

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

**IMPLEMENTATION OF A VULNERABILITY ANALYSIS SERVICE OF
PUBLIC HOSTS WITH SAAS MODALITY**

Thesis

**ALEJANDRO COSTA RUEDA
TUTOR: LÉONARD JANER-GARCIA**

2021-2022

Abstract

Today almost all companies base their business models on web applications and online services. These services are public to everyone, including malicious agents that can attack these business models. Many of these companies do not have the knowledge, time or resources to analyze if their service is free of possible attacks. With this project, we try to create a tool that can help these companies to know the possible dangers and attacks to which their service is open, by analyzing the vulnerabilities that the host may have.

Resum

Actualment gairebé totes les empreses basen els seus models de negoci en aplicacions web i serveis en línia. Aquests serveis són públics per a tothom, inclosos els agents maliciosos que poden atacar aquests models de negoci. Moltes d'aquestes empreses no disposen del coneixement, del temps o dels recursos per analitzar si el seu servei està lliure de possibles atacs. Amb aquest projecte intentem crear una eina que pugui ajudar a aquestes empreses a conèixer els possibles perills i atacs als quals està obert el seu servei, mitjançant l'anàlisi de les vulnerabilitats que pot tenir el host.

Resumen

Hoy en día casi todas las compañías basan sus modelos de negocio en aplicaciones web y servicios online. Estos servicios son públicos para todo el mundo, incluidos agentes maliciosos que pueden llegar a atacar estos modelos de negocio. Muchas de estas empresas no tienen los conocimientos, el tiempo necesario o los recursos para analizar si su servicio está libre de posibles ataques. Con este proyecto intentamos crear una herramienta que pueda ayudar a estas empresas a conocer los posibles peligros y ataques a los que está abierto su servicio, mediante un análisis de las vulnerabilidades que puede llegar a tener el host.

Table of Contents

List of Figures	III
1 Introduction	1
2 Objectives and scope	3
3 Theoretical Framework	5
3.1 Cybersecurity	5
3.2 Similar solutions	5
3.2.1 Installed solutions	5
3.2.2 Online Solutions	6
3.3 Vulnerability Analysis Tools	7
3.3.1 Nessus	7
3.3.2 OWASP ZAP	12
3.3.3 Nmap.....	17
3.3.4 Metasploit	23
3.3.5 Sqlmap.....	26
3.3.6 WPScan.....	28
3.3.7 Joomscan	30
3.3.8 TLSSLed.....	35
3.4 Web Technologies	36

//		
	3.4.1 Front-end	36
	3.4.2 Back-end.....	39
4	Methodology	41
5	Development	43
5.1	Definition of the functional and technological requirements	43
5.1.1	Functional Requirements	43
5.1.2	Technological Requirements	44
5.2	Development Process.....	45
5.2.1	Design of the web application.....	45
5.2.2	Code Tools Integration	47
5.2.3	Code Front-end	53
5.2.4	Code Back-end.....	56
5.2.5	Scan Server.....	59
5.2.6	Database.....	61
6	Conclusions	64
6.1	Difficulties.....	64
6.2	Future investigations.....	65
6.3	Personal reflections	65
7	Bibliography	67

List of Figures

3.1 Scan folder and new scan option	8
3.2 Scan templates	9
3.3 Configure new scan	9
3.4 Scan configured.....	10
3.5 Scan finished	10
3.6 General results of the scan	11
3.7 Detailed scan results.....	11
3.8 Welcome page.....	13
3.9 Introduce url to attack	14
3.10 Results of the scan.....	15
3.11 Detailed results of the scan.....	16
3.12 Result of a scan to the port 22, 113 and 139. Source [8].....	18
3.13 Example of SYN request and open port. Source [8].....	18
3.14 Example of SYN request and closed port. Source [8]	19
3.15 Example of SYN request and filtered port. Source[8].....	19
3.16 Result of scan metasploitable2	21
3.17 Result of scan metasploitable2 with services version.....	22
3.18 Entry message of metasploit.....	24
3.19 Results of vsftpd 2.3.4	24

3.20 Results of vnc 3.3.....	25
3.21 Intercept request with burpsuite.....	27
3.22 Save request in burpsuite.....	27
3.23 Execute sqlmap with saved request.....	28
3.24 Results of sqlmap.....	28
3.25 Results of wpscan	29
3.26 joomscan command to attack host.....	32
3.27 Results of joomscan.....	33
3.28 Enumeration of the components.....	34
3.29 TLSSled command on host 192.168.56.1 443.....	35
3.30 Files where the results are stored	35
3.31 Result of ssl scan	36
3.32 Poll from stateofjs [14]	37
3.33 Poll from stateofjs [14]	38
3.34 Performance comparison. Source [15]	39
3.35 Poll from stackoverflow[19]	40
5.1 Diagram of the application.....	45
5.2 Import and usage of python-nmap. Source [20].....	48
5.3 Example of result of a scan. Source [20].....	48
5.4 Window to activate the API	49

5.5	Example of capturing the output of traceroute.....	50
5.6	The script finds where is the output and saves the path to use it later.....	51
5.7	The [+] signs define what are they checking, and the [++] signs are the results of that check	51
5.8	Structure that returns the wpscan_out_parse library	52
5.9	Welcome page.....	53
5.10	Register page	53
5.11	Login page	54
5.12	Main page	54
5.13	Create host page.....	55
5.14	Host in detail	55
5.15	Create scan page	56
5.16	Diagram of the scan back end	57
5.17	Example of result.....	59
5.18	Diagram of the database	62

1. Introduction

This Final Degree Project aims to create a service able to analyse defence vulnerabilities in public hosts. The project is part of a bigger project developed with ESED Ltd.¹ with a Software as a Service model.

The security analysis is normally made manually and is a process that takes a lot of time. Some businesses don't have the time or the resources to carry out a complete vulnerability analysis exam. Some of these tests can not be made with the desired periodicity and intensity, because several of these audits are really time-consuming and require multiple steps. At the end of the day, companies work with a tight schedule that they have to fulfil, and the ones that make the decisions do not use to have a technical profile to notice that their businesses might be in grave danger. This situation can lead to some security breaches that can gravely harm the company, like exposing information about the clients, being unable to use their equipment or losing control of its data.

Through lots of analysis, frameworks of attack can be developed in order to eliminate the need for a person manually testing those day-to-day vulnerabilities. All these frameworks save a lot of time to pen-testers but they can also be programmed to be made periodically and extract daily reports of the vulnerabilities present in the host. Those reports can be presented to justify the need for more resources, therefore saving time for the members of the security team.

There are lots of companies that do not check the security of their public hosts and this leaves them vulnerable to attacks from malicious agents. But, if they could have an instant security report of the safety of their hosts once this is published, this would warn them about the possible threats that they may encounter with the actual state of the host. Also it can help to visualize to a non-technical profile all the vulnerabilities in order to make better decisions.

¹<https://www.esedsl.com/>

2. Objectives and scope

The main objective of this Final Degree Project is to create a full framework to detect vulnerabilities in public hosts and adapt the framework to work within a larger project. To accomplish this main goal, it is mandatory to fulfil these subgoals:

- The user must be able to register a host.
- The user must be able to start a scan with a simple command.
- The user must be able to check the status of the scan
- The user must be able to retrieve the information of the scan once it is finished
- The user must be able to make different types of scans.
- The service must be able to deliver a report about the key vulnerabilities found in the host.
- The service must be able to save all previous reports from each host.
- The report must be understood by a manager profile.
- The report must have enough complexity to be used for a technical profile.

3. Theoretical Framework

3.1 Cybersecurity

Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks [1]. The attacks come from malicious agents that try to get some type of benefit from it. In order to protect from an attack, it is necessary to prepare the people and the equipment. The preparation of the equipment can be done by assessing if there are any vulnerabilities.

Nowadays there are multiple solutions to protect or prevent attacks from malicious agents. Some of the solutions similar to the one that we are proposing in this final degree project can be found in the next section.

3.2 Similar solutions

There exist different types of solutions, some need to be installed, and others are online and ready to use.

3.2.1 Installed solutions

This solutions need to be installed in a machine in order to work. They need to be administrated, and much more knowledge is needed in order to make them work. They are also specialized to check vulnerabilities in all devices inside a network.

Nessus

Nessus is part of the company Tenable [2]. Its features are explained in more detailed in section 3.3.1

Netsparker

Netsparker is an online solution owned by Invicti [3]. Some of its features are:

- Vulnerability Trend Report.
- Technology Version Tracking.
- Close security gaps with automated WAF rules.
- Integration into CI/CD tools.
- Simplify Compliance.
- Unlimited Role-Based Access

3.2.2 Online Solutions

These kinds of solutions are ready to use for each user, most of them just need a target and they are ready to check its vulnerabilities.

