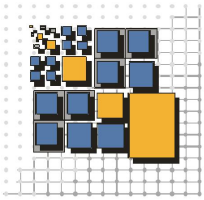


University of Bamberg

Distributed Systems Group

Master Thesis

in the degree programme International Software Systems Science
at the Faculty of Information Systems and Applied Computer Sciences,
University of Bamberg

Topic:

Open-Source Software Discovery and Vulnerability Analysis

Author:

Rajendran, Hari Prashanth

Reviewer:
Prof. Dr. Guido Wirtz

Date of submission:
01.10.2021

Abstract

In this technology era, most software developers and enterprises are bound to use open-source software components to reduce development efforts. Which also says 96% of the applications used in enterprises is built through open-source software[Maa19]. These OSS components are shared with the help of portals like GitHub and GitLab. The main strength of the OSS component, which is highly shareable and reusable, will be the most significant weakness. Due to its rising IT security threats, the security part is still being improved from the beginning of ”shared code culture”. An unsecured OSS component may lead to substantial security threats. Most of the known security issues and threats are registered in vulnerability databases. There are few active vulnerability databases available in the market, and users can search OSS components/software in the vulnerability database.

This thesis builds an OSS vulnerability scanning web application to extract and analyse the OSS component used inside software projects. At the same time, the scanning takes place on the client-side due to the source code privacy of the user's software project. The automation of extracting the OSS components is mandatory because a software project can consist of a lot of OSS packages/libraries. The first step is to create an automated solution to extract the OSS components. The OSS scanner can scan most of the known software projects, and it is also built by considering the scalability of the software projects. The application will analyse the extracted components from the scanner to

find the known vulnerability with the NVD vulnerability database's help and retrieve all the known vulnerabilities of the OSS components found in the project. Finally, the results will be analysed and converted into a simple pdf report to the end-user.

Contents

List of Figures

List of Tables

Listings

Abbreviations

OSS Open-source Software

NVD National Vulnerability Database

CVE Common Vulnerabilities and Exposure

CPE Common Platform Enumeration

CVSS Common Vulnerability Scoring System

CERT/CC Computer Emergency Response Team Coordination Center

JSON JavaScript Object Notation

API Application Programming Interface

XML Extensible Markup Language

DOM Document Object Model

1 Introduction

Open-source software is a kind of computer program or application available to the public and has been developed collaboratively. It stands as free and open-source software (FOSS). Open source development could be seen as a paradigm where the end-user can choose between varying shades of licensing types. It could also be seen as an economically prudent way for project developers to fund their work. The open-source movement was spurred by developers who wanted access to code because they wanted more control over what they created. Apart from the high benefits, the OSS also has many threats that make a drawback, but it can be resolved by taking precautions.

1.1 Motivation

Nowadays, firms and individuals use different forms of computer software running on other platforms, from a simple mobile application to a sophisticated distributed enterprise system. Even though the software is created using different methodologies based on a wide variety of technologies, still each has its advantages and disadvantages [Tur38a][NMED14]. Software security is an essential concern in the software development process because not only to reduce the additional cost but also it can cause severe damage for developers or organizations [Tur38b]. In some recent years, there have been many incidents in which software vulnerability imposed vital damage to companies and individuals. To make a clear idea of this issue, we have mentioned incidents that happened in recent years. A prominent example of an open-source software vulnerability is the Equifax breach in 2017, which exposed 145 million users' data due to outdated open-source software. The outcome of this incident relies openness because the same code has seen by all users, which includes the attackers[**?**]. Software vulnerabilities are imposed by the importance of the threats, which depend on the factors like exploitation complexity and attack-surface[8]. Likewise, there have been a lot of incidents before in companies and individuals have been affected by software vulnerability with significant damage [Tur38a]. In addition, vulnerabilities are common and fundamental. Therefore, choosing software is essential to risk management for information security. Selecting software libraries with solid security features can reduce risk management to individuals or organizations. OSS has been referred to as a potential answer for software security issues and vulnerabilities because most of the open-source software licenses give full access to any part of the code to examine and modify [AUS05].

1.2 Problem Statement

In general open-source software is used massively in modern Software Development, which says around 96% of the application in the enterprise market uses open-source software[Maa19]. Today standard components are not re-written; instead, they are shared as packages around the world. Open Source Portals like Github and Gitlab make it very easy to share those components. The advantage of this "Shared Code Culture" is also the most significant disadvantage because the whole Community and the components are constantly changing. Releases of components are often pushed daily if the project is active. However, if the community switches or the primary Authors leave, those projects depreciate very fast.

Moreover, due to rising IT security threats, a component can become insecure if not patched frequently. Whether an OSS component will become critical also depends on the context where the component will be used. The context will define which actions need to be taken to use an OSS component within the project. We propose to build a scanner to extract all the possible open-source software components and their dependencies from software projects and perform data analysis with the help of CVE and NVD to find the severity of each vulnerable component [AH16]. The first challenge for building this scanner will be scanning for open-source software components, which should happen on the client-side of the system due to the source code privacy of the projects.

Along with the scanning, the system should extract the open-source component meta information from various software projects by using the best text analysis approach. After the extraction of the open-source software components, the next challenge will be assessing the vulnerability and its security issues of each component by choosing a suitable predictive analysis model. The final output of this scanning will be a PDF report which gives precise detail of each open-source software component used inside the project.

2 Background

2.1 Open-source Software

At a fundamental level, it is a software code available for all users to inspect, modify, copy and use in almost any way they choose. The first evolution of open-source software happened in the 1970s by Richard Stallman [Wei11]. Commercial software often called proprietary software, also has additional licensing that further prohibits people from attempting to reverse engineer or modify this process. However, in open-source software, the source code is made available alongside the final executable program. It means any developers or programmers can modify the application to improve or customize it. The students also can study how the programs were written, and the programs can be easily copied and distributed over the internet with the help of cloud repositories [HHR13]. Supporters of open-source believe open collaboration allows the software to evolve via the contribution of many users. Also, they believe people should have the right to use their software in whatever way they want without licensing restrictions. The term open-source has made many impacts in computer science practising, whereas many technologies have generally been built and distributed by a permissive license. The open-source software focuses more on the term ”free”, which helps to build open and transparent software systems. It also helps to improve the project becoming reliable and scalable so that it will help the digital economy to grow [HHR13].

2.2 National Vulnerability Database

The NVD is one of the largest and most effective databases, which is used to report all known vulnerabilities for both commercial and open-source components. It was governed and operated under the US National Institute of Standards and Technology(NIST) from 2005. This organization is sponsored by the Department of Homeland Security's National Cybersecurity and Communications integration centre and Network Security deployment[13]. The information which has been provided by the NVD database helps the developer or security member to help track the software security. The information from NVD helps to analyze the software security vulnerability of an OSS component. It also helps the user know what type of issues are in it and helps the user move further. A section of NVD also provides information on CVE as well as the information source, which is usually from the MITRE corporation [ss18]. Figure 1 will show us how far an OSS component has impacted all these years. This information is gathered by NVD with the help of the CVSS V2 and CVSS V3 data. The NVD also gives us the history of all OSS components with the help of MITRE's CVE dictionary[13]. The security researchers or organization discovers the vulnerabilities of each component, and they will report this vulnerability to CVE. Once the rectified vulnerability is reported to the product owner or open-source project community to resolve it, the report will stay private for 60 to 90 days with CVE before going public. The information from the NVD is beneficial to the consumers who use it so that they can protect themselves from vulnerability. However, at the same time, a hacker can also see the vulnerability of software before the consumer does. Once the threat of an OSS component is received by the CVE, then this information is passed on to the NVD database and will be available in search.

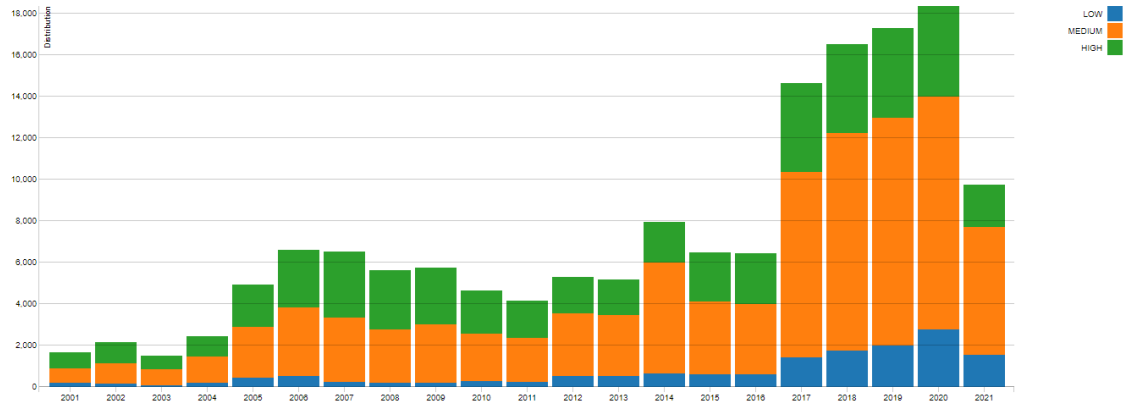


Figure 1: CVSS Severity Distribution Over Time [DAT]

2.3 Common Vulnerabilities & Exposure

The CVE is built for one primary purpose, which is to identify, describe and catalogue cybersecurity vulnerabilities. Each vulnerability will have one CVE record in the catalogue. All vulnerabilities are discovered by the security expert who works for the organization or independent, which is partnered with CVE. The CVE program was found in 1999 first, and it is run by the Department of Homeland Security(DHS) and Cybersecurity and Infrastructure Security Agency(CISA). The CVE and NVD sound like they are the same, but they are two different programs. Whereas the CVE record contains the id number, description, and a public reference of a known vulnerability, NVD is a database that is used to list all the CVE records of each vulnerability. The sponsors of the CVE program decided to leave this information publicly accessible. The CVE can also be used to integrate with security tools and services by an individual or organization. The users have all access to copy the record, reference the information, or analyze it without modifying it. The users can also link a CVE record to another vulnerability under some terms and conditions. The CVE provides a generic identifier for vulnerabilities which helps the user to identify the information to access accurately and efficiently [cve].

2.4 Data Scraping

Data exists everywhere in different formats like web pages to printed materials, and as we established before, there is much value that can be found in the correct data set. Data scraping is a process that extracts data from tables or other structured document formats and converts them into a format that can be quickly processed or analyzed. It is helpful because not all data sources provide ready-to-analyze, pre-formatted data. Data scraping plays a part in unlocking this value. Data scraping refers to retrieving data from one format into a more useful format for further processing or analysis. In some scenarios, the data might be extracted from similar data sets from two different sources. In that case, the data has to be reviewed and processed to make sure they are both formatted equally. When it comes to different types of data sources, there are endless ways to format data. The two of the most significant categories of data sources are digital and physical sources [NJM18].

Data extraction can be categorized into two types: Logical and Physical extraction. Logical extraction will extract less information because it will consume less time and is a lot easier, and it is categorized into two types: Full and Incremental extraction. Physical extraction is the opposite of logical extraction, where it consumes more time, but the outcome of this extraction will give you more information. Physical extraction is categorized into two types: Online and Offline extraction [S18].

2.5 Vulnerability Analysis

Today, businesses increase their dependence on information technology, including the cloud, IoT devices, mobile and social, their cyber risk continues to rise. However, a vulnerability management program can help to identify the weakness before it creates any significant issues. Stats say 95% of all cyber attacks exploit known vulnerabilities, and on an average of 15,000 new vulnerabilities are discovered each year[Gla19]. So constant vigilance is required to evaluate IT security posture, discover weaknesses and respond appropriately. The key to responding to this more dangerous threat environment is a robust vulnerability analysis program. A formal process identifies and quantifies the security weakness, including the application software, hardware and network. A vulnerability analysis should give a clean and transparent report of what the environment needs attention and where it lies on the list of priorities. Identifying vulnerabilities is vital because, unlike the targeted attacks, which dominated the landscape previously. Today's advanced attacks are programmed to search for vulnerabilities in a system and automatically start their attack process. Therefore it is critical to defending even if the organization is not a high priority target. The vulnerabilities can be scored based on the risk, impact and potential exploitation of the weakness.

The vulnerability analysis for open-source software should be imposed because software selection is essential for information security. The quality of software selection should be strong so that we can reduce the risk of information security [AUS05]. Vulnerability is also known as vulnerability assessment, "it is a mechanism that defines, identifies and classifies the security holes" [AUS05]. The vulnerability assessment will also provide deep insight, knowledge and threats about the software to an organization or individual. There are several types of vulnerability analysis: Network-based analysis, Host-based analysis, Wireless network analysis, Application analysis and database analysis.

2.6 Dependency Manager

A good software design concerns the software to be built from smaller, single-purpose modules with well-defined interfaces and, on top of that, keeping with the concept of software reuse. As a result of the broad adoption of open-source software, most of the software built now depends on software built by others. So to monitor all these dependencies used inside a software requires a dependency manager. A dependency is an external software module that can contain one single file or a group of files clubbed together and can be used to perform specific tasks. Software modules, known as dependency managers, coordinate external libraries or packages into bigger applications. All dependency managers use a specific configuration file which consists of dependency name, dependency version and repository of the dependency. The most common dependency manager's configura-

tion files are composer.json, package.json, build.gradle and pom.xml. All the initialised dependencies of the project are fetched from the repository by using their name and version. In some cases, few dependency managers have their repositories like maven central for maven and Gradle projects, npm for npm based projects and packages for composer projects [Mat17]. The dependency manager is required for two main reasons: I, to confirm that both the development and production environment uses the same dependency and its version. ii, to keep the dependency up to date [201].

2.6.1 Components of Dependency Manager

The following are typical components of a dependency management system [S18]:

Module: This component can be used in all sorts of projects, and these can be called packages or libraries based on the programming language. The modules come with the information of what dependencies are required for a specific module.

Manifest file: This file records all the dependencies of the project. It contains all the meta information of the project.

Lock file: This file usually captures all the meta information of the dependencies, converts it into a dependency source tree, and makes the versions immutable.

Repository: In most cases, each dependency has its repository where the dependencies will be fetched directly from the repository to the project. For example, maven and Gradle project dependencies are fetched from Maven Central.

Dependency Constraint: There will be a dependency constraint in the manifest file where it allows only the required version of the project.

Resolution Rule: It is a rule where every dependency manager has to select the correct dependency version to the project.

3 Literature Survey

This chapter provides brief information about the literature surveys specific to Data scraping and vulnerability analysis. All these existing surveys are particular about specific implementations. However, this implementation combines multiple methods to attain a complete automated OSS component extraction and analysis of each component.

3.1 Evolution of Vulnerability Analysis

Vulnerability analysis is an essential operation to perform in all domains, which also includes the software product. One vulnerability can cause a catastrophe in an active system and be an open gateway to the attackers to exploit the system. A good vulnerability assessment should define, identify and categorize the issues in the component[Ros]. Neumann and Parker say that the new vulnerabilities of IT systems are evolved from the attacks that happened in the past by using long-known techniques [PD89]. Therefore building a solid and secure application is merely an impossible task or will be more expensive. Earlier the vulnerability assessment happens manually in all organizations where this is huge and overwhelming work for the IT security teams.

