability scanner projects offer.

# 4. Experiments and Evaluation of the EWAS

This chapter of the thesis is focused on demonstration of functionalities of the project, listing results of the experiments, evaluating the proposed idea and making a comparison of results between related works done. For testing the prototype of the EWAS, functional testing was used (Ammann & Offutt, 2016), when the EWAS was tested against the functional requirements/specifications. Moreover, functions (or features) of the system were checked by feeding them input and testing output. Functional testing justified that the specifications were adequately satisfied by the application. This form of research is not about how processing occurs, but rather concerned the processing outcomes. It simulates real use of the system but does not make assumptions about any structure of the application.

## 4.1. Implemented Environment

To evaluate the proposed solution, we configured an environment on the Windows 10 virtual machine (see Figure 21). The system itself configured on the XAMPP which is freely available open-source, cross-platform web server solution (Apache Friends, 2017) and installed on the Windows 10. On the vulnerable virtual machine, python script installed (Roman Inflianskas, 2019). The script is gathering information regarding installed applications in a simplified way and sends data to the system via API.

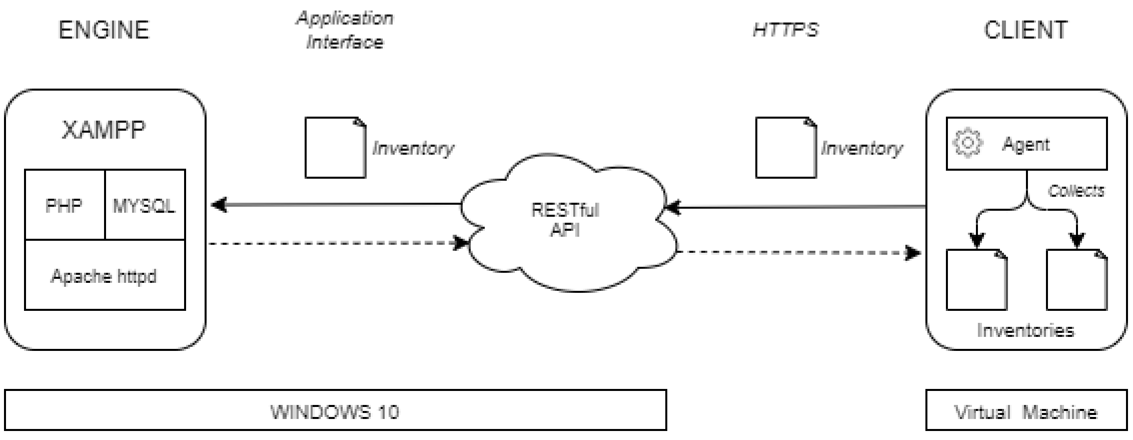


**Figure 21.** Testing environment

Source: by author

In the lab environment, 24 (twenty-four) software products were installed (Table 6). Some of them were with known vulnerable versions (Sanguino & Uetz, 2017).

**Table 6.** List of installed software products in the lab environment

| | Vendor | Product | Version |
|---|---|---|---|
| **1** | Microsoft Corporation | Microsoft .NET Framework 4 Client Profile | 4.0.30319 |
| **2** | Microsoft Corporation | Microsoft .NET Framework 4 Extended | 4.0.30319 |
| **3** | Mozilla | Mozilla Firefox 48.0.2 (x64 en-GB) | 48.0.2 |
| **4** | Oracle | Java(TM) 6 Update 45 (64-bit) | 6.0.450 |
| **5** | Apple Inc. | Apple Mobile Device Support | 9.0.0.26 |
| **6** | Oracle | Java(TM) SE Development Kit 6 Update 45 (64-bit) | 1.6.0.450 |
| **7** | Apple Inc. | Bonjour | 3.0.0.10 |
| **8** | Apple Inc. | iTunes | 12.1.3.6 |
| **9** | Oracle Corporation | MySQL Server 5.7 | 5.7.15 |
| **10** | Adobe Systems Incorporated | Adobe AIR | 20.0.0.260 |
| **11** | Adobe Systems Incorporated | Adobe Flash Player 23 NPAPI | 23.0.0.207 |
| **12** | Google LLC | Google Chrome | 78.0.3904.108 |
| **13** | Mozilla | Mozilla Thunderbird 38.6.0 (x86 en-US) | 38.6.0 |
| **14** | Mozilla | Mozilla Maintenance Service | 38.6.0 |
| **15** | Mozilla | SeaMonkey 2.35 (x86 en-US) | 2.35 |
| **16** | The Wireshark developer community, https://www... | Wireshark 2.0.0 (64-bit) | 2.0.0 |
| **17** | Oracle Corporation | MySQL Installer - Community | 1.4.17.0 |
| **18** | Adobe Systems Incorporated | Adobe Reader XI (11.0.17) | 11.0.17 |
| **19** | Apple Inc. | Apple Application Support (32-bit) | 3.1.3 |
| **20** | Python Software Foundation | Python Launcher | 3.7.6386.0 |
| **21** | Apple Inc. | Apple Software Update | 2.1.4.131 |
| **22** | Zoom Video Communications, Inc. | Meetings | 4.6 |
| **23** | Microsoft | Internet Explorer | 8 |

Source**:** by author

## 4.2. Results of Data Collection from NVD Feeds and Analysis

As mentioned before, the project can keep up to date with the NVD data feeds. During the experiment on the 28th of April 2020, the system collected **143,278** CVEs (see Table 7) and **2,484,591** CPE identifiers. In addition, the analysis showed that **540** CVEs do not have CPE identifiers, and **2,348,766** CPEs were marked as vulnerable.

**Table 7.** Last ten collected CVEs

| CVE_ID | Base Score | Base Severity | Exploitability Score | Impact Score | Published Date | Problem Type |
|---|---|---|---|---|---|---|
| CVE-2020-9785 | 7.8 | HIGH | 1.8 | 5.9 | 04/01/2020 18:15 | CWE-119 |
| CVE-2020-9784 | 4.3 | MEDIUM | 2.8 | 1.4 | 04/01/2020 18:15 | NVD-CWE-Other |
| CVE-2020-9783 | 8.8 | HIGH | 2.8 | 5.9 | 04/01/2020 18:15 | CWE-416 |
| CVE-2020-9781 | 5.3 | MEDIUM | 3.9 | 1.4 | 04/01/2020 18:15 | CWE-281 |
| CVE-2020-9780 | 3.3 | LOW | 1.8 | 1.4 | 04/01/2020 18:15 | CWE-200 |
| CVE-2020-9777 | 5.3 | MEDIUM | 3.9 | 1.4 | 04/01/2020 18:15 | CWE-20 |
| CVE-2020-9776 | 3.3 | LOW | 1.8 | 1.4 | 04/01/2020 18:15 | CWE-200 |
| CVE-2020-9775 | 5.3 | MEDIUM | 3.9 | 1.4 | 04/01/2020 18:15 | CWE-665 |
| CVE-2020-9773 | 3.3 | LOW | 1.8 | 1.4 | 04/01/2020 18:15 | CWE-200 |
| CVE-2020-9770 | 6.5 | MEDIUM | 2.8 | 3.6 | 04/01/2020 18:15 | CWE-326 |

Source: by author

Overall, Figure 22 illustrates how many vulnerabilities belongs to vendors from the extracted CPEs. For example, the highest percentage belongs to Google which was 26.9% and followed by Linux (10.9%),  Mozilla (6.3%), Cisco (5.5%), Apple (3.4%) and other vendors (47.1%), accordingly.

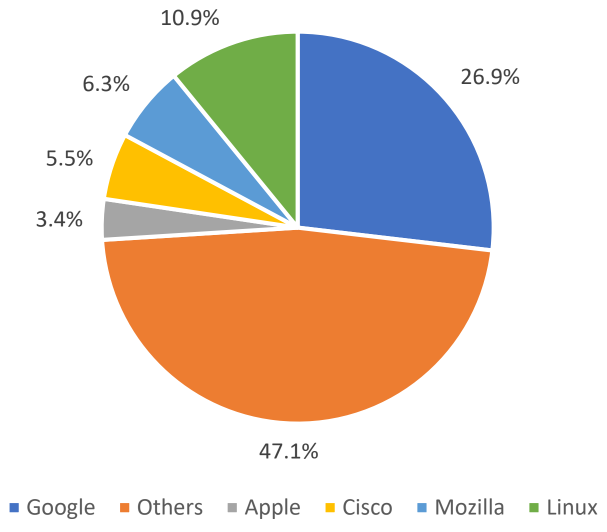


**Figure 22.** The Most Vulnerable Vendors

Source: by author

Furthermore, if we look at data as a product (see Figure 23), we can observe from the pie chart, that the highest vulnerable configurations were related to chrome which was 26.18 per cent. By contrast, seamonkey shared 1.86% of total portion; meanwhile, linux_kernel, ios, firefox and others were 10.87%, 3%, 2.57% and 55.52% respectively.