Intruder.io

Intruder.io [4] offers a 30 day free trial but does not offer a free version to be tested for more than 30 days. It is able to identify:

- Common mistakes and configuration weaknesses
- Missing patches
- Application Bugs
- Attack surface reduction
- Encryption weaknesses

Probely

Within its free features [5], these are some of them:

- Verify SSL security
- Security Headers
- Cookie Flags

3.3 Vulnerability Analysis Tools

After looking at some vulnerability analysis tools, it has been decided that these are going to be the selected ones:

3.3.1 Nessus

It is an open-source network vulnerability scanner that checks computers to find vulnerabilities that hackers could exploit. It is able to:

- Search vulnerabilities that could allow unauthorized control or access to sensitive data on a system
- Scan for misconfigurations
- Search Denials of Service vulnerabilities
- Schedule security audits
- Simulate attacks
- Show vulnerabilities in a graphical and comprehensive way

In this example it is going to be analyzed a metasploitable2 machine:

1. Into the My Scans Folder, we select the New Scan option (Figure 3.1)
2. We select the Web Application Tests option (Figure 3.2)
3. We have to fill the fields (Figure 3.3)
4. We can start the scan with the play button (Figure 3.4)

5. When it is finished we will see a tick before the last modified column (Figure 3.5)
6. If we enter, we can see the results of the analysis (Figure 3.6)
7. Inside the report, the results can be ordered by severity and its score (Figure 3.7)