All the newly identified vulnerabilities should be stored in excel or a CSV file for later use. When compared to the no. of vulnerabilities today, the '90s and late 2000 had significantly less number of vulnerabilities [Gla19]. The early vulnerability scanning software gives a simple report of found vulnerabilities in the system. Later, this report has to be given to the IT security team to analyze the possible threats of the vulnerability. Then the report is sent to a higher authority for review and approval. It is such a manual process that has been used earlier to detect vulnerabilities in an active system [Gla19]. Manual scanning and repair strategies would soon become impractical as the number of vulnerabilities grew in the following years and the necessity of vulnerability management became more apparent to companies. Now the future of vulnerability analysis is focusing on fully automated assessment.

3.1.1 Benefits

Lutz Lowis describes that attackers may usually repeat their exploits by reusing a susceptible service[20]. Despite many old and new threats, there are few advantages to performing vulnerability analysis operations. Here are some advantages that can be achieved through a vulnerability assessment [[LA11], [Gla19], [Vul]]:

Identify All Vulnerabilities: A vulnerability analysis can identify all the possible vulnerabilities in the system. It can be identified by both organization and the attacker(hacker) if they use the same software for scanning. The organization has a higher advantage by fixing this issue before the attackers initiate.

Regular Scanning: If the IT security person conducts regular system scanning, the scanning can give the level of risk available in the system. So this helps to figure out the health of the overall system.

Frequent Assessment: Vulnerability assessment will save money and time because if an organization or individual fails to complete the vulnerability analysis procedure, there is a greater possibility that an attacker will be able to exploit the system, resulting in the system having to be rebuilt.

Past Reports: The previous evaluation reports will help improve the present system in the future.

3.1.2 Vulnerability Attacks

A vulnerability is an attribute of a software component that can exploit or damage an active system. Most of the vulnerabilities have a high tendency of causing damage to security policy by internal or external persons(hacker or insider). There is a past event that happened in 2017 where a significant data breach occurred in Equifax. This data breach caused more than 100 million user data to be leaked [Maa19]. In this vast area of IT, there are still new and unknown vulnerabilities emerging every day.

Ryohei Koizumi and Ryoichi Sasaki have found that whenever a vulnerability assessment operation takes place, the IT security team should always use the latest scanning tool because sometimes the old software is effective against some minor virus threat. However, it does not exploit the vulnerability [KS15]. Some vulnerability attacks have to be focused more on because these types of attacks are listed as high in risk level. Here are some major attack types that are focused on by few researchers [AKAA15]:

Configuration-based: This type of vulnerability occurs when a misconfiguration in the system or running any unwanted services in the background. The configuration-based vulnerability has a central weak point. The hackers can quickly get into the organization network and find the active system by any form of misconfiguration. This type of vulnerability is based on weak management protocols, weak permissions and weak encryption.

Security patches: A security patch is an essential update for all active systems because the role of security patches is to fix or remove the flaw or issue found from the vulnerability report. The software patches play a vital role in rectifying and fixing the components for commercial and open-source software components. Generally, the security patches will be released every month; for example, Microsoft sends security patches to its Windows operating system every month. A system that fails to update its security patches will lead to significant security breaches. As everyone knows, the best example of security patches failures is ransomware which is called wannacry [Kas17].

Zero-Day Vulnerability: It is the most challenging vulnerability to rectify and fix the issue. It is because the vulnerability is new and unknown to the organization. The attackers will take advantage of this loophole and will cause a significant security risk. The term "Zero-day" is used because the vendor was unaware of the threat which affected the software, and the vendors had "0" days worked on the security patch or fixing the vulnerability.

Faulty Open-Source Package: It is the vulnerability where hackers have been using for several years. Generally, the hackers used to inject a credential sniffer inside an instrumental shared library or package. This kind of attack happens when the vendor does not update the open-source packages regularly. This threat will stay in the package for several

days or until the vendor finds it.

3.1.3 Types of Vulnerability Analysis

The vulnerability analysis is also called vulnerability assessment, where the primary purpose is to keep the organization safe from digital threats. It is a methodology that is used to find the IT application and the infrastructure. It also involves intense scanning by the security expert or team of the organization. Few types of vulnerability analysis are used exclusively for some parts of IT:

Network-based Analysis: A network-based analysis is a mechanism to identify the network defects and issues in the network. They mostly scan and analyze the network endpoint and device network for security issues. The failure of vulnerability analysis will allow the hackers to take advantage of the network issue. The organization will invest more time to improve its existing framework, which is used for network vulnerability analyses. In 2008 Hai L Vu etl developed a vulnerability analysis framework for scanning network vulnerabilities, and along with that, they have also proposed a scalable algorithm. Both framework and algorithm are used to evaluate the network vulnerabilities without generating a full-scale graph [VKCK08]. The table 1 gives the results of each network component used in the network by using the framework. The results come with a brief description of the effects of the listed vulnerability. This information about the vulnerabilities have been taken with the help of vulnerability databases [VKCK08].

Network Components	Nessus Vulnerability Output	Vulnerability
SIP Proxy(H_1)	**CVE Reference: CVE-2005-0449** **Description:** Allows attacker to bypass net filer/iptables or initiate a denial of service attack. CVSS Base Score: 5.0 Pre-Condition: Linux, Kernel 2.6.x Post-Condition: Denial of Service, bypassing firewall	V (source, H_1, 0.50, Linux-kernel-2.6.1, bypassing_firewall)
	CVE Reference: CVE-2008-1483 **Description:** Attackers are able to hijack a SSH Session CVSS Base Score: 6.2 Pre-Con OpenSSH-4.3p2 and earlier Post-Condition: Allows remote connection, Gain access to system.	V (source, H_1, 0.62, OpenSSH-4.3p2, hijack_session)
Asterisk Server (H_2)	**CVE Reference: CVE-2008-1483** **Description:** Attackers are able to hijack a SSH Session CVSS Base Score: 6.2 Pre-Condition: OpenSSH-4.3p2 and earlier Post-Condition: Allows remote connection, Gain access to system.	V (source, H_2, 0.62, OpenSSH-4.3p2, hijack_session)
	CVE Reference: CVE-2008-0095 **Description:** Allows remote attackers to cause Denail of Service to Asterisk via a BYE message with an Also header. CVSS Base Score: 5.0 Pre-Condition: Asterisk, open source, 1.4.0 and previous. Post-Condition: Denial of Asterisk/VoIP services.	V (source, H_2, 0.50, Asterisk 1.2.1, DoS)
	CVE Reference: CVE-2007-1306 **Description:** Allows remote attackers to cause Denial of Service to Asterisk using a SIP packet without a SIP-version header. CVSS Base Score: 7.8 Pre-Condition: Asterisk, open source, 1.4.16 and previous. Post-Condition: Denial of Asterisk/VoIP services.	V (source, H_2, 0.78, Asterisk 1.2.1, DoS)

Table 1: Sample results of the network vulnerability analysis framework [VKCK08].

Host-based Analysis: Sometimes a vulnerability can be found in the vendor's resources, and with this vulnerability, there are high chances where even an insider can be an attacker for the system. The attackers mainly cause damage by making an improper configuration setting in the host. This vulnerability assessment takes place in servers, workstations or other network hosts. The host-based analysis will give a detailed insight into configuration settings in the network, patches and update history. The insight gathered from the analysis will also give us the potential damage caused by the attackers or intruders. Anil Sharma et al. created a software tool, "Ferret", by using Perl language that identifies the vulnerabilities present in the host [CAS+04]. This software tool helps the system administrator to identify the vulnerabilities and take action based on the threat. The host vulnerability is checked using a different plug-in module, and the end output will also mention which plug-in module is used for the assessment.

Database Analysis: Misconfigurations often occur in databases and Big Data systems. Database vulnerability analysis is mainly used for identifying the known risk in the databases. The most common risks are missing patches, weak passwords and default vendor accounts [Imp]. Sartaj Singh described the importance of inherent dangers of the database, like how data theft is happening in the internet era, and he also mentioned that the existing encryption methods are not foolproof for the high-end professionals[31]. A vulnerability attack can exploit file permission and database configuration files and can potentially steal sensitive information like credit card details, personal details, etc. There is still research going to secure the database more effectively and efficiently. In 2008 Ghassan Jabbour and Daniel A. Menasce presented a framework that provides self-protection to the database from unauthorized or intensional security parameter changes. Also, they proposed that this framework can be implemented in an Oracle 10g Release 2 database [JM08].

Application Analysis: Vulnerabilities are frequently identified in third-party apps that are built and managed. The vulnerability assessment is essential to an organization's security team because they have to identify the vulnerability in the application before it exploits the system. This process is used to identify misconfiguration vulnerabilities in applications, outdated software packages and weak authentication. Sultan S. Alqahtani [Alq17] has researched a modelling approach that improves traceability and trust in software products by linking the security knowledge with the software artefacts. He also introduced a scanner called Semantic Global Problem Scanner(SE-GPS), which is created by integrating the modelling approach. With the modelling approach, the tool can now link the NVD [nis] security database to the maven build repository.

The application vulnerability analysis is a crucial security process to be considered by the organization. The process's primary goal is to identify the vulnerability and report it to the security authority to mitigate it before the vulnerability exploits the system. Most application vulnerability analysis tools use the proper guidelines to make a good scanning tool. Here is the main guideline that a good vulnerability scanner tool should use [Sre]:

Setup: The setup should begin with proper documentation about the application, with perfect security permissions and configuration tools.

Test Execution: Run the scanner tool, which can be an existing tool or a self-created one. The scanner should identify all the software packages and their dependencies used inside the software product.

Vulnerability Analysis: Once the execution is finished, the extracted software packages and their dependencies should be analysed by using any existing vulnerability databases like NVD, ISS(Internet Security Systems), etc. Each software package is searched in the database by using its name and version.

Reporting & Remediation: After the previous process, the analysis will give a formal report of each software package. The report includes the risk level of each package which is categorized as LOW, MEDIUM and HIGH. Sometimes the report will also provide a brief description of the threat. The security department of the organization takes the remediation action. Mostly, the remediation will have two possibilities: to change the version of the package or find an alternate software package.

3.1.4 Vulnerability Databases

A vulnerability database is a collection of information about all known defects of a software application. This database is aimed to maintain all the information like name, version, threat description, level of risk and history of the components. New vulnerability information is generated every day to be fed into the database to be a known vulnerability for future users. The respective vulnerability database adequately reviews each information before being generated into the database [GS17]. The software owner, security researcher, and public users of the software can give the vulnerability information. Most of the databases use CVE's information to be integrated with the vulnerability database[14]. The Common Vulnerabilities and Exposures(CVE) is a program that the MITRE's corporation owns. The CVE is built for one primary purpose, which is to identify, describe and catalog cybersecurity vulnerabilities.

There are a lot of public and private vulnerability databases which both government and private organizations maintain. Each database is selected based on the user's choice, where some like to go with public or proprietary databases. In most cases, the user always prefers to use the public database for two main reasons, it is maintained by the government and its open-source. Here is a list of significant vulnerability databases [LA11]:

NVD: Among all the databases National Vulnerability Database is the most commonly used database by users. The NVD was first found in 2005 by the US National Institute of Standards and Technology(NIST) [nis]. Most of the known vulnerabilities for both commercial and open-source are fed into the NVD databases, and that is why **i!**s one of the largest and most efficient vulnerability databases. NVD is a completely open-source project so that anybody can use it without any restrictions. The NVD integrates CVE's information for every component present inside the database [cve]. The CVE is mapped in the database by using the CVE id so that the user can easily retrieve the information from the CVE dictionary if needed. All the data stored in the database have a unique id for each vulnerability, history date, and short vulnerability description. The NVD also integrates CVSS(Common Vulnerability Scoring System) information to each vulnerability.

The figure 2 will give a clear idea about how components can be searched in the NVD database. The result list will be provided based on the keyword entered in the search bar along with information like CVE id, a summary of the vulnerability and CVSS score.

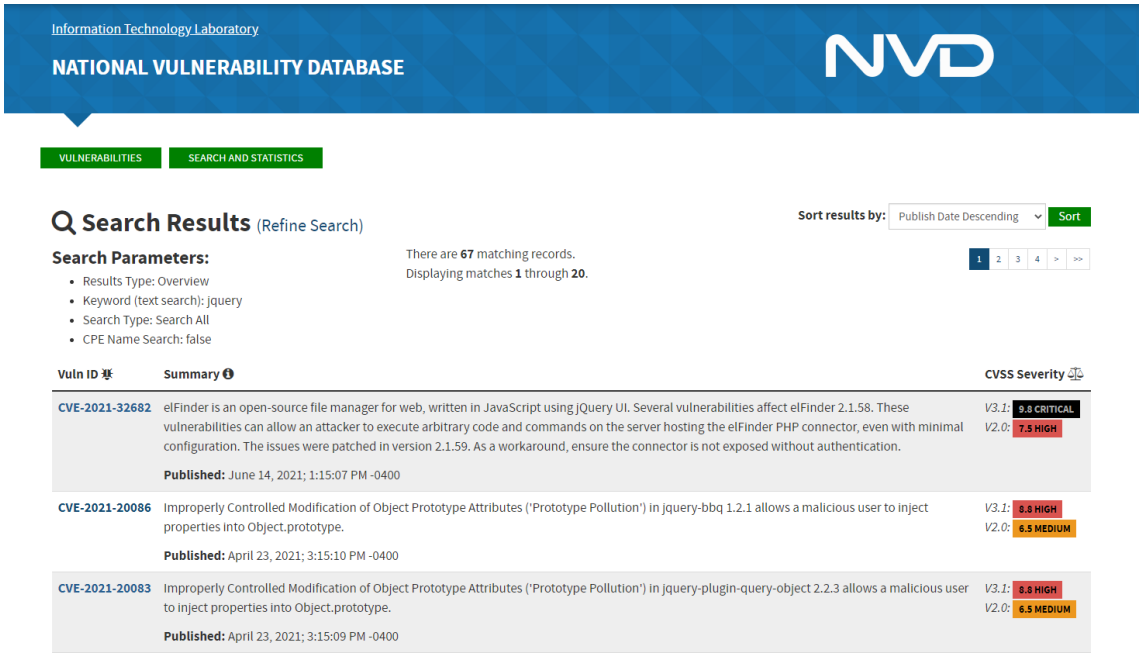


Figure 2: NVD result for "jquery" component [nis].

SecurityFocus: The SecurityFocus is a traditional news portal where the community discusses the new security issues of the vulnerabilities. The Bugtraq is the most common product of the SecurityFocus. The Bugtraq tool is exclusively used to list all security issues, like new vulnerabilities, exploitation methods and security related announcements by the Vendor. Bug traq was created in 1993 by Scott Chasin. On Tuesday, May 14th, 1996, Aleph One gained control of BugTraq. BugTraq has evolved into a well-respected security mailing list with over 27,000 subscribers over the years [Bug].

Ashish Arora [AKTY06] made an empirical analysis on Vendor's patching behaviour by evaluating the gap between the date of vulnerability disclosed and the date of security patch rollout. They have discovered that a vendor takes an average of 29 days to fix the issue from the date of vulnerability is disclosed. To achieve this, the authors have compiled data from SecurityFocus and CERT/CC. The author also confirmed that the vendors take more time to fix the vulnerabilities they get from CERT/CC. It means the authors say that the Vendor responds faster to SecurityFocus vulnerabilities than the CERT/CC. The results of SecurityFocus will give information like Bugtraq ID, and it is class of error, publish date and its CVE ID. It also uses CVE information to list all vulnerabilities and give a discussion forum to report any other errors after the security patches. Figure 3 is an example of the vulnerability publication of SecurityFocus.

