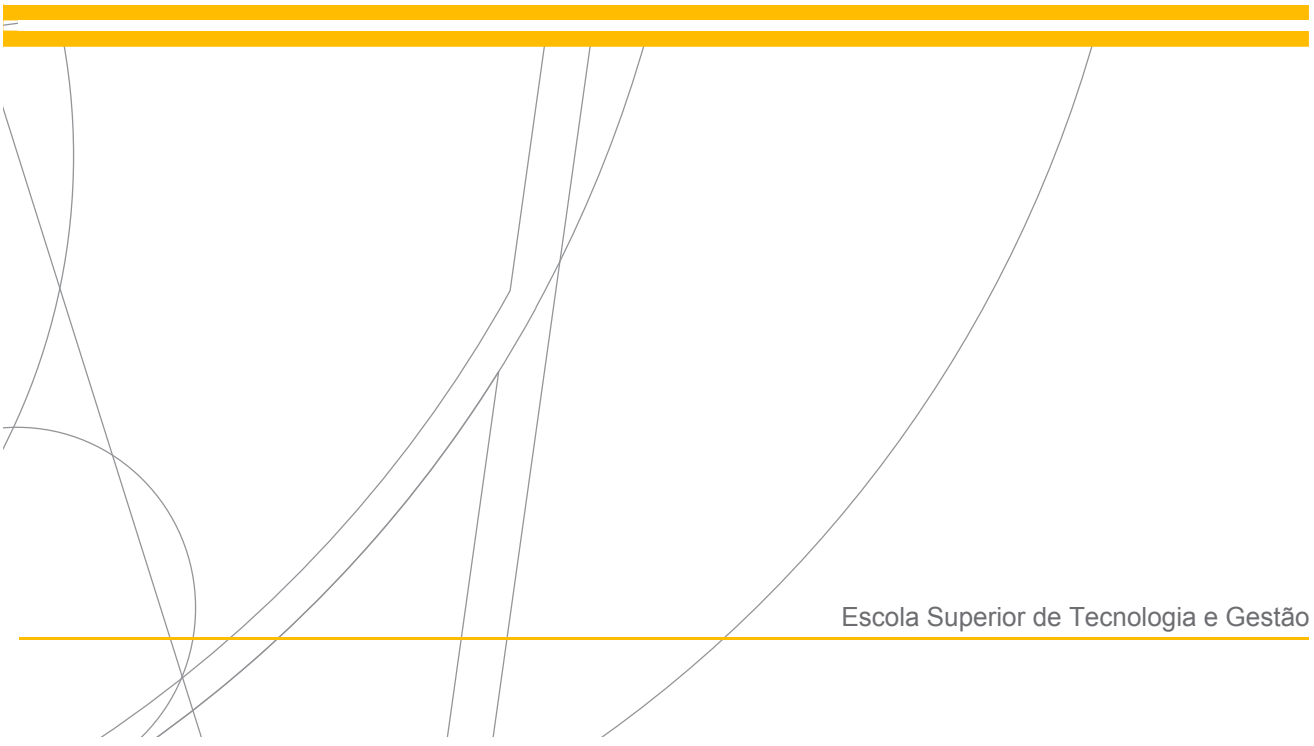
Instituto Politécnico
de Viana do Castelo

ASSESSING THE ACCURACY OF VULNERABILITY SCANNERS AND DEVELOPING A TSUNAMI SECURITY SCANNER PLUG-IN

Ricardo Araújo



Escola Superior de Tecnologia e Gestão

Instituto Politécnico
de Viana do Castelo

Ricardo dos Santos Araújo

ASSESSING THE ACCURACY OF VULNERABILITY SCANNERS AND DEVELOPING A TSUNAMI SECURITY SCANNER PLUG-IN

Nome do curso de Mestrado

Mestrado em Cibersegurança

Trabalho efetuado sob a supervisão de

Professor Pedro Pinto

Professor António Pinto

Janeiro de 2023

Mestrado em Cibersegurança
Master in Cybersecurity

Assessing the Accuracy of Vulnerability Scanners and Developing a Tsunami Security Scanner Plug-in

a master's thesis authored by

Ricardo dos Santos Araújo

and supervised by

Pedro Filipe Cruz Pinto
Professor Adjunto, Instituto Politécnico de Viana do Castelo

António Alberto dos Santos Pinto
Professor Coordenador, Instituto Politécnico do Porto

This thesis was submitted in partial fulfilment of the requirements for the Master's degree in Cybersecurity at the Instituto Politécnico de Viana do Castelo

ipvc

14 of February, 2023

Abstract

Digital transformation is a key factor for a company's success. Recently this digital transformation was accelerated in many companies due to the Covid-19 pandemic, requiring more changes in people, systems, and data. In some cases, these changes in systems and procedures uncover new vulnerabilities that could be early detected and mitigated. In this context, the vulnerability scanner tools may prevent configuration errors and known vulnerabilities at an early stage.

The release of the Tsunami Security Scanner, an open-source vulnerability scanner released by Google, opens the opportunity to analyze and compare the commonly used, free-to-use vulnerability scanners. The wide choice of Vulnerability Scanning Tools can be a time-consuming task for a company that needs to take into consideration complex and numerous variables such as accuracy and precision to be able to choose the right tool.

This thesis aims to assess the accuracy of vulnerability scanner tools. In the first stage resources usage and performance assessment regarding different vulnerabilities and systems. In the second stage, a plugin is developed for the Tsunami Security Scanner with the purpose of detecting a specific vulnerability (CVE-2019-12815).

The precision assessment is accomplished by placing multiple virtual machines in a network with different vulnerable scanners and other machines with different vulnerable and non-vulnerable operating systems. This enables the validation that the features and performance of these scanners are different or vary accordingly to the target systems. This work can be particularly helpful to organisations with lower resources such as Small and Medium-sized Enterprises (SMEs) since it reviews a set of these tools that are available for use. The development of the Tsunami Security Scanner plugin is also important as an effort to increase the range of plugins available.

Keywords: Vulnerability Scanning. Comparison. Open-source. Tsunami

Resumo

A transformação digital é um fator chave para o sucesso das empresas. Recentemente a transformação digital foi acelerada em muitas empresas devido à pandemia de Covid-19, exigindo mudanças de pessoas, sistemas e dados. Em alguns casos, essas mudanças nos sistemas e procedimentos revelam novas vulnerabilidades que devem ser detectadas e mitigadas com antecedência. Neste contexto, as ferramentas de verificação de vulnerabilidades podem evitar erros de configuração e vulnerabilidades conhecidas numa fase antecipada.

A disponibilização do Tsunami Security Scanner, um verificador de vulnerabilidades de código aberto lançado pelo Google, abre a oportunidade de analisar e comparar os verificadores de vulnerabilidades comumente usados e gratuitos. A ampla escolha de ferramentas de verificação de vulnerabilidades pode ser uma tarefa demorada para uma empresa que precisa levar em consideração variáveis complexas e numerosas, como exatidão e precisão, para poder escolher a ferramenta certa.

Esta tese visa avaliar a precisão de ferramentas de verificação de vulnerabilidades. Numa primeira fase, avaliação do uso de recursos e desempenho em relação a diferentes vulnerabilidades e sistemas. Numa segunda fase, é desenvolvido um plugin para o Tsunami Security Scanner com o objetivo de detectar uma vulnerabilidade específica (CVE-2019-12815).

A avaliação da precisão das ferramentas é realizada colocando múltiplas máquinas virtuais em uma rede com diferentes verificadores de vulnerabilidades e outras máquinas com diferentes sistemas operativos vulneráveis e não vulneráveis. Isso permite validar que as características e desempenho desses verificadores são diferentes ou variam de acordo com os sistemas-alvo. Este trabalho pode ser particularmente útil para organizações com recursos mais limitados, já que revê um conjunto dessas ferramentas que estão disponíveis para uso. O desenvolvimento do plugin para o Tsunami Security Scanner também é importante como um esforço para aumentar a gama de plugins disponíveis.

Palavras-chave: Vulnerability Scanning. Comparison. Open-source. Tsunami

Aknowledgements

First of all, I would like to thank Professors Pedro Pinto and António Pinto for their support and for their dedicated time to making this work possible. Then thank my wife for her patience and support. Finally, I would like to thank all the professors of the cybersecurity master's degree for the knowledge transmitted, and family and friends for their motivation and support.

Thank you all!

Contents

List of Figures

List of Tables

List of Abbreviations

CDN Content Delivery Network

CLI Command Line Interface

CMDI command injection

CPU Central Processing Unit

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

DB DataBase

DDoS Distributed Denial of Service

DOM Document Object Model

DoS Denial of Service

FN False Negatives

FP False Positives

GCF Greenbone Community Feed

GSF Greenbone Security Feed

GUI Graphical User Interface

GVM Greenbone Vulnerability Manager

IP Internet Protocol address

LDAP Lightweight Directory Access Protocol

MCyber Master in Cybersecurity

NASL Nessus Attack Scripting Language

NVD National Vulnerability Database

NVT Network Vulnerability Tests

OS Operating System

OSP Open Scanner Protocol

OWASP Open Web Application Security Project

PME Pequenas e Médias Empresas

PoC Proof of Concept

PR privileges Required

RAM Random-access memory

RCE Remote code execution

SCAP Security Content Automation Protocol

SME Small and Medium-sized Enterprise

SQL Structured Query Language

SQLI SQL Injection

SSRF Server-Side Request Forgery

TI tecnologia da informação

TN True Negatives

TP True Positives

UI User Interaction

VMP Vulnerability Management Program

WAVSEP Web Application Vulnerability Scanner Evaluation Project

WN Wireless Network

XML Extensible Markup Language

XSS Cross-site scripting

XXE XML External Entity

ZAP Zed Attack Proxy

Chapter 1

Introduction

Hackers launch new and more sophisticated attacks every day exploiting the failures and vulnerabilities of computer networks and systems [60]. These vulnerabilities can be explored by attacks with different impacts on systems and information. Thus, security-related standards, regulations and recommendations are proposed to avoid these vulnerabilities. Even if a given system or service is secure-by-design, i.e. the devices or services have been designed to be secure, zero-day vulnerabilities can be found.

In order to detect vulnerabilities at early stages and, hopefully, before they are explored, security teams use vulnerability scanners. There are a plethora of vulnerability scanning tools available, each offering a unique combination of features and capabilities.

In this chapter, section 1.1 presents the context and motivation for this research work. Section 1.2 presents the objectives. Section 1.3 details the contribution made by this research work to the community. Finally, Section **??** explains the organization of the thesis.

1.1 Context and Motivation

Vulnerability scanning tools are automated tools that scan applications and networks, to identify security vulnerabilities such as improper data validation on outdated or non-patched software. A list of vulnerabilities can be found on the Open Web Application Security Project (OWASP) foundation website at [1].

[1] https://owasp.org/www-community/vulnerabilities/

Finding vulnerabilities is possible through two methods, automatic or manual vulnerability analysis. Both have their pros and cons and the most efficient is to be used together. Manual vulnerability is always important and should never be discarded as these professionals know the most common vulnerabilities/errors better than anyone else, but require human and financial resources, making difficult the adoption in Small and Medium-sized Enterprises (SMEs). As an alternative, open-source or free-to-use automated vulnerability scanning tools may be used by organisations to better improve their cybersecurity resilience. To reduce testing time and take advantage of the repetitive nature of testing, tools have been devised to automatically perform many of the same tasks that one does in manual penetration testing [10].

The detection efficiency of vulnerability scanning tools is heavily dependent on their vulnerability database. A large database will enable a more thorough detection. New vulnerabilities are discovered frequently, which means that these tools are only efficient if they maintain a steady pace of updates to their vulnerability databases. One would expect that, if such a tool is developed by a large company or organisation, its vulnerability database would also be a large one, it would see frequent updates and would be a single tool that would be sufficient for SMEs. It is assumed that SMEs will have a small in-house support team with only periodic availability to pursue vulnerability assessments.

A set of vulnerability scanners are already available and the list was recently updated. The Tsunami Security Scanner[2] is an open-source vulnerability scanner that was released on June 9, 2020, by Google. Despite being marked as a non-official product, it triggered the research work described herein. First, the release of this scanner triggered an assessment comparing the Tsunami Security Scanner to other vulnerability scanners already available. Second, it triggered the development of a Tsunami Security Scanner plugin.