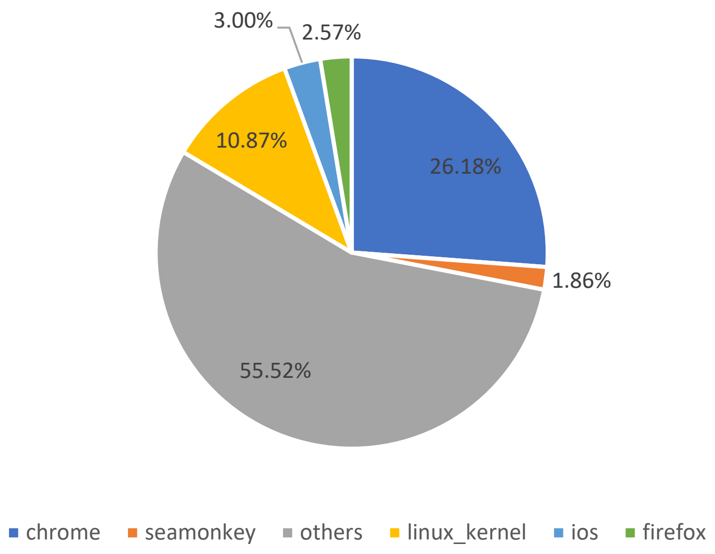


**Figure 23.** The Most Vulnerable Products

Source: by author

From Figure 24, we can see that 69.70 percentage of vulnerable configurations belonged to software products, operating systems were 25.59%, and hardware configurations hold 4.71 percentage.

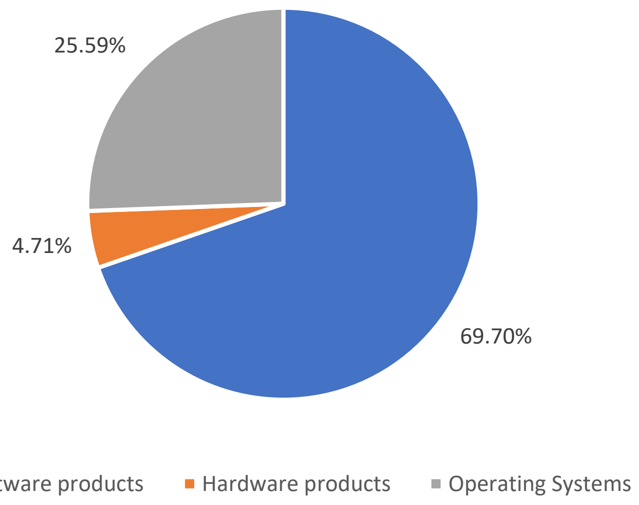


**Figure 24.** Gathered information by attributes

Source: by author

The bar graph (Figure 25) shows the distribution of vulnerabilities by severity over time. The choice of LOW, MEDIUM and HIGH is based upon the CVSS Base score. Vulnerabilities marked as HIGH level are increasing unlike MEDIUM level vulnerabilities are fluctuating. Although CRITICAL vulnerabilities increased during the last three years, there are still less in 2020 than in 2019. To sum up, vulnerabilities with LOW and not defined (Null) severities are less than other severities.

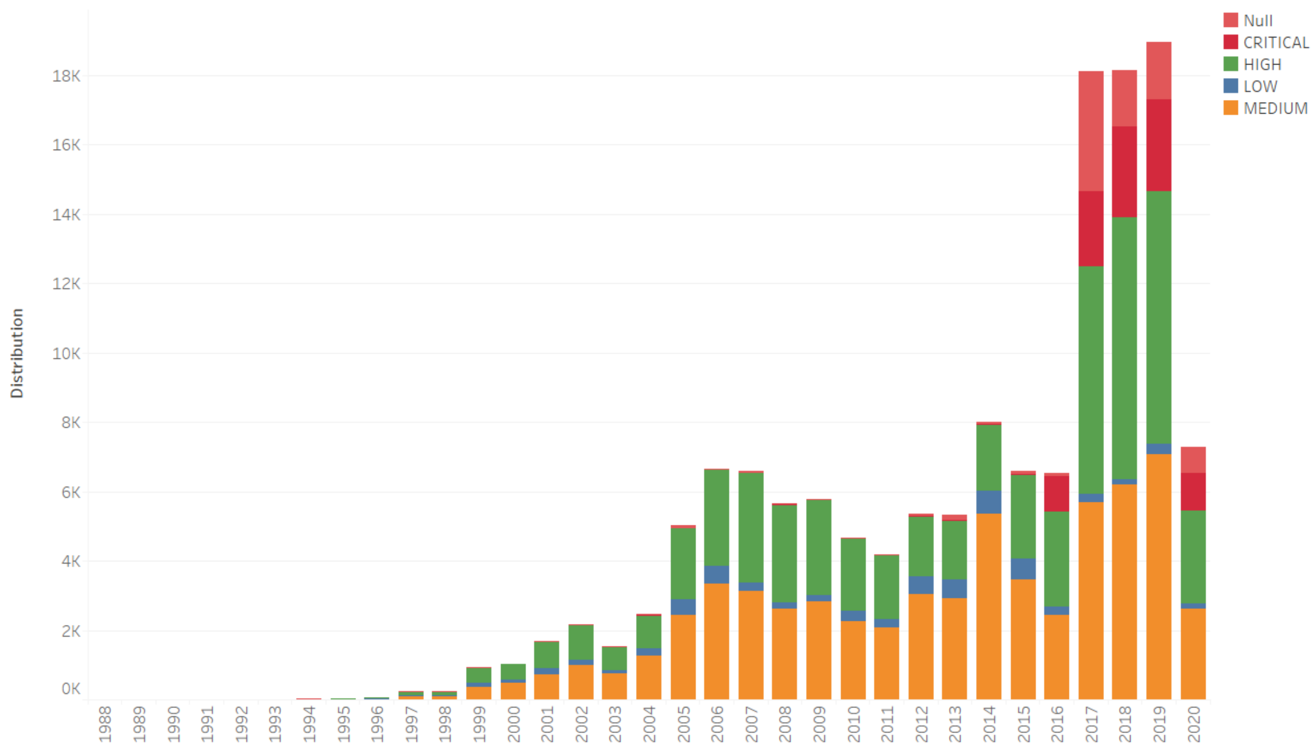


**Figure 25.** CVSS Severity Distribution Over Time

Source: by author

Additionally, as seen from the line chart (Figure 26), published vulnerabilities from 1988 to 1996 were approximately at the same level. On the other hand, from 1998, for 18 years, changes in vulnerabilities started to increase, and some years slightly dropped before the raising. After fluctuating, published vulnerabilities raised to its highest level in 2019 which **18,938** CVEs were published. Finally, the number of vulnerabilities till 28th of April 2020 is **7,266**.

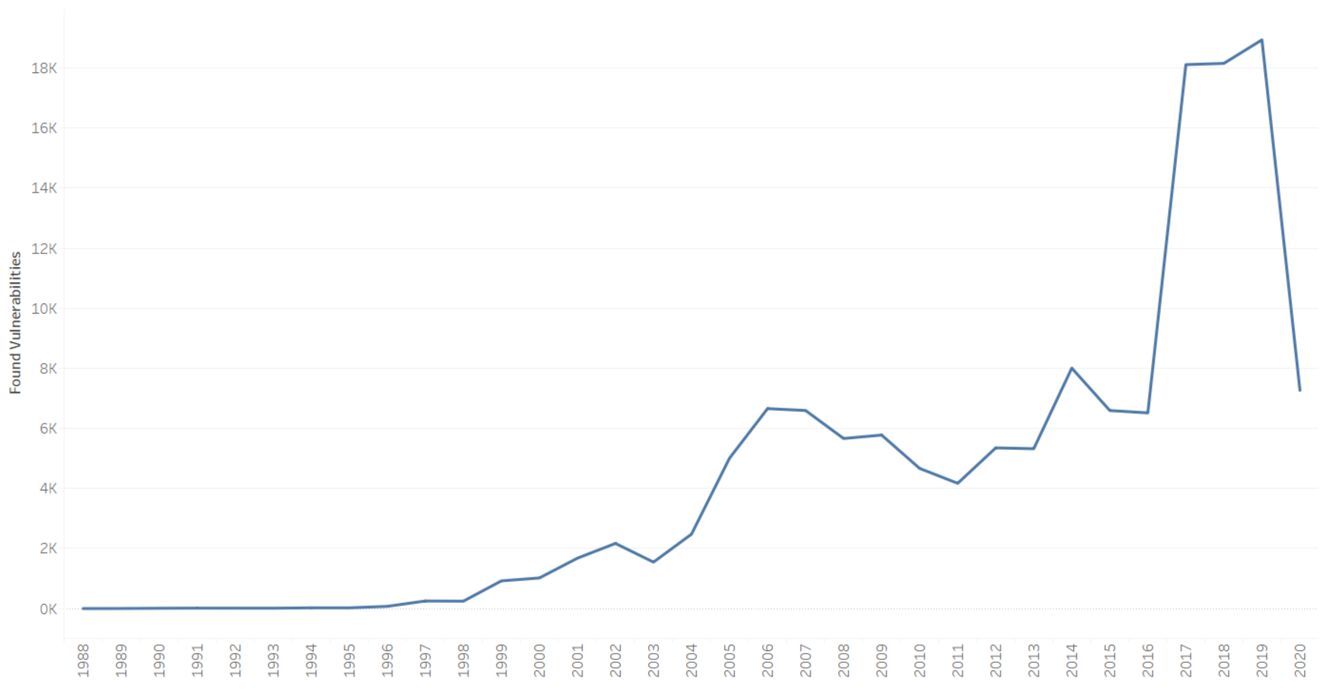


**Figure 26.** Published vulnerabilities by year

Source: by author

## 4.3. Identified Vulnerabilities

Identified vulnerabilities by alerts and early warnings are demonstrated in this subchapter. The result of analysis expresses the capability of a system to detect vulnerabilities and early warnings without human interaction. As seen from Table 8, the system discovered 2270 vulnerabilities and marked 15 of them as an early warning. Moreover, the system gave 196 false positives which are relatively less than true positives.