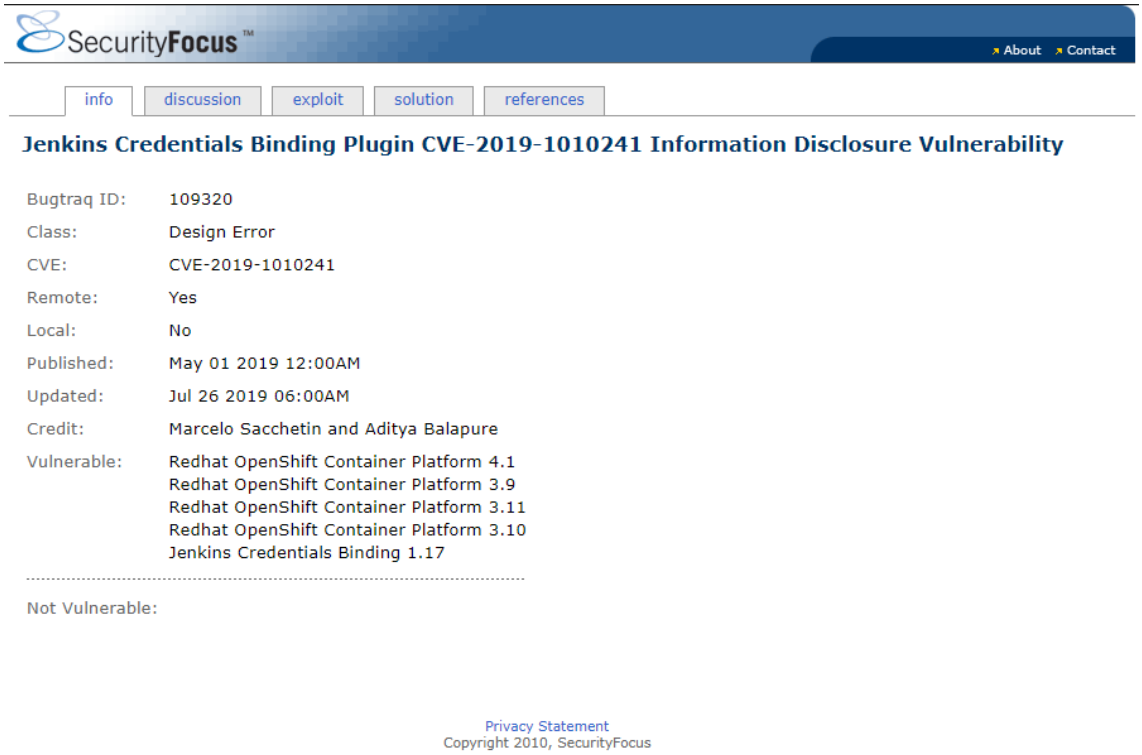


Figure 3: Example of vulnerability publication by SecurityFocus [Sec].

IBM X-Force: The IBM X-Force is an elite team of security experts that contains hackers, and incident responders [Ibm]. They are dedicated to resolving some of the toughest cybersecurity challenges in the world. The IBM X-Force is led by former FBI Cybercrimes Division Assistant Director Chris Soghoian, who brings over 20 years of experience to this role. Unlike NVD, the X-force is proprietary software, whereas NV is completely open-source. Despite being proprietary software, the X-force gets its vulnerabilities information from the CVE dictionary. Apart from this, X-force is also providing API services for this software. X-force requires an IBM id to get full access, which will be a paid service later. X-force also uses CVSS scoring system to give a clear risk level of the vulnerability. Though it is a proprietary database, not many researchers will use this database for research purposes. Figure 4 is an example of vulnerability publication by IBM X-Force [Ibm].

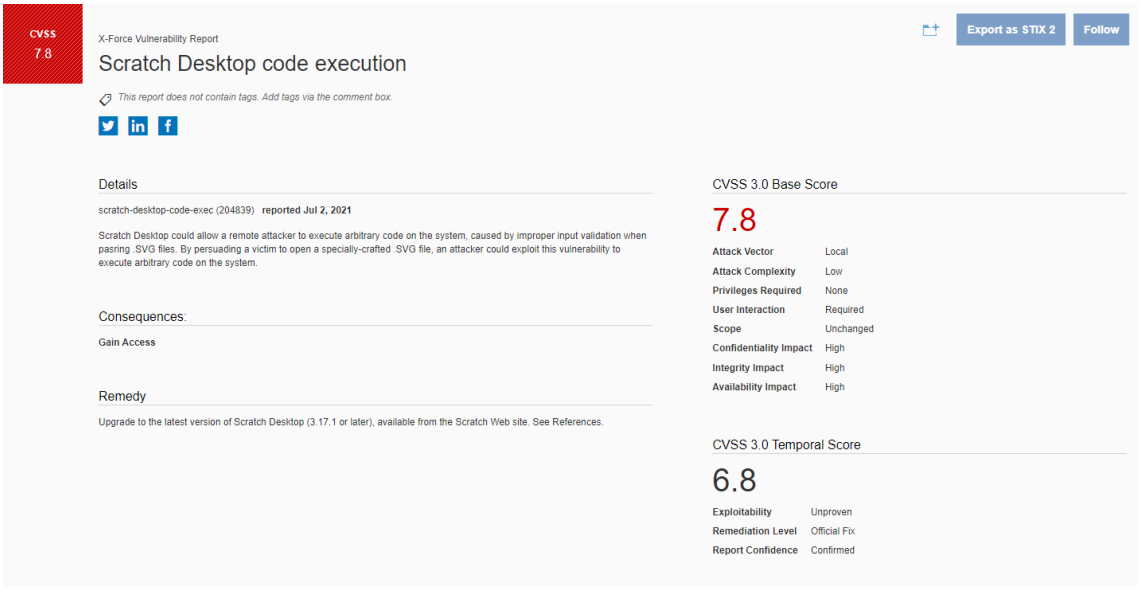


Figure 4: Example of vulnerability publication by IBM X-Force [Ibm].

CERT/CC: The CERT/CC stands for Computer Emergency Response Team Coordination Center [Web21]. CERT/CC is also one of the databases which provide information about ongoing vulnerabilities, and it also tries to resolve incidents like data breaches and denial-of-service attacks. CERT/CC was founded in 1988 by Carnegie Mellon University in Pittsburgh, Pennsylvania and supported by the Defense Advanced Research Projects Agency, which was part of the U.S. Department of Defense [Ins]. The main characteristic of CERT/CC is to resolve the security incident and try to regain control and minimize the damage. Later on, they will assist in reporting the incident response to prevent the issue from happening again. Figure 5 is an example of vulnerability publication by CERT/CC.

Figure 5: Example of vulnerability publication by CERT/CC [Ins].

In general, the main goal of the incident response team is to protect the organization from any vulnerability, which can be software, network or cybersecurity-related issues. The CERT/CC follows a universal incident response model, which is "protect, detect and respond" [DWG04].

Protect: The first step of this model is to protect the organization by taking some security measures before the incident happens. This model focuses on more protection than reacting to it, such as creating an incident response plan, providing security awareness, performing risk analysis, etc.

Detect: It is a process that happens before responding to the vulnerabilities. A proper response can be given to solve the vulnerability if the detection of the vulnerability is proper. Sometimes the detection duration might take a month, week or a day, and it entirely depends on the security incidents. Before performing a detection strategy, the responsible person should provide solutions, for example, applications that often run, how regular network traffic looks, what the network protocol should be avoided, etc. The standard technique of detection is routers, firewalls, network monitors, etc.

Respond: Once the security incident is detected, the final step will be to provide the perfect solution. Generally, responding to the security incident has a few steps. The first step will be getting the information about the security incident from the vendor or business partner. The next step is for the team to analyse the information to find quick and practical solutions. Once the solution is found, the team will create an immediate

strategy to stop the issue before it causes more damage. The last step will be responding to the incident and will be published to others so that the affected users can regain control.

3.2 Data Scraping

Data scraping is an automation process used to extract data from files, websites, databases or any application. With the help of data scraping, a user can get a considerable amount of relevant data such as product reviews, business contact information, social media post or specific content from a file. The content mentioned above can be extracted using existing data scraping tools or build a custom program with the help of programming language support. Sometimes data scraping is also called web scraping just because it is widely used in the web domain. After the birth of the world wide web in 1989, businesses started showing interest in creating websites to show their business-related information such as product details, upcoming events and contact forms. Since then, the evolution of data scraping has been tremendous, whereas now, the data scraping techniques are used in cloud-based applications. Ram Sharan Chaulagain et al. proposed a cloud-based web scraper architecture that can handle storage and computing resources with elasticity by using Amazon web service [SB16]. This architecture was proposed concerning the previous drawbacks of scraping large amounts of data, such as reliability of data scraping, storage issue of extensive data and intensive computation. So, therefore, this clearly explains that large amounts of data cannot be extracted by using the traditional data scraping methods.

3.2.1 Types of Data Scraping

Generally, data scraping techniques are used in the area of the web or any enterprise application. Depending upon the data extraction requirements of the system, the data scraping is categorised into the following types:

Web Scraping: Web scraping is also called “web harvesting”, “web data extraction”, or even “web data mining”. Web scraping is an automatic process of getting the web data from a web page and parsing the data to get the required information to organize a separate database, and this is called web scraping. The main goal of web scraping is to lower the need for human involvement in downloading web pages, manually organizing the web data to a database or spreadsheet by using copy-paste technique. The automated process of web scraping is much efficient and cost-effective than manual web scraping. Apart from this, automated web scraping can be configured to have higher accuracy for data extraction than human accuracy. Web scraping came into the picture when the web was invented. Figure 6 shows the workflow of a standard web scraper. The Scrapehub is the part where it takes a URL as an input, and the given input is processed based on the client’s configuration. The scrape engine uses different libraries or self-made programs for parsing the web data and converts it into meaningful information to organize in the database [SB16].

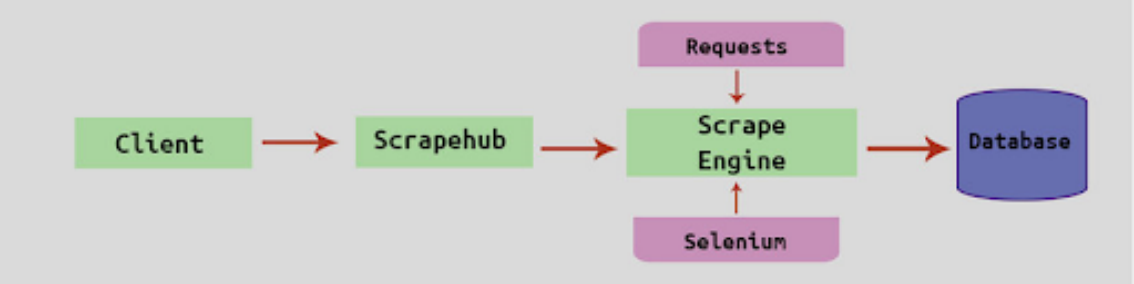


Figure 6: Traditional Web Scraper [SB16].

The process of web scraping is a combination of two operations. First, the web data extraction and next will be processing or cleaning the extracted raw data to provide insightful information. The extraction part is more accessible when the source data is in a database or an ontological structure. However, in most cases, all web data extraction deals with unstructured or semi-structured data. The researchers are coming up with new methods to improve the data scraping more efficiently and fast. In 2021 Jaebeom You [YLK21] proposed a method called "DeepScrap" for collecting tweets. The DeepScrap method is used to scrape all the recent tweets quickly by using Twitter's standard API. The multiprocessing of deepScrap helps to get refined information rather than going with single processing. This scraping technique can help the OSS scanner to extract the name and version of a component.

Screen Scraping: In short, screen scraping means reading text data from a computer terminal screen or a piece of programming that mediates legacy applications, and modern user interfaces [Gil]. Both web scraping and screen scraping have many similarities, except a few key differences, such as where the data is gathered and how it is used. This scraper technique is mainly used for scanning old sources because of the quick speed of technological change. Specific legacy systems, software, and applications become outdated and expensive to maintain. It would be a big and complicated process when the organization decides to migrate the old resource to the latest one without the help of a screen scraper. In 2017 there was a report where SnapLogic and the independent research firm Vanson Bourne stated that more than 500 U.S IT companies data are completely trapped in their legacy system [Sna]. A $140 million loss was incurred as a result of the data trap. Not every company needs to use screen scraping. It is more needed for companies like companies that hold their client's data for very long time record-keeping purposes like companies that produce CRM services, crypto or stock exchanges, etc. Even without the source code of any legacy application, the screen scraper can still extract the data.

Sergio Flores-Ruiz et al. have explained black-box solution for the migration process of a mainframe by using screen scraping technique [FRPCDP18]. The author came up with a back-box solution to JavaFX and relational database mainframe systems because most of these legacy systems are consolidated on mainframes. The author also mentioned that the previous migrating solution was inefficient due to a shortage of systematicity and lack of business rule verification. As we know, applying the screen scraper technique to retrieve legacy information from an old source is a compelling idea, but some research has tried to take the screen scraping next step. An operation of screen scraper is, once the information is retrieved from the legacy information system and moved to the new data storage system. After this operation, the legacy information system will be deactivated by the organization. Alex van Oostenrijk [Oos04] has tried to implement a web

service between a website and the legacy information system by using the screen scraper technique. He believed that a continuous screen scraping system would be essential to achieve it. Continuous screen scraping is nothing but keeping the old information system active and connecting the new system(website) with the help of web services. After the implementation, the author experienced that this is not a robust idea because it took four minutes to search 132 pages per request, but it can be overcome by using a caching technique.

3.2.2 Challenges

The usage of data scraping is expanding every day, and along with that, the challenges also travels with it. Data scraping helps extract information based on the requirements when consumed, but creating a data scraper is not an easy task. There are few challenges that have to be taken into consideration before creating a data scraper, and the listed challenges are considered for only data scraping [[Dem21],[Joh20]]:

Complicated Data Structure: It means the structure of the data source can be varied from one another. This challenge will occur when a system is trying to extract data from multiple data sources. This can be resolved if the developer designs a scraper effectively by focusing on the target data source.

Scraping Time: The challenge will be faced when the system is trying to extract significant data sources frequently. This may cause system overloading, and the scraping will be not effective. The system should maintain a proper balance in scraping time to overcome this issue, like avoid frequent data extraction and perform an extraction with time intervals.

Data Quality: It is a significant challenge to focus on because data scraping is all about extracting meaningful information. The quality of data is evaluated by cross verifying with the predefined requirements. This challenge can be easily solved by making sure that all the predefined requirements match the validation process.

Legal Challenges: This kind of challenge affects only web data scraping because most corporate or private websites will not accept any data scraping bot to extract their information without their knowledge. It does not mean that data scraping is illegal, but it may cause problems if it is not done with proper permission from the respected owners of the data source or website.

Large-scale Extraction: One of the significant challenges in data scraping is that academics are currently working to improve the technology for large-scale data extraction. This issue leads to a breakdown if the data scraping is continuously being performed. In most cases, the data scraping techniques are used for minor operations.