1.2 Objectives

The purpose of this thesis is to analyze in detail how different scanners behave in different environments (Windows and Linux). For this, it is needed to analyze the vulnerabilities detected by each scanner and confirm if they exist. The objective is to collect the

[2]https://github.com/google/tsunami-security-scanner

usage of resources and develop a performance assessment regarding different vulnerabilities and systems. For this, it is necessary to mount the machines on an internal network and then install the chosen scanners on different machines so there is no interference in their performance. When the environments are prepared each scanner will be tested on all the vulnerable and non-vulnerable machines. To run the tests, three scanners should be installed on each machine and the percentage of RAM, CPU and packets per second will be analyzed. After everything is tested, is required to analyze the collected data and determine the level of each scanner for each operating system. The vulnerabilities will be tested for each scan to determine the accuracy and precision In the end, given the new Tsunami tool, contribute to this open tool by proposing a plugin for an existing vulnerability, found during testing, for the tsunami-security-scanner to understand how they work and understand the level of difficulty.

1.3 Contributions

This thesis is specially developed to understand the differences between vulnerability scanner tools in different operating systems and provides the following contributions:

1. A performance study on vulnerability analysis tools.
2. Assess how the Tsunami Security Scanner tool compares with similar tools.
3. A plugin for Tsunami Security Scanner to detect a specific vulnerability.

This thesis work resulted in a scientific publication presented at the IFIP SEC 2021 conference [7].

1.4 Organization

This thesis is organised into the following chapters and sections. Chapter 2 presents the background in the context of vulnerabilities and vulnerability tools. Chapter 3 presents the related work regarding vulnerability tools. Chapter 4 presents the assessment, including the scanners selection and the results and analysis obtained for each. Chapter 5 explains the internals of Tsunami and how to develop a plugin. Finally, chapter 6 concludes this work.

Chapter 2

Background

This chapter details the concepts of vulnerability in Section 2.1, the definition of Common Vulnerabilities and Exposures (CVE) 2.2 and how the Common Vulnerability Scoring System (CVSS) 2.3 is calculated using Base Score Metrics, Temporal Score Metrics and Environmental Score Metric. Next, explain the different types of Vulnerability scanning tools 2.4.

2.1 Security Vulnerabilities

A security vulnerability is a flaw in hardware or software that runs on the hardware that weakens the overall security of the device/system. New vulnerabilities are constantly discovered and a threat actor, such as an attacker, can take advantage of a given security vulnerability to cross privilege boundaries (i.e., carry out unauthorized actions) within a computer system. Vulnerabilities and attack surfaces are terms that can be used in this context.

There are different types of vulnerabilities. As collected in OWASP Top Ten 2021 [41] which is a periodically updated list of the most critical security risks to web applications [57] and there are set vulnerabilities that can be described as follows:

- Broken Access Control - When a regular user can access places that should be protected for users with elevated permissions
- Cryptographic Failures - Not using cryptography or using weak cryptography can lead to credential theft

- Injection - when a user sends data to the server but does not have any validation, filtering, or sanitization. This can lead to:
  - SQL Injection (SQLI) - allows the execution of Structured Query Language (SQL) commands in a given DataBase (DB)
  - Cross-site scripting (XSS) - allows the execution of javascript [28] on the server, which can be reflected, stored or Document Object Model (DOM)-based;
  - command injection (CMDI) - allows the execution of arbitrary commands on the host operating system;
- Insecure Design - is a broad category representing different weaknesses, expressed as missing or ineffective control design.
- Security Misconfiguration - Occurs when security settings are not adequately defined in the configuration process or maintained and deployed with default settings, for example:
  - XML External Entity (XXE), that an input containing a reference to an external entity and is processed by a weakly configured Extensible Markup Language (XML) parser;
  - Path Traversal - aims to access files and directories that are stored outside the web root folder;
- Vulnerable and Outdated Components - If the software is vulnerable, unsupported, or out of date.
- Software and Data Integrity Failures - This is related to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and Content Delivery Networks (CDNs).
- Security Logging and Monitoring Failures - Insufficient logging, detection, monitoring, and active response;
- Server-Side Request Forgery (SSRF) - This flaw occurs whenever a web application is fetching a remote resource without validating the user-supplied URL.

Security vulnerabilities can be indicated by a vulnerability value which denotes the severity of risk or loss because of the vulnerability. For instance, password files and Microsoft Word are used to store information on computer systems, but the vulnerability related to the password file typically has high severity due to the importance of the password [52]. In order to monitor and control the types and severity of vulnerabilities, the National Vulnerability Database (NVD) has been created. The NVD is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables the automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics. In order to identify and monitor various types of vulnerabilities in a service, it can be used both Mitre and NIST as reference sources, since they identify the associated CVE, check the affected versions and present the vulnerability details as both contain external references for the resolution or the exploit.

2.2 CVE

The CVE is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known cyber security issues, maintained by The Mitre Corporation. The system was officially launched for the public in September 1999. The goal of CVE is to make it easier to share data across separate vulnerability capabilities (tools, repositories, and services) with this "common enumeration" [11]. CVE does not include a solution, impact level, or vendor technical details because this information can already be found in numerous vulnerability sources such as NVD which help the security team provide solutions and other advisories for identifiers on the CVE List, as is explained by the authors [11].

CVE consists of CVE + Year + Arbitrary Digits, the arbitrary digits will begin at four fixed digits and expand with arbitrary digits only when needed in a calendar year, for example, CVE-YYYY-NNNN and if needed CVE-YYYY-NNNNN, CVE-YYYY-NNNNNN, and so on. CVE IDs are assigned to flaws that meet a specific set of criteria. They must be:

1. Independently fixable: The flaw can be fixed independently of any other bugs.

2. Acknowledged by the affected vendor OR documented: The software or hardware vendor acknowledges the bug and that it has a negative impact on security. Or, the reporter must have shared a vulnerability report that demonstrates the negative impact of the bug AND that it violates the security policy of the affected system.

3. Affecting one codebase: Flaws that impact more than one product get separate CVEs. In cases of shared libraries, protocols or standards, the flaw gets a single CVE only if there's no way to use the shared code without being vulnerable. Otherwise, each affected codebase or product gets a unique CVE.

There are several sites that allow checking information for a given CVE. An example of these is the site [51], if you search for the CVE associated with the vulnerability, it returns all the publications made, either by the software that suffers the vulnerability or by other services, such as Tenable or *redhatcve* or others. Exploit-DB [21] includes a database of many Proof of Concept (PoC), and it is possible to use these crypts to prove or exploit a vulnerability.

2.3 CVSS

CVE can also be applied to other applications like CVSS which provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity, as well as a textual representation of that score. CVSS has become the industry standard supported by most vendors. It solves the problem of chaos in the process of vulnerability evaluation, gives a concise vulnerability evaluation model, unifies the evaluation criteria and makes the majority of security information compatible [59].

CVSS is now on its third major version (v3.1), which was designed to address some of the shortcomings in its predecessor, v2. Most notably, version 3 introduces looks at the privileges required to exploit a vulnerability, as well as the ability for an attacker to propagate across systems ("scope") after exploiting a vulnerability. Updates to the CVSS version 3.1 specification include clarification of the definitions and explanation of existing base metrics such as Attack Vector, Privileges Required, Scope, and Security

Requirements. A new standard method of extending CVSS, called the CVSS Extensions Framework, was also defined, allowing a scoring provider to include additional metrics and metric groups while retaining the official Base, Temporal, and Environmental Metrics. The additional metrics allow industry sectors such as privacy, safety, automotive, healthcare, etc., to score factors that are outside the core CVSS standard.

The scores used in the different versions of CVSS are presented in Table 2.1. In version two are three base scores ranging from Low (0.0-3.9), Medium (4.0-6.9) and High (7.0-10.0). In the new version, there are five base score ranges None (0.0), Low (0.1-3.9), Medium (4.0-6.9), High (7.0-8.9) and Critical (9.0-10.0).

Table 2.1: CVSS Score

	CVSSv2	CVSSv3
Critical	N/D	9.0-10.0
High	7.0-10.0	7.0-8.9
Medium	4.0-6.9	4.0-6.9
Low	0.0-3-0	0.1-3.9
None	N/D	0.0

Another difference between CVSS version 2 and 3 is that the Confidentiality, Integrity, and Availability metrics changed to have scoring parameters of None, Low, or High. The Attack Vector metric added the Physical (P) value, which indicates a vulnerability where the adversary must have physical access to a system in order to exploit the vulnerability. A new metric, User Interaction (UI), was added. This metric indicates whether or not the cooperation of a legitimate user is needed to conduct an exploit. And also the privileges Required (PR) was added to indicate that administrative or other escalated privileges on the target machine must be achieved in order to successfully exploit the system. CVSS scores are used by the NVD, CERT and others to assess the impact of vulnerabilities. Many security vendors have created their own scoring systems, as well. A CVSS score is composed of three sets of metrics (Base, Temporal, Environmental), each of which has an underlying scoring component.

The Base metric group represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments. It is composed of two sets of metrics: the Exploitability metrics and the Impact metrics. The Exploitability metrics reflect the ease and technical means by which the vulnerability can be exploited. That is, they

represent characteristics of the thing that is vulnerable, which is referred to formally as the vulnerable component. On the other hand, the Impact metrics reflect the direct consequence of a successful exploit and represent the consequence to the thing that suffers the impact, which refers to formally as the impacted component. Figure 2.1 it is presented an example of Base Score Metrics. This example includes attack vector, attack complexity, privileges required, user interaction, and scope as the exploitability metrics, and confidentiality, integrity, and availability impact as the impact metrics.

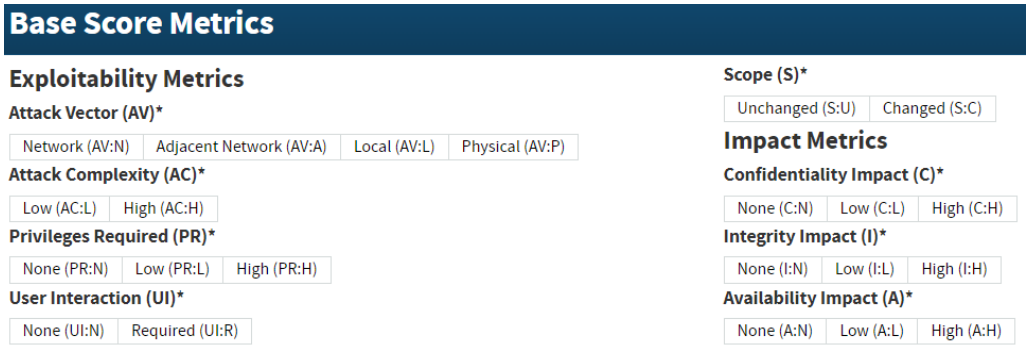


Figure 2.1: Base Score Metrics

The Temporal metrics measure the current state of exploit techniques or code availability, the existence of any patches or workarounds, or the confidence that one has in the description of a vulnerability. Temporal metrics will almost certainly change over time. Temporal Score Metrics details in Figure 2.2.

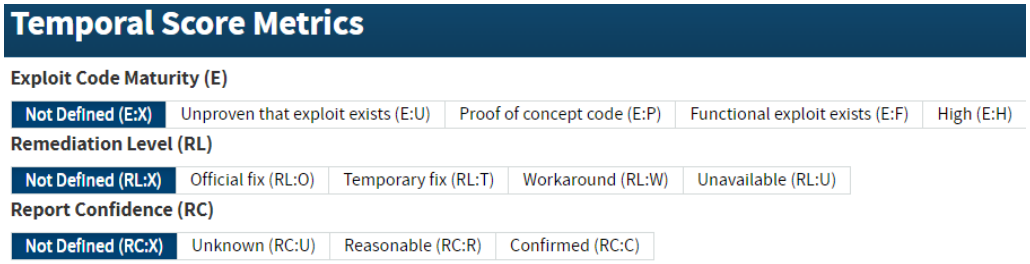


Figure 2.2: Temporal Score Metrics

The Environmental Score Metrics are dependent on the importance of the affected IT asset to a user's organization, and they are measured in terms of complementary/alternative security controls in place, Confidentiality, Integrity, and Availability. The metrics are the modified equivalent of base metrics and are assigned metrics values based on the component placement in the organization infrastructure. An example of Environmental

Score Metrics is presented in Figure 2.3.