**Table 8.** Found vulnerabilities by system

| | Vendor | Product | Vulnerable Versions | Detected Vulnerabilitie s | False Positives | Early Warnings |
|---|---|---|---|---|---|---|
| **1** | Mozilla | Firefox | 48.0.2 | 510 | 0 | **10** |
| **2** | Adobe | Flash Player | 23.0.0.207 | 116 | 0 | 0 |
| **3** | Mozilla | Thunderbird | 38.6.0 | 280 | 0 | **5** |
| **4** | Mozilla | SeaMonkey | 2.35 | 1 | 0 | 0 |
| **5** | Adobe | Adobe Reader | 11.0.17 | 222 | 0 | 0 |
| **6** | Oracle | Java SE Development Kit | 1.6.0.450 | 8 | 0 | 0 |
| **7** | Wireshar k | Wireshark | 2.0.0 | 82 | **0** | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **8** | Apple Inc. | iTunes | 12.1.3.6 | 436 | 0 | 0 |
| **9** | Oracle | MySQL Server | 5.7.15 | 167 | 0 | 0 |
| **10** | Adobe | Adobe AIR | 20.0.0.260 | 15 | **1** | 0 |
| **11** | Apple | Bonjour | 3.0.0.10 | 1 | **1** | 0 |
| **12** | Google | Chrome | 78.0.3904.108 | 117 | 0 | 0 |
| **13** | Oracle | MySQL Installer - Community | 1.4.17.0 | 193 | **193** | 0 |
| **14** | Python | Python Launcher | 3.7.6386.0 | 1 | **1** | 0 |
| **15** | Zoom | Meetings | 4.6 | 3 | 0 | 0 |
| **16** | Microsoft | Internet Explorer | 8.0.7601.17514 | 118 | 0 | 0 |
| | | | **Total:** | 2270 | 196 | 15 |

Source: by author

From the bar chart (see Figure 27) it is clear that a number of found CVEs regarding software products were rather equal to correct results, false-positives in other word incorrect results were related to MySQL Installer – Community, Python Launcher, Bonjour and Adobe AIR.

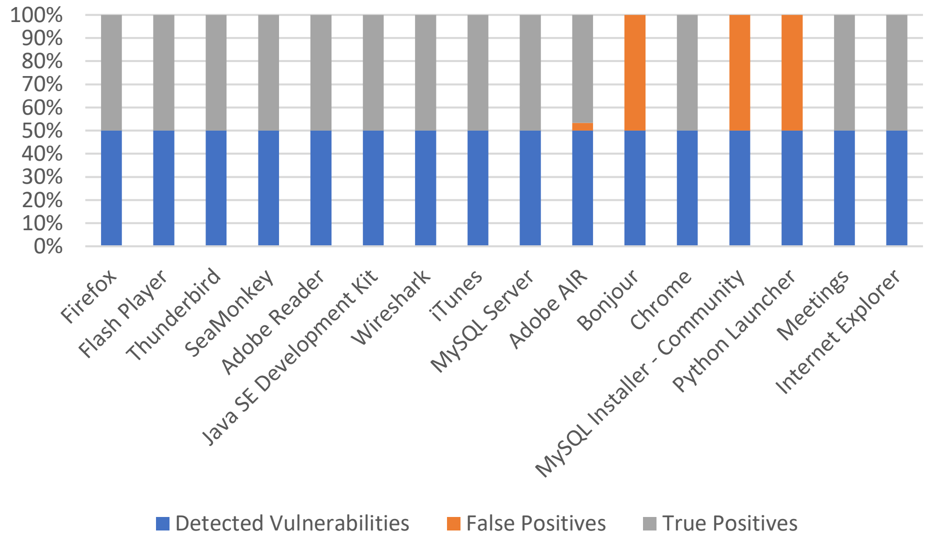


**Figure 27.** Comparison of False and True Positive results

Source: by author

Table 9 demonstrates weaknesses found on the Mozilla Firefox v48.0.2. Due to the complex analysis of versions of software products, the system found this software product among affected products which were their versions below than 67. If the program checks only exact versions, the system will skip these vulnerabilities mentioned in the vulnerability database.

**Table 9.** Example of Found vulnerabilities related to Mozilla Firefox v48.0.2

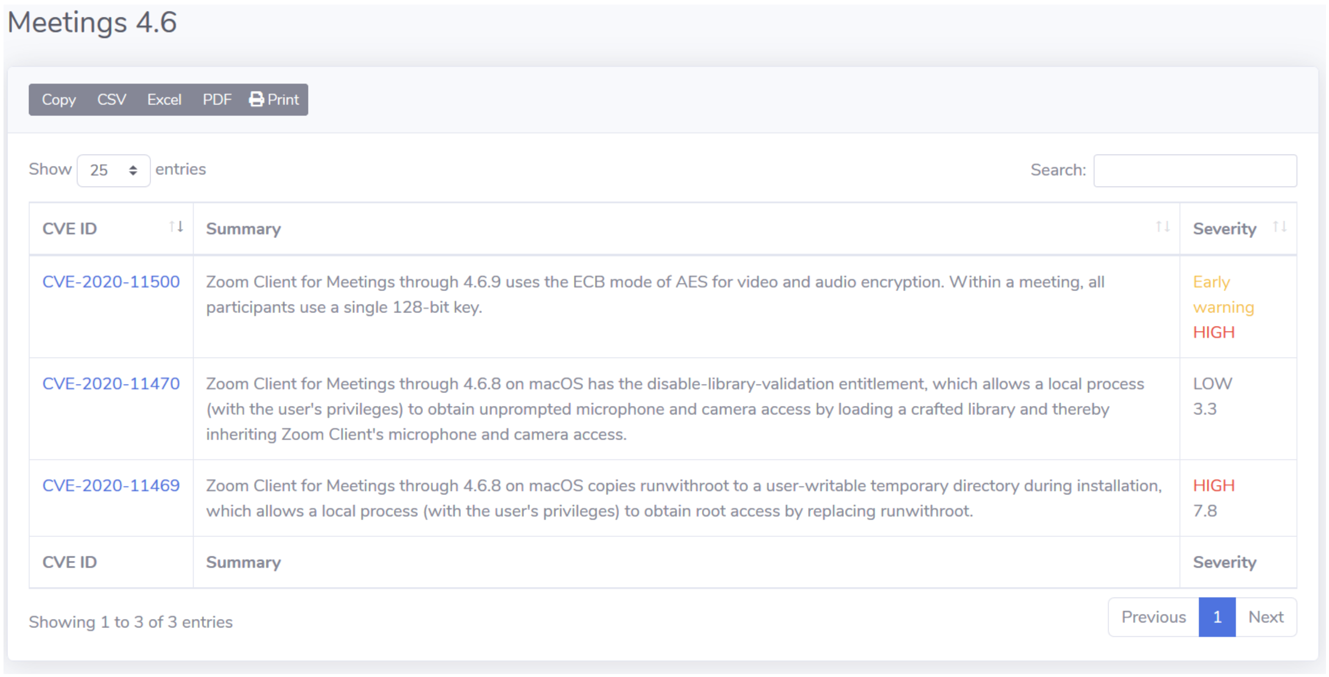
| CVE ID | DESCRIPTION | CVSS Severity |
|---|---|---|
| **CVE-2019-11691** | A use-after-free vulnerability can occur when working with XMLHttpRequest (XHR) in an event loop, causing the XHR main thread to be called after it has been freed. This results in a potentially exploitable crash. This vulnerability affects Thunderbird < 60.7, Firefox < 67, and Firefox ESR < 60.7. | 9.8 CRITICAL |
| **CVE-2019-11692** | A use-after-free vulnerability can occur when listeners are removed from the event listener manager while still in use, resulting in a potentially exploitable crash. This vulnerability affects Thunderbird < 60.7, Firefox < 67, and Firefox ESR < 60.7. | 9.8 CRITICAL |
| **CVE-2019-11693** | The bufferdata function in WebGL is vulnerable to a buffer overflow with specific graphics drivers on Linux. This could result in malicious content freezing a tab or triggering a potentially exploitable crash. *Note: this issue only occurs on Linux. Other operating systems are unaffected.*. This vulnerability affects Thunderbird < 60.7, Firefox < 67, and Firefox ESR < 60.7. | 9.8 CRITICAL |
| **CVE-2019-11694** | A vulnerability exists in the Windows sandbox where an uninitialized value in memory can be leaked to a renderer from a broker when making a call to access an otherwise unavailable file. This results in the potential leaking of information stored at that memory location. *Note: this issue only occurs on Windows. Other operating systems are unaffected.*. This vulnerability affects Thunderbird < 60.7, Firefox < 67, and Firefox ESR < 60.7. | 7.5 HIGH |

| | | |
|---|---|---|
| **CVE-2019-11695** | A custom cursor defined by scripting on a site can position itself over the addressbar to spoof the actual cursor when it should not be allowed outside of the primary web content area. This could be used by a malicious site to trick users into clicking on permission prompts, doorhanger notifications, or other buttons inadvertently if the location is spoofed over the user interface. This vulnerability affects Firefox < 67. | 4.3 MEDIUM |
| **…** | … | … |
| **CVE-2017-7845** | A buffer overflow occurs when drawing and validating elements using Direct 3D 9 with the ANGLE graphics library, used for WebGL content. This is due to an incorrect value being passed within the library during checks and results in a potentially exploitable crash. Note: This attack only affects Windows operating systems. Other operating systems are unaffected. This vulnerability affects Thunderbird < 52.5.2, Firefox ESR < 52.5.2, and Firefox < 57.0.2. | 7.5 HIGH |
| **CVE-2017-7843** | When Private Browsing mode is used, it is possible for a web worker to write persistent data to IndexedDB and fingerprint a user uniquely. IndexedDB should not be available in Private Browsing mode and this stored data will persist across multiple private browsing mode sessions because it is not cleared when exiting. This vulnerability affects Firefox ESR < 52.5.2 and Firefox < 57.0.1. | 8.8 HIGH |

Source: by author

## 4.4. Identified Early Warnings

During the analysis of collected data from the NVD, CVEs showed that there are some vulnerabilities or CVEs which they do not have any CPE identifiers. There are several reasons, one of them is that after publishing new vulnerabilities, they are waiting for analysis. For example, on the 4th of April, the system detected early warning related to Zoom Meetings with version 4.6 (see Figure 28).