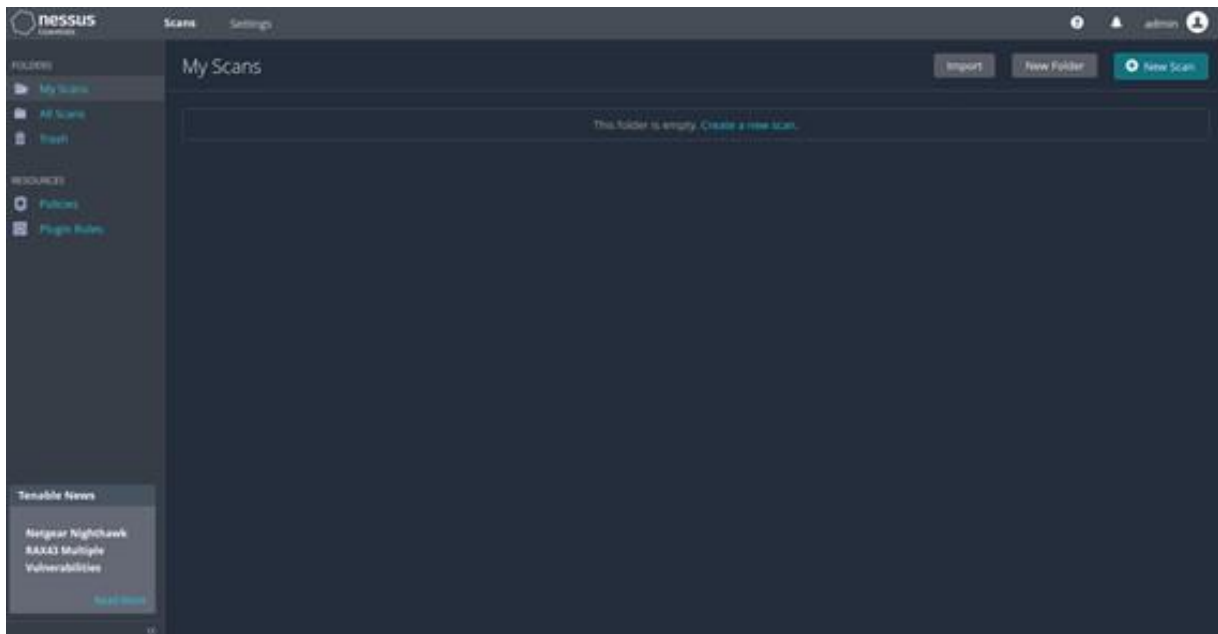


Figure 3.1: Scan folder and new scan option

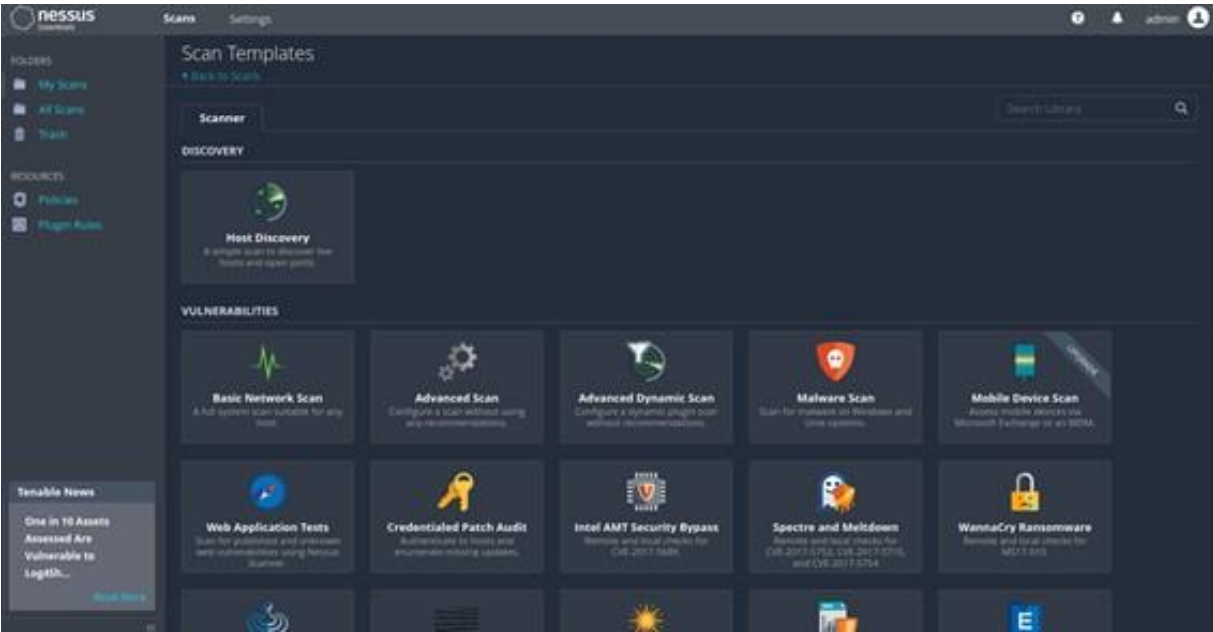


Figure 3.2: Scan templates

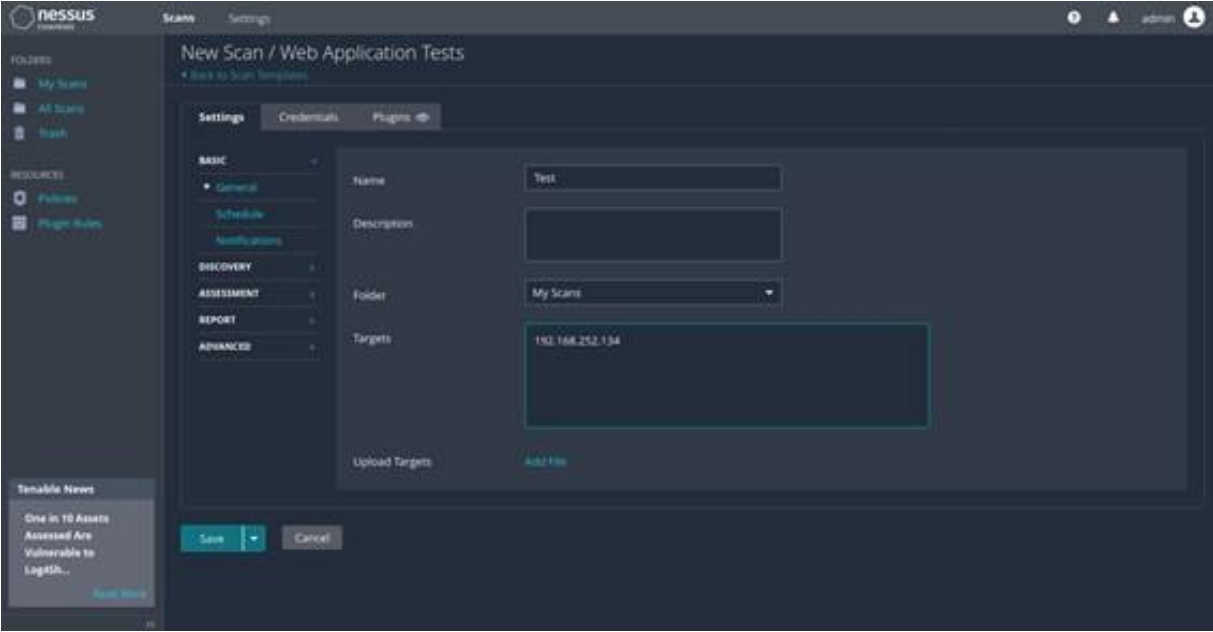


Figure 3.3: Configure new scan

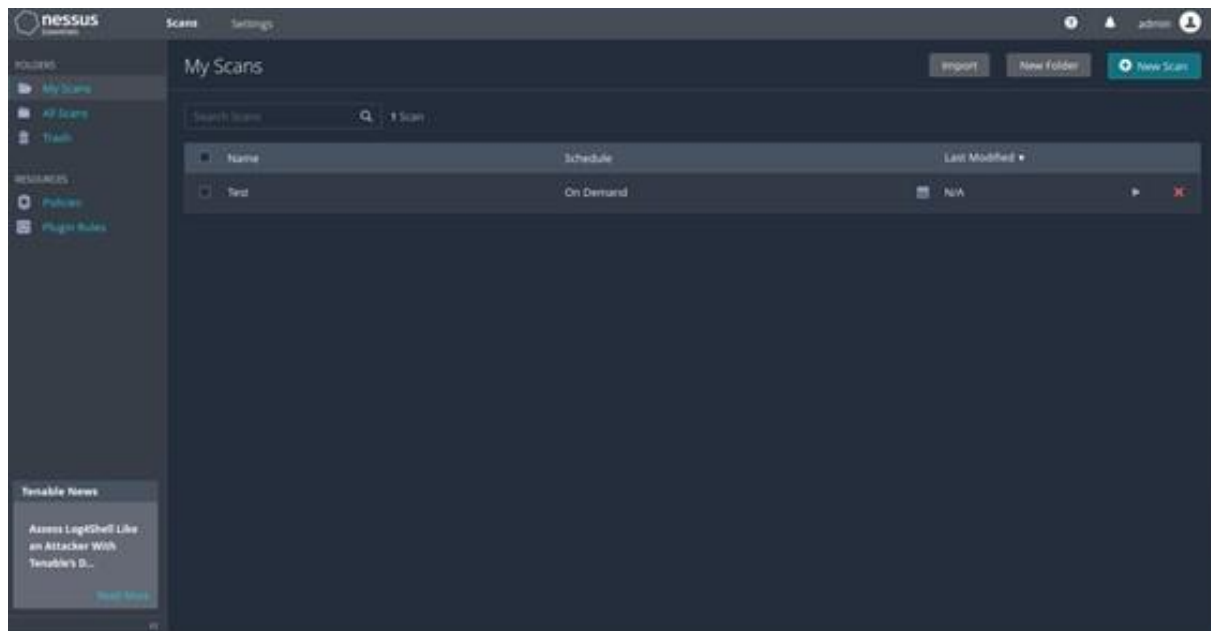


Figure 3.4: Scan configured



Figure 3.5: Scan finished

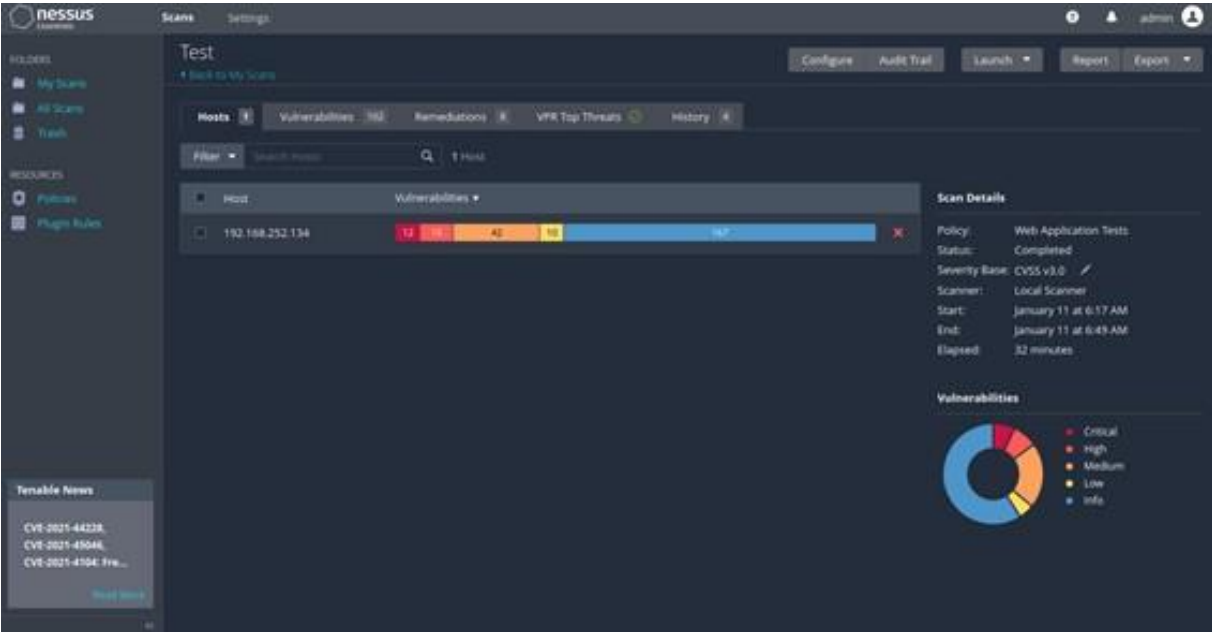


Figure 3.6: General results of the scan

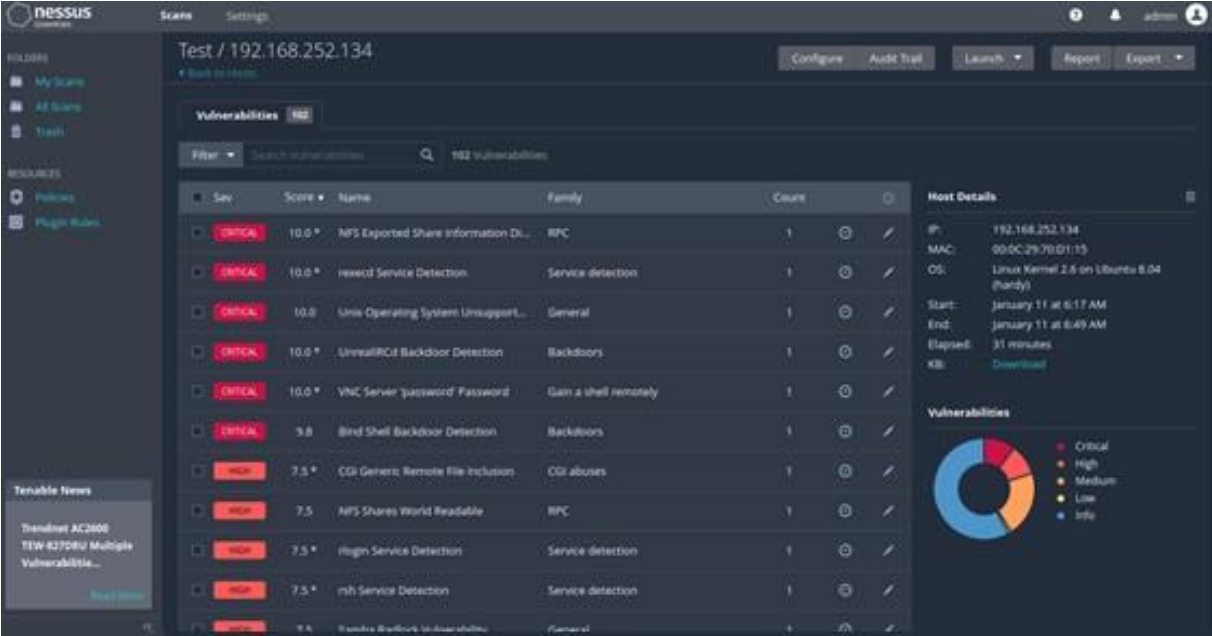


Figure 3.7: Detailed scan results

3.3.2 OWASP ZAP

It is an open-source web application security scanner developed by the Open Web Application Security Project (OWASP) Foundation [6]. It is a highly recognized tool used by a lot of security experts and maintained by them with new add-ons that can be downloaded from the marketplace. It is written mostly in Java and it is available in Windows, Linux and Mac OS X. Some of its features are:

- It can work as an intercepting proxy
- It is able to perform automated scanners
- Brute force scanning
- Fuzzing
- Port scanning
- Has an advanced SQL Injection Scanner
- It allows users to interact with API REST

We are going to attack a vulnerable host:

1. Select option Automated Scan (Figure 3.8)
2. Enter the url of the web application that you want to test (Figure 3.9)
3. In the alerts section we have all the possible vulnerabilities that we can encounter in this web application (Figure 3.10)
4. In this part, we can see all the possible vulnerabilities that the web application can have and some information of how it has been detected (Figure 3.11)