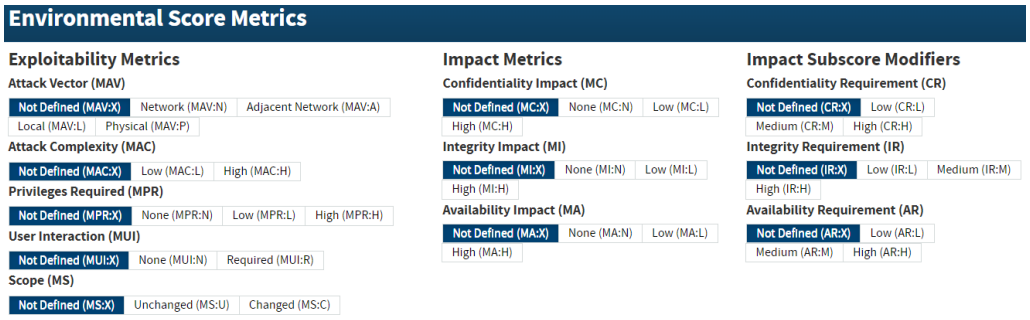


Figure 2.3: Environmental Score Metrics

The items in these metrics change according to the CVSS version. Table 2.2 presents an example with the details for the version CVSSv2 and CVSSv3.1, for a given CVE, CVE-2019-12815. For this CVE the CVSSv2 presents a score of 7.5 and the attack vector is:

AV:N/AC:L/Au:N/C:P/I:P/A:P

The CVSSv3.1 presents a score of 9.8 and the attack vector is:

AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 2.2: Severity and Metrics

CVSS v2.0	**CVSS v3.1**
Base Score: 7.5 HIGH	Base Score: 9.8 CRITICAL
Impact Subscore: 6.4	Impact Score: 5.9
Exploitability Subscore: 10.0	Exploitability Score: 3.9
Access Vector (AV): Network	Attack Vector (AV): Network
Access Complexity (AC): Low	Attack Complexity (AC): Low
Authentication (AU): None	Privileges Required (PR): None
Confidentiality (C): Partial	User Interaction (UI): None
Integrity (I): Partial	Scope (S): Unchanged
Availability (A): Partial	Confidentiality (C): High
	Integrity (I): High
	Availability (A): High

2.4 Vulnerability scanning tools

A Vulnerability Scanner is a standalone application or program using a Graphical User Interface (GUI) or a Command Line Interface (CLI) with procedures to detect vulnerabilities and exploits in a given machine that is being analysed. These procedures and their

effects depend on multiple factors such as Operating System (OS), installed programs, existing services, their versions and configurations. Thus, the scanners rely on signatures of known vulnerabilities and exploits and either maintain them in a local database that may be updated online or require a set of detection plugins or scripts that must be installed before scanning.

A vulnerability scanning tool scans a network or system for weaknesses and security vulnerabilities that could be exploited by a threat actor. By using automation, an organization can systematically strengthen its security posture by uncovering and addressing potentially threatening issues. The scanner has a database with information on vulnerabilities and the respective software. Scanners start by enumerating the ports and services, thus identifying the open ports and what is running on them. After knowing which services are running and which versions it will look for vulnerabilities and if the installed version is vulnerable, the scanner can run a simple script to validate if it is really vulnerable and thus reduce false positives. Vulnerability scanners can be categorized into 5 types based on the type of assets they scan.

1. Network-based scanners: Network-based vulnerability scanners identify possible network security attacks and vulnerable systems on wired or wireless networks. Network-based scanners discover unknown or unauthorized devices and systems on a network and help determine if there are unknown perimeter points on the network.

2. Host-based scanners: Host-based vulnerability scanners are used to locate and identify vulnerabilities in servers, workstations, or other network hosts, and provide greater visibility into the configuration settings and patch history of scanned systems.

3. Wireless scanners: Wireless vulnerability scanners are used to identify rogue access points and also validate that a company's network is securely configured.

4. Application scanners: Application vulnerability scanners can find misconfiguration of service as well as page errors that can lead to an exploit. They also analyze the algorithms used to encrypt the data.

5. Database scanners: Database vulnerability scanners can detect entry points into a

company's databases.

This thesis assesses the following set of network scanners: Openvas, Nessus, Nexpose and Tsunami Security Scanner. However, other scanners for different purposes, exist. In the case it is required a Web scanner, scanners such as OWASP Zed Attack Proxy (ZAP) [42], BurpSuit [13], Nikto[53] and WPScan[64] for WordPress can be used. Some of these such as BurpSuit and ZAP operate as a web proxy server between the browser and target applications and lets you intercept, inspect, and modify the raw traffic passing in both directions.

2.4.1 OpenVAS

OpenVAS as known or more recent Greenbone Vulnerability Manager (GVM) is a widely used vulnerability scanner, open source and distributed by Greenbone Networks. The GVM is the central service that consolidates plain vulnerability scanning into a full vulnerability management solution. GVM controls the OpenVAS Scanner via Open Scanner Protocol (OSP). The service itself offers the XML-based, stateless Greenbone Management Protocol. GVM also controls an SQL database (PostgreSQL) where all configuration and scan result data is centrally stored, also handles user management including permissions control with groups and roles and the service has an internal runtime system for scheduled tasks and other events. The main scanner OpenVAS Scanner is a full-featured scan engine that executes vulnerability tests against target systems.

OpenVAS has grown a broad community of security experts and when is flag a false positive to the OpenVAS mailing list, the feedback is usually prompt and knowledgeable. The OpenVAS scanner uses regularly updated feeds. Feeds may include the commercial Greenbone Security Feed (GSF) or the free Greenbone Community Feed (GCF). The GSF is a paid service utilizing updates from security experts worldwide. Updates are delivered automatically via the Greenbone Security Manager and the Greenbone Cloud Services. These feeds form a stream of small procedures that the scanner uses to check all the devices in your network for known and potential security problems. Its capabilities include unauthenticated and authenticated testing, various high-level and low-level internet and industrial protocols, performance tuning for large-scale scans and a powerful internal

programming language to implement any type of vulnerability test [40]. The OpenVAS Scanner works with Network Vulnerability Testss (NVTs) and is mostly implemented in the programming language Nessus Attack Scripting Language (NASL). A set of NVTs are wrappers for external tools. As new vulnerabilities are published every day, new NVTs appear in the Greenbone Security Feed. This feed is commercial and requires a respective subscription key. In case no subscription key is present, the update synchronisation will use the Community Feed instead. The script *greenbone-nvt-sync* will fetch all new and updated security checks and install them at the proper location. Once this is done it will send a signal to the OpenVAS Scanner, *openvassd* so that the new NVTs are loaded and considered for new security scans.

2.4.2 Nessus

Nessus is a proprietary vulnerability scanner developed by Tenable, Inc. The tool is free for non-enterprise use, however, for enterprise consumption, there are options that are priced differently. The company produces updates for new vulnerabilities within 24 hours of a new vulnerability's release. Tenable Research designs programs to detect vulnerabilities. These programs are named plugins and are written in the NASL. The plugins contain vulnerability information, a simplified set of remediation actions and the algorithm to test for the presence of the security issue. Nessus identifies the vulnerabilities that need attention with high-speed, accurate scanning. According to the G2 company Website[1], Nessus is the leader of the Vulnerability Scanners category with the highest score, where this score is based on reviews gathered from their user community, as well as data aggregated from online sources and social networks. According to G2, "a unique algorithm is applied to this data to calculate the satisfaction and Market Presence scores in real time". It is also possible to use Nessus scripting language to be able to write tests to a specific system [54]. Examples of this family of plugins for NESSUS are AIX Local Security Checks, Alma Linux Local Security Checks, Backdoors, CISCO, Databases, Denial of Service, FTP, Firewalls, Mobile Devices, SNMP, Web Servers, Windows. In this latest Windows plugin family, it is possible to see that it contains 5518 plugins sorted by date. In this case, the first one

[1] https://www.g2.com/categories/vulnerability-scanner

is Remote Desktop client for Windows Multiple Vulnerabilities (May 2022) [2] published on 05/10/2022 and it is possible to see the vulnerability description for which this plugin was created, as well as the possible solution and respective references. Tenable Research has published 171225 plugins, covering 69124 CVE IDs and 30940 Bugtraq IDs. On the OpenVAS website [39] it is possible to verify that OpenVAS has more than 100000 feeds for vulnerabilities.

2.4.3 Nexpose

Nexpose is a vulnerability scanner which aims to support the entire vulnerability management lifecycle, including discovery, detection, verification, risk classification, impact analysis, reporting and mitigation. It integrates with Rapid7's Metasploit for vulnerability exploitation. It is sold as standalone software, an appliance, a virtual machine, or as a managed service or private cloud deployment. Nexpose allows the creation of asset groups based on divvying up remediation duties and uses those groups to create remediation reports for the teams responsible for those assets. It uses Nmap [37] to perform basic TCP port scanning and runs additional scanner modules to gather more information about the target hosts. Nexpose has a special feature known as Live monitoring, which collects the available data and then converts that data into action plans. Nexpose has various editions with different deployment options [49]. Nexpose uses Metasploit which is possible to write, test, and execute exploit code. The Metasploit Framework contains a suite of tools that can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. At its core, the Metasploit Framework is a collection of commonly used tools that provide a complete environment

2.4.4 Tsunami Security Scanner

Tsunami Security Scanner has been made available on GitHub as version 0.0.1 on June 9, 2020, as open source, is a general-purpose network security scanner with an extensible plugin system for detecting high-severity vulnerabilities with high confidence. This scanner relies on a plugin system to provide basic scanning capabilities. According to [23], the Tsunami Security Scanner is announced to:

[2] https://www.tenable.com/plugins/nessus/160941

- support a small manually curated set of vulnerabilities
- detect high severity, RCE-like vulnerabilities, which are often actively exploited in the wild
- generate scan results with high confidence and minimal false-positive rate
- implement detectors that are easy to implement
- be scalable, be executed fast and perform non-intrusive scans

Tsunami Security Scanner also uses a set of tools known as Nmap which is used for network discovery, and Ncrack [35] which is a high-speed network authentication cracking tool, both tools required to be pre-installed before running the scanner. Regarding the plugins developed for the Tsunami Security Scanner, there are several repositories, including the google repository [3], which contains plugins published by google, which include detectors, web fingerprinters and portscan through Nmap. The last commit for the detectors folder is for CVE-2017-7615 which checks if a MantisBT application [32] is vulnerable to arbitrary password reset and unauthenticated admin access. There is also the community repository [4], where the last plugin created was for CVE-2022-1388 for the vulnerability in the equipment of F5 BIG-IP [12] that may bypass iControl REST authentication. The community has started developing plugins since the public release of the Tsunami Security Scanner, at the date of 28 November 2021, a set of plugins were available and ready to be used to scan for the following CVEs:

- CVE-2020-3452: Web services interface of Cisco Adaptive Security Appliance;
- CVE-2020-17519: Apache Flink 1.11.0 to 1.11.2;
- CVE-2021-25646: Apache Druid 0.20.0 and earlier
- CVE-2021-41773: Apache HTTP Server 2.4.49
- CVE-2021-22205: GitLab CE/EE 11.9
- CVE-2021-3129: Ignition before 2.5.2, as used in Laravel before 8.4.2

[3] https://github.com/google/tsunami-security-scanner-plugins/tree/master/google
[4] https://github.com/google/tsunami-security-scanner-plugins/tree/master/community

- CVE-2021-29441: Nacos before version 1.4.1
- CVE-2017-7615: MantisBT through 2.3.0
- CVE-2021-35464: ForgeRock AM server before 7.0

According to [55], the fingerprinter web service system, that identifies software and versions, has also been improved and it is now possible to detect around 21 applications depending on their version, such as the Gitlab version 10.0.0 [24] until 13.4.1 [25], Jenkins from version 1.359 until 2.251 [14], phpMyAdmin version 4.5.3.1 until 5.0.4 [45] and a few more. The authors in [56] provide a discussion about future work on developing a "Dynamic Scanning Orchestra" that allows users to put one or several plugins and this one will not need to be recompiled. The authors suggest compiling an execution graph when the scanner starts, based on the input/output data dependencies across all installed plugins.

2.4.5 Other Vulnerability Scanners

Other vulnerability scanners are also available, either for web-based or network-based contexts. These tools offer alternative methods for identifying security weaknesses in systems and applications. Regarding these contexts, the following vulnerability scanners can be depicted:

- Qualys Vulnerability Management [48]
- AT&T Cybersecurity [9]
- Alibaba Cloud Managed Security Service [31]
- Nikto [53]
- W3AF [58]
- Arachni [6]
- Acunetix [2]

These vulnerability scanners were not considered for the current research since most are web application scans or not as popular as selected ones.