Meetings 4.6

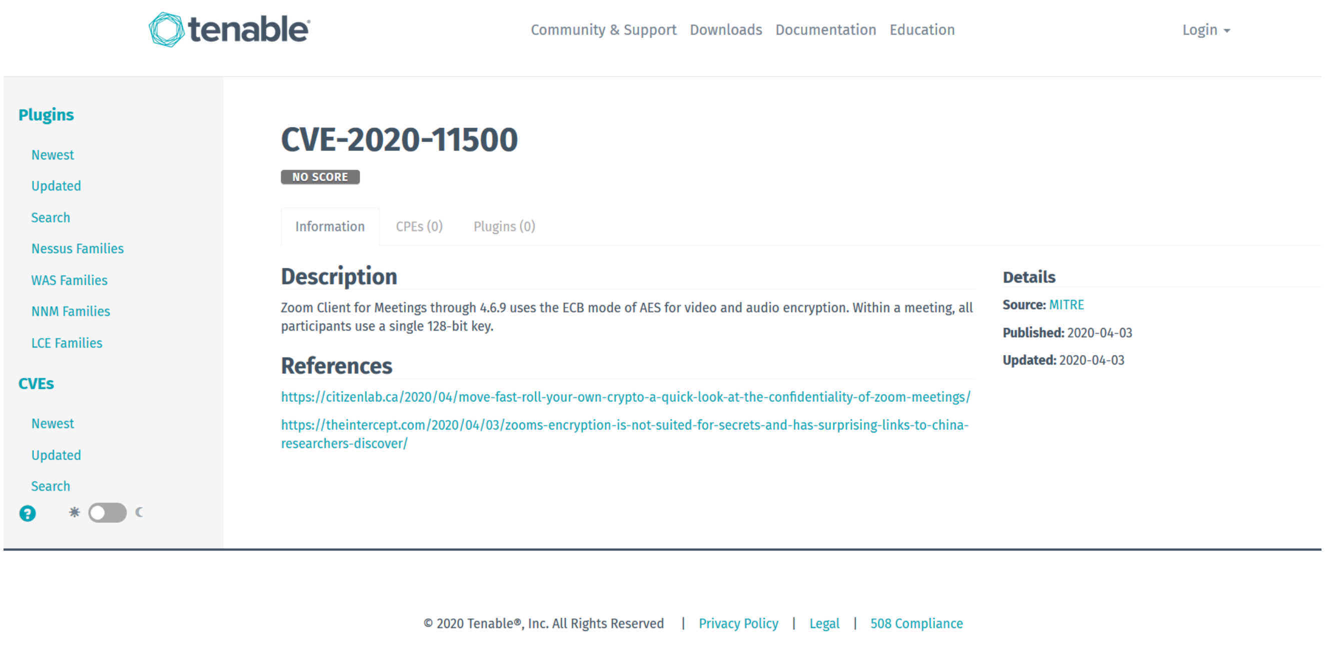
| CVE ID | Summary | Severity |
|---|---|---|
| CVE-2020-11500 | Zoom Client for Meetings through 4.6.9 uses the ECB mode of AES for video and audio encryption. Within a meeting, all participants use a single 128-bit key. | Early warning HIGH |
| CVE-2020-11470 | Zoom Client for Meetings through 4.6.8 on macOS has the disable-library-validation entitlement, which allows a local process (with the user's privileges) to obtain unprompted microphone and camera access by loading a crafted library and thereby inheriting Zoom Client's microphone and camera access. | LOW 3.3 |
| CVE-2020-11469 | Zoom Client for Meetings through 4.6.8 on macOS copies runwithroot to a user-writable temporary directory during installation, which allows a local process (with the user's privileges) to obtain root access by replacing runwithroot. | HIGH 7.8 |
| CVE ID | Summary | Severity |

Showing 1 to 3 of 3 entries

**Figure 28.** Early Warning related to Zoom Meetings with version 4.6

Source: by author

However, other commercial products for vulnerability management such as Nessus (Tenable) had not published plugins or CPE identifiers (see Figure 29) in order to detect vulnerabilities related to CVE-2020-11500 till 6th of April 2020 10:00 pm GMT (*CVE-2020-11500*, 2020).



CVE-2020-11500

Description

Zoom Client for Meetings through 4.6.9 uses the ECB mode of AES for video and audio encryption. Within a meeting, all participants use a single 128-bit key.

References

https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/
https://theintercept.com/2020/04/03/zooms-encryption-is-not-suited-for-secrets-and-has-surprising-links-to-china-researchers-discover/

Details

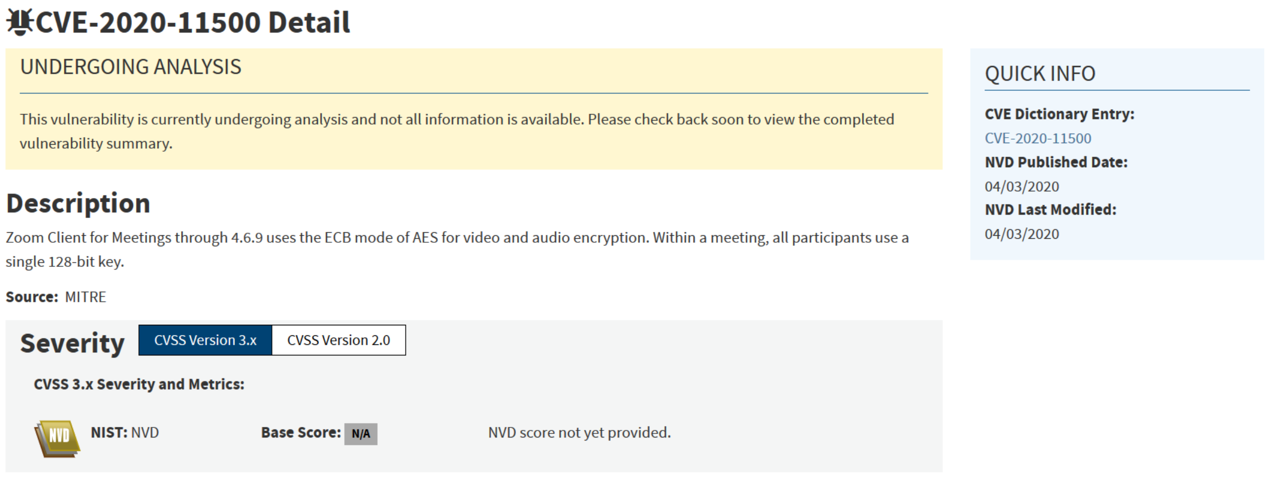
**Source:** MITRE
**Published:** 2020-04-03
**Updated:** 2020-04-03

© 2020 Tenable®, Inc. All Rights Reserved | Privacy Policy | Legal | 508 Compliance

**Figure 29.** CVE-2020-11500 from the Tenable

Source: Official Tenable site (last accessed on 06th of April 2020 10:00 pm GMT)

In that way, if the system does not find any CPE identifier or Plugin for matching, there is a possibility that the program will miss this vulnerability. This situation may increase the number of false negatives in the results. One of the main reasons for that is that newly published vulnerability (*NVD - CVE-2020-11500 Detail*, 2020) whose status was undergoing analysis by the NVD, and CPE identifiers were not assigned (see Figure 30).


CVE-2020-11500 Detail

UNDERGOING ANALYSIS

This vulnerability is currently undergoing analysis and not all information is available. Please check back soon to view the completed vulnerability summary.

Description

Zoom Client for Meetings through 4.6.9 uses the ECB mode of AES for video and audio encryption. Within a meeting, all participants use a single 128-bit key.

Source: MITRE

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD Base Score: N/A NVD score not yet provided.

QUICK INFO

CVE Dictionary Entry: CVE-2020-11500

NVD Published Date: 04/03/2020

NVD Last Modified: 04/03/2020

**Figure 30.** CVE-2020-11500 from the NVD

Source: NVD official site (last accessed on 06th of April 2020 10:00 pm GMT)

## 4.5. Comparison of results between related work and thesis

During vulnerability analysis, the same twelve software products were used, which also examined in the related work (Sanguino & Uetz, 2017). In Table 10, the results of two researches were compared, as well as in order to distinguish results were titled as "Related Research" and "Thesis Results". Found CVEs are divided into two pieces such as correct and incorrect results.