3.2.3 Data Scraping Methods

Mostly data scraping used purposes like marketing and price research, monitoring, analyzing and retrieving information for decision making and CRM tools. If a user or organization configures the data scraping technique effectively, the tool will be pretty powerful for retrieving meaningful information. The data scraping has been classified into

two scraping techniques which will be utilized based on the requirements:

Manual Scraping: As everyone knows, data scraping is done chiefly automated but can also be manually scraped from a resource. It is the traditional way of data scraping. This process is nothing but seeing through the vision and replicating it into the desired format like a spreadsheet. The advantage of manual scraping is that it is straightforward to implement, with no additional skills and human intervention. At the same time, manual scraping also has its drawbacks. Compared to automated scraping, it is slow, high cost, and errors may occur with human intervention. Under manual scraping, there is only one technique available [Rad]:

Copy-Pasting: In Manual scripting, all the user wants to know is copy-pasting, which takes much effort. Most of the time, manual scripting will have data reputation issues because of human involvement. On the other hand, manual scraping is seldom seen in reality because automated scraping is considerably faster and less expensive.

Automated Scraping: The evolution of manual scraping is automated scraping, and the current IT era is moving through automation to reduce human effort and make it cost-effective with high accuracy. Because of their simplicity of use and time and cost advantages, automated web scraping technologies have grown in popularity. Automated scraping has more advantages than manual scripting like very fast data scraping and extraction, time and cost efficiency, easy to use and supports API services. Eventually, it also has its negative side, which is not that bad, like requiring light training; in some cases, scraping is illegal and lacks human checks. Automated scraping has many methods to scrape data based on the requirement [Rad]:

HTML Parsing: It is the easiest and fastest way of extracting data from a file. This technique is mainly used for extracting data from HTML files. The parsing is done with the help of a programming language like JavaScript to extract data from the HTML file.

DOM Parsing: The document model object is an interface used to parse XML or HTML file source code into a string. The primary purpose of the DOM parser is to get the in-depth structure of the HTML or XML file. The representation of a document is shown in the tree view. With the help of a DOM parser, the scraper can extract the information from the node data. The positive side of the DOM parser is more effective than standard parsing methods like the data persisting in the memory, forward and backward traversing is possible, and immediate changes are possible.

XPath: XPath stands for XML path, and it is an exclusive querying language for XML files. The xpath is used to navigate the XML file because the XML documents are based on the tree structure. There are few reasons why XPath is preferable when parsing XML files, like the queries are compact, easy to use, simple syntax, do not return repeated values and work with both HTML and XML attributes. Due to its interoperability, the XPath can be used in any programming language like C, C++, C-Sharp, Java, Javascript, etc. Xpath also can retrieve the relevant information from any complicated XML file by allowing different types of expressions. The essential expressions are Root, Element, Attribute, Text and Comment.

Text Pattern Matching: Pattern matching is a powerful tool for extracting information from natural language. It can be used to perform tasks from simple text segmentation to complex parsing and machine translation. In technical terms, it is verifying the given

sequence of characters to find similarities with a pattern. The patterns to verify the given string are designed or created with the help of regular expression. Most programming languages support regular expression. Unlike other methods, pattern matching can be implied on all data types. Pattern matching can be performed simultaneously with the help of parallel pattern matching.

3.3 String Distance Metrics

Both supervised, and unsupervised learning can benefit from distance measures. Distance measurements can be used for text mining, medical analysis, document categorization, and similarity analysis, among other things. The primary purpose of string metrics is to correct a spelling mistake by calculating the distance between two data points or strings; the calculation will give the string similarity between the given inputs. In simpler terms, it calculates the distance between one text to another text to find similarities. The string metrics are frequently used as information integration in areas like fraudulent detection, fingerprint analysis, plagiarism detection, ontology merging, DNA analysis, RNA analysis, image analysis, etc. The distance between strings may be calculated using a variety of algorithms. Each algorithm calculates the distance between strings, but apart from this, it also has some unique features [Wu20]. The following algorithms are:

3.3.1 Hamming Distance

Hamming distance is a simpler algorithm to calculate the distance between two strings found by Richard Hamming, who introduced this technique for Hamming codes. The algorithm performs calculations to find the similarity in the given strings by comparing the changes in the position of the two strings [Tok15]. The equation of hamming distance will look like Equation 1, and the 2 illustrates the distance value of the words [Say].

$$\begin{aligned} D_H &= \sum_{i-1}^{k} |x_i - y_i| \\ x = y &\Rightarrow D = 0 \\ x \neq y &\Rightarrow D = 1 \end{aligned} \tag{1}$$

X	Y	Distance
Male	Male	0
Male	female	1

Table 2: Distance between the word ”Male” and ”Female”.

Equation 1 explains how the distance is calculated between the two strings to find the similarity. The benefits of using hamming distance are simpler and effective to detect spelling mistakes or errors. They are also quite good and effective in data streams. The only disadvantage of using Hamming distance is that the bandwidth usage is more.

3.3.2 Levenshtein distance

Levenshtein distance algorithm is also used to find the distance between two strings and a few operations. Vladimir Levenshtein from Russia discovered the algorithm. The Levenshtein distance is also known as the Edit-distance based algorithm because it computes the number of edits to find the distance. The edit contains three operations which are Insertion, Deletion and Substitution. To find less similarity between two strings, it requires more operation [ZS19]. For example, the distance between GILY and GEELY is two, and two operations have been done: substitution and deletion. The equation of Levenshtein distance looks like Equation 2 [Cue17].

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } mini(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i-1-j)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & otherwise. \end{cases} \tag{2}$$

The algorithm can be used for word suggestion and autocorrection by measuring how different two strings are by counting the number of character edits. Implementing the Levenshtein distance algorithm as a recursive implementation will cause massive complexity, but this can be solved by using a proper memorization technique.

3.3.3 Damerau-Levenshtein distance

The Damerau-Levenshtein distance algorithm is an updated version of the Levenshtein distance algorithm. The one difference which separates Damerau-Levenshtein from Levenshtein is that it includes another operation: transposition. It takes only a minimum number of operations to find the similarity of the strings by changing one word to another. The name Damerau-Levenshtein was derived from the scientists Frederick J. Damerau and Vladimir I. Levenshtein. The main benefit of using the Damerau-Levenshtein algorithm is that it is comparatively faster than its predecessor algorithm, and it also saves time by avoiding Regex expression to calculate the similarity [ZS19]. The equation of Damerau-Levenshtein looks like Equation 3 [Wika].

$$d_{a,b}(i,j) = \begin{cases} 0 & \text{if } i = 0 j = 0 \\ d_{a,b}(i-1,j)+1 & \text{if } i > 0 \\ d_{a,b}(i,j-1)+1 & \text{if } j > 0 \\ d_{a,b}(i-1,j-1)+1_{a_i \neq b_j} & \text{if } i,j > 0 \\ d_{a,b}(i-2,j-2)+1_{a_i \neq b_j} & \text{if } i,j > 1 and a[i] = b[j-1] and a[i-1] = b[j] \end{cases} \tag{3}$$

In natural language processing, the Damerau–Levenshtein distance is a crucial factor. They are mainly considered in the area of DNA analysis and Fraudulent detection.

3.3.4 Jaro Distance

Jaro distance is a metric for comparing the similarity of two strings, and it is defined using Equation 4 [Wikb].

$$d_{a,b}(i,j) = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{s_1} + \frac{m}{s_2} + \frac{m-t}{m}\right) & otherwise \end{cases} \tag{4}$$

Here s1 and s2 are the two strings. m is considered the number of matching characters, and t is considered half the number of matching characters. Unlike the other string distance algorithm, the Jaro distance ranges from 0 to 1, where 0 means no similarity and 1 has similarity [Kun21]. The result of the algorithm will be given as a float value. It can be used to check whether the two strings are the same or not. The improved version of the Jaro similarity algorithm is the Jaro-Winkler distance algorithm. The Jaro-Winkler algorithm is also similar to the Jaro similarity. However, the Winkler is built using prefix scale p for finding precise distance, and the equation looks like Equation 5 [End19].

$$Sim_{JW}(s_1, s_2) = Sim_J(s_1, s_2) + lp[1 - Sim_J(s_1.s_2)] \tag{5}$$

4 Design & Architecture

This chapter will explain the scanner architecture, things that we have considered while designing the component and some of the valuable techniques I have analyzed to make an efficient scanner to extract OSS component.

4.1 Scanner Architecture

The first task of developing this scanner is creating an automated OSS component extractor because manual extraction of OSS component is a tedious and redundant task; finding an automated system will reduce the time consumption and make it more cost-effective. The manual OSS extraction is suitable for small projects with less OSS components consumption which is a very rare case. The OSS scanner is built as a web application in this project by considering some useful advantages like accessibility across the devices, less maintenance and increased flexibility and scalability. Like traditional web applications, the OSS scanner has front-end and back-end applications. We have decided to break down the OSS scanner into three major parts, which are OSS component Analyzer, Evaluator and Reporter.

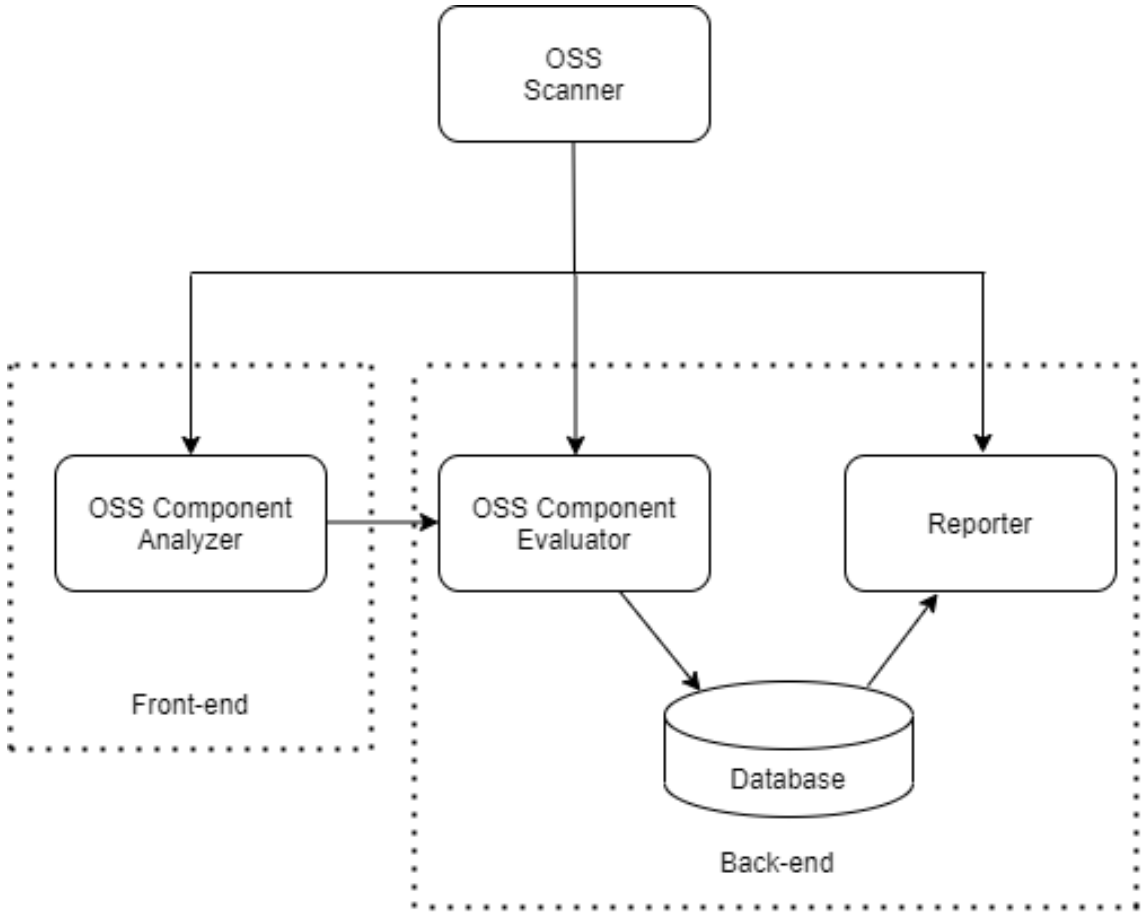


Figure 7: OSS scanner architecture

The figure 7 shows where the three major parts have been placed in the web -application. The OSS component analyzer will be developed in the front-end because, as per the research question, the OSS components should be scanned in the front-end(web browser). Once the OSS components are extracted from the project, the meta-information(name and version) will be sent to the OSS component evaluator for finding the vulnerabilities. The OSS component evaluator and reporter are developed in the back-end. To make a clear understating, the figure 8 will show clear interaction between the components.

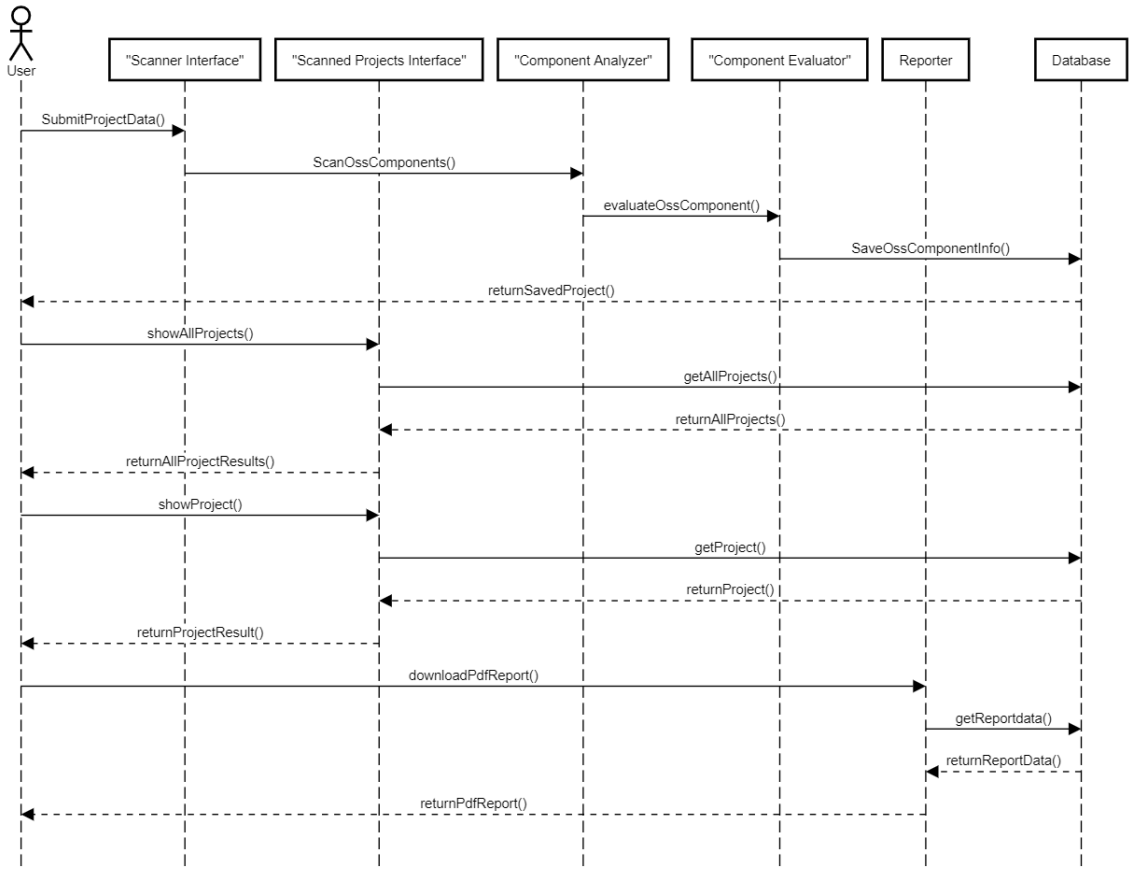


Figure 8: OSS scanner sequence diagram

4.2 Scanner Components

As shown in figure 7 there are three main components involved in the design and architecture of the OSS scanner. Following are the components which are interrelated to each other and gives a big picture of their functionality. OSS Component Analyzer, OSS Component Evaluator and OSS Reporter.