Chapter 3

Related work

There are research works that focus on comparing tools that evaluate a specific type of vulnerability, such as web application scanning tools. Authors in [5, 19, 18] perform different tests to evaluate different Web Vulnerability Scanners. The authors in [18] used the following vulnerability scanners, Nessus, Acunetix Vulnerability Scanner and OWASP ZAP, for the tests they used two different projects called Project A and Project B. Project A with the Centos operating system and for Project B Ubuntu, both with the apache web server. Finally, the authors mention that different scanners detect different vulnerabilities and that Acunetix is the best vulnerability scanning tool for the web application and Nessus is a good scanner for network scanning.

In [5] the authors describe tests done on two web applications WebGoat and Damn. Eight vulnerability scanners were used for these tests; HP WebInspect; IBM AppScan; OWASP ZAP; Skipfish; Arachni; Vegas; and Iron WASP. The authors recommended improving the vulnerability detection capabilities of both the open-source and commercial scanners to enhance code coverage and the detection rate, and to reduce the number of false positives.

Authors in [19] present an evaluation of eleven black box web vulnerability scanners, Acunetix, AppScan, Burp, Grendel-Scan, Hailstorm, Milescan, N-Stalker, NTOSpider, Paros, w3af, Webinspect, both commercial and open sources. they demonstrate that many classes of vulnerabilities are completely overlooked by these tools, and thus research is required to improve the automated detection of these flaws.

In [33], the authors compare the performances of Arachni and OWASPZAP, open-

source web vulnerability scanners that compare the results from the OWASP benchmark and Web Application Vulnerability Scanner Evaluation Project (WAVSEP) benchmark. They concluded that the ZAP scanner performed better than the Arachni scanner for the SQLI, XSS and CMDI categories. Arachni scanner, on the other hand, performed better in the Lightweight Directory Access Protocol (LDAP) category.

The research in [22] focuses on the vulnerabilities of SQL injection and Cross-Site. Three anonymous scanning tools, non-disclosed allegedly to assure neutrality and to respect their commercial licenses, were evaluated. The results showed that overall coverage of vulnerabilities is low and the percentage of false positives is very high.

In [29] the authors developed a modular web vulnerability scanner and to verify the accuracy of SecuBat, they will select one hundred interesting sites from the list of potential victims for further analysis and confirm exploitable flaws in the identified pages. They hope to have provided a tool available to website administrators and web developers to proactively audit the security of their applications.

In [4] the authors have tested the MySQLlInjector web scanning tool with enhanced features that can perform efficient penetration tests on PHP-based websites to detect SQL injection vulnerabilities. The authors conclude that the tools have a combination of attacking patterns, vectors and modes to help web developers run various types of tests.

In [62], the authors compare different analysis tools such as Tenable Security Center 3.0, Skybox Secure and Amenaza SecurITree. The selected tools are compared in terms of the correlated analysis they provide one by one and placed in separate subsections. The authors mention that Skybox's Secure and Amenaza's SecurITree both correlate vulnerabilities with each other by creating attack paths or scenarios as they call them. SecurITree's scenarios are from the viewpoint of the attackers while the scenarios from Secure are from the organization's viewpoint. The problem with Skybox Secure and SecurITree is that scalability is quite low, so they cannot be used in large networks. The Tenable tool correlates vulnerabilities in a completely different way. It shows correlated vulnerabilities with IDS alerts in order to see which vulnerabilities an organization is exposed to and which are actually being exploited at a certain moment in time. They conclude that standards are well supported and compliance seems to be a major selling point. However, only three tools supported correlated analysis. And the comparison shows that scalability and the

amount of manual input required are the biggest concerns at this moment for supporting tools correlated analysis. While valid research at the time, it is now mostly outdated. Some of the referenced tools are no longer available.

In [27], the authors also present a large quantitative comparison of vulnerability scanning tools(AVDS, Patchlink scan, Nessus, NeXpose, QualysGuard, SAINT and McAfee VM). They created an environment with 20 physical servers running a total of 28 virtual machines, divided into four VLAN segments. Various operating systems and versions, e.g. Windows XP SP2, Debian 5.0 and Windows Server 2003 SP1. Each host had several different network services HTTP, HTTPS, SMTP, FTP, Streaming Media Server, RDP, SSH, SMB and VNC. Their focus was on the direct output of the tools or, in other words, the number of vulnerabilities these tools identify. They focused on functionality and on accuracy. This work differs by focusing on tools to be usable by SMEs. Moreover, newer tools were launched, not available at the time, some being open-source and free-to-use tools. Additionally, they did not perform resource usage comparisons. They conclude that some tools are better at detecting vulnerabilities in Windows systems, and others at detecting vulnerabilities in Linux systems.

In [61], the authors compare the following set of network vulnerability scanning tools: Nessus, Nmap, Open VAS, Retina CS Community, Microsoft Baseline Security Analyzer, and Nexpose Community Edition. These scanners are analyzed and discussed network vulnerability scanning in a hands-on laboratory class with students. The laboratory featured three virtual machines each one with a different OS installed: BT5, Windows XP, and Kali Linux. The authors conclude that the feedback from the students was 90% positive regarding the execution of the tools and that the students had problems mainly with the installation of OpenVAS.

Authors in [57] explain the basics of vulnerability scanning and its advantages(automation, speed, cost-effectiveness, scalability, compliance, accuracy). Also, they describe how it integrates with Vulnerability Management Program (VMP), helping to choose an appropriate type of Vulnerability Scanner, and understand when and how to employ vulnerability scanning to identify assets and choose which assets to scan and when.

In a more recent work [30], the authors presented a performance-based comparison between two tools: Nessus and Retina. The authors start by comparing the graphical

environment of each tool. It was concluded that Nessus and Retina have almost the same vulnerability detection ability, and Nessus has a small advantage since it includes a web mirroring tool that is very helpful on the web, that can extract the website and analyze it locally. In terms of scanning time, Nessus performed faster (approximately 6 times) than Retina. But if the scanner runs with a Web application module, Nessus performs much slower than Retina. The main comparison used in this article was the ability to search, Scanning Time, and ability to detect vulnerabilities. They conclude that both scanners performed very well in vulnerability identification, in terms of speed without an active Web Application feature, Nessus performed faster than Retina, on the other hand, with an active Web Application module, Nessus performs much slower than Retina. Of the two, Retina has now been discontinued by its developer, which issued a notification stating its end of life by December 31, 2019. The goal is to perform a similar study on Kushe but be a more up-to-date, more complete and more thorough one.

Similar work was presented in [15], where the authors opted for comparing a dedicated, hardware-based commercial tool against an open-source, free-to-use, software-based tool. While they conclude that the commercial solution is faster at presenting results, they did not assess the efficiency of their findings. Moreover, it was a small comparison of just two tools.

In [26], the author questioned the performance of vulnerability scanning tools as a method to remedy the security issues these tools identify. The author concludes that manual effort will always be needed to reach complete accuracy and the remediation guidelines outputted by the tools is very cumbersome to address.

This thesis is focused on the use of larger spectrum vulnerability scanning tools as these would require less time and resources to implement, while still being able to detect web application vulnerabilities. Additionally, studies comparing vulnerability scanning tools have not been updated recently.

Chapter 4

Assessment of Vulnerability Scanners

This chapter introduces the methodology for the Assessment of the Vulnerability Scanners defined and for each section each stage of the methodology is detailed. It starts by presenting a set of research questions that are achieved in developing the work and answered at the end. In Section 4.1 and 4.2 the selected tools and the environment deployed for testing are explained as well as the tests performed. Finally, the data collected with the respective results are presented in Section 4.3.

The methodology adopted for the assessment of vulnerability scanners is presented in Figure 4.1. In the first stage, it will be necessary to formulate research questions. In the second stage, technical information about the vulnerability scanners is gathered. In the third step, evaluation tests are executed. In the last stage, the results are collected and analyzed taking into account the data from the tests.

The following three research questions were designed to follow this assessment:

- Q1: What is the most efficient, free-to-use, vulnerability scanning tool currently available?
- Q2: How does Tsunami Security Scanner compare to similar tools currently available?
- Q3: Is Tsunami Security Scanner well suited to be used by SMEs?

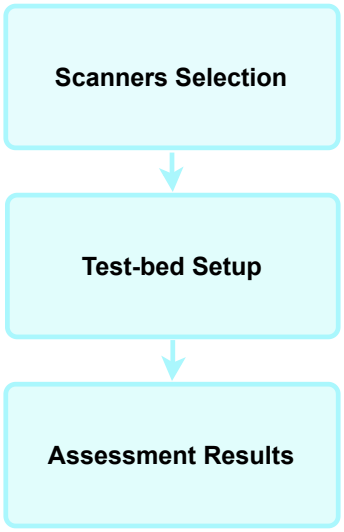


Figure 4.1: Methodology adopted

These research questions are answered after the assessment is completed.

4.1 Scanners Selection

In this stage, and given the research questions Q1 and Q2, the selected set of Vulnerability Scanners to analyse was narrowed to the following: OpenVAS [3], Nessus [54], Nexpose [49], and Tsunami Security Scanner [16]. To answer these three questions, it was necessary to collect the technical features of this set of vulnerability scanners.

Then, the technical features of these vulnerability scanners were collected. Table 4.1 presents the selected vulnerability scanners and their main properties regarding their license, the availability of their source code, their mode of operation and the update process of their vulnerability lists. Also, each vulnerability scanner contains local and online databases, the difference is represented in Figure 4.2.

All selected vulnerability scanners are free-to-use. Nessus Essentials is free for personal use but limits scanning to 16 different IPs. Nexpose offers a 1-year trial, after which turns into a paid tool. Nexpose was included in the current analysis to detect if there is a significant difference between free-to-use and paid scanners. OpenVAS and Tsunami Security Scanner provide their versions as open source. OpenVAS, Nessus and Nexpose use a GUI while the Tsunami Security Scanner operates in the CLI environment. Regarding the update process, Nessus and Nexpose have a local database with the vulnerabilities

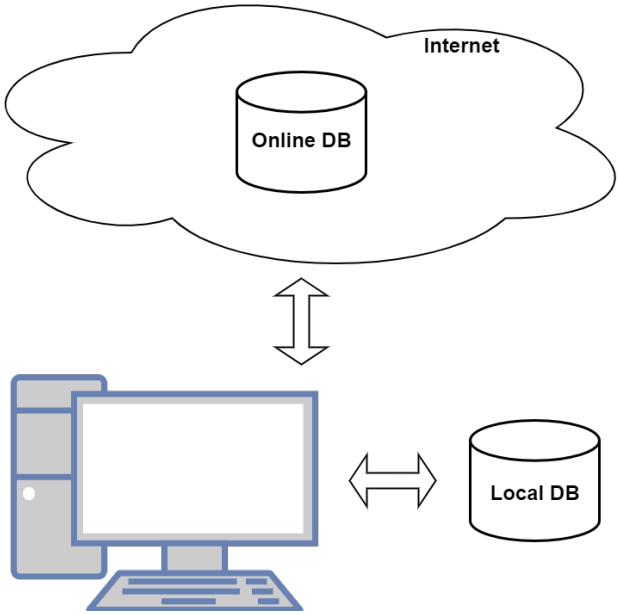


Figure 4.2: Database online vs local

signatures that are updated online, while Tsunami Security Scanner uses detection plugins.

Table 4.1: Selected Vulnerability Scanners

	License and source code			Operation		Vulnerabilities list update		
	Free to use	Trial period	Open source	GUI	CLI	Local DB	Online DB	Plugins or scripts
OpenVAS	x		x	x		x	x	x
Nessus	(x)			x		x	x	x
Nexpose	(x)	x		x		x	x	x
Tsunami Security Scanner	x		x		x			x

The output of the selected vulnerability scanners is a PDF file with a non-standard organisation containing a set of potential vulnerabilities identified by a CVE identification. The list of these vulnerabilities and their CVE ID is maintained publicly in [17]. Each CVE record comprises the identification number, a description, and at least one public reference. These CVE records are sent to NVD that extends their classification with additional information, severity scores and impact ratings. The severity scores are expressed by CVSS, an open framework used for communicating the characteristics and severity of vulnerabilities. The score is obtained by using three metric groups: Base, Temporal, and Environmental. The Base metrics produce a score ranging from 0 to 10, which can then be modified by the scoring of the Temporal and Environmental metrics.

4.2 Test-bed Setup

A test bed was set up in order to test and evaluate all the vulnerability scanning tools identified in Section 4.1. The test-bed topology is depicted in Figure 4.3 and comprises multiple virtual machines hosted on a laptop with an Intel core i7-4710HQ CPU @ 2.50GHz processor, 12 GB of RAM, a 256GB SSD, running the Windows 10 64bit OS 4.2.