**Table 10.** Comparison results between thesis results and related work

| | Products | Versions | Related Research | | Thesis Results | |
|---|---|---|---|---|---|---|
| | | | Correct | Incorrect | Correct | Incorrect |
| 1 | Mozilla Firefox | 48.0.2 | 18 | 1 | 510 | 0 |
| 2 | Adobe Flash Player | 23.0.0.207 | 17 | 22 | 116 | 0 |
| 3 | Microsoft Internet Explorer | 8 | 340 | 9 | 118 | 0 |
| 4 | Mozilla Thunderbird | 38.6.0 | 9 | 18 | 280 | 0 |
| 5 | Mozilla SeaMonkey | 2.35 | 1 | 474 | 1 | 0 |
| 6 | Adobe Adobe Reader | 11.0.17 | 75 | 162 | 222 | 0 |
| 7 | Oracle Java | 8.0.1120.15 | 0 | 0 | 0 | 0 |
| 8 | Oracle Java SE Development Kit | 8.0.1120.15 | 0 | 0 | 0 | 0 |
| 9 | Wireshark Wireshark | 2.0.0 | 84 | 1 | 82 | 0 |
| 10 | Apple Inc. iTunes | 12.1.3.6 | 3 | 143 | 436 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | Oracle MySQL Server | 5.7.15 | 0 | 0 | 167 | 0 |
| 12 | Adobe AIR | 20.0.0.260 | 23 | 41 | 15 | 1 |

Source: by author and (Sanguino & Uetz, 2017)

From the above Table 10 and Figure 31, we can see that thesis results were rather comprehensive than related work. Addition to getting more correct results, system decreased incorrect results nearly to 0 (zero) by using complex matching phases. Only, results regarding Adobe AIR v. 20.0.0.260 gave false-positive, and it was less than "Related Research". In other words, the outcome of the thesis clearly outweighed related work limitations in many cases.

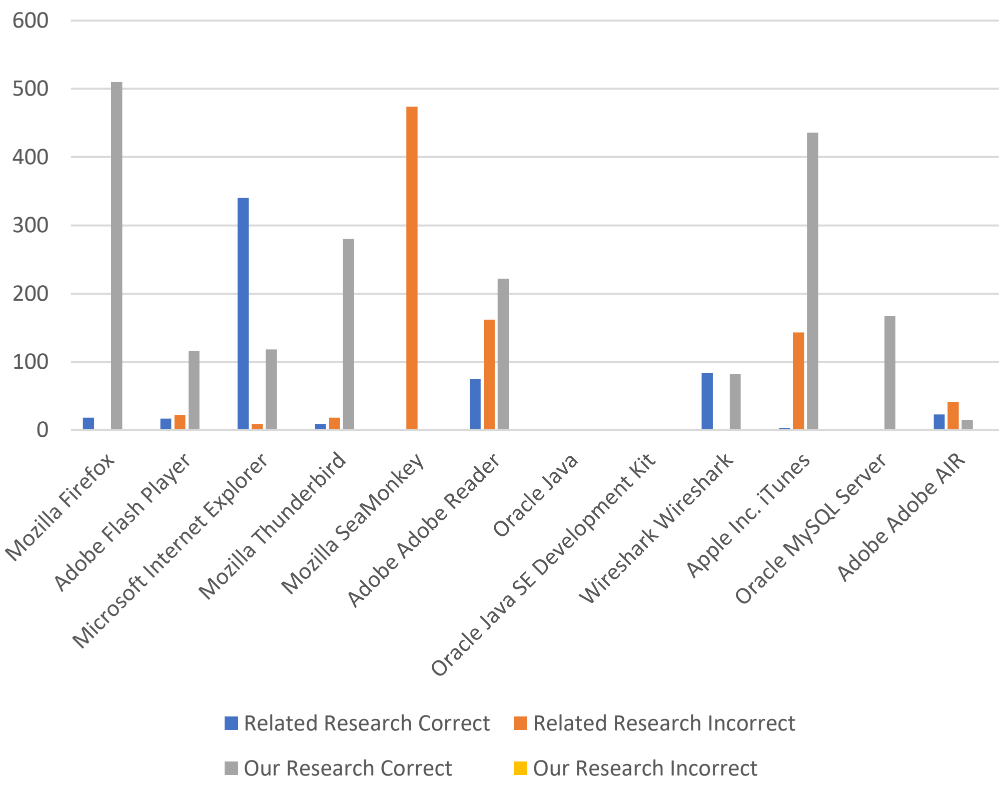


**Figure 31.** Comparison results between thesis results and related work

Source: by author

## 4.6. Evaluation Approach based on OWASP benchmark

For the evaluation of the proposed solution, I used the OWASP benchmark for scoring the prototype of the EWAS (Mburano & Si, 2019), which is one of the important approaches in order to assess vulnerability scanners. The benchmark metrics used to test and evaluate results are as follows:

- True Positives (TP) - Tool correctly identifies a real vulnerability;
- False Positives (FP) - Tool fails to identify a real vulnerability;

- True Negatives (TN) - Tool correctly ignores a false alarm;
- False Negatives (FN) - Tool fails to ignore a false alarm.

**True Positive Rate (TPR)**

$$TPR = \frac{TP(number\ of\ true\ positives)}{TP + FN}$$

**False Positive Rate (FPR)**

$$FPR = \frac{FP(number\ of\ false\ positives)}{FP + TN}$$

**Score**

$$Score = TPR - FPR$$

Source: OWASP Benchmark Project (last accessed on 30th of April 2020)

The score produced by OWASP Benchmark is based on the Youden Index (*OWASP Benchmark*, 2020), which is one of the standard methods that summarize the accuracy of the test set. OWASP benchmark accuracy score is Normalized distance from the "guess line".

In order to evaluate the system, 2300 fake vulnerabilities were created and put the vulnerability database. Then, the system triggered for detection of vulnerabilities to give alerts. As a result, the system ignored false alarms and did not provide alerts.

**TP =** **2074** (*True Positives*)
**FN =** **0** (*False Negatives*)
**TN =** **2300** (*True Negatives*)
**FP =** **196** (*False Positives*)

**Example of calculation:**

$$TPR = \frac{2074}{2074 + 0} = 1$$

$$FPR = \frac{196}{196 + 2300} \approx\ 0.079$$

$$Score = 1 - 0.079 \approx\ 0.92$$

The formula was implemented during evaluation and the system score was 92% which was relatively sufficient in terms of thesis.

## 4.7. The graphical user interface of the system (Web-based console)

This section represents GUI of the application which was used to during experiment in order to fulfil functional requirements. The Web-based console has modules for making ease of using the system like Dashboard, Create Alerts, List Alerts, Found Vulnerabilities and reporting which were designed and developed.

**Dashboard** – This module gives an overall view regarding the results of the analysis and identification of vulnerabilities. When a user logins to the system, the user can see metrics related to the outcome of the system (see Figure 32).

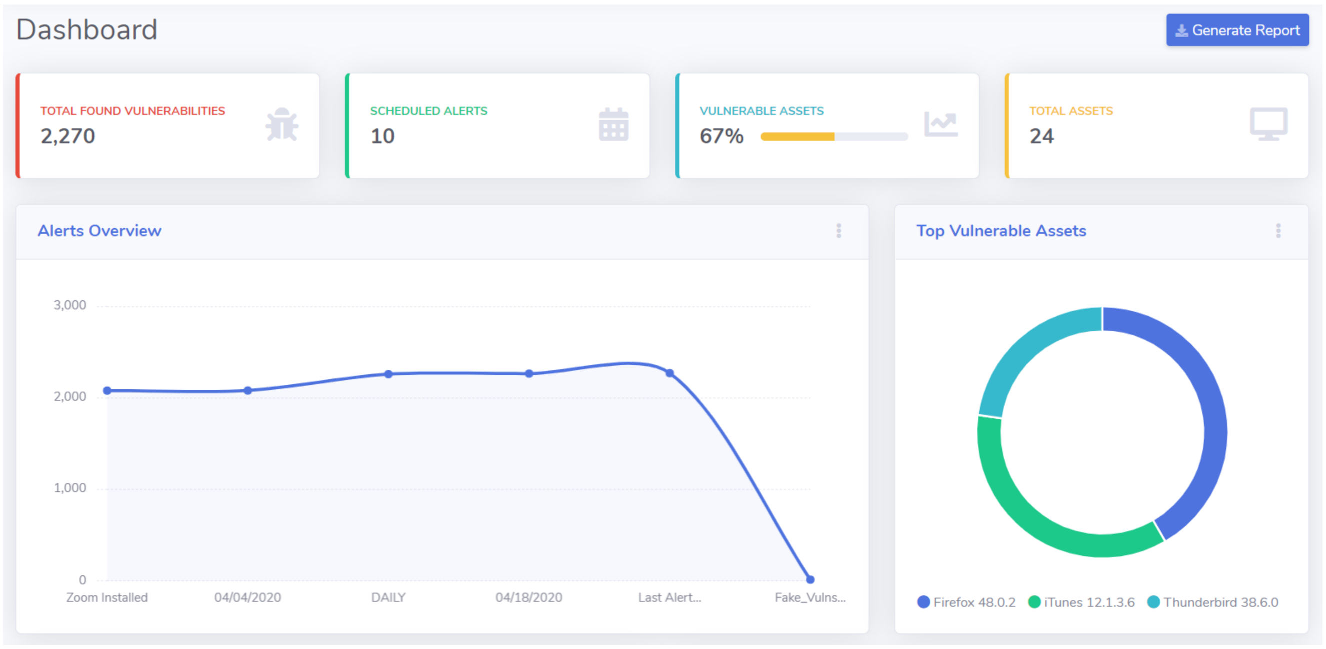
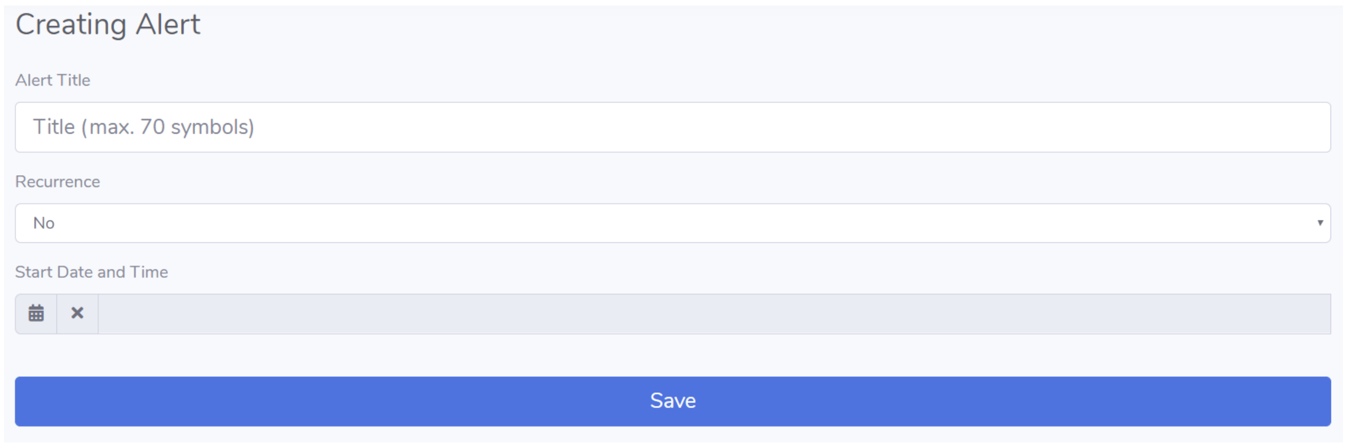