OSS Component Analyzer: This component is mainly responsible for extracting the OSS components used in the projects. This particular task should be performed on the client-side, where the scanning should take place in the browser. The user's inputs required in this interface are Project name, description, members, and project directory. After receiving the inputs, there are two stages of the automation process. First, the scanner will detect what type of application framework is given as input so that the scanner can parse the targeted file for the further process. Once the target file is parsed, the second

process will extract the OSS component name and the version used in the project. It creates JSON data with basic project information along with the list of OSS components and its version. Overall the primary goal of this component is to extract all the OSS components, and It will send the data evaluation process.

OSS Component Evaluator: This component evaluates the OSS components which are extracted by the analyzer. The evaluation process component will be developed as a back-end application to perform the process on the server-side. This evaluation is performed with the help of NVD database API service. The extracted OSS components will be searched in the NVD database by using its name and version to find the known vulnerabilities registered under the respective component version. This component will be running asynchronously in the server because a project can have n-number of OSS components. Once the OSS components are evaluated, the information which is collected from the NVD database is used for analyzing the risk level of each OSS component. Finally, all the information is converted as a JSON file and stored as a container in Azure data storage.

OSS Reporter: This component is a part of the scanner's user interface where the user can download the OSS report of the project. This report shows all the basic information of the project and its OSS component along with the vulnerability and the risk level of each component. This component simply generates a pdf report with all the above information retrieved from the database.

4.3 Design Challenge

Almost every software product development project has a unique set of problems. These issues can be a roadblock to development throughout system design and, more importantly, during implementation. There were several difficulties faced throughout the implementation of this thesis. The major problems that need to be addressed and a solution formed are listed below.

Application framework: Because I had never worked with Angular framework before, the first hurdle I had while starting implementation was familiarizing myself with it. I got acquainted to the Angular environment with the aid of several online tutorials and hands-on programming. The are few reasons why we chose to develop frontend application with Angular framework because it has faster development process like having neat documentation for understanding Angular with a large developer community and it also helps us with efficient problem-solving patterns where the angular services helps to integrate the business logic with app user-interface.

File structure: The next challenge for us was trying to create a generic function that can extract the OSS component name version from respected config files of the software project. This challenge was time-consuming because each software project can be built using a different application framework, and also each application framework has its dependency manager. For instance, the Ruby on Rails application framework uses RubyGems as a dependency manager. The Ruby on Rails application framework generates a file called Gemfile, where all the required dependencies of the project will be mentioned there along with the version of the dependency. The file type of the Gemfile is .gemfile. Likewise, each application framework has a different file type as refereed in the

table 3. To overcome this challenge, when the project data is submitted at the beginning, the project data should pass a condition to identify which application framework it is. Once identifying the application, the OSS component extraction will be performed based on the file type of config file. So for each application framework, there should be different extraction functions.

Finding the right regex: Another challenge was finding the correct regex for each config file. The more challenging part was extracting the OSS component names and versions from config files like Gemfile and requirements.txt. It was not that difficult to overcome this challenge because the same solution, like the file structure, is required for this challenge. When the user submits a Django or Ruby on Rails project, firstly, as we said, the condition will identify the application framework and based on the file type, the extraction will be performed. The regex is implemented in the extraction function, for instance, when we see the figure 14 which lists all the OSS components used in a Ruby on Rails project. Each file line will be sent for a cleaning process and delivers the component name and version, which is extracted from each line. To extract the name and version from each line, a proper regex is required. Like Django and Ruby on Rails, the other application framework was not that difficult because most file types were XML and JSON files. The component name version can be easily extracted with the help of JSON and XML DOM parsing functions.

Verification: Another challenge was finding the correct regex for each config file. The more challenging part was extracting the OSS component names and versions from config files like Gemfile and requirements.txt. It was not that difficult to overcome this challenge because the same solution, like the file structure, is required for this challenge. When the user submits a Django or Ruby on Rails project, firstly, as we said, the condition will identify the application framework and based on the file type, the extraction will be performed. The regex is implemented in the extraction function, for instance, when we see the figure 14 which lists all the OSS components used in a Ruby on Rails project. Each file line will be sent for a cleaning process and delivers the component name and version, which is extracted from each line. To extract the name and version from each line, a proper regex is required. Like Django and Ruby on Rails, the other application framework was not that difficult because most file types were XML and JSON files. The component name version can be easily extracted with the help of JSON and XML DOM parsing functions.

5 Implementation

This chapter explains how the component is developed. The ways to overcome the design challenges, some of the findings during the development have been written down.

5.1 System Characteristics

When beginning the implementation, I decided to use a simple yet powerful programming language for both frontend and backend applications to obtain my results. The programming languages which I selected was javascript for the frontend and C# for the backend. I used the Visual Studio Code IDE for the frontend and Visual Studio 2019 for the backend to simplify my programming work. The configuration of the computer used for implementation is as follows:

Operating System: Windows 10 Enterprise 2016 - 64-bit
Processor: Intel Core i7-6700 @ 3.4GHz
RAM: 16GB
HDD: 500GB

The software support tools used for frontend implementation are listed as follows:

Programming Language: Javascript
IDE: Visual Studio Code
Packet Manager: npm
Version Control: Azure Repos

The software support tools used for backend implementation are listed as follows:

Programming Language: C#
IDE: Visual Studio 2019
Packet Manager: NuGet
Database: Azure Data Storage - Containers
Version Control: Azure Repos

All the infrastructure and software that I used are provided by evosoft GmbH.

5.2 OSS Component Analyzer

The OSS component analyzer is a starting point in the process of scanning the OSS components from a project. It is a front-end module, and it is implemented in the Angular framework using typescript language. The GUI interface of this module requires four inputs from the user: the project name, description, members, and project source directory. It has two essential tasks before it proceeds to the evaluation process. In the first task, the module should find the required configuration file in the project. This configuration file identification was made with the help of a dependency manager. Each application

framework has its dependency manager, and it generates a unique configuration file for the application framework. The table 3 shows all the config files of each application framework. The OSS component names and versions will be listed in the config files.

Application Framework	Dependency Manager	Config file	File Type
.NET(Console Application)	Nuget	.csproj file	XML
Angular Framework	npm	package.json	JSON
Microsoft TFS	Nuget	app.config	XML
Django	pip	requirements.txt	Text
Ruby on Rails	Rubygem	gemfile	File
Laravel	composer	composer.json	JSON
Gradle projects	gradle	build.gradle	GRADLE
Maven projects	POM	pom.xml	XML
.NET	Nuget	packages.config	XML

Table 3: Configuration file of each application framework.

Once the respective config is identified from the project, the targeted data should be scrapped from config files. To achieve this, automated scrapping methods are used based on the config file type. For instance, the user is scanning the Maven project, and then according to the table, the config file is an XML file, so, therefore, to scrap the targeted data DOM parsing method is used. Likewise, for JSON file, JSON parsing methods can be used, text pattern matching methods can be used for a text file, Gemfile, FIle and GRADLE files. After scrapping the targeted data, the next task is to generate a JSON object with the general project information and scrapped Open-source Software (OSS) information and will be ready to send it to the OSS evaluation process.

The figure 9 will illustrate the technical flow of how the OSS component is extracted from the software project and send the meta-information like component name and version for the evaluation process. Firstly, the user must give the public input such as project name, project description, project members and the complete source code directory. Once the data is submitted, the first validation stage is to check whether the required config file is available in the directory to identify the project's application framework. Once the scanner identifies the application framework, the OSS components will be extracted based on the config file's file type. For instance, the scanner will search for the build if the user submits a Gradle project for scanning .gradle file, which looks like the figure 15 and will start the extraction process. In the extraction process, the build.gradle file is iterated and read line by line with the help of FileReader class in Angular framework until the line begins with a string "compile". The dependencies section is where all the project dependencies, which is the OSS components, are registered, and we know that every dependency under the dependency section starts with the string "compile". So whenever the strings start with the string "compile", that particular line will consist of the component name and version. Then the component name and version will be extracted out by using text pattern matching functions. Likewise, the same type of extraction is performed for Django and Ruby on Rails projects. For the other projects, simple DOM parser methods are used to extract the components because their file types are JSON and XML. Finally, the extracted components will be converted to a JSON object and the general information and sent to the backend application for the evaluation process.

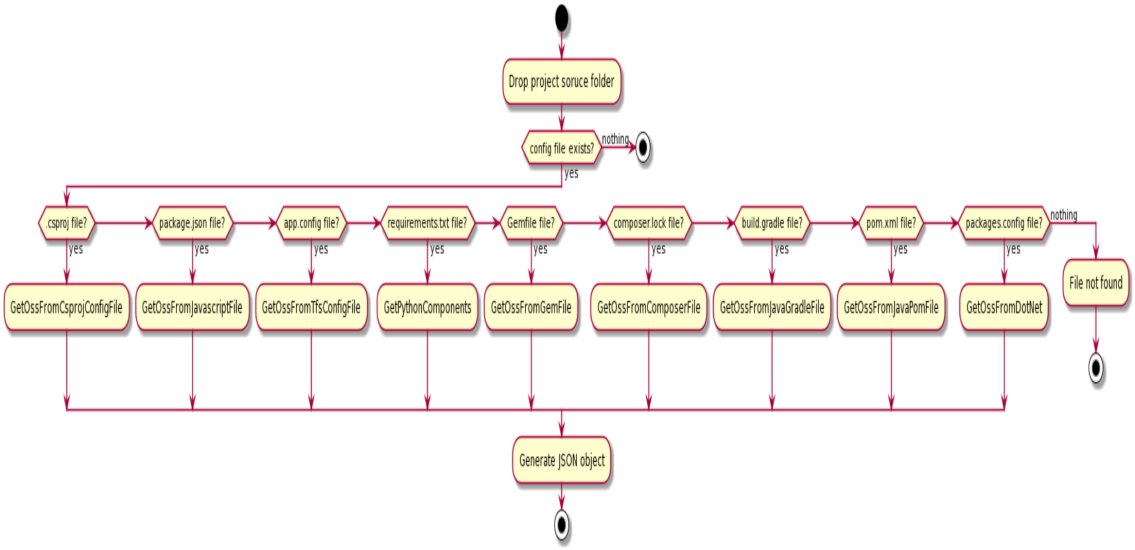


Figure 9: OSS Component Analyzer Activity Diagram

5.3 Component Evaluator

The OSS component evaluator is the final process of finding the vulnerabilities with the help of a vulnerability database. This backend application is developed by using ASP.NET. The searching can be done with the help of NVD's API services [NVD]. Once the backend application receives the JSON objects from the client application, the application will start searching the OSS components in the NVD database by using the endpoints provided in the API documentation [NVD]. First component names and versions will be searched under the CPE dictionary to verify the product has been registered in the CPE. If the component is found in the CPE dictionary, the service returns all the results of the component name. To verify the component name and the version is not a normal process because the CPE name has its naming specification. The figure 10 is an example of the CPE naming specification. From the figure 10 it is visible that the component name and

```
cpe:/a:microsoft:internet_explorer:8.0.6001:beta

wfn:[part="a",vendor="microsoft",product="internet_explorer",
version="8\.0\.6001",update="beta"]
```

Figure 10: CPE name specification

version are in the 3rd and fourth position of the CPE string. The given component name and version must be verified with the CPE name. As the second stage of verification, the given component name and CPE name is verified again by using the Levenshtein distance algorithm. Having this second stage of verification can make sure the component does not have any spelling mistakes. Once the CPE name is validated with the given component, the next step will be retrieving the CVE details using the CPE name to find out the

vulnerability information. CVE retrieval API services are used to find the vulnerability information [NVD].

The figure 11 will illustrate the technical side of how the OSS components are evaluated with the help of the NVD database. Firstly, the backend application will receive the request sent from the client application, and the OSS components in the JSON object will be iterated to evaluate each component. In the iteration, the first process is to search the component name in the CPE dictionary to verify the component is registered in the CPE dictionary by using the "https://services.nvd.nist.gov/rest/json/cpes/1.0?keyword=" API endpoint. The results from the CPE dictionary will be iterated to find the exact component name of its version. The component name and version will be matched with the CPE name that looks like in the figure 10. We already know that the vendor, product name and version are separated using the ":" character. In this case, first, the CPE name will be split using the ":" character and converted to an array. Now every 3rd and fourth position of the array will be matched with the given component name and version respectively for every iteration until finding the exact match. As the second stage of verification, the component name and the CPE product name will have a string distance evaluation using the Levenshtein distance algorithm to spellcheck. If the distance result is 0, it means the spelling match is perfect, but if the result is 1 to 2, still the component will be taken to the result, but it will be flagged as "recommended". If it is greater than or equal to 3, then the iteration will be skipped to the next component and flagged as "Not found". Once the OSS component name is verified against the CPE name, the next task is to search for the vulnerabilities in the CVE database by using the CPE name string. The vulnerabilities are searched using the "https://services.nvd.nist.gov/rest/json/cves/1.0?cpeMatchString=" API endpoint. The API endpoint requires cpeMatchString as an input parameter where we can give the CPE name as input and gives the vulnerability information of the particular CPE name like the figure 20. After collecting the vulnerability results of all the possible OSS components, the results will be converted as a JSON object. It will be saved in the Azure data storage as a blob container.