CPU	Intel core i7-4710HQ 2.50GHz
RAM	12GB
SSD	256GB
OS	Windows 10 64bit

The use of virtualisation was selected in order to produce comparable results. VirtualBox 6.1 was the adopted virtualisation solution. Five virtual machines were deployed, one to act as the scanner, and the remaining four to act as targets. Kali Linux 2020.4 was selected due to the simple installation process of the required tools for scanning. In order to minimise the impacts of the installation of the tools, after the initial setup of the Kali Linux OS, a snapshot was taken and all tools were installed over that initial snapshot. This was made to maintain the same exact configuration on the system, prior to each tool installation. While the tests were executed, the target virtual hosts were disconnected from the Internet.

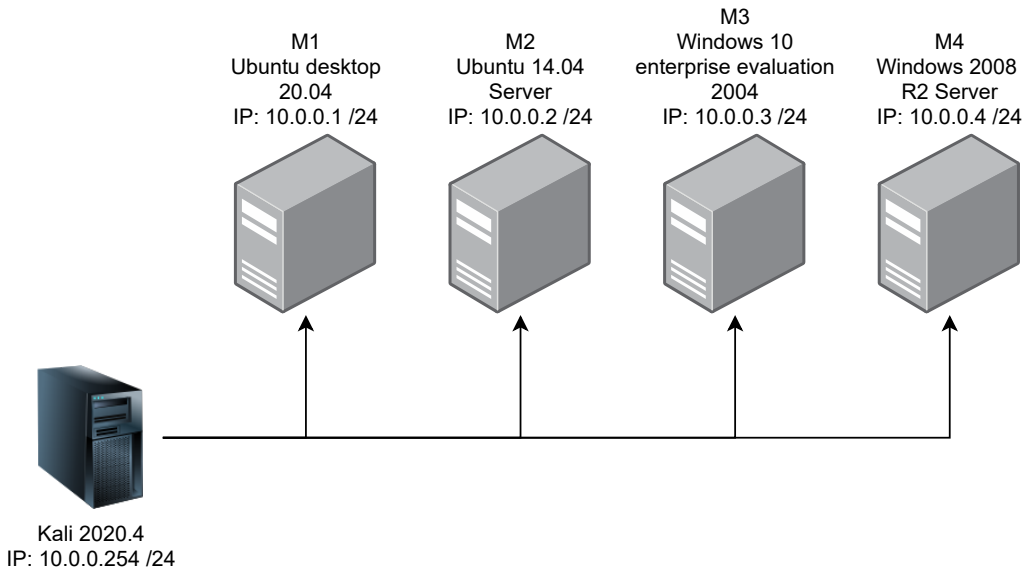


Figure 4.3: Test-bed topology

The targets were selected in order to be as diverse as possible. Of the four virtual machines, two were Linux-based and two were Windows-based. Each set of two machines per platform was selected to represent a client version and a server version of each platform. As an example of a server was installed the Ubuntu Server 14.04 with the Internet Protocol address (IP) 10.0.0.2/24 (*M2*) and Windows 2008 R2 Server with the IP 10.0.0.4/24 (*M4*), as an example of machines used by users, it was installed Ubuntu Desktop 20.04 with the IP 10.0.0.1/24 (*M1*) and Windows 10 Enterprise with the IP 10.0.0.3/24 (*M3*), finally, the machine that runs the scanner will have the IP 10.0.0.154/24. Android targets were also considered at the beginning but, because the first tests showed that, due to strict firewall configurations, no results were reported by the selected tools, Android targets were dropped. Thus, the set of target virtual machines comprised:

- a Ubuntu Desktop 20.04 (as *M1*);
- a Ubuntu Server 14.04 (as *M2*);
- a Windows 10 Enterprise (as *M3*);
- a Windows 2008 R2 Server (as *M4*).

The machines *M1* and *M3* targets machines are standard installations of the respective OS, whereas *M2* and *M4* were deployed using metasploitable, version 3 [1]. Metasploitable virtual machines are machines specifically configured with software that includes sets of vulnerabilities to be tested. To assess the detection capabilities of the scanning tools, the later virtual machines were considered the most relevant. These were tested without modifications or configurations, aside from disabling the MySQL server of *M2* and enabling ping replies on *M4*. MySQL was disabled because of the excessive time taken by the Tsunami Security Scanner password brute-forcing with Ncrack. Ping was enabled to ease the use of scanning tools that first checked the target's liveliness with a ping request.

To run the tests a bash script was developed in order to monitor and record the execution time (duration), RAM memory and CPU usage on the Kali virtual machine, in a systematic way. For the network usage monitoring, on the same Kali virtual machine, and whenever a vulnerability scan was started, the **tcpdump** command was executed with

[1] https://github.com/rapid7/metasploitable3

arguments to identify the packets sent by the Kali virtual machine to the target machine, i.e. using 10.0.0.154 as the source IP and 10.0.0.X as the destination IP, where the "X" is the IP address of each target presented in Figure 4.3.

In accordance with the topology depicted in Figure 4.3, four vulnerability scanning exercises were conducted on each of the four target systems, utilizing each of the four vulnerability scanners. A total of 64 vulnerability scans were carried out, and the average and standard deviation values were determined for each scanner.

4.3 Assessment Results

The initial results showed that Tsunami Security Scanner was the fastest, using the least resources. At the time of these tests, it was concluded that the Tsunami Security Scanner does not contain enough vulnerability detection plugins and because of this it detects almost no vulnerabilities and requires low resources to do so. For this reason, the Tsunami Security Scanner was not included in the results of this section.

Figure 4.4 shows the average duration of the performed scan tasks. From these results, it can be highlighted that the standard targets (*M1* and *M3*) are scanned fastest by all tools due to having the least vulnerabilities and the least services available through the network. On the other hand, *M2* and *M4*, being metasploitable-based targets, took the most time to scan. It can be observed that of the three shown, Nessus was the fastest and OpenVAS was the slowest. OpenVAS took almost 5 times more to scan *M4* when compared to the other tools.

Figure 4.5 a) shows the average network usage in terms of the number of packets, per second, sent by the Kali machine to the target machines. All vulnerability scanning tools report more network usage when scanning the *M2* target, which runs a metasploitable Ubuntu Server. This is expected as this target is the one with the most services available through the network. When comparing tools, OpenVAS is the tool that uses more network resources, followed by the Nessus tool.

Figure 4.5 b) shows the average CPU used by the Kali machine during the execution of the different tools. The tool that uses the most CPU is OpenVAS. This was expected as this tool also used more network resources and took the most time to complete. Nonetheless,

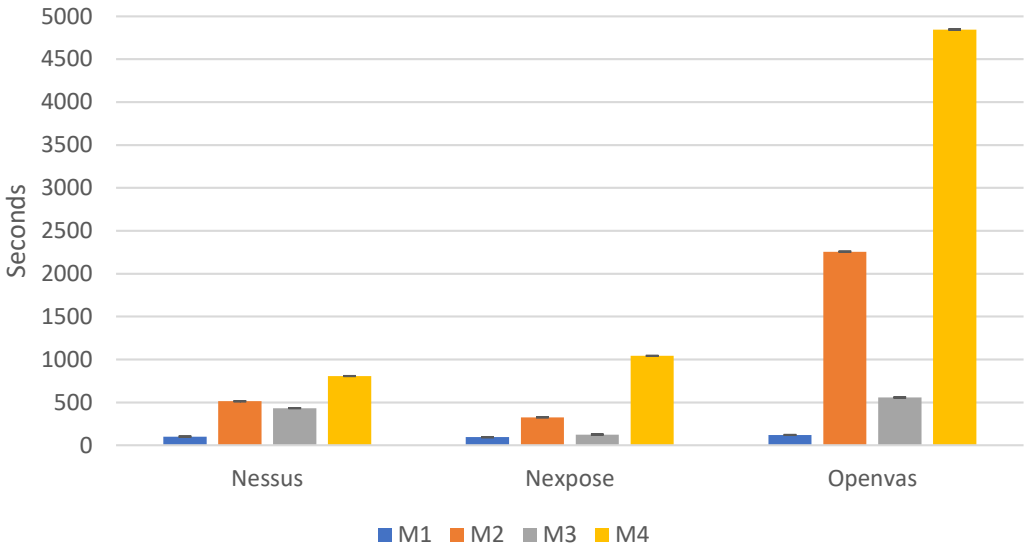


Figure 4.4: Scan duration in seconds

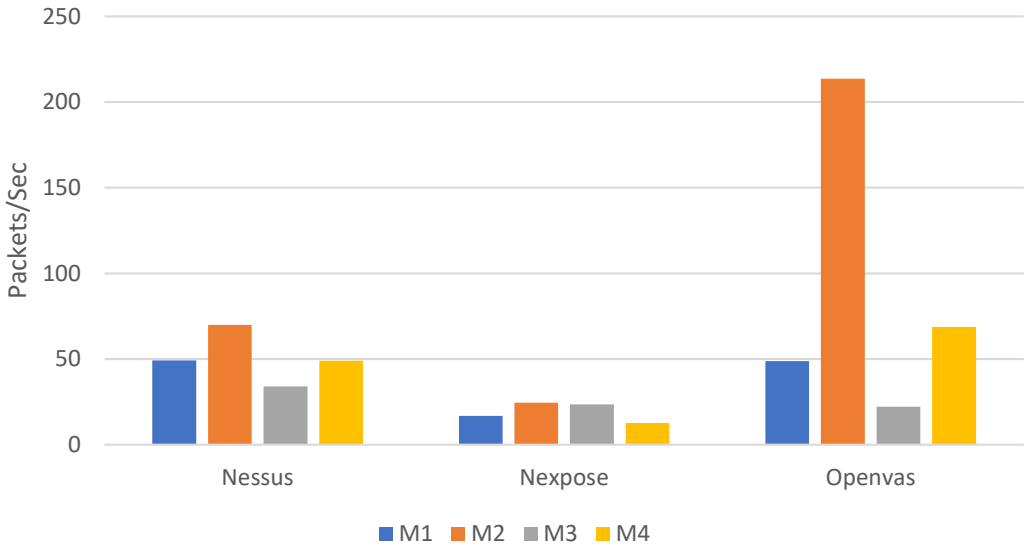
the overall CPU usage of all tools is very low, maxing below 3,5%. Worthy of note, and because of so low average values, is the fact that this result was the one that has shown the greater standard deviation.

Figure 4.5 c) shows the average memory used by the Kali machine during the execution of the different tools. One conclusion that can be made is that all tools use a similar amount of memory independently of the scanned target. The tool that requires the most amount of memory is Nexpose, using almost four times the memory needed by the other two tools.