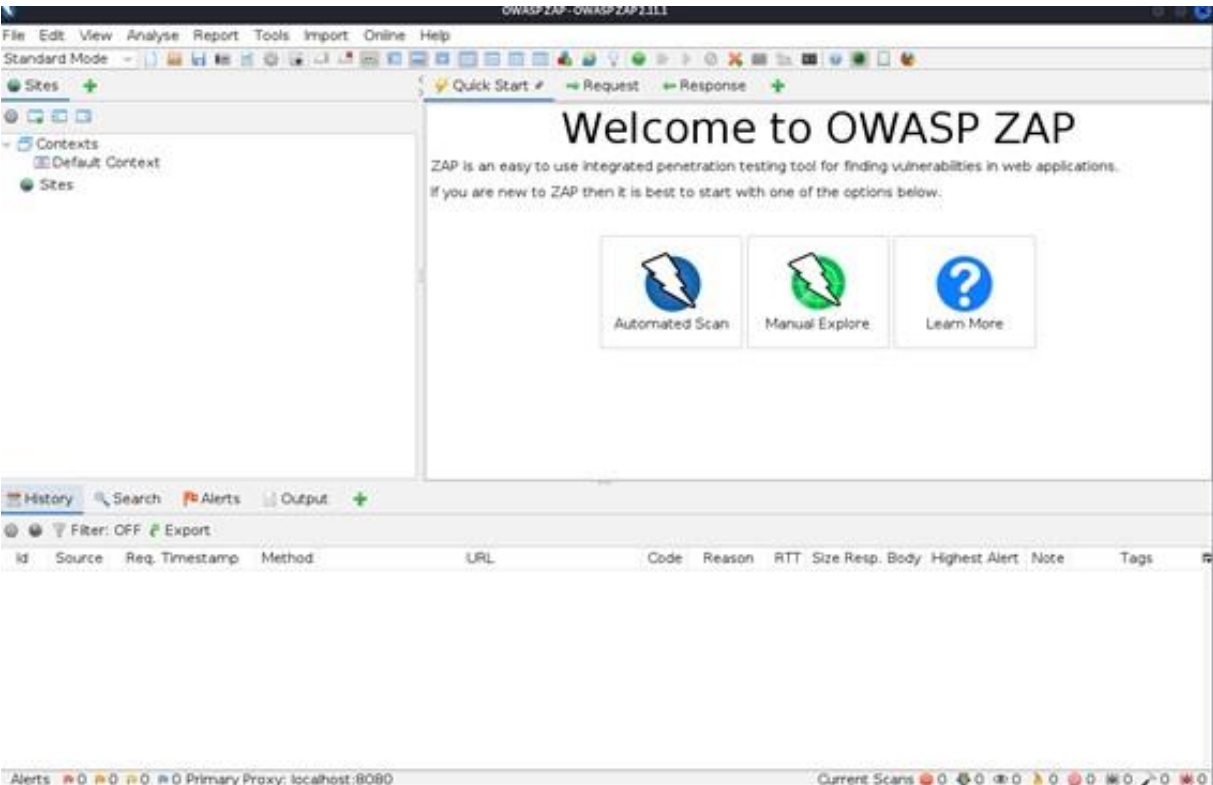


Figure 3.8: Welcome page

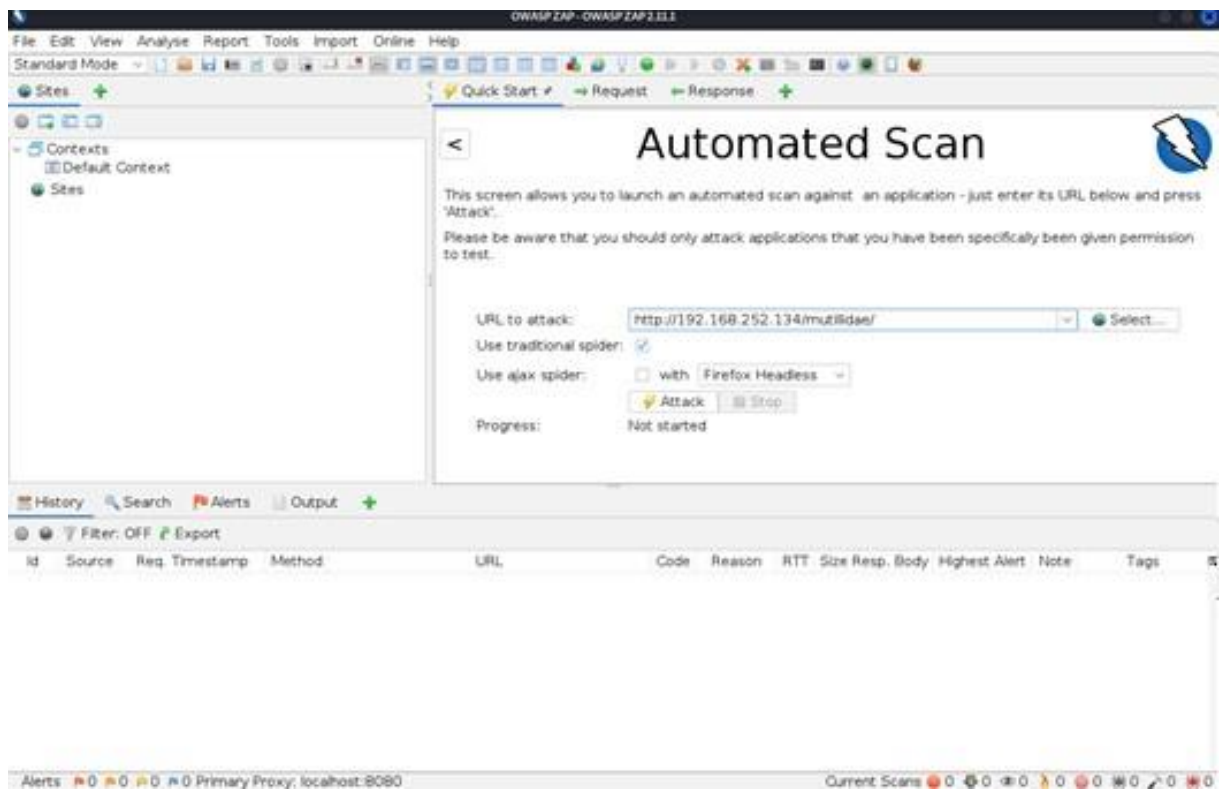


Figure 3.9: Introduce url to attack

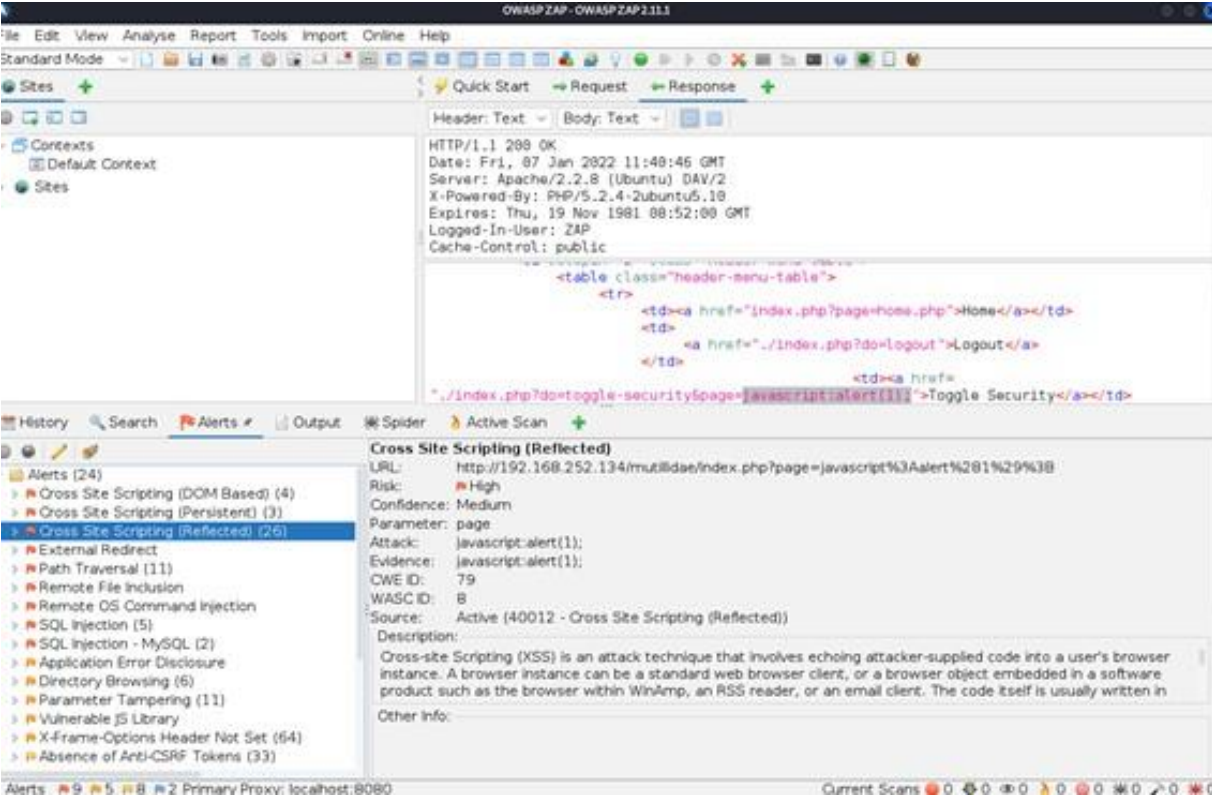


Figure 3.10: Results of the scan

- ▼ 📁 Alerts (24)
 - > 🚩 Cross Site Scripting (DOM Based) (4)
 - > 🚩 Cross Site Scripting (Persistent) (3)
 - > 🚩 Cross Site Scripting (Reflected) (26)
 - > 🚩 External Redirect
 - > 🚩 Path Traversal (11)
 - > 🚩 Remote File Inclusion
 - > 🚩 Remote OS Command Injection
 - > 🚩 SQL Injection (5)
 - > 🚩 SQL Injection - MySQL (2)
 - > 🚩 Application Error Disclosure
 - > 🚩 Directory Browsing (6)
 - > 🚩 Parameter Tampering (11)
 - > 🚩 Vulnerable JS Library
 - > 🚩 X-Frame-Options Header Not Set (64)
 - > 🚩 Absence of Anti-CSRF Tokens (33)
 - > 🚩 Cookie No HttpOnly Flag (19)
 - > 🚩 Cookie without SameSite Attribute (19)
 - > 🚩 Information Disclosure - Debug Error Message
 - > 🚩 Private IP Disclosure (7)
 - > 🚩 Server Leaks Information via "X-Powered-By" h
 - > 🚩 Timestamp Disclosure - Unix (428)
 - > 🚩 X-Content-Type-Options Header Missing (95)
 - > 🚩 Information Disclosure - Sensitive Information
 - > 🚩 Information Disclosure - Suspicious Comments