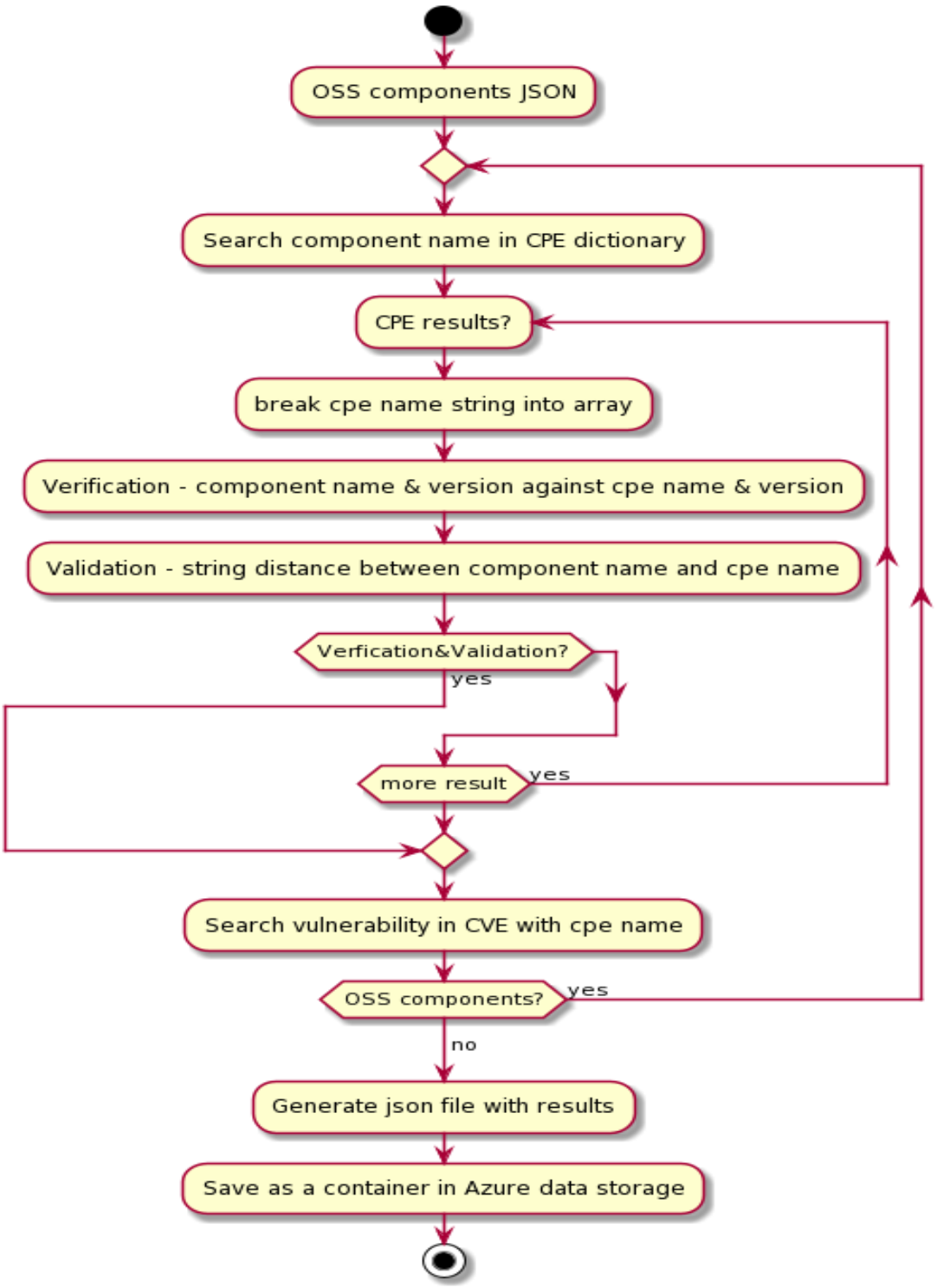


Figure 11: OSS Component Evaluator Activity Diagram

5.4 Reporter

The Reporter is the final module of the OSS scanner, and it is implemented in the client-side of the application. This module is implemented in the Angular framework by using the jspdf library. The primary purpose of this module is to generate a report of the scanned projects by containing the CVSS information of each OSS component. The report will consist only of the necessary and human-understandable information in the report, so this makes the user have a clear understanding of the report and helps to make a decision. The information for the report will be retrieved from the Azure data storage.

The figure 12 shows the technical flow of how the report is downloaded. Once the user requests the backend application, the backend application will retrieve the blob container of the specific project which is requested. In the blob container, the JSON object is stored as blob data. Then the blob data is converted to a JSON object with relevant properties like both CVSS V2 and V3 data. Finally, with the help of the jspdf library, the results retrieved from the backend application is downloaded as a pdf file.

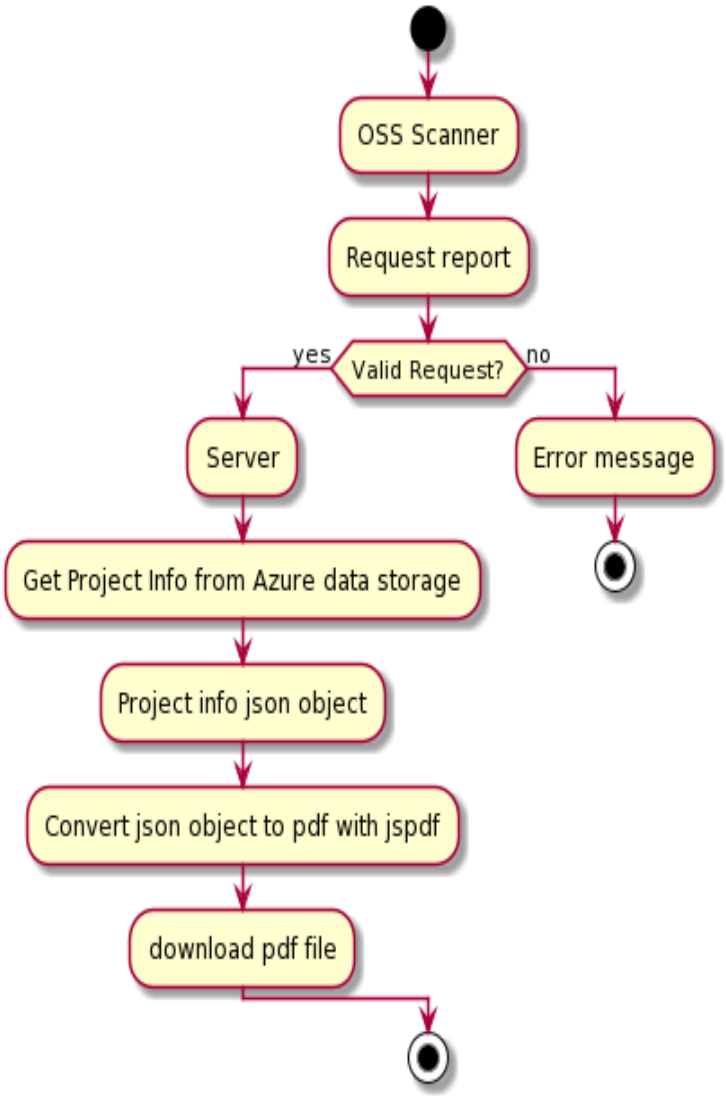


Figure 12: OSS Component Evaluator Activity Diagram

6 Results & Discussion

In this chapter, we summarize the results obtained from our experiments. Our experiment is mainly focused on automating the open-source software discovery and finding a suitable vulnerability database for vulnerability analysis.

6.1 Open-Source Software Discovery

6.1.1 Manual Scrapping

As defined previously, the scanner has to scan OSS components from different application framework projects. It means that the scanner should have a standard output function even if it scans different projects. In this experiment, first, we have tried using manual scrapping to understand how the OSS components can be identified and where it is stored inside the project. Manual scrapping will be an easy solution if it is a small project, but it will be time-consuming if performed in a large scale project. Figure 13, 14, 15, 16, 17 and 18 shows where exactly the OSS components are registered in the project. With the help of manual scrapping from these files, we can extract the component name and version by traditional copy & pasting technique. Then we have to search vulnerability information of each component in the vulnerability database. Therefore manual scrapping for big projects will be a time-consuming task, and also there is a possibility for human errors.

Figure 13 shows the composer.json file of the laravel framework project. In the composer.json file, the "require" and "require-dev" properties have the libraries and versions used in the project. Whenever the developer installs a library through the composer dependency manager, the installed library's name and version will be registered in the composer.json file.

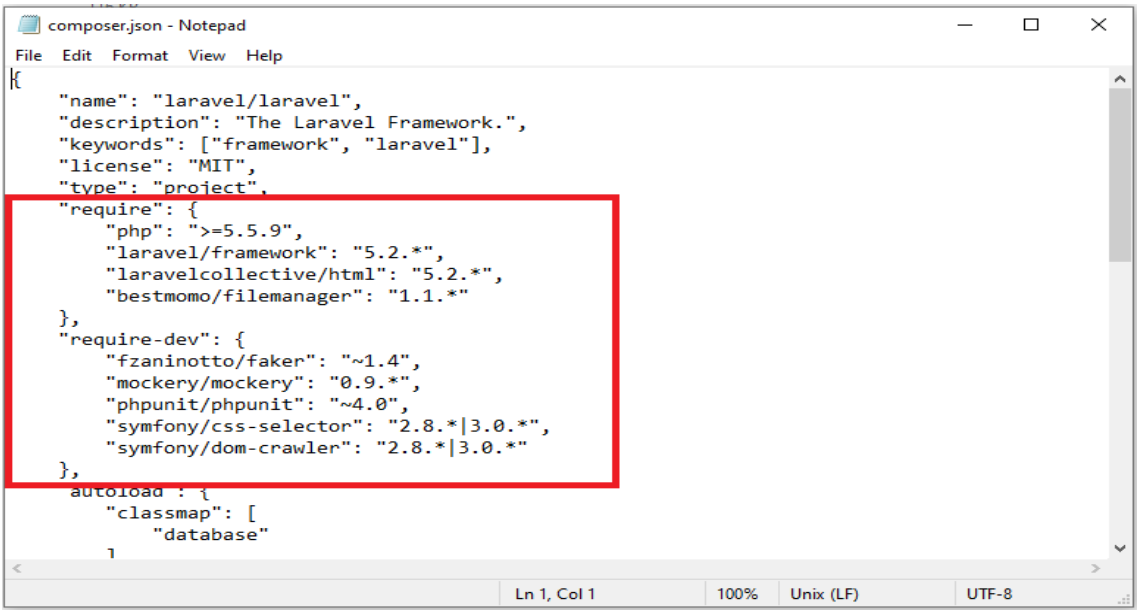


Figure 13: Laravel project config file

The figure 14 shows the Gemfile present inside a Ruby on Rails project. The developers will include all the required libraries and its version in the Gemfile, and the following way to include a library in a Gemfile is ”gem <component_name>, <version>”.

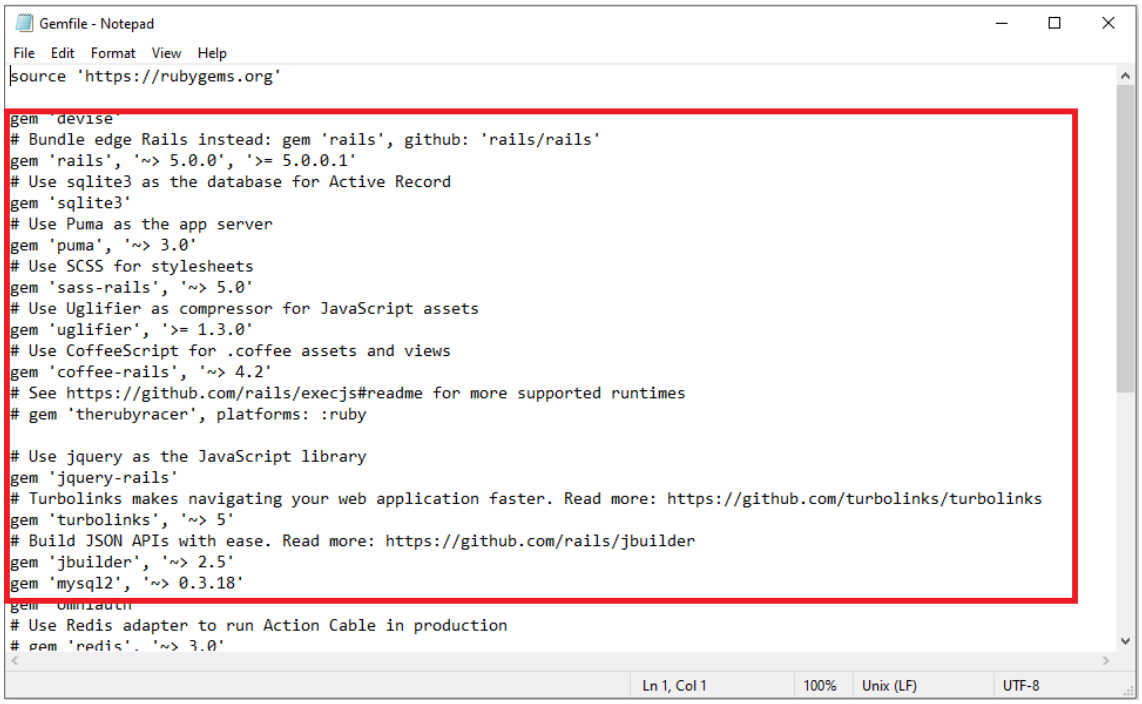


Figure 14: Ruby on Rails project config file

The figure 15 shows the build.gradle file present inside a Gradle project. All the required libraries of the project are listed under the ”dependencies” section.

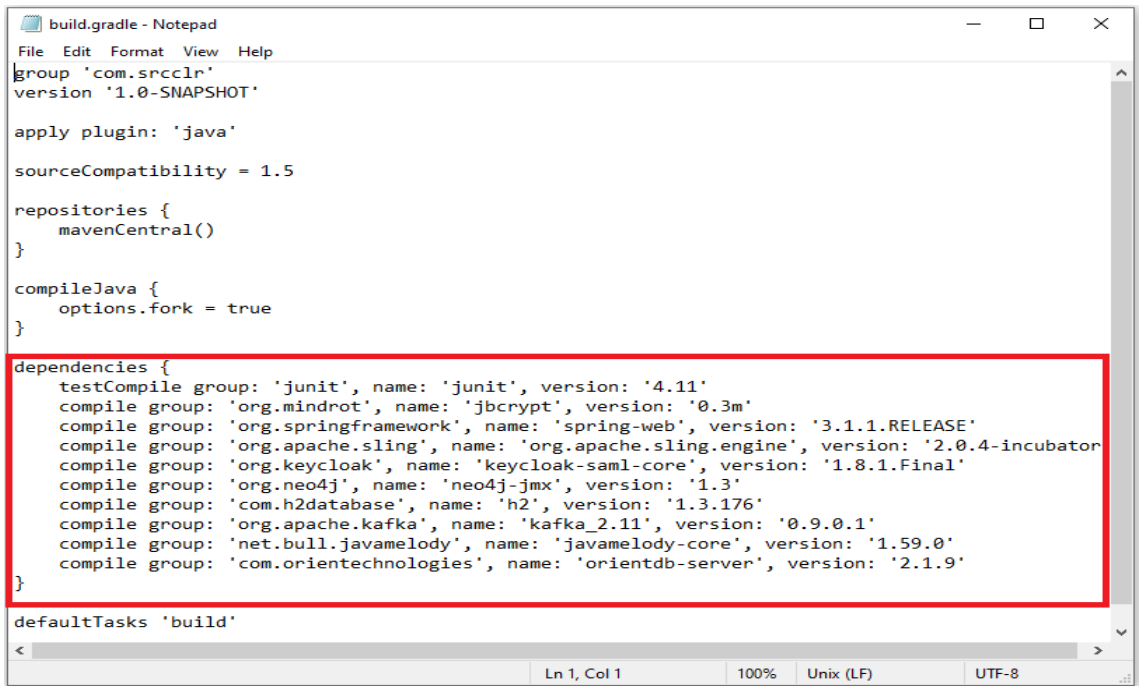


Figure 15: Gradle project config file

Figure 16 shows the requirements.txt file present inside a Django project. All the libraries' names and versions will be stored in the requirements.txt file when installed through the pip packet-management system. The requirements.txt is a simple text file that contains only the component names and versions.