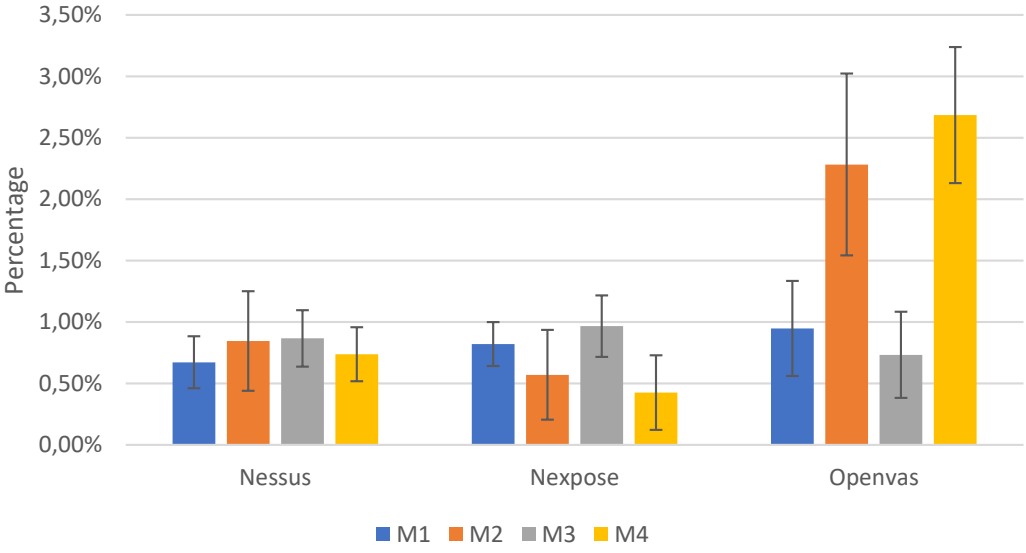
Table 4.2: Vulnerability identification results for M2

Vulnerability	CVSS	Nessus	Nexpose	OpenVAS
CVE-2010-1574	10 (v2)		FP	
CVE-2015-3306	10 (v2)	TP		TP
CVE-2015-5377	9.8			FP
CVE-2017-3167	9.8	FP	FP	
CVE-2017-3169	9.8		FP	
CVE-2017-7679	9.8		FP	
CVE-2018-1312	9.8		TP	
CVE-2018-5337	9.8			FP
CVE-2018-5341	9.8			FP
CVE-2019-12815	9.8		TP	
CVE-2017-9788	9.1		FP	
CVE-2016-5387	8.1		TP	
CVE-2017-15715	8.1		TP	

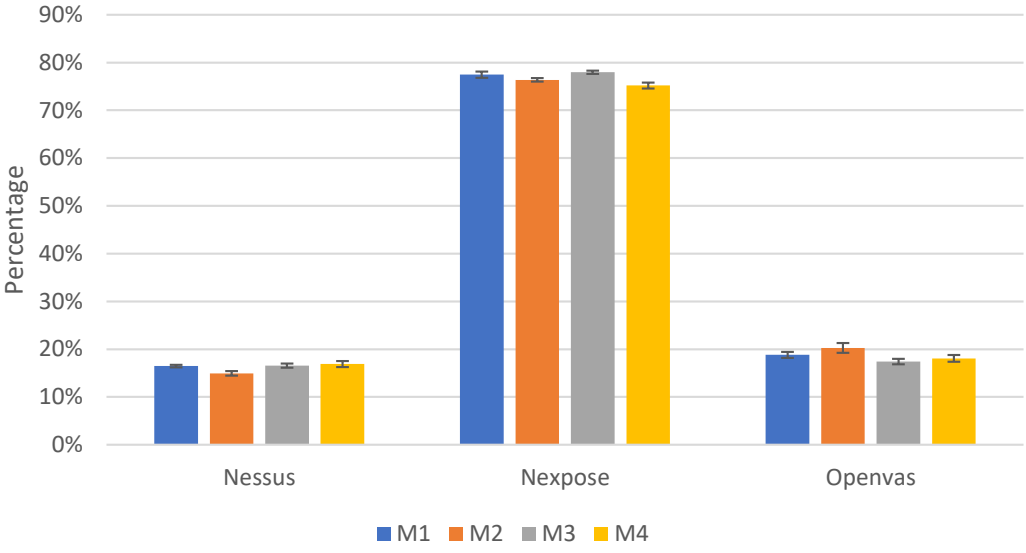
Tables 4.2 and 4.3 show the vulnerability identification results achieved by the different tools. In these tables, only vulnerabilities with an assigned CVE identification were



(a) Network usage in packets per second



(b) CPU usage in percentage



(c) RAM usage in percentage

Figure 4.5: Network, CPU and RAM usage

considered. Vulnerabilities with an assigned CVE identification are published online and known by all vulnerability scanner tools, thus becoming a ground truth to which results of other tools can be compared. A list comprising all vulnerabilities reported by all tools, separated by target (*M2* and *M4*), was compiled. The real presence of each vulnerability was then manually confirmed. In order to avoid such manual verification to become cumbersome, the full list of vulnerabilities was reduced to the ones that presented a score above 7.5, based on CVSS in version 3, plus the ones that presented a maximum score of 10, independently of the CVSS version. Then a description of the vulnerable True Positives (TP) found with a score equal to 10 is made for better knowledge.

- CVE-2015-3306: ProFTPD 1.3.5 allows remote attackers to read and write to arbitrary files.
- CVE-2010-0219: Apache Axis2, as used in *dswsbobje.war* in SAP BusinessObjects Enterprise XI 3.2 [50], CA ARCserve D2D r15 [8], and other products, has a default password of axis2 for the admin account, which makes it easier for remote attackers to execute arbitrary code.
- CVE-2012-2688: Unspecified vulnerability in the *_php_stream_scandir* function in the stream implementation in PHP before 5.3.15 and 5.4.x before 5.4.5 [43] has unknown impact and remote attack vectors, related to an overflow.
- CVE-2015-1635: HTTP.sys in Microsoft Windows 7 SP1, Windows Server 2008 R2 SP1, Windows 8, Windows 8.1, and Windows Server 2012 Gold and R2 [34], allow remote attackers to execute arbitrary code via crafted HTTP requests.
- CVE-2017-7213: Zoho ManageEngine Desktop Central before build 100082 allows remote attackers to obtain control over all connected active desktops via unspecified vectors.

In order to evaluate the performance of the set of vulnerability scanners, both the accuracy and precision of the detected vulnerabilities were analysed. For this specific analysis, only *M2* and *M4* results were considered since these were the ones that had multiple identifiable vulnerabilities. *M1* and *M3* are standard, recent and fully updated installations of Ubuntu and Windows 10, respectively.

Table 4.3: Vulnerability identification results for M4

Vulnerability	CVSS	Nessus	Nexpose	OpenVAS
CVE-2010-0219	10 (v2)			TP
CVE-2010-1574	10 (v2)		FP	
CVE-2012-2688	10 (v2)	TP		
CVE-2015-1635	10 (v2)		TP	TP
CVE-2017-7213	10 (v2)			TP
CVE-2015-5377	9.8			FP
CVE-2015-8249	9.8			TP
CVE-2017-11346	9.8			TP
CVE-2017-3167	9.8	FP	FP	
CVE-2017-3169	9.8	TP		
CVE-2017-7668	9.8	TP		
CVE-2017-7679	9.8	TP		
CVE-2018-5337	9.8			FP
CVE-2018-5338	9.8			TP
CVE-2018-5339	9.8			TP
CVE-2018-5341	9.8			FP
CVE-2020-10189	9.8	TP		
CVE-2017-5648	9.1			TP
CVE-2017-9788	9.1	TP		
CVE-2016-10012	7.8			TP

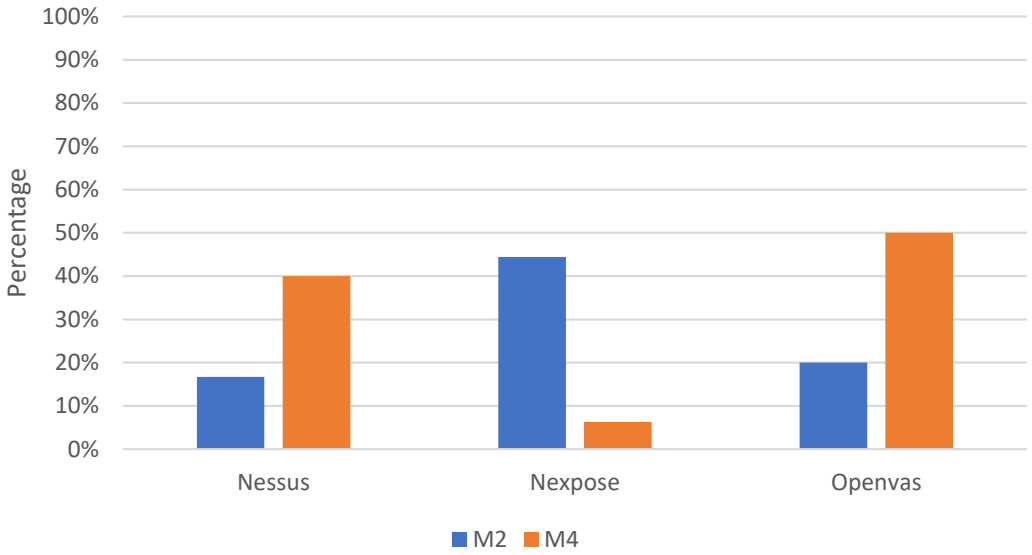
Equations 4.1 and 4.2 were used to calculate accuracy and precision, respectively. These equations consider the number of TP, False Positives (FP) and False Negatives (FN). TP being the number of vulnerabilities identified that are really present in the target. FP being the number of vulnerabilities identified that are not present in the target. FN being the number of vulnerabilities that are present in the target but not identified.

$$\text{Accuracy } (\%) = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} (\times 100) \tag{4.1}$$

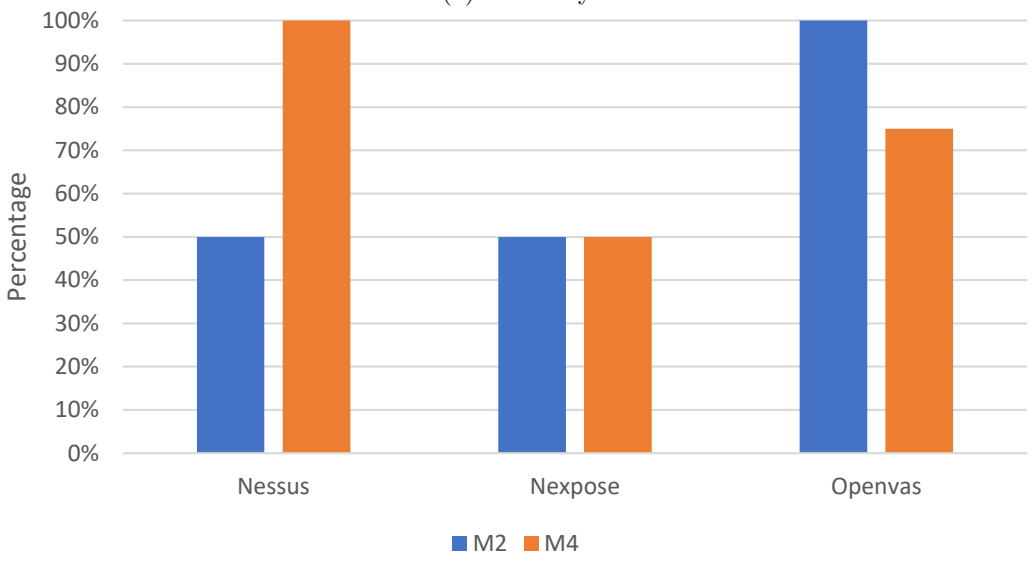
$$\text{Precision } (\%) = \frac{\text{TP}}{\text{TP} + \text{FP}} (\times 100) \tag{4.2}$$

Equation 4.1 evaluates if a tool is capable of detecting all available vulnerabilities within a target. i.e. its accuracy. Equation 4.2 is expected to evaluate if a tool only detects existing vulnerabilities and not false ones, and this can be named precision. The results shown in Figure 4.6 resulted from calculating these equations from the data of the

vulnerability identification results shown in Tables 4.2 and 4.3. From these results, the overall obtained accuracy is at most 50% for the case of the *M4* scan with OpenVAS, this means that multiple vulnerabilities were not detected by all tools. Regarding accuracy, OpenVAS and Nessus performed better for *M4*, a Windows machine, while Nexpose was more accurate for *M2*, a Ubuntu machine. In terms of precision, the overall better-performing tool was OpenVAS with 100% accuracy for *M2*, and almost 80% for *M4*. The least precise tool was Nexpose, with 50% accuracy for both *M2* and *M4*.



(a) Accuracy



(b) Precision

Figure 4.6: Comparison of detection capabilities

After evaluating the performance, resource usage, accuracy and precision of the selected

vulnerability scanning tools, conclusions regarding the three questions listed in Section 1 can be drawn.

- Q1: What is the most efficient, free-to-use, vulnerability scanning tool currently available?

The answer to Q1 can state that there is no one tool that can be classified as the best at all evaluated criteria. While OpenVAS is the tool that uses more CPU and network resources and takes the most time, it also uses less memory and has better overall precision. Nonetheless, in terms of accuracy, it is better when scanning Windows-based targets, and not so good when scanning Linux-based targets. Nexpose, for instance, has an average precision of 50% for both Linux and Windows-based targets but, when analysing its accuracy, the results show poor overall results.

- Q2: How does Tsunami compare to similar tools currently available?

It has been determined that, in its current state, the Tsunami Security Scanner is not yet equipped to serve as a replacement for other established tools such as Nessus, Nexpose, or OpenVAS. Tsunami Security Scanner is relatively recent and currently, it lacks openly available detection plugins. Tsunami architecture is plugin oriented, where each plugin will detect the presence of a specific vulnerability. When this research started, the number of plugins available was almost nonexistent, meaning that Tsunami was unable to detect vulnerabilities that were present in the targets. It can be envisioned that Tsunami, may become a relevant candidate if the community release a number of detection plugins comparable to the remaining tools.