Figure 3.11: Detailed results of the scan

3.3.3 Nmap

Is a free open-source utility for network discovery and security auditing [7]. It was released on September 1st 1997 by Gordon Lyon in the fifty-first issue of Phrack magazine. At the beginning Nmap only supported Linux operating system and had fewer functionalities than it does nowadays. Thanks to the Open Source development, this software has grown to support more functionalities, different operating systems, and it has become one of the world's most popular network security scanners.

Nowadays Nmap can perform various tasks such as:

- Searching hosts available on the network
- The OS version of the hosts
- Check which services does a host offer
- See which applications are being run on a host and its version
- Recognise which types of packets and filters/firewalls are in use

To gather all this information, Nmap sends packets to the hosts and analyses and verifies the information of the answer. For example, in order to perform a port scan by default, Nmap sends a TCP SYN scan which is really fast and gives information like the port number, the state (open, closed, filtered,...) and the service that it has associated (Figure 3.12).

```

krad# nmap -p22,113,139 scanme.nmap.org
Starting Nmap ( http://nmap.org )
Nmap scan report for scanme.nmap.org (64.13.134.52)
PORT      STATE      SERVICE
22/tcp    open      ssh
113/tcp   closed    auth
139/tcp   filtered  netbios-ssn
Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds

```

Figure 3.12: Result of a scan to the port 22, 113 and 139. Source [8]

This TCP SYN scan starts with a TCP packet sent to a host with the SYN Flag set to 1 and the number of the port that it is going to be checked. As it is a TCP SYN request, the sender wants to create a TCP connection with that port and the receiver has to respond to that connection.

If the receiver responds with a SYN/ACK packet, it means that the port is open (Figure 3.13).

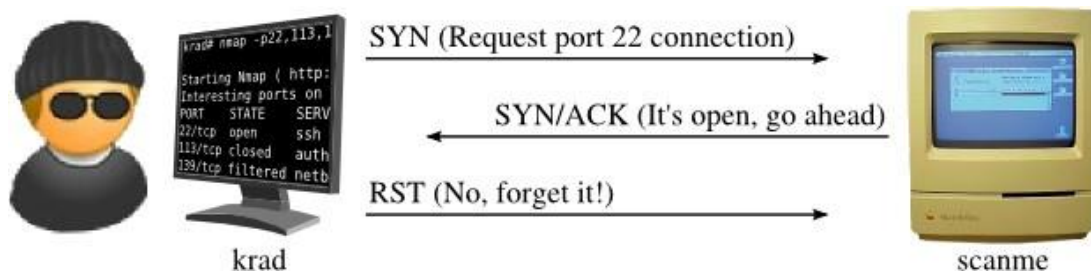


Figure 3.13: Example of SYN request and open port. Source [8]

If the receiver responds with a RST packet, the port it is closed (Figure 3.14).

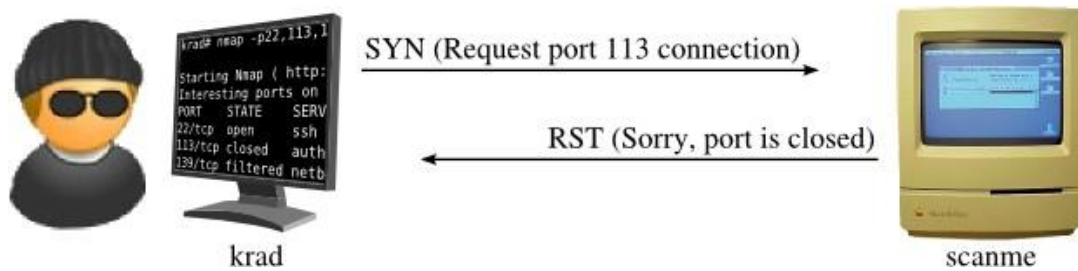


Figure 3.14: Example of SYN request and closed port. Source [8]

But if there is no respond after a few tries of connection, the port it is filtered by a firewall or other type of protection that denies the respond to the sender request (Figure 3.15).

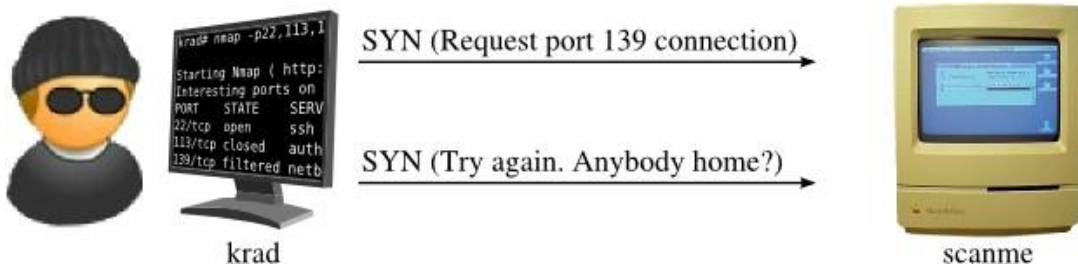


Figure 3.15: Example of SYN request and filtered port. Source[8]

Although a TCP SYN scan is the default option, Nmap is able to perform multiple types of scans:

- **TCP connect scans:** Nmaps asks the OS to make a connection if it does not have privileges to create a packet by itself
- **UDP scans:** scans for UDP ports, though take much more time because it is necessary to wait for an answer that may not come.
- **SCTP INIT scans:** SCTP equivalent scan to TCP SYN.
- **TCP NULL, FIN, and Xmas scans:** Exploit a subtle loophole in the TCP RFC to differentiate between open and closed ports.
- **TCP ACK scans:** It is used to map firewall rule sets.

- **TCP Windows scans:** Like the TCP ACK but using vulnerabilities from Windows machines.
- **TCP Maimon scans:** The same as NULL, FIN and Xmas scans but using FIN/ACK.
- **Custom TCP scans:** Allows to personalize the scanflags for advanced users.
- **SCTP COOKIE ECHO scans:** More advanced SCTP scan. Takes advantage that the drop of packets containing COOKIE ECHO chunks should be silent in the SCTP implementation.
- **Idle scans:** Uses a zombie host and does not use the address of the attacker.
- **IP protocol scans:** Allows to determine the IP protocols that are being used.
- **FTP bounce scans:** Scanning ports via a remote FTP server.

To exemplify the usage of this tool, it has been used a metasploitable2 machine as a scanned machine, and a Kali machine as the scanning machine:

1. With the command `nmap [IP-Address]`, it is possible to make a quick scan to see what ports and services does the machine have open(Figure 3.16).

This is a machine that provides lots of services such as FTP, SSH, HTTP, etc.

2. With the option `-sV` Nmap detects the versions of these services and with the `-sC` option, nmap will run a default script to grab more detailed information (Figure 3.17).

```
File Actions Edit View Help
(kali@kali)-[~]
└─$ nmap 192.168.252.134
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-12 05:54 EST
Nmap scan report for 192.168.252.134
Host is up (0.00055s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
```

Figure 3.16: Result of scan metasploitable2

```

kali@kali:~$ nmap -vv -sC 192.168.252.134
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-12 10:15:57 EST
Nmap scan report for 192.168.252.134
Host is up (0.0026s latency).
Net shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-syst
|_STAT:
|_FTP server status:
|_  Connected to 192.168.252.134
|_  Logged in as ftp
|_  TYPE: ASCII
|_  No session bandwidth limit
|_  Session timeout in seconds is 300
|_  Control connection is plain text
|_  Data connections will be plain text
|_  vsFTPd 2.3.4 - secure, fast, stable
|_End of status
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh          OpenSSH 4.7p1 Debian Aubuntu1 (protocol 2.0)
|_ssh-hostkey:
|_  1824 88:8f:cf:e1:c0:5f:6a:74:d8:98:24:fa:ca:d5:8c:cd (DSA)
|_  2848 56:56:2a:0f:21:1d:dca:7:2b:ae:01:01:2a:3d:e8:f3 (RSA)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
|_sslv2:
|_SSLv2 supported
|_ciphers:
|_  SSL2_DES_64_CBC_WITH_MD5
|_  SSL2_DES_192_EDE3_CBC_WITH_MD5
|_  SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|_  SSL2_RC3_128_CBC_WITH_MD5
|_  SSL2_RC4_128_EXPORT40_WITH_MD5
|_  SSL2_RC4_128_WITH_MD5
|_ssl-cert: Subject: commonName=ubuntu@04-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX
|_Not valid before: 2010-03-17T14:07:45
|_Not valid after: 2010-04-16T14:07:45
|_smtp-command: metaspl0itable.localdomain, PIPELINING, SIZE 1024000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
|_ssl-date: 2022-01-12T15:59:09+00:00; +10s from scanner time.
23/tcp    open  domain      ISC BIND 9.4.2
|_dns-nsid:
|_  bind.version: 9.4.2
88/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
131/tcp   open  rpcbind    2 (RPC #10000)
|_rpcinfo:
|_  program version  port/proto  service
|_  100000 2          111/tcp    rpcbind
|_  100000 2          111/udp    rpcbind
|_  100003 2 3 4        7000/rpc   nfs