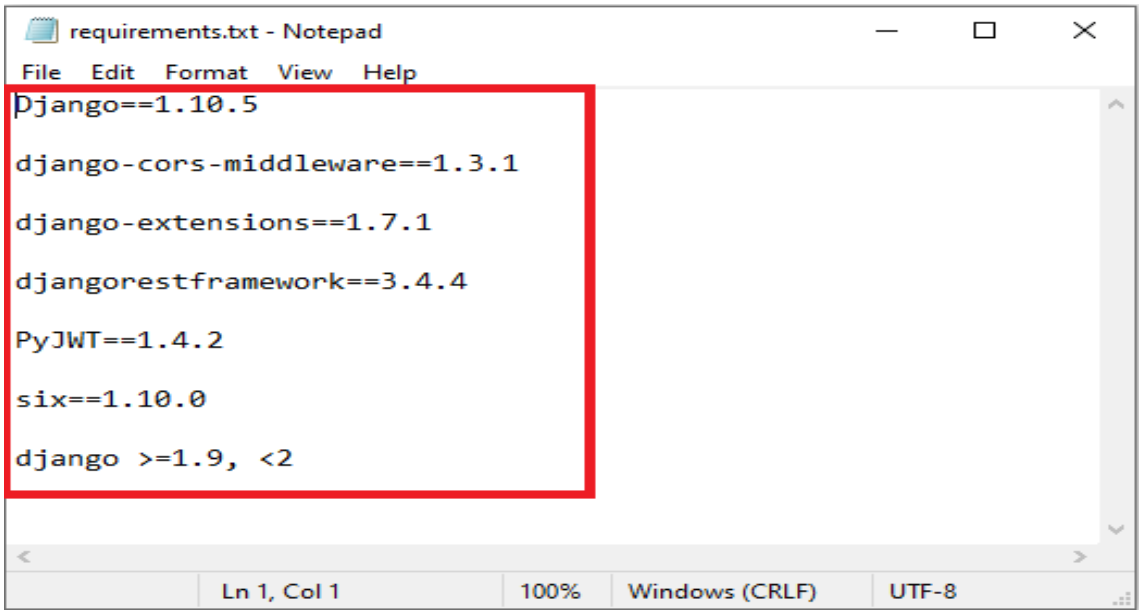


Figure 16: Django project config file

Figure 17 shows the pom.xml file present inside a Maven project. All the required libraries of the project are listed under the "<dependencies>" tag. Each component will be listed under the "<dependency>" tag and under that component name, and the version will be mentioned under the "<artifactId>" and "<version>" tag, respectively.

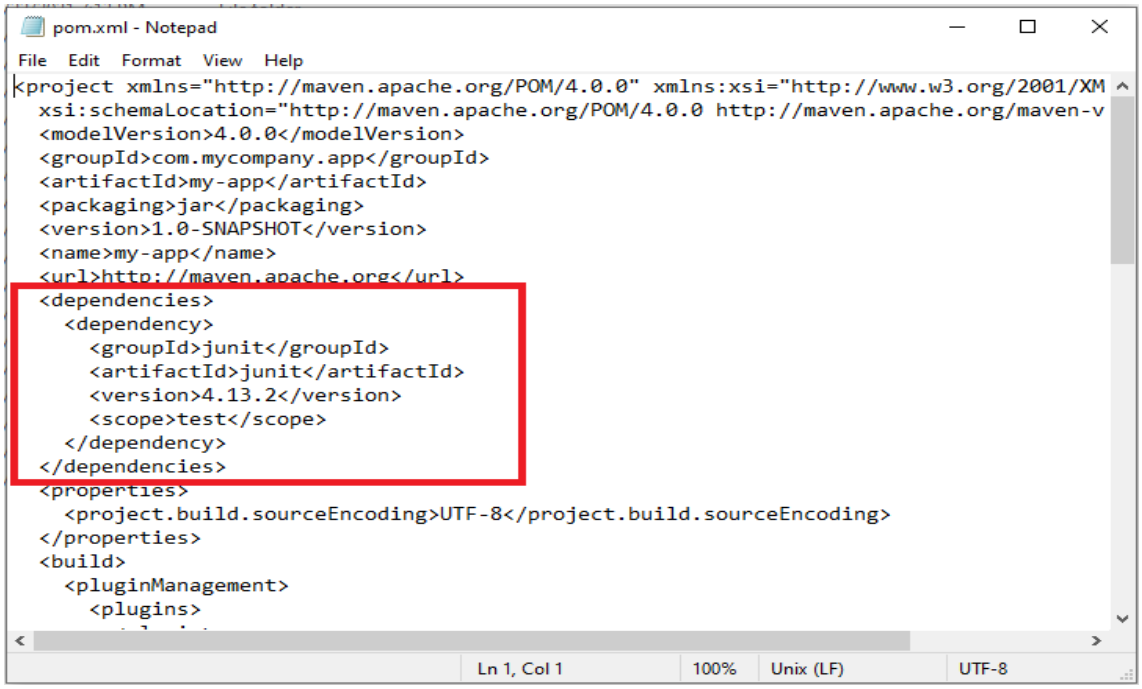


Figure 17: Maven project config file

Figure 18 shows the .csproj file present inside a .Net project. The libraries are installed through the Nuget dependency manager. The installed libraries' names and versions will be registered in the .csproj file, an XML file. The component information will be addressed in the "PackageReference>" tag, and the component will be registered under the "ItemGroup>" tag.

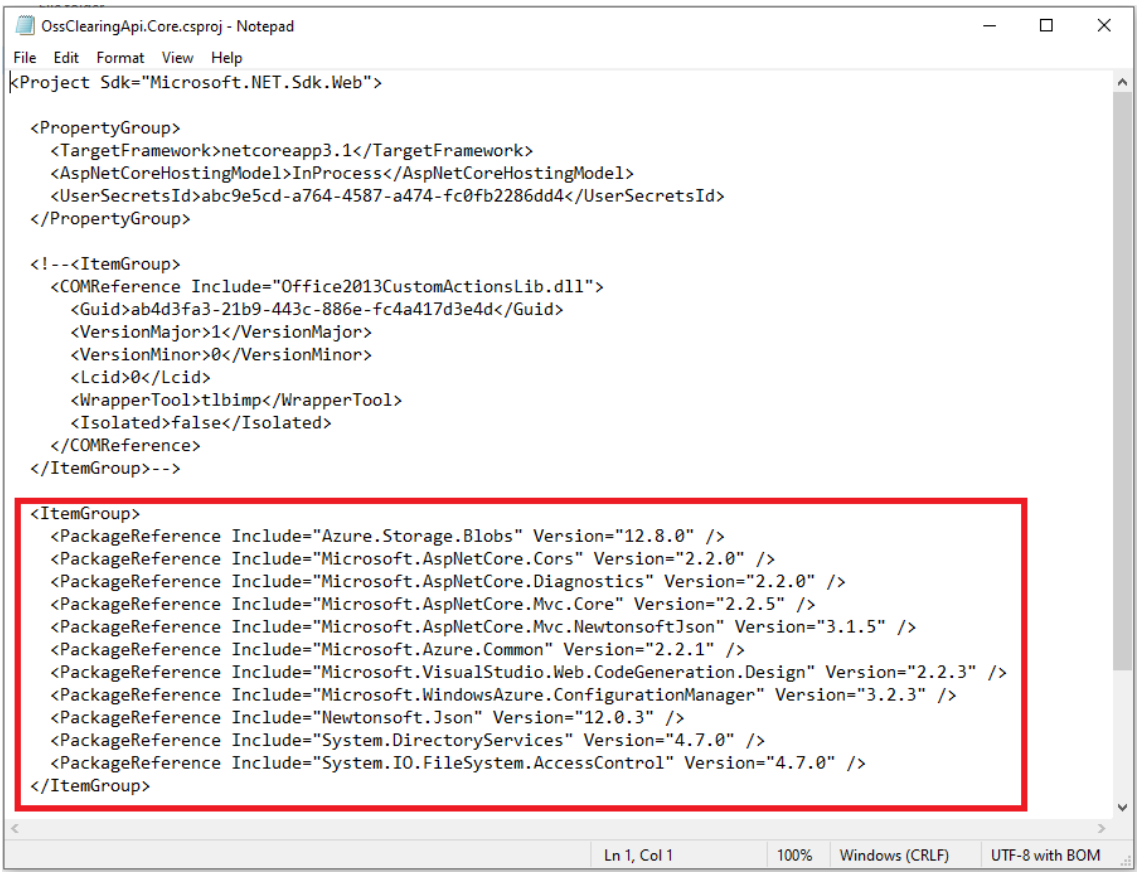


Figure 18: Dotnet project config file

6.1.2 Automated Scrapping

After experimenting the manual scraping, we tried automating the process by using Automated scraping methods. DOM parsing and text pattern matching methods are used for automated extraction. The main reason to use these two scraping methods is that the scanning takes place on the client-side. These two methods of scrapping can be achieved by using the Angular framework. For projects like Dotnet, Gradle, Laravel and Maven DOM parsing methods are used because the config files of these projects are XML and JSON files. The text pattern matching methods are used for Django and Ruby on Rails projects. Finally, by using these two scraping methods, a generic function is created to give a final output as a JSON object. Figure 19 will illustrate the extracted component from the Ruby on Rails project within the browser. Along with the component name and version, we have also shown from which file it is extracted.

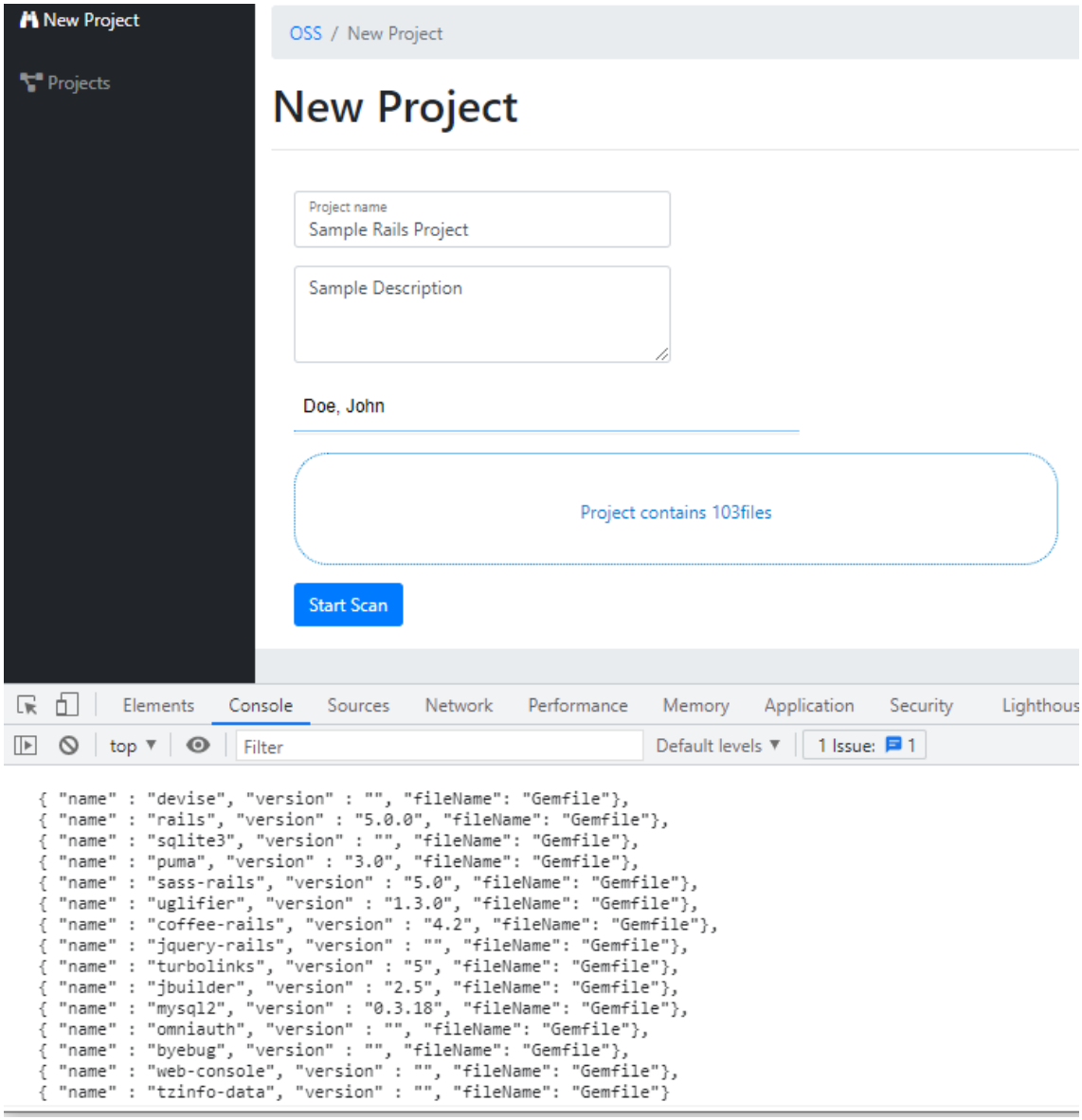


Figure 19: OSS component name and version extraction from Ruby on Rails project

6.2 Vulnerability Databases

After scraping the component name and its version, the following process is to find the vulnerabilities of each OSS component. Before starting the process, we must first find a reliable vulnerability database with programmatic access to automate the process. The IBM X-Force was the first database I found, and apart from the regular user interface access, it also has programmatic access like API services. Though IBM X-Force has all the utilities to automate the process, it is accessible only for non-commercial use, not for commercial purposes. The following database, which we experimented with, is CERT/CC, but before exploring it further, the database has only user interface access and no programmatic access.

The next database was Bugtraq(Security Focus), but unfortunately, it has been shut down from January 31st 2021[Cim21]. Still, the website is live with old data, and it can

be used for manual searching. On top of the, accessing a shut downed service will open for other security threats. All three databases mentioned above uses CVE dictionary to search and register the vulnerabilities. Considering the requirements like secured, reliable, programmatic access and commercial use, the NVD database is a suitable fit for searching vulnerability. The NVD database is completely open-source, and we can use it for both commercial and non-commercial use. When comparing NVD with IBM X-Force, the key advantage is that it is completely open-source. The NVD database has individual API documentation for both CVE and CPE. Therefore the OSS component can be verified with the help of CPE. Like IBM X-Force, the NVD database also give CVSS scores for each component so that the users will find it easy to assess the OSS component. Figure 20 shows the CVSS both V2 and V3 scores of PyJwt 0.3.2 OSS component vulnerability information.