- Q3: Is Tsunami well suited to be used by SMEs?

At the time of this work, it seems that Tsunami Security Scanner is not suitable for a SME context. Despite being open source and free-to-use, their current lack of detection plugins plus its mode of operation makes it unsuitable for use in SMEs that do not have human resources capable of developing their own plugins for Tsunami. Moreover, the way the results are reported by Tsunami (JSON format) makes it best suited for use in automatic assessments of a development pipeline in a product development life cycle.

Chapter 5

Development of a Tsunami Plugin

This chapter presents in more detail how the tsunami works 5.1. Then it is explained how to develop a plugin for the tsunami 5.2 and the creation of a plugin for a specific vulnerability 5.3.

5.1 Tsunami internals

The Tsunami Security Scanner uses a hardcoded 2-step process. Hard coding is the software development practice of embedding data directly into the source code, which means that data can only be modified by editing the source code and recompiling the executable. The two processes used are Reconnaissance and Vulnerability Verification and can be described as follows:

- Reconnaissance: In the first step, the scanner will identify the open ports and their protocols, services and other software used in the system to be scanned.
- Vulnerability Verification: After the first step ends, the Tsunami will select only those plugins whose service was identified in the first step, and thus achieve the minimum of false positives.

Figure 5.1 details the Tsunami workflow, beginning with the reconnaissance step, Tsunami probes the scanned target and gathers as much information about the target as possible, including:

- open ports

- protocols
- network services & their banners
- potential software & corresponding version

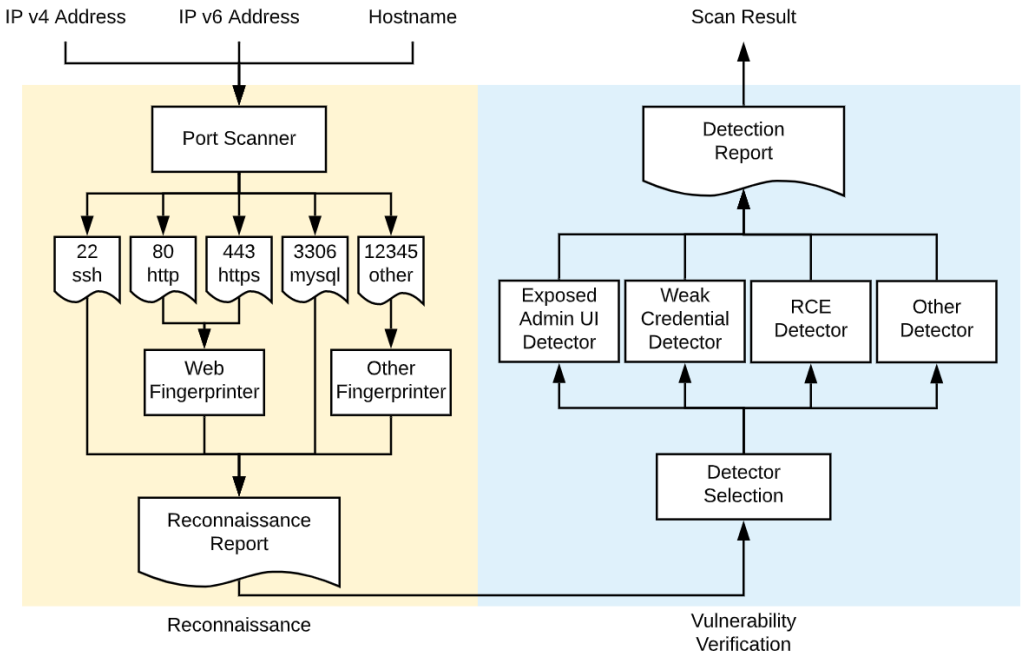


Figure 5.1: Scanning Workflow

In the port scanning phase, the Tsunami Security Scanner performs port sweeping in order to identify open ports, protocols and network services on the scanned target. Usually, port scanners only provide the service and the version. When needing more information about a host, the scanner needs to perform further fingerprinting work. If the scan target might choose to serve multiple web applications on the same TCP port 443 using Nginx [36] for reverse proxy, /blog for WordPress [63], and forum for phpBB [44], etc. The Port scanner will only be able to indicate that port 443 is running Nginx. A Web Application Fingerprinter with a comprehensive crawler is required to identify these applications.

At the end of the reconnaissance step, the Tsunami Security Scanner compiles both the port scanner outputs and service fingerprinter outputs into a single *ReconnaissanceReport protobuf* for Vulnerability Verification. Protocol buffers are Google's language-neutral, extensible mechanism for serializing structured data, which is like XML, but smaller, faster, and simpler [47].

In the Vulnerability Verification step, the Tsunami Security Scanner executes the *VulnDetector* plugins in parallel to verify certain vulnerabilities on the scan target based on the information gathered in the Reconnaissance step. *VulnDetector's* detection logic could either be implemented as plain Java code or as a separate binary/script using a different language like python or go.

Usually, one *VulnDetector* only verifies one vulnerability and the vulnerability often only affects one type of network service or software. In order to avoid doing wasteful work, Tsunami Security Scanner allows plugins to be annotated by some filtering annotations to limit the scope of the plugin.

Then, before the Vulnerability Verification step starts, Tsunami Security Scanner will select matching *VulnDetectors* to run based on the exposed network services and running software on the scan target. Non-matching *VulnDetectors* will stay inactive throughout the entire scan [23].

5.2 Developing a Tsunami Plugin

The version used in the tsunami security scanner tests was Tsunami v0.0.2. It is possible to develop two types of plugins, fingerprinting and vulnerability verification. To develop a plugin there is an example in the Tsunami Security Scanner repository [1]. Opening the *ExampleVulnDetector.java* file what is found first is *pluginInfo* which tells the Tsunami the basic information about the plugin. If Jenkins service is taken as an example, the developed plugin will have to say that this plugin will only be called if there is this service on the target, ForSoftware(name = "Jenkins"). The information about the target will be acquired through the *DetectionReportList* function in the *targetInfo*, collected by the tsunami. To check if it is vulnerable, a code must be programmed on *ServiceVulnerable* to check if the value "true" is returned. The goal is to do a fingerprinting plugin and then go to the file *ExampleCallingCommand.java* in the *isServiceVulnerable* function.

The chosen vulnerability was CVE-2019-12815/CVE-2015-3306, over ProFTP, the description for this vulnerability is on [38]. The mod_copy module in ProFTPD 1.3.5 allows remote attackers to read and write to arbitrary files via the *SITE CPFR* and *SITE CPTO*

[1] https://github.com/google/tsunami-security-scanner-plugins/tree/master/examples

commands that can be used in ProFTP[46]. The *mod_copy* module implements *SITE CPFR* and *SITE CPTO* commands, which can be used to copy files/directories from one place to another on the server without having to transfer the data to the client and back.

The first step was to install and configure Tsunami Security Scanner, this script automates it by downloading the source code compiling it, and creating the directory, where the custom-made plugin, will be placed.

For the Tsunami Security Scanner to work it is necessary to have Nmap (version >= 7.80) and ncrack (version >= 0.7) softwares installed.

```
1 {
2     bash -c "$(curl -sfL https://raw.githubusercontent.com/google/
    tsunami-security-scanner/master/quick_start.sh)"
3 }
```

Listing 5.1: installation

To create the custom plugin, an existing example in their repository was used:

```
1 {
2     $HOME/tsunami/repos/tsunami-security-scanner-plugins/examples/
    example_vuln_detector/
3 }
```

Listing 5.2: Existing example

The java file containing the developed source code can be found at:
/src/main/java/com/google/tsunami/plugins/example/ExampleVulnDetector.java

5.3 Testing the Developed Plugin

Eclipse IDE [20] was used to open and compile the plugin. In *@PluginInfo* was placed the information about the vulnerability to be developed.

As this vulnerability is about FTP then it has to be written *@ForServiceName("ftp")*, and then Tsunami will only call this plugin if it detects that the machine being scanned has this service active.

After that, the *isServiceVulnerable* function will have to confirm if it is or it is not vulnerable.

In this case, start by connecting to the FTP service via socket:

```
1 {
2     Socket socket = new Socket(ipAdress,21);
3 }
```

Listing 5.3: Existing example

Then create a variable to read and write the content in the FTP server.

```
1 {
2     BufferedReader reader = new BufferedReader(new
      InputStreamReader(socket.getInputStream()));
3
4     OutputStreamWriter writer = new OutputStreamWriter(socket.
      getOutputStream());
5 }
```

Listing 5.4: Read and write the content in the FTP server

If the response is as expected a confirmation is needed with an "if" statement:

```
1 {
2     String response = reader.readLine();
3
4     if (!response.startsWith("220 "))
5 }
```

Listing 5.5: Expect code 220

The code 220 indicates that the server is ready for the new client, it is possible to check all response codes in [1].

If the code is different from 220 then this machine is not vulnerable, if it is 220, then a second check needs to be done. The command SITE CPFR[2] is for copying from one place to another directly on the server.

```
1 {
2     writer.write("SITE CPFR /etc/passwd\r\n");
```

[2] http://www.proftpd.org/docs/contrib/mod_copy.html

```
3     writer.flush();
4     response = reader.readLine();
5     if (!response.startsWith("350 "))
6 }
```

Listing 5.6: Example of a data query

For the second step indicated above, if the answer is 350 it indicates that the service is vulnerable and the function returns true. A 350 response code is sent by the server in response to a file-related command that requires further commands, in the case of confirmation of the vulnerability it will not be necessary.

The last step is to fill in the *buildDetectionReport* function which is to get the vulnerability details in the Tsunami report.

The full script can be found in Annex A.

Chapter 6

Conclusions

Organisations may benefit from a systematic and periodic vulnerability assessment using free-to-use scanning tools. Using automated vulnerability scanning tools also reduces the required human, technical and financial resources when compared to manual penetration testing. This thesis provides information on cybersecurity in general, it is explained vulnerabilities, CVE, CVSS and vulnerability scanning tools. This can be an entry point for anyone interested in cybersecurity and wants to know how the area itself works, also for SMEs where it is possible to identify different tools either for web or networks, some paid others open-source. Paid versions manage to have a better response to new vulnerabilities with active support, but they can be quite expensive. On the other hand, we have open-source ideas for SMEs, free and with the support of the community always willing to help. For both cases, the vulnerability scanning tool that runs periodically can prevent malicious attacks on the organization.

With the release of Tsunami, yet another free-to-use vulnerability scanning tool, it was decided to perform an updated evaluation of the existing similar tools with the possibility of creating an explanatory plugin. The evaluation considered both the performance of the tools, but also their accuracy and precision. The obtained results show that OpenVAS was the tool that achieved the best overall precision and the best accuracy when scanning Windows-based systems. Nexpose was the tool that achieved the best accuracy when scanning Linux-based systems. In terms of CPU, memory and network usage, the results differ greatly from tool to tool but a common trait, of requiring more resources to scan systems with more vulnerabilities, was also identified. It was also concluded that Tsunami

has very short detection capabilities and is still far from the detection capabilities of the other free-to-use tools. The manual confirmation of vulnerabilities reported by all tools was focused on the critical ones, with a CVSS above 7.5. For future work, manual confirmation was carried out of all vulnerabilities reported by all tools to have a better understanding of both the accuracy and precision of the evaluated tools.

After this assessment, a plugin was designed to contribute to the recent Tsunami with a plugin that identifies a specific ProFTPD vulnerability.

The plugin operates for the CVE-2019-12815 and allows for remote code execution and information disclosure without authentication and with that is able to explain how a plugin for Tsunami is made. The purpose of a plugin is not to exploit the vulnerability but to identify it through the service and version, or with certain commands that do not affect the integrity of the system.

References

[1] *214, 215, 220, 221 FTP Response Codes — Serv-U.* (accessed on 10 October 2022). URL: https://www.serv-u.com/resources/tutorial/214-215-220-221-ftp-response-codes.

[2] *Acunetix — Web Application Security Scanner.* (accessed on 10 October 2022). URL: https://www.acunetix.com/.

[3] M Ugur Aksu, Enes Altuncu, and Kemal Bicakci. "A First Look at the Usability of OpenVAS Vulnerability Scanner". In: *Workshop on Usable Security (USEC) 2019*. NDSS. 2019.

[4] Abdul Bashah Mat Ali et al. "SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks". In: *Procedia Computer Science* 3 (2011), pp. 453–458.

[5] Richard Amankwah et al. "An empirical comparison of commercial and open-source web vulnerability scanners". In: *Software: Practice and Experience* 50.9 (2020), pp. 1842–1857.