```

Figure 3.17: Result of scan metasploitable2 with services version

3.3.4 Metasploit

Is one of the most used open-source penetration testing frameworks and it was released in 2003 by H. D. Moore as a portable network tool using Perl [9]. In 2007 it was rewritten in Ruby and in 2009 it was bought by Rapid7, a security company that provides vulnerability management solutions. It has a huge amount of exploits to use. Metasploit helps us with:

- Check vulnerabilities
- Using exploits for those vulnerabilities
- Setting and configuring an exploit
- Choosing and configuring a payload
- Execute the exploit

Let's check for vulnerabilities from the previous nmap scan:

1. After we gather this information, we can go to Metasploit console and search for possible vulnerabilities (Figure 3.18).
2. In this case, we have searched for a vulnerability in vsftpd v2.3.4, and we have found that it exists an exploit from 2011 ranked as excellent (Figure 3.19).
3. We can search for more vulnerabilities like vnc 3.3. In this case we can find that they exist some exploits for this protocol too, but they have a lower rank (Figure 3.20).

```

File Actions Edit View Help
(kali@kali)-[~]
└─$ msfconsole

      .:ok000kz~
      .00000000000c      'c0k000ko:
      :0000000000000k,      ,k0000000000000:
      '00000000kkk00000; :000000000000000'
      00000000.      .000000000l.      ,00000000
      00000000.      .c00000c.      ,00000000k
      |0000000.      ;d;      ,00000000l
      ,00000000.      ;:      ;      ,00000000.
      <0000000.      ,00c.      'e00.      ,0000000c
      e000000.      ,0000.      :0000.      ,000000e
      100000.      ,0000.      :0000.      ,000000l
      ,0000'      ,0000.      :0000.      :0000;
      ,d000 .0000eccc0000.      x00d,
      ,k0l ,0000000000000.      ,00k,
      :kk; ,0000000000000.      c0k:
      ;k000000000000000k;
      ,x00000000000k,
      ,10000000l.
      ,d0d,
      .
      .

+ -- --[ metasploit v6.1.14-dev ]
+ -- --[ 2180 exploits - 1155 auxiliary - 399 post ]
+ -- --[ 592 payloads - 45 encoders - 10 nops ]
+ -- --[ 9 evasion ]

Metasploit tip: To save all commands executed since start up
to a file, use the makerc command

```

Figure 3.18: Entry message of metasploit

```

msf6 > search vsftpd 2.3.4
Matching Modules
-----
#  Name                                     Disclosure Date  Rank     Check  Description
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
0  exploit/unix/ftp/vsftpd_234_backdoor     2011-07-03     excellent No      VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor
msf6 >

```

Figure 3.19: Results of vsftpd 2.3.4

```
File Actions Edit View Help
msf6 > search vnc 3.3
Matching Modules
-----
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  exploit/windows/vnc/realvnc_client        2001-01-29      normal No      RealVNC 3.3.7 Client Buffer Overflow
1  auxiliary/scanner/vnc/vnc_login          2001-01-29      normal No      VNC Authentication Scanner
2  exploit/windows/vnc/winvnc_http_get      2001-01-29      average No      WinVNC Web Server GET Overflow

Interact with a module by name or index. For example info 2, use 2 or use exploit/windows/vnc/winvnc_http_get
msf6 > |
```

Figure 3.20: Results of vnc 3.3

3.3.5 Sqlmap

It is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws [10]. It is also able to exploit the found vulnerabilities to look Databases and take their information.

Some of its features are:

- Full Support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Maria DB ...
- It supports six SQL Injection techniques:
 - Boolean-based blind
 - Time-based blind
 - Error-based
 - UNION query-based
 - Stacked queries and out-of-band
- Can enumerate users, password hashes, privileges, roles, databases, tables and columns
- Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.

We are going to attack the metasploitable2 machine, the mutillidae application:

1. Get a request with a proxy, in this case it has been used Burp Suite (Figure 3.21).
2. Save the request in a .txt to use it with sqlmap (Figure 3.22).
3. Use the request previously saved in the sqlmap -r instruction (Figure 3.23).
4. If it is injectable, there should be something like this, where it shows us which is the database type and the types of injections that can be made (Figure 3.24).