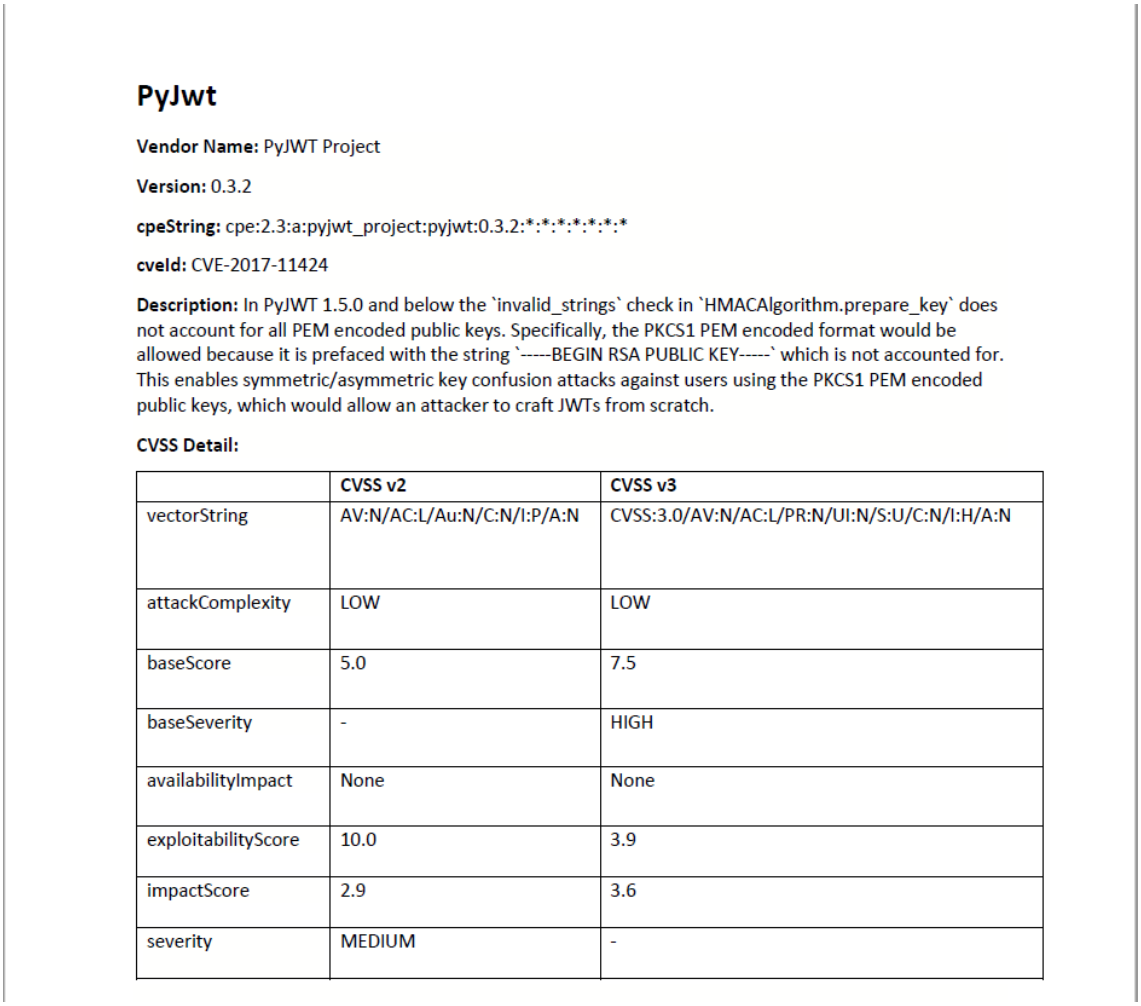
Schema: <No Schema Selected>

```
            ]
          },
          "impact": {
            "baseMetricV3": {
              "cvssV3": {
                "version": "3.0",
                "vectorString": "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N",
                "attackVector": "NETWORK",
                "attackComplexity": "LOW",
                "privilegesRequired": "NONE",
                "userInteraction": "NONE",
                "scope": "UNCHANGED",
                "confidentialityImpact": "NONE",
                "integrityImpact": "HIGH",
                "availabilityImpact": "NONE",
                "baseScore": 7.5,
                "baseSeverity": "HIGH"
              },
              "exploitabilityScore": 3.9,
              "impactScore": 3.6
            },
            "baseMetricV2": {
              "cvssV2": {
                "version": "2.0",
                "vectorString": "AV:N/AC:L/Au:N/C:N/I:P/A:N",
                "accessVector": "NETWORK",
                "accessComplexity": "LOW",
                "authentication": "NONE",
                "confidentialityImpact": "NONE",
                "integrityImpact": "PARTIAL",
                "availabilityImpact": "NONE",
                "baseScore": 5.0
              },
              "severity": "MEDIUM",
              "exploitabilityScore": 10.0,
              "impactScore": 2.9,
              "acInsufInfo": false,
              "obtainAllPrivilege": false,
              "obtainUserPrivilege": false,
              "obtainOtherPrivilege": false,
              "userInteractionRequired": false
            }
          },
          "publishedDate": "2017-08-24T16:29Z",
          "lastModifiedDate": "2019-10-03T00:03Z"
        }
      ]
```


Figure 20: Vulnerability information of PyJwt 0.3.2 component from NVD database

6.3 Vulnerability Report:

The report will be ready to download once the vulnerability assessment is over. The main task of this module is to give a simple report of the OSS component discovered and its vulnerability information. The report will have both CVSS V2 and V3 properties so that the user can assess the component based on their requirements. The CVE and CPE id will also be in the report. Therefore the user can look up further information externally if it is needed. This report will not make any decisions on behalf of the user. It will just share the vulnerability details of each OSS component. Based on the vulnerability information, the user can decide whether to keep the OSS component or move to the next possible version. Figure 20 will illustrate how the report indicates vulnerability information of each OSS component.



PyJwt

Vendor Name: PyJWT Project

Version: 0.3.2

cpeString: cpe:2.3:a:pyjwt_project:pyjwt:0.3.2:*:*:*:*:*:*:*

cveId: CVE-2017-11424

Description: In PyJWT 1.5.0 and below the `invalid_strings` check in `HMACAlgorithm.prepare_key` does not account for all PEM encoded public keys. Specifically, the PKCS1 PEM encoded format would be allowed because it is prefaced with the string `-----BEGIN RSA PUBLIC KEY-----` which is not accounted for. This enables symmetric/asymmetric key confusion attacks against users using the PKCS1 PEM encoded public keys, which would allow an attacker to craft JWTs from scratch.

CVSS Detail:

	CVSS v2	CVSS v3
vectorString	AV:N/AC:L/Au:N/C:N/I:P/A:N	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N
attackComplexity	LOW	LOW
baseScore	5.0	7.5
baseSeverity	-	HIGH
availabilityImpact	None	None
exploitabilityScore	10.0	3.9
impactScore	2.9	3.6
severity	MEDIUM	-

Figure 21: Sample Report of PyJwt 0.3.2 component from NVD database

7 Conclusion & Future Work

7.1 Conclusion

In this thesis, we have focused on automating the open-source software discovery by developing a client-side scanner and finding a suitable vulnerability database to find the OSS component's vulnerability. Software security is one of the most significant quality assurance measures for software. It has been shown that software security is one of the most critical and essential parts to focus on before the application gets delivered to the end user[4]. This proposed scanner is developed to find all the OSS components used in a software project, and this solution will help the developers to find the vulnerabilities present in the OSS component. The scanner can scan the following application framework projects: Django, Laravel, Ruby on Rails, .Net core, Gradle and Maven.

This implementation has the advantage that the OSS component analyzer module can be reused for integrating with another vulnerability database. Also, it can be modified by adding a new application framework project for scanning. The end report provided by the system will not take or suggest any decisions on behalf of the user. The report will give the vulnerability findings of each component where the developer can decide whether to keep the same version or move to a different version of the OSS component. This thesis implementation will be helpful for the first level of security check before delivering the application to the end-user.

7.2 Future Work

Our work shows a direction for further development and research in the domain of open-source software security. We have developed a scanner that extracts the OSS components from software projects with the help of dependency managers. There are numerous possibilities to improve or customize this scanner according to the application scenario.

The extracted OSS component from the software projects can be used for creating a license clearing tool. The license clearing tool is another crucial process in an organization to maintain transparency towards software usage. If the license is adequately maintained, then the organization can avoid the unwanted cost.

Apart from automating the extraction of OSS components and delivering the vulnerability information of each OSS component, we can also develop an extra feature like recommending the nearest version with less vulnerability of the same component. In this case, the user can have an idea of the nearest version of the component.

There could be many other ways to improve the scanner, other than the ones pointed out by us. However, we believe that the scanner so far has yielded satisfactory results, as is evident from the evaluation. We hope to see more improvements in this scanner, or in general, in the field of open-source software security.

References

[201] Devopedia. 2018. Dependency manager.

[AH16] Rashad Aliyev and Lourdes Herrero. Analyzing vulnerability databases. 10 2016.

[AKAA15] O. B. Al-Khurafi and M. A. Al-Ahmad. Survey of web application vulnerability attacks. In *2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, pages 154–158, Los Alamitos, CA, USA, dec 2015. IEEE Computer Society.

[AKTY06] Ashish Arora, Ramayya Krishnan, Rahul Telang, and Yubao Yang. An empirical analysis of software vendors' patching behavior: Impact of vulnerability disclosure. page 22, 01 2006.

[Alq17] S. S. Alqahtani. Enhancing trust â€“ software vulnerability analysis framework. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 563–564, Los Alamitos, CA, USA, mar 2017. IEEE Computer Society.

[AUS05] Kemal Altinkemer, Jackie Ulmer, and Sanjay Sridhar. Vulnerabilities and risk management of open source software: An empirical study. 01 2005.

[Bug] Bugtraq. `https://www.securityfocus.com/archive/1/description#0.2.1`. Accessed: 01.03.2021.

[CAS+04] M. Cukier, N. Anand, W. H. Sanders, J. R. Martin, and A. Sharma. Ferret: A host vulnerability checking tool. In *Pacific Rim International Symposium on Dependable Computing, IEEE*, pages 389–394, Los Alamitos, CA, USA, mar 2004. IEEE Computer Society.

[Cim21] Catalin Cimpanu. Iconic bugtraq security mailing list shuts down after 27 years, 2021.

[Cue17] Cuelogic. The levenshtein algorithm, 2017.

[cve] Cve - faq. `https://cve.mitre.org/about/faqs.html`. Accessed: 01.03.2021.

[DAT] NATIONAL VULNERABILITY DATABASE. Cvss severity distribution over time.

[Dem21] Andrew Demchenko. How to deal with the most common challenges in web scraping, 2021.

[DWG04] G. Dietrich, G. B. White, and T. Goles. Cyber security exercises: Testing an organization?s ability to prevent, detect, and respond to cyber security events. In *2014 47th Hawaii International Conference on System Sciences*, volume 8, page 70170a, Los Alamitos, CA, USA, jan 2004. IEEE Computer Society.

[End19] Statistical Odds & Ends. Jaro & jaro-winkler similarity, 2019.

[FRPCDP18] S. Flores-Ruiz, R. Perez-Castillo, C. Domann, and S. Puica. Mainframe migration based on screen scraping. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 675–684, Los Alamitos, CA, USA, sep 2018. IEEE Computer Society.

[Gil] Alexander S. Gillis. Screen scraping.

[Gla19] Rhett Glauser. A history of the vulnerability management lifecycle, 2019.

[GS17] Anna Granova and Marco Slaviero. Chapter 83 - cyber warfare. In John R. Vacca, editor, *Computer and Information Security Handbook (Third Edition)*, pages 1085–1104. Morgan Kaufmann, Boston, third edition edition, 2017.

[HHR13] Michael Heron, Vicki Hanson, and Ian Ricketts. Open source and accessibility: Advantages and limitations. *Journal of Interaction Science*, 1, 01 2013.

[Ibm] Ibm x-force. `https://www.ibm.com/in-en/security/xforce`. Accessed: 01.03.2021.

[Imp] Inc. Imperva. Database security assessment.

[Ins] Software Engineering Institute. The cert division.

[JM08] G. Jabbour and D. A. Menasce. Policy-based enforcement of database security configuration through autonomic capabilities. In *2008 4th International Conference on Autonomic and Autonomous Systems*, pages 188–197, Los Alamitos, CA, USA, mar 2008. IEEE Computer Society.

[Joh20] Mic Johnson. Main challenges in web scraping, 2020.

[Kas17] Kaspersky. The biggest ransomware threats of 2017, 2017.

[KS15] R. Koizumi and R. Sasaki. Study on countermeasures using mitigation software against vulnerability attacks. In *2015 Fourth International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, pages 28–33, Los Alamitos, CA, USA, oct 2015. IEEE Computer Society.

[Kun21] Arnab Kundu. Jaro & jaro-winkler similarity, 2021.

[LA11] L. Lowis and R. Accorsi. Vulnerability analysis in soa-based business processes. *IEEE Transactions on Services Computing*, 4(03):230–242, jul 2011.

[Maa19] Gilad David Maayan. The dangers of open-source vulnerabilities, and what you can do about it, 2019.

[Mat17] Seun Matt. What are dependency managers?, 2017.

[nis] Nist- national vulnerability database. `https://nvd.nist.gov/general`. Accessed: 01.03.2021.

[NJM18] Z. Nasar, S.W. Jaffry, and M.K Malik. Information extraction from scientific articles: a survey. *International Journal for all Quantitative Aspects of the Science of Science*, 117, 09 2018.

[NMED14] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitraş. Some vulnerabilities are different than others. In Angelos Stavrou, Herbert Bos, and Georgios Portokalidis, editors, *Research in Attacks, Intrusions and Defenses*, pages 426–446, Cham, 2014. Springer International Publishing.

[NVD] NVD. New nvd cve/cpe api and legacy soap service retirement.

[Oos04] Alex van Oostenrijk. Screen scraping web services. 12 2004.

[PD89] Neumann P.G. and Parker D.B. A summary of computer misuse techniques. *12th Nat'l Computer Security Conf*, pages 396–406, 10 1989.

[Rad] Radware. An overview of web scraping techniques.

[Ros] Linda Rosencrance. vulnerability assessment (vulnerability analysis).

[S18] Vithal S. Different extraction methods in data warehouse, 2018.

[Say] Dr. Saed Sayad. K nearest neighbors - classification.

[SB16] A. Sampaio and J. G. Barbosa. A study on cloud cost efficiency by exploiting idle billing period fractions. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 138–143, Los Alamitos, CA, USA, nov 2016. IEEE Computer Society.

[Sec] Securityfocus. `https://www.securityfocus.com/bid/109320`. Accessed: 01.03.2021.

[Sna] Snaplogic. Study: Businesses losing 140 billion annually to disconnected data.

[Sre] Syamini Sreedharan. What is vulnerability assessment. testing process, vapt scan tool.

[ss18] White source software. The national vulnerability database explained, 2018.

[Tok15] Natalia Tokareva. Chapter 11 - distances between bent functions. In Natalia Tokareva, editor, *Bent Functions*, pages 89–96. Academic Press, Boston, 2015.

[Tur38a] Alan Mathison Turing. *Systems of Logic Based on Ordinals: a Dissertation*. Ph.d., Cambridge University, Cambridge, UK, 1938.

[Tur38b] Alan Mathison Turing. *Systems of Logic Based on Ordinals: a Dissertation*. Ph.d., Cambridge University, Cambridge, UK, 1938.

[VKCK08] H. L. Vu, K. K. Khaw, T. Chen, and Fei-Ching Kuo. A new approach for network vulnerability analysis. In *2008 33rd IEEE Conference on Local Computer Networks (LCN 2008)*, Los Alamitos, CA, USA, oct 2008. IEEE Computer Society.

[Vul] The importance of vulnerability testing. `https://ancgroup.com/the-importance-of-vulnerability-testing/`. Accessed: 01.03.2021.

[Web21] Webopedia. Cert/cc, 2021.

[Wei11] Zhaohui Wei. Research on the application of open source software in digital library. *Procedia Engineering*, 15:1662–1667, 12 2011.

[Wika] Wikipedia. Damerau–levenshtein distance.

[Wikb] Wikipedia. Jaro–winkler distance.

[Wu20] Gang Wu. String similarity metrics – edit distance, 2020.

[YLK21] J. You, J. Lee, and H. Kwon. A complete and fast scraping method for collecting tweets. In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 24–27, Los Alamitos, CA, USA, jan 2021. IEEE Computer Society.

[ZS19] Chunchun Zhao and Sartaj Sahni. String correction using the damerau-levenshtein distance. *BMC Bioinformatics*, 20:277, 06 2019.

In accordance with § 9 Para. 12 APO, I hereby declare that I wrote the preceding master's thesis independently and did not use any sources or aids other than those indicated. Furthermore, I declare that the digital version corresponds without exception in content and wording to the printed copy of the master's thesis and that I am aware that this digital version may be subjected to a software-aided, anonymised plagiarism inspection.

Bamberg, 17.09.2021 Hari Prashanth Rajendran