[6] *Arachni/arachni: Web Application Security Scanner Framework.* (accessed on 10 October 2022). URL: https://github.com/Arachni/arachni.

[7] Ricardo Araújo, António Pinto, and Pedro Pinto. "A Performance Assessment of Free-to-Use Vulnerability Scanners - Revisited". In: *IFIP Advances in Information and Communication Technology* 625 (2021), pp. 53–65. ISSN: 1868422X. DOI: 10.1007/978-3-030-78120-0_4/COVER. URL: https://link.springer.com/chapter/10.1007/978-3-030-78120-0_4.

[8] *Arcserve - Data Protection and Business Continuity Solutions*. (accessed on 10 October 2022). URL: https://www.arcserve.com/.

[9] *AT&T Vulnerability Scanning Service (VSS)*. (accessed on 10 October 2022). URL: https://cybersecurity.att.com/resource-center/solution-briefs/att-external-vulnerability-scanning-service.

[10] Andrew Austin and Laurie Williams. "One technique is not enough: A comparison of vulnerability discovery techniques". In: *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE. 2011, pp. 97–106.

[11] Thanapon Bhuddtham and Pirawat Watanapongse. "Time-related vulnerability lookahead extension to the CVE". In: *2016 13th International Joint Conference on Computer Science and Software Engineering, JCSSE 2016* (Nov. 2016). DOI: 10.1109/JCSSE.2016.7748927.

[12] *BIG-IP application services, hardware, and software — F5*. (accessed on 10 October 2022). URL: https://www.f5.com/products/big-ip-services.

[13] *Burp Suite - Application Security Testing Software - PortSwigger*. (accessed on 10 October 2022). URL: https://portswigger.net/burp.

[14] *Changelog Archive*. (accessed on 10 October 2022). URL: https://www.jenkins.io/changelog-old/.

[15] Sanon Chimmanee et al. "A performance comparison of vulnerability detection between NetClarity Auditor and Open Source Nessus". In: *Proceeding of the 3rd European Conference of Communications (ECCOM'12)*. 2012, pp. 280–285.

[16] Catalin Cimpanu. *Google open sources Tsunami vulnerability scanner*. ZDNet. July 2020. URL: https://www.zdnet.com/article/google-open-sources-tsunami-vulnerability-scanner/ (visited on 10/18/2020).

[17] MITRE Corp. *Common Vulnerabilities and Exposures (CVE)*. (accessed on 10 February 2020). URL: https://cve.mitre.org/.

[18] Nor Izyani Daud, Khairul Azmi Abu Bakar, and Mohd Shafeq Md Hasan. "A case study on web application vulnerability scanning tools". In: *2014 Science and Information Conference*. IEEE. 2014, pp. 595–600.

[19] Adam Doupé, Marco Cova, and Giovanni Vigna. "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners". In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer. 2010, pp. 111–131.

[20] *Eclipse Desktop & Web IDEs — The Eclipse Foundation*. (accessed on 10 October 2022). URL: https://www.eclipse.org/ide/.

[21] *Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers*. (accessed on 10 October 2022). URL: https://www.exploit-db.com/.

[22] Jose Fonseca, Marco Vieira, and Henrique Madeira. "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks". In: *13th Pacific Rim international symposium on dependable computing (PRDC 2007)*. IEEE. 2007, pp. 365–372.

[23] *GitHub - google/tsunami-security-scanner: Tsunami is a general purpose network security scanner with an extensible plugin system for detecting high severity vulnerabilities with high confidence.* URL: https://github.com/google/tsunami-security-scanner (visited on 10/19/2020).

[24] *GitLab 10.0 released with Auto DevOps and Group Issue Boards — GitLab*. (accessed on 10 October 2022). URL: https://about.gitlab.com/releases/2017/09/22/gitlab-10-0-released/.

[25] *GitLab Patch Release: 13.4.1 — GitLab*. (accessed on 10 October 2022). URL: https://about.gitlab.com/releases/2020/09/24/gitlab-13-4-1-released/.

[26] Hannes Holm. "Performance of automated network vulnerability scanning at remediating security issues". In: *Computers & Security* 31.2 (2012), pp. 164–175.

[27] Hannes Holm et al. "A quantitative evaluation of vulnerability scanning". In: *Information Management & Computer Security* (2011).

[28] *JavaScript.com*. (accessed on 10 October 2022). URL: https://www.javascript.com/.

[29] Stefan Kals et al. "Secubat: a web vulnerability scanner". In: *Proceedings of the 15th international conference on World Wide Web*. 2006, pp. 247–256.

[30] R Kushe. "Comparative Study of Vulnerability Scanning Tools: NESSUS vs RETINA". In: *Security & Future* 1.2 (2017), pp. 69–71.

[31] *Managed Security Service: Security Management of the Online Business - Alibaba Cloud*. (accessed on 10 October 2022). URL: https://www.alibabacloud.com/product/mss.

[32] *Mantis Bug Tracker*. (accessed on 10 October 2022). URL: https://www.mantisbt.org/.

[33] Balume Mburano and Weisheng Si. "Evaluation of web vulnerability scanners based on OWASP benchmark". In: *2018 26th International Conference on Systems Engineering (ICSEng)*. IEEE. 2018, pp. 1–6.

[34] *Microsoft: Cloud, Computadores, Aplicações e Jogos*. (accessed on 10 October 2022). URL: https://www.microsoft.com/pt-pt/.

[35] *Ncrack - High-speed network authentication cracker*. (accessed on 10 October 2022). URL: https://nmap.org/ncrack/.

[36] *NGINX - High Performance Load Balancer, Web Server, & Reverse Proxy*. (accessed on 10 October 2022). URL: https://www.nginx.com/.

[37] *Nmap: the Network Mapper - Free Security Scanner*. (accessed on 10 October 2022). URL: https://nmap.org/.

[38] *NVD - CVE-2015-3306*. (accessed on 10 October 2022). URL: https://nvd.nist.gov/vuln/detail/CVE-2015-3306.

[39] *OpenVAS - Learn All About the OpenVAS Vulnerability Scanner — Bugcrowd*. (accessed on 10 October 2022). URL: https://www.bugcrowd.com/glossary/openvas-vulnerability-scanner/.

[40] *OpenVAS - Open Vulnerability Assessment Scanner*. (accessed on 10 October 2022). URL: https://www.openvas.org/.

[41] *OWASP Top Ten - OWASP Foundation*. (accessed on 10 October 2022). URL: https://owasp.org/www-project-top-ten/.

[42] *OWASP ZAP*. (accessed on 10 October 2022). URL: https://www.zaproxy.org/.

[43] *PHP: Hypertext Preprocessor*. (accessed on 10 October 2022). URL: https://www.php.net/.

[44] *phpBB - Free and Open Source Forum Software*. (accessed on 10 October 2022). URL: https://www.phpbb.com/.

[45] *phpMyAdmin - Downloads*. (accessed on 10 October 2022). URL: https://www.phpmyadmin.net/downloads/.

[46] *ProFTPD module mod$_c$opy*. (accessed on 10 October 2022). URL: http://www.proftpd.org/docs/contrib/mod_copy.html.

[47] *Protocol Buffers - Google Developers*. (accessed on 10 October 2022). URL: https://developers.google.com/protocol-buffers.

[48] *Qualys VMDR 2.0: - Vulnerability Management Tool — Qualys*. (accessed on 10 October 2022). URL: https://www.qualys.com/apps/vulnerability-management-detection-response/.

[49] Rapid7. *Free Nexpose Community 1-Year Trial*. (accessed on 10 October 2022). URL: https://www.rapid7.com/info/nexpose-community.

[50] *SAP BusinessObjects - Business Intelligence (BI) Platform & Suite*. (accessed on 10 October 2022). URL: https://www.sap.com/products/technology-platform/bi-platform.html.

[51] *Search Engine for Security Intelligence — Vulners*. (accessed on 10 October 2022). URL: https://vulners.com/.

[52] Ankur Shukla, Basel Katt, and Livinus Obiora Nweke. "Vulnerability discovery modelling with vulnerability severity". In: *2019 IEEE Conference on Information and Communication Technology, CICT 2019* (Dec. 2019). DOI: 10.1109/CICT48419.2019.9066187.

[53] *sullo/nikto: Nikto web server scanner*. (accessed on 10 October 2022). URL: https://github.com/sullo/nikto.

[54] Tenable. *Nessus Vulnerability Assessment Tool*. (accessed on 10 February 2020). URL: https://www.tenable.com/products/nessus (visited on 02/10/2021).

[55] *tsunami-security-scanner-plugins/google/fingerprinters/web at master - google/tsunami-security-scanner-plugins*. (accessed on 10 October 2022). URL: https://github.com/google/tsunami-security-scanner-plugins/tree/master/google/fingerprinters/web.

[56] *tsunami-security-scanner/future$_w$ork.mdatmaster − google/tsunami − security − scanner*. (accessed on 10 October 2022). URL: https://github.com/google/tsunami-security-scanner/blob/master/docs/future_work.md#dynamic_orchestration.

[57] "Vulnerability Scanning Tools and Services - NCSC.GOV.UK". In: (). URL: https://www.ncsc.gov.uk/guidance/vulnerability-scanning-tools-and-services.

[58] *w3af - Open Source Web Application Security Scanner*. (accessed on 10 October 2022). URL: http://w3af.org/.

[59] Ruyi Wang et al. "An improved CVSS-based vulnerability scoring mechanism". In: *Proceedings - 3rd International Conference on Multimedia Information Networking and Security, MINES 2011* (2011), pp. 352–355. DOI: 10.1109/MINES.2011.27.

[60] Y. Wang and J. Yang. "Ethical Hacking and Network Defense: Choose Your Best Network Vulnerability Scanning Tool". In: *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 2017, pp. 110–113.

[61] Yien Wang and Jianhua Yang. "Ethical hacking and network defense: Choose your best network vulnerability scanning tool". In: *Proceedings - 31st IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2017* (May 2017), pp. 110–113. DOI: 10.1109/WAINA.2017.39.

[62] SM Welberg. "Vulnerability management tools for COTS software-A comparison". In: *Hg. v. University of Twente* 1 (2008). URL: http://doc.utwente.nl/64654/1/Vulnerability_management_tools_for_COTS_software_-_a_comparison_v2.

[63] *WordPress.com: Fast, Secure Managed WordPress Hosting*. (accessed on 10 October 2022). URL: https://wordpress.com/.

[64] *wpscanteam/wpscan: WPScan WordPress security scanner. Written for security professionals and blog maintainers to test the security of their WordPress websites.* (accessed on 10 October 2022). URL: `https://github.com/wpscanteam/wpscan`.

Appendices

Appendix A

Tsunami plugin for CVE-2019-12815

```
{
   public DetectionReportList detect(
      TargetInfo targetInfo, ImmutableList<NetworkService>
   matchedServices) {
    logger.atInfo().log("ProFTP starts detecting.");

    // An example implementation for a VulnDetector.
    return DetectionReportList.newBuilder()
        .addAllDetectionReports(
            matchedServices.stream()
                // Check individual NetworkService whether it is
   vulnerable.
                .filter(unused -> isServiceVulnerable(targetInfo))
                // Build DetectionReport message for vulnerable
   services.
                .map(networkService -> buildDetectionReport(
   targetInfo, networkService))
                .collect(toImmutableList()))
        .build();
  }

```

```
18 private boolean isServiceVulnerable(TargetInfo targetInfo) {
19
20       String ipAdress = targetInfo.getNetworkEndpoints(0).
    getIpAddress().getAddress();
21       System.out.println(ipAdress);
22
23       try {
24         String serverImportantOutput;
25
26           Socket socket = new Socket(ipAdress,21);
27           BufferedReader  reader = new BufferedReader(new
    InputStreamReader(socket.getInputStream()));
28           OutputStreamWriter writer = new OutputStreamWriter(socket
    .getOutputStream());
29
30           String line;
31
32           String response = reader.readLine();
33
34           System.out.println(response);
35           if (!response.startsWith("220 ")) {
36             return false;
37           }
38
39
40           writer.write("SITE CPFR /etc/passwd\r\n");
41           writer.flush();
42
43           response = reader.readLine();
44           System.out.println(response);
45           if (!response.startsWith("350 ")) {
46             return false;
47           }
48
49           System.out.println(socket.isConnected());
```

```
50            socket.close();
51          }catch(Exception e){
52              System.out.print("Error connecting!");
53          }
54      return true;
55  }
56  }
```

Listing A.1: code