Figure 3.21: Intercept request with burpsuite

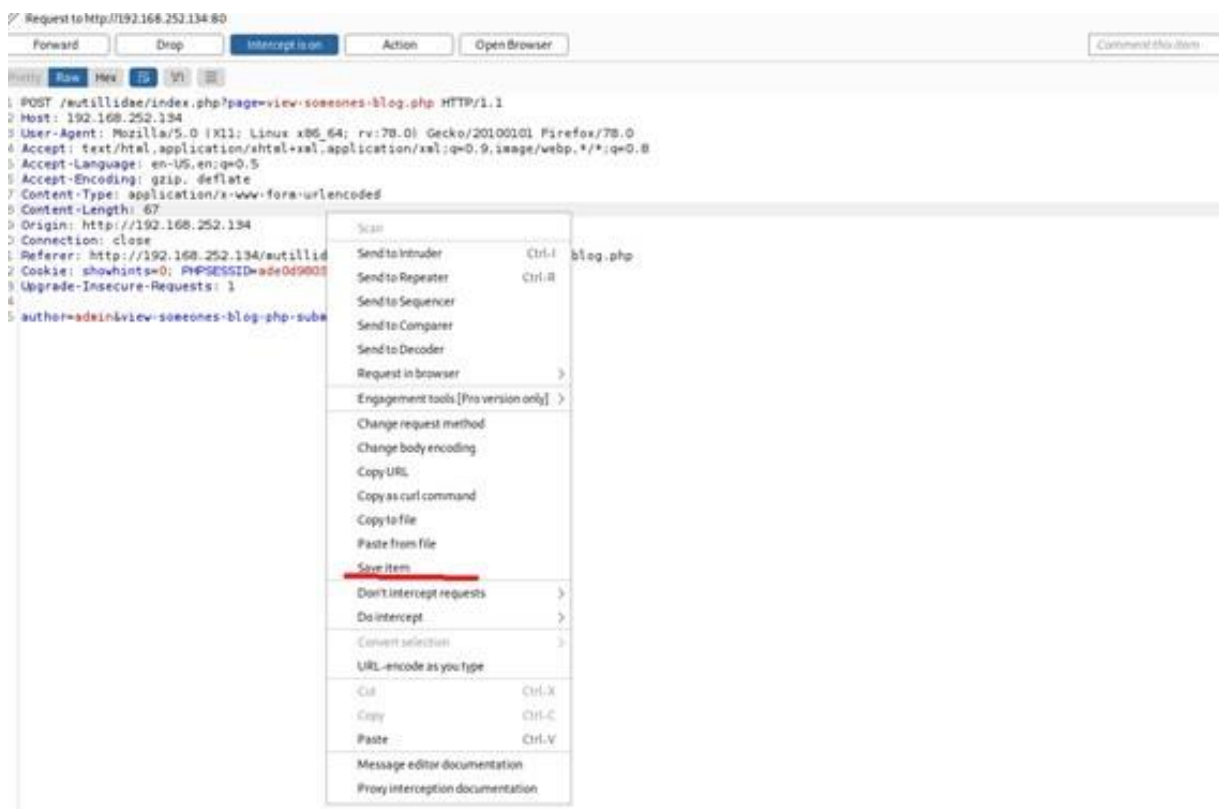


Figure 3.22: Save request in burpsuite

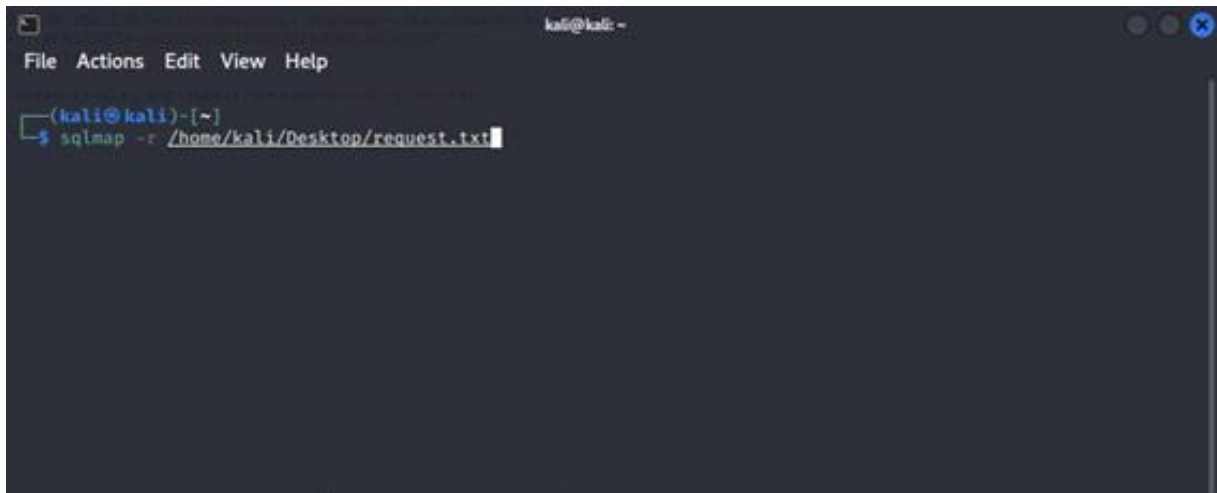


Figure 3.23: Execute sqlmap with saved request

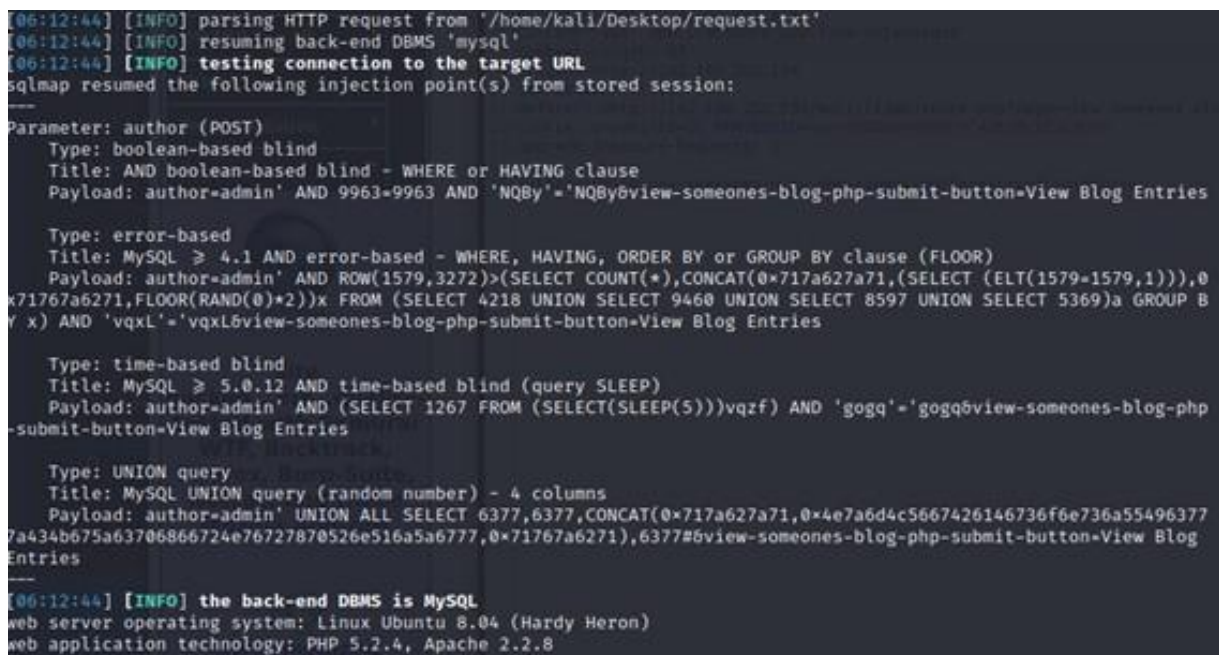


Figure 3.24: Results of sqlmap

3.3.6 WPScan

The WPScan CLI tool is a free, for non-commercial use, black box WordPress security scanner written for security professionals and blog maintainers to test the security of their sites [11].

It checks for:

- WordPress version installed and associated vulnerabilities
- Which plugins and themes are installed and if they are vulnerable
- Username enumeration
- Users with weak password via password brute forcing
- Backed up and public accessible wp-config.php files
- ...

An example of the usage of this tool can be seen below (Figure 3.25).

```
root@kali:/home/kali# wpscan --url pentest.id --enumerate tt
-----
  W P S C A N
-----
WordPress Security Scanner by the WPScan Team
Version 3.7.6
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

-----
[+] URL: http://pentest.id/
[+] Started: Fri May 29 10:48:59 2020

Interesting Finding(s):

[+] http://pentest.id/
  Interesting Entry: Server: Apache
  Found By: Headers (Passive Detection)
  Confidence: 100%

[+] http://pentest.id/robots.txt
  Interesting Entries:
  - /wp-admin/
  - /wp-admin/admin-ajax.php
  Found By: Robots Txt (Aggressive Detection)
  Confidence: 100%

[+] http://pentest.id/xmlrpc.php
  Found By: Link Tag (Passive Detection)
  Confidence: 30%
  References:
  - http://codex.wordpress.org/XML-RPC_Pingback_API
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access

[+] http://pentest.id/readme.html
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%

[+] http://pentest.id/wp-cron.php
  Found By: Direct Access (Aggressive Detection)
```

Figure 3.25: Results of wpscan

3.3.7 Joomscan

JoomScan is an open-source project, developed with the aim of automating the task of vulnerability detection and reliability assurance in Joomla CMS deployments[12]. It was released in 2018 by OWASP (Open Web Application Security Project) and it is implemented in Perl.

It is only capable of detecting already known vulnerabilities and detect many misconfigurations that may led to vulnerable situations. Some of its features are:

- Version enumeration
- Vulnerability enumeration
- Components enumeration
- Components vulnerability enumeration
- Firewall detection
- Finding common log files
- Finding common backup files

Even if it has all this features, it has its limitations:

- The database still lacks of unknown exploit checks
- Lacks IDS (Intrusion Detection System) evasion bypass
- Lacks sophisticated fuzzing
- It is not a full fledged SQL Injection tool

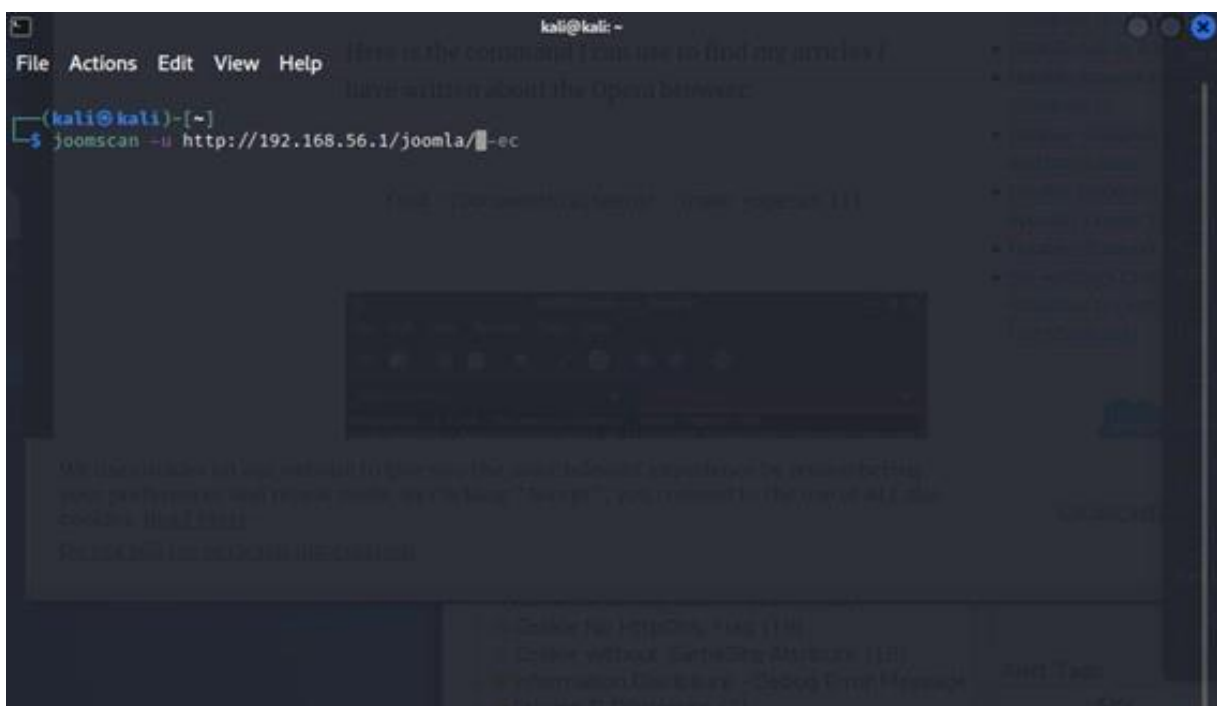
To perform a scan of a Joomla website, the scanner create a series of GET Requests to identify possible vulnerabilities. For example, to get the Joomla version that is being used, a request with the url:

`https://[HOST]/administrator/manifests/files/joomla.xml` is sent and the returned XML

file contains the version among other things. If the asked resource does exist, Joomscan checks for a possible vulnerability, depending of the type, it may require the usage of a sample exploit string, but if the exploit string is not available, it works out the vulnerability state with version deduced.