**Figure 32.** The dashboard of the system

Source: by author

The dashboard offers quantitative measurements related to Total found vulnerabilities, Scheduled Alerts, Vulnerable assets and Total assets. Furthermore, user can get information with bar chart which depicts Alerts Overview, and Top vulnerable assets by a pie chart.

**Create Alerts** – Through that option, the user can create alerts and schedule which will be monitoring on the specific date and time or repeatedly with recurrence option (see Figure 33).


Creating Alert

Alert Title

Title (max. 70 symbols)

Recurrence

No

Start Date and Time

Save

**Figure 33.** Creating Alerts

Source: by author

The user can define the title of alert for distinguishing the results, and besides the start date and time, user can put the recurrence which alert will be triggered during pre-defined intervals.

**List Alerts** – List of alerts gives a clue regarding triggered alerts which scheduled by the user and helps to users to check the status of results (see Figure 34).

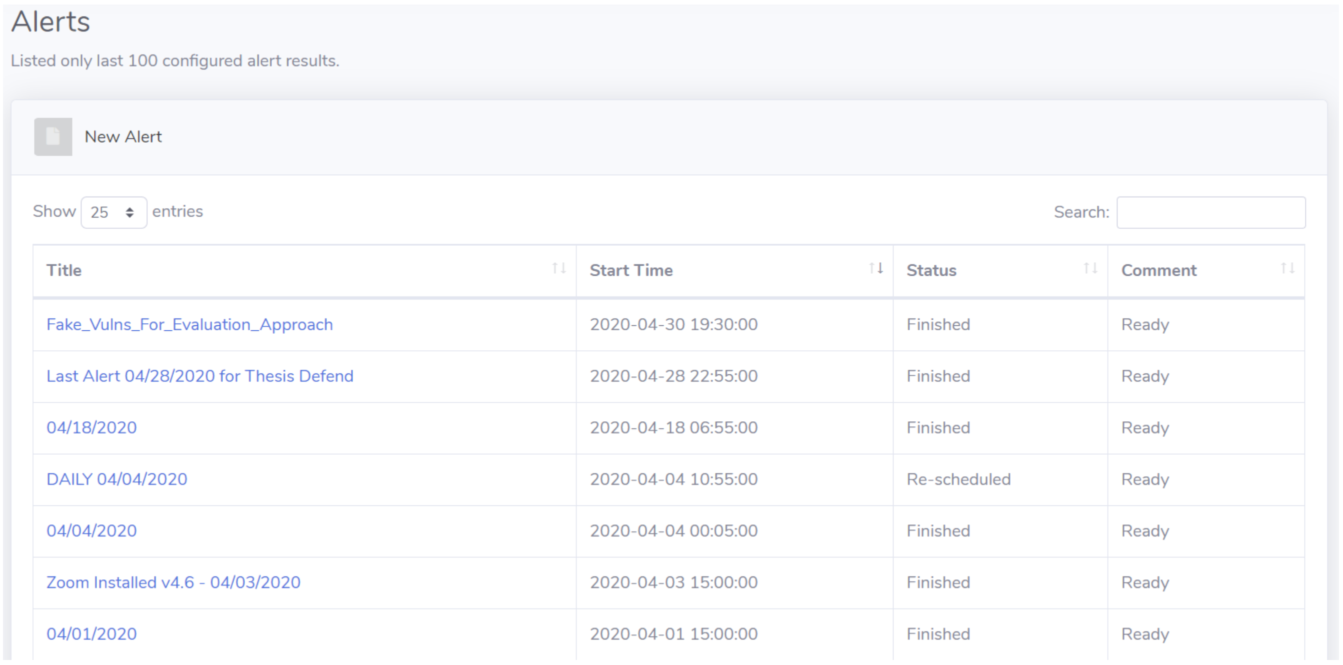
Finished – It means analysis process finished by the system;

Running – It expresses that analysis is going on currently;

Re-scheduled – It means analysis finished and due to recurrence alert scheduled again;

Failed – Means something went wrong during analysis and need to be checked log files;

Ready – Means report generated by the system, and it can be reviewed by the user.



Alerts

Listed only last 100 configured alert results.

New Alert

Show 25 entries Search:

| Title | Start Time | Status | Comment |
|---|---|---|---|
| Fake_Vulns_For_Evaluation_Approach | 2020-04-30 19:30:00 | Finished | Ready |
| Last Alert 04/28/2020 for Thesis Defend | 2020-04-28 22:55:00 | Finished | Ready |
| 04/18/2020 | 2020-04-18 06:55:00 | Finished | Ready |
| DAILY 04/04/2020 | 2020-04-04 10:55:00 | Re-scheduled | Ready |
| 04/04/2020 | 2020-04-04 00:05:00 | Finished | Ready |
| Zoom Installed v4.6 - 04/03/2020 | 2020-04-03 15:00:00 | Finished | Ready |
| 04/01/2020 | 2020-04-01 15:00:00 | Finished | Ready |

**Figure 34.** Alert List

Source: by author

By selecting and opening the alert titles, users can review and check the found vulnerabilities.

**Found Vulnerabilities and reporting** – If the report generated by the system, users can check the visual results of the alerts (see Figure 35). Users are able to revise results regarding found vulnerabilities and total analyzed assets by numbers.

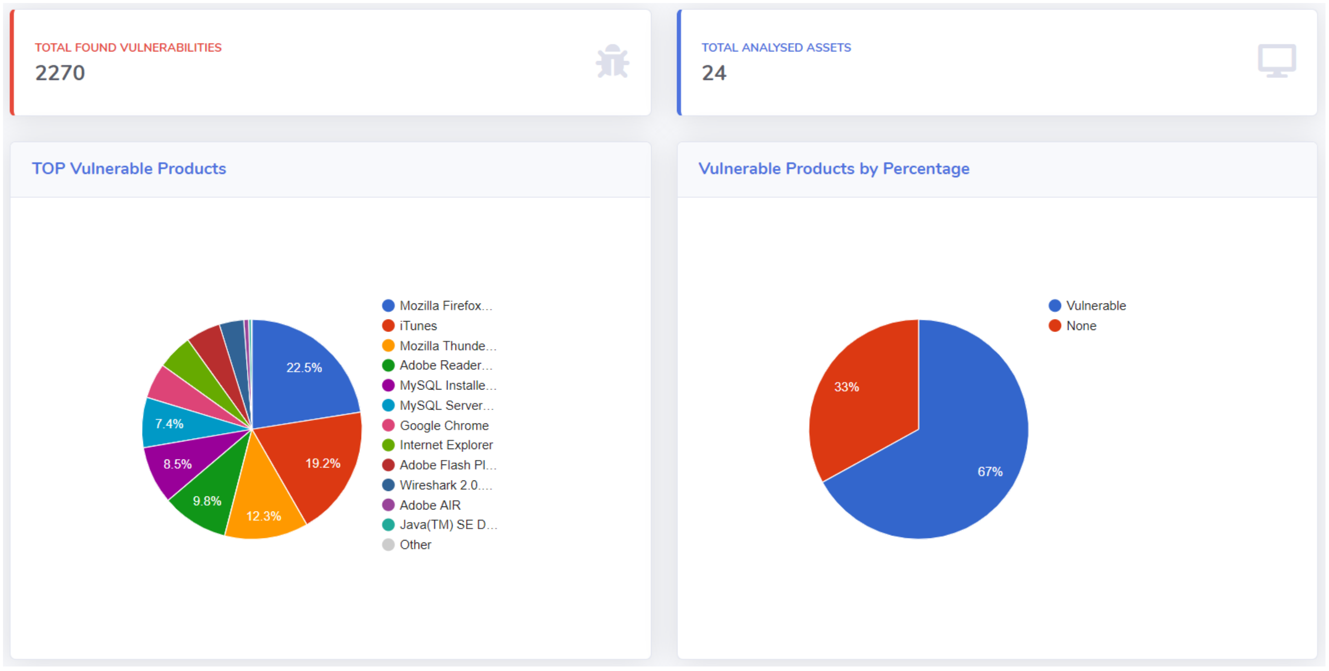
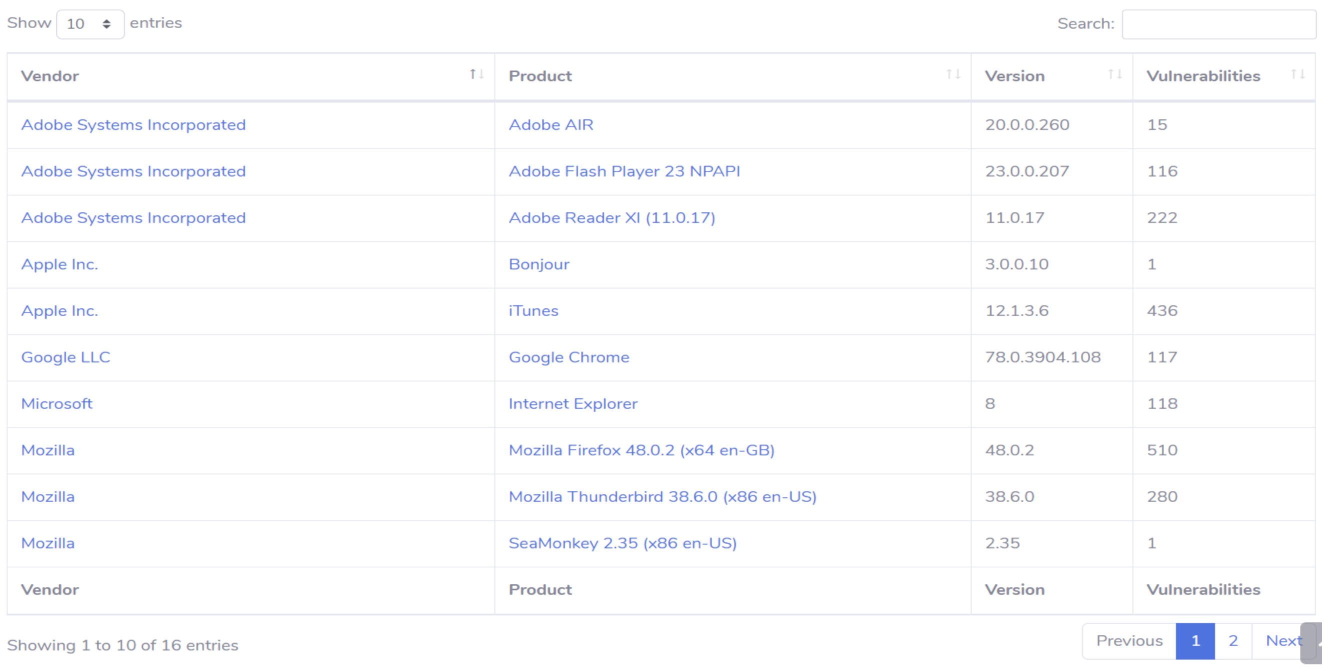


**Figure 35.** Example reports of the result

Source: by author

Also, users can check the Top and vulnerable products by percentage via the help of pie charts. Moreover, this reporting functionality (see Figure 36), also aggregates and groups total found vulnerabilities by products names which user can review detected vulnerabilities by clicking the software products.



| Vendor | Product | Version | Vulnerabilities |
|---|---|---|---|
| Adobe Systems Incorporated | Adobe AIR | 20.0.0.260 | 15 |
| Adobe Systems Incorporated | Adobe Flash Player 23 NPAPI | 23.0.0.207 | 116 |
| Adobe Systems Incorporated | Adobe Reader XI (11.0.17) | 11.0.17 | 222 |
| Apple Inc. | Bonjour | 3.0.0.10 | 1 |
| Apple Inc. | iTunes | 12.1.3.6 | 436 |
| Google LLC | Google Chrome | 78.0.3904.108 | 117 |
| Microsoft | Internet Explorer | 8 | 118 |
| Mozilla | Mozilla Firefox 48.0.2 (x64 en-GB) | 48.0.2 | 510 |
| Mozilla | Mozilla Thunderbird 38.6.0 (x86 en-US) | 38.6.0 | 280 |
| Mozilla | SeaMonkey 2.35 (x86 en-US) | 2.35 | 1 |
| Vendor | Product | Version | Vulnerabilities |

Showing 1 to 10 of 16 entries

**Figure 36.** Example of found vulnerabilities by-products

Source: by author

Finally, by clicking the product or vendor title, the user can view detected vulnerabilities and early warnings.

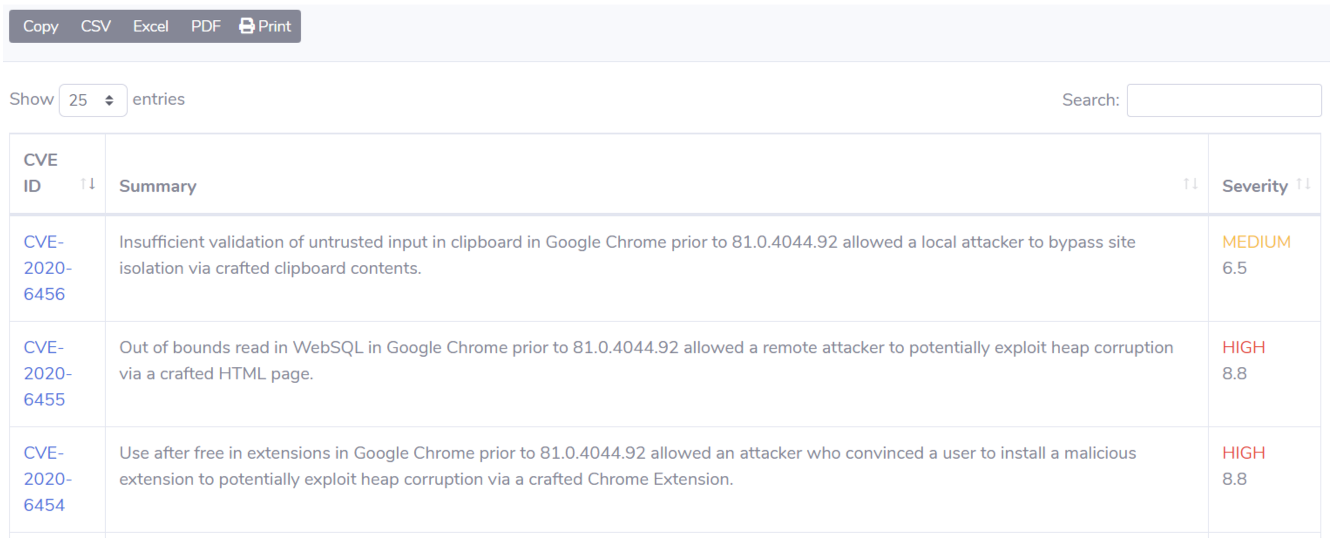
Information in the data table includes:

CVE ID – it links data to NVD database which user can get more detailed information regarding found vulnerability;

Summary – Or a description of vulnerability explains the actual state and impact of weakness;

Severity – It helps users to decide which vulnerabilities should be fixed first;

Early Warning – It marks vulnerability for paying more attention due to the newest information regarding vulnerability.



| CVE ID | Summary | Severity |
|---|---|---|
| CVE-2020-6456 | Insufficient validation of untrusted input in clipboard in Google Chrome prior to 81.0.4044.92 allowed a local attacker to bypass site isolation via crafted clipboard contents. | MEDIUM 6.5 |
| CVE-2020-6455 | Out of bounds read in WebSQL in Google Chrome prior to 81.0.4044.92 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. | HIGH 8.8 |
| CVE-2020-6454 | Use after free in extensions in Google Chrome prior to 81.0.4044.92 allowed an attacker who convinced a user to install a malicious extension to potentially exploit heap corruption via a crafted Chrome Extension. | HIGH 8.8 |

**Figure 37.** Example of found vulnerabilities related to Google Chrome v78.0.3904.108

Source: by author

In addition, the user can search for any vulnerability by using keywords, as well as copy and extract data to different formats or print.

## 4.8. Conclusions of the fourth chapter

During the testing process, the agent collected a total of 23 pre-installed applications, including 12 vulnerable products which used related work. The system gathered 143,278 CVEs from the NVD, and last import was at 28th of April 2020. Then extracted 2,484,591 CPEs (known vulnerable configurations) from the CVEs which 2,348,766 of them marked as vulnerable by NVD and analysis showed that 540 CVEs do not have CPE identifiers. Generally, 60.70 percentage of vulnerable configurations belong to applications, operating systems are 25.59%, and hardware configurations make up 4.71 percentage. This experiment focused only on applications/software products. Distribution of vulnerabilities by severity over time depicted that vulnerabilities marked as a HIGH level is increasing, and MEDIUM level vulnerabilities are fluctuating, the published vulnerabilities raised to the highest level in 2019 as well. Finally, the collected data represented that the most top shares by vendors belong to Google which was 26.9% and followed by Linux (10.09%), Mozilla (6.3%), Cisco (5.5%) and other vendors (47.1%), accordingly.

Furthermore, matching software products information against vulnerable products Natural Language Full-Text Searches and Fuzzy string matching was used and given scores for eliminating irrelevant results. Addition to that, version comparison was complex analysis because the system is looking at ranges between the vulnerable version of software products, not only at exact versions.

After matching data and identification of vulnerabilities, the system detected 2270 vulnerabilities and 196 of them were false positives. Additionally, the system was able to identify 15 early warnings related to newly published vulnerabilities before the commercial products. On the other hand, the system leads to making slight mistakes in finding the proper weaknesses related to software products.

For tool evaluation, the OWASP benchmark was used to score a tool's security analysis results against it, and evaluation score was 92% that can be a significant outcome.

To conclude, the system successfully collected data from the NVD and pre-installed inventories from the virtual machine, classified, aggregated information and stored on its database.  Analysis of gathered data showed that the system achieved this without any error-prone. The generated report demonstrates that using "Natural Language Full-Text Searches and Fuzzy string matching" gave the best results and eliminated irrelevant results. Besides providing true positives, the system was able to detect early warnings. In addition to that, using sophisticated algorithms for comparison of software versions, increased accurate outcomes and dropped incorrect results almost to zero.

# 5. Conclusions

First of all, a theoretical study was carried out: relevant information was collected, analyzed, and research opportunities and limitations of VMSs were identified, as well as the most suitable algorithms for the EWAS were chosen.

Secondly, based on the analyzed information regarding limitations of existed solutions, a framework was prepared, which consists of phases of the data collection and parsing of data, vulnerability identification and an early warning selection, populating data for the Web-based console. In common sense, the proposed framework is a solution for handling the cons of vulnerability assessment tools.

Thirdly, for the practical part, prototyping methodology, comparative and experimental study were chosen. The evaluation of the system was divided into two phases, such as testing against functional requirements and using the OWASP Benchmark scoring system. In the initial phase, the system was tested against functional requirements and accomplished these goals with slight errors:

- Successfully collected data from NVD without any error-prone;
- Detected Early warnings with minor mistakes;
- Removed False Negatives;
- Gave more true positives than false positives during the identification of vulnerabilities;
- Decreased Human interaction;
- Finally, increased accurate outcomes and dropped incorrect results almost to zero.

Furthermore, the generated report by the Web-based console demonstrated that using “Natural Language Full-Text Searches and Fuzzy string matching” gave accurate results and eliminated irrelevant results during the vulnerability identification process.

In the second evaluation phase, the overall, the benchmark score of the prototype in the identification of vulnerabilities was 92%, and the system detected 2270 vulnerabilities and 196 of them were false positives. Additionally, the system was able to identify 15 early warnings related to newly published vulnerabilities before the commercial products.

In conclusion, the comparison of thesis results with related work and evaluation with OWASP benchmark scoring system characterized that system gained more proper results with slight mistakes which overcome the limitation of vulnerability analysis by using sophisticated algorithms and methods. The EWAS will help individuals or organizations to respond to active threats and create compensation measures on time, especially in terms of software products.

# 6. References

*About CVE*. (2019). https://cve.mitre.org/about/index.html

Ammann, P., & Offutt, J. (2016). Introduction to Software Testing. In *Introduction to Software Testing*. https://doi.org/10.1017/9781316771273

Apache Friends. (2017). *XAMPP Installers and Downloads for Apache Friends*. Https://Www.Apachefriends.Org/Index.Html. https://doi.org/10.2212/spr.2011.1.4

Apel, M., Biskup, J., Flegel, U., & Meier, M. (2010). Towards early warning systems - Challenges, technologies and architecture. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6027 LNCS*(October), 151–164. https://doi.org/10.1007/978-3-642-14379-3_13

Carnegie Mellon University for the operation of the Software Engineering, & Institute. (2016). *CRR Supplemental Resource Guide, Volume 4: Vulnerability Management, Version 1.1*. *4*. https://www.us-cert.gov/sites/default/files/c3vp/crr_resources_guides/CRR_Resource_Guide-VM.pdf

Cheikes, B. A., & Scarfone, K. (2011). *Common Platform Enumeration : Naming Specification Version 2 . 3*. http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf

*CPE - Common Platform Enumeration: CPE Specifications*. (2020). https://cpe.mitre.org/specification/

*CVE-2013-4869 Detail*. (2013). https://nvd.nist.gov/vuln/detail/CVE-2013-4869

*CVE-2020-11470*. (2020). https://www.tenable.com/cve/CVE-2020-11470

*CVE-2020-11500*. (2020). CVE-2020-11500

*Database Design and Modeling*. (2020). https://dev.mysql.com/doc/workbench/en/wb-data-modeling.html

*Fuzzy string matching for PHP*. (2020). https://github.com/wyndow/fuzzywuzzy

Granova, A., & Slaviero, M. (2017). Cyber Warfare. In *Computer and Information Security Handbook* (Vol. 13, Issue April 2008). Elsevier Inc. https://doi.org/10.1016/B978-0-12-803843-7.00083-1

Haldar, R., & Mukhopadhyay, D. (2011). *Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach*. *June*. http://arxiv.org/abs/1101.1232

Harrell, C. R., Patton, M., Chen, H., & Samtani, S. (2018). Vulnerability assessment, remediation, and automated reporting: Case studies of higher education institutions. *2018 IEEE International*

*Conference on Intelligence and Security Informatics, ISI 2018*, 148–153. https://doi.org/10.1109/ISI.2018.8587380

*How to keep up-to-date with the NVD data*. (n.d.). https://nvd.nist.gov/vuln/data-feeds

Kassem Saleh, G. E. (2009). Modeling Security Requirements for Trustworthy Systems. In *Encyclopedia of Information Science and Technology, Second Edition* (p. 8). https://doi.org/10.4018/978-1-60566-026-4.ch424

Mburano, B., & Si, W. (2019, February 8). Evaluation of web vulnerability scanners based on OWASP benchmark. *26th International Conference on Systems Engineering, ICSEng 2018 - Proceedings*. https://doi.org/10.1109/ICSENG.2018.8638176

Mell, P., Scarfone, K., & Romanosky, S. (2006). Common vulnerability scoring system. *IEEE Security and Privacy*, *4*(6), 85–89. https://doi.org/10.1109/MSP.2006.145

Mladenović, D. D. (2017). Vulnerability assessment and penetration testing in the military and IHL context. *VOJNOTEHNIČKI GLASNIK / MILITARY TECHNICAL COURIER*, *65*(2), 464–480.

Mulero Chaves, J., & De Cola, T. (2017). Public Warning Applications: Requirements and Examples. *Wireless Public Safety Networks 3: Applications and Uses*, 1–18. https://doi.org/10.1016/B978-1-78548-053-9.50001-9

*MySQL 8.0 Reference Manual*. (2020). https://dev.mysql.com/doc/refman/8.0/en/

NIST. (2012). Guide for Conducting Risk Assessments SP800-30rev1. *NIST Special Publication 800-30 Revision 1*, *September*, 95. https://doi.org/10.6028/NIST.SP.800-30r1

*NVD - CVE-2020-11500 Detail*. (2020). https://nvd.nist.gov/vuln/detail/CVE-2020-11500

*NVD | General Information*. (2019). https://nvd.nist.gov/general

*NVD Data Feeds*. (n.d.). https://nvd.nist.gov/vuln/data-feeds

*OWASP Benchmark*. (2020). https://owasp.org/www-project-benchmark/

Powers, D. (2019). *PHP 7 solutions : dynamic web design made easy*.

Quinn, S. D., Souppaya, M., Cook, M., & Scarfone, K. (2018). *National checklist program for IT products - guidelines for checklist users and developers*. https://doi.org/10.6028/NIST.SP.800-70r4

Ramaki, A. A., & Atani, R. E. (2016). A survey of IT early warning systems: architectures, challenges, and solutions. *Security and Communication Networks*, *9*(17), 4751–4776. https://doi.org/10.1002/sec.1647

*Requirements and Recommendations for CVE Compatibility*. (2017). https://cve.mitre.org/compatible/questionnaires/96.html

Roman Inflianskas. (2019). *Python library for managing installed applications on Windows* (0.1.6). https://pypi.org/project/winapps/

Ruohonen, J. (2019). A look at the time delays in CVSS vulnerability scoring. *Applied Computing and Informatics*, *15*(2), 129–135. https://doi.org/10.1016/j.aci.2017.12.002

Sanguino, L. A. B., & Uetz, R. (2017). *Software Vulnerability Analysis Using CPE and CVE*. http://arxiv.org/abs/1705.05347

Scarfone, K. A., Souppaya, M. P., Cody, A., & Orebaugh, A. D. (2008). *Technical guide to information security testing and assessment.* https://doi.org/10.6028/NIST.SP.800-115

scip ag. (2020). *vulscan - Vulnerability Scanning with Nmap*. https://github.com/scipag/vulscan

*Top 10 Vulnerability Assessment Scanning Tools*. (2018). https://cwatch.comodo.com/blog/website-security/top-10-vulnerability-assessment-scanning-tools/

Waltermire, D., & Fitzgerald-McKay, J. (2018). *Transitioning to the Security Content Automation Protocol (SCAP) Version 2*. https://doi.org/10.6028/NIST.CSWP.09102018

*WMI Tasks*. (n.d.). https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-tasks--computer-software