Using this tool does not require lots of arguments, with just the url, it performs a basic scan that checks all the vulnerabilities previously mentioned. In the next example, a Joomla website is being scanned with a kali machine:

1. To check a web application, we have to introduce the parameter `-u` to and its url (Figure 3.26).
2. After that, it should appear a report from the scan. As we can see it shows information about the joomla version, which are the directories that we can find ... (Figure 3.27).
3. With the `-ec` option we can enunmerate all the components that are being used (Figure 3.28).
4. If there is any vulnerability it should appear.



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)~  
$ joomscan -u http://192.168.56.1/joomla/ -ec
```

The screenshot shows a terminal window with a dark background. At the top, the window title is "kali@kali: ~". Below the title bar, there is a menu bar with "File", "Actions", "Edit", "View", and "Help". The terminal prompt is "(kali@kali)~". The user has entered the command "joomscan -u http://192.168.56.1/joomla/ -ec". The output of the command is partially visible, showing a progress bar and some text, but it is mostly obscured by a large, dark, semi-transparent box in the center of the terminal. The bottom of the terminal shows some faint text, likely the output of the joomscan command, including "Check for HTTP(S) ...", "Check without ...", and "Information ...".

Figure 3.26: joomscan command to attack host


```
File Actions Edit View Help
[+] Enumeration component (com_contact)
[++] Name: com_contact
Location : http://192.168.56.1/joomla/components/com_contact/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_contact/

[+] Enumeration component (com_content)
[++] Name: com_content
Location : http://192.168.56.1/joomla/components/com_content/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_content/

[+] Enumeration component (com_contenthistory)
[++] Name: com_contenthistory
Location : http://192.168.56.1/joomla/components/com_contenthistory/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_contenthistory/

[+] Enumeration component (com_fields)
[++] Name: com_fields
Location : http://192.168.56.1/joomla/components/com_fields/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_fields/

[+] Enumeration component (com_finder)
[++] Name: com_finder
Location : http://192.168.56.1/joomla/components/com_finder/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_finder/

[+] Enumeration component (com_media)
[++] Name: com_media
Location : http://192.168.56.1/joomla/components/com_media/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_media/

[+] Enumeration component (com_newsfeeds)
[++] Name: com_newsfeeds
Location : http://192.168.56.1/joomla/components/com_newsfeeds/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_newsfeeds/

[+] Enumeration component (com_users)
[++] Name: com_users
Location : http://192.168.56.1/joomla/components/com_users/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_users/

[+] Enumeration component (com_wrapper)
[++] Name: com_wrapper
Location : http://192.168.56.1/joomla/components/com_wrapper/
Directory listing is enabled : http://192.168.56.1/joomla/components/com_wrapper/
```

Figure 3.28: Enumeration of the components

3.3.8 TLSSLed

TLSSLed is a Linux shell script whose purpose is to evaluate the security of a target SSL/TLS (HTTPS) web server implementation [13]. Some of the tests that it performs are:

- Check which versions of SSL support
- Check which versions of TLS support
- NULL cipher
- The strength of the cipher method based on key length
- The availability of strong ciphers
- If the digital certificate is MD5 signed
- Renegotiation capabilities of the server

We are going to check the SSL/TLS from our Joomla server:

1. You run the command `tlssled [IP address] [Port]` (Figure 3.29).



```
(kali@kali)-[~]
└─$ tlssled 192.168.56.1 443
```

Figure 3.29: TLSSLed command on host 192.168.56.1 443

2. Once the command has finished, we have the results in different files (Figure 3.30).



```
openssl_HEAD_1.0_192.168.252.134_00_20220112-140413.err  openssl_HEAD_192.168.252.134_00_20220112-140413.log  openssl_RENEG_LEGACY_192.168.252.134_00_20220112-140413.err
openssl_HEAD_1.0_192.168.252.134_00_20220112-140413.log  openssl_RENEG_192.168.252.134_00_20220112-140413.err  openssl_RENEG_LEGACY_192.168.252.134_00_20220112-140413.log
openssl_HEAD_192.168.252.134_00_20220112-140413.err  openssl_RENEG_192.168.252.134_00_20220112-140413.log  sslscan_192.168.252.134_00_20220112-140413.log
```

Figure 3.30: Files where the results are stored

3. For example, if we open the `sslscan` file, we can see which TLS versions are enabled and if there are yellow, it might be a problem (Figure 3.31).

```
(kali@kali)-[~/TLSSled_1.3_192.168.56.1_443_20220112-145530]
└─$ cat sslscan 192.168.56.1 443 20220112-145530.log
Version: 2.0.10-static
OpenSSL 1.1.1l-dev  xx XXX xxxx

Connected to 192.168.56.1

Testing SSL server 192.168.56.1 on port 443 using SNI name 192.168.56.1

SSL/TLS Protocols:
SSLv2      disabled
SSLv3      disabled
TLSv1.0    enabled
TLSv1.1    enabled
TLSv1.2    enabled
TLSv1.3    enabled
```

Figure 3.31: Result of ssl scan

3.4 Web Technologies

This section details the possible web technologies and determines which is the best for the project.

3.4.1 Front-end

The Front-end is the part of the web application users interact to make requests and receive data. This is the most visible part of the website and there are different solutions:

- **HTML:** or HyperText Markup Language, it is the basic of the web. Defines the structure and the content of all the web.
- **CSS:** or Cascading Style Sheets, it is used to describe how all the elements of the page are rendered.
- **JavaScript:** is a programming language very popular in the web environment because it is used as a scripting language. It is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting, imperative, and declarative styles. JavaScript has very powerful libraries and Frameworks that can help to build dy-

namic front-end:

- **Vue.js:** is a JavaScript framework for building user interfaces. The two main features of Vue.js are Declarative Rendering, which allows changing the HTML state depending on the JavaScript state, and Reactivity, meaning that Vue.js can track the JavaScript state to efficiently update the DOM.
- **React.js:** is a JavaScript library developed by Facebook Inc. As Vue.js, it also uses a Declarative Rendering and Reactivity. A peculiar thing about React is that uses a language called JSX for the rendering. This language is a mix of HTML and JavaScript.
- **Angular:** is a JavaScript framework developed by Google. It has a similar functionality as Vue.js and React, but instead of working with a virtual DOM, Angular uses the real DOM. It is also known for a pronounced learning curve.

The front-end of the application uses HTML, CSS, JavaScript and Vue.js. After comparing Vue.js, React.js and Angular, the framework that better suits the project is Vue.js. As it can be seen in figure3.32 and in figure 3.33 Angular is not well regarded in the community, excluding it from the viable options.



Figure 3.32: Poll from stateofjs [14]



Figure 3.33: Poll from stateofjs [14]

This leaves the contest to Vue.js and React.js. Both are flexible and scalable tools. But Vue.js uses plain HTML which is easier to migrate than JSX, being the first more versatile. Also in the next figure (Figure 3.34), it can be seen that Vue.js performs a little better in general and decanting the balance in favour of Vue.js.

Vue and React Performance Comparison		
Name	 React.js (ms)	 Vue.js (ms)
Create rows Duration for creating 1000 rows after the page loaded	187.6 ± 4.3	169.2 ± 3.6
Replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations)	165.2 ± 7.0	161.8 ± 3.9
Partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows	93.6 ± 5.6	168.1 ± 7.4
Select row Duration to highlight a row in response to a click on the row (with 5 warmup iterations)	12.4 ± 4.1	9.8 ± 2.5
Swap rows Time to swap 2 rows on a 1K table (with 5 warmup iterations)	19.6 ± 4.7	21.8 ± 4.5
Remove row Duration to remove a row (with 5 warmup iterations)	51.5 ± 2.0	52.5 ± 1.8
Create many rows Duration to create 10,000 rows	2033.7 ± 32.0	1521.4 ± 55.7
Append rows to large table Duration for adding 1000 rows on a table of 10,000 rows	271.8 ± 9.9	338.4 ± 10.3
Clear rows Duration to clear the table filled with 10,000 rows	224.4 ± 6.0	240.9 ± 11.4
Startup time Time for loading, parsing and starting up	49.4 ± 0.7	48.4 ± 2.4

Figure 3.34: Performance comparison. Source [15]

3.4.2 Back-end

The back-end responds to what the front end has initiated [16]. It is the most computational part of a web page, and some of the most used languages are:

- **Python:** is a multiparadigm, general-purpose, interpreted, high-level programming language [17]. It is useful to process text, which is necessary in this project because some of the previous mentioned vulnerability analysis tools return outputs in raw text.

- **PHP:** or Hypertext Preprocessor, it is one of the top ten most popular programming languages as it can be seen in figure 3.35. Also the Laravel framework provides easy to build web applications. It helps with the connection to the database, session based authentication and more [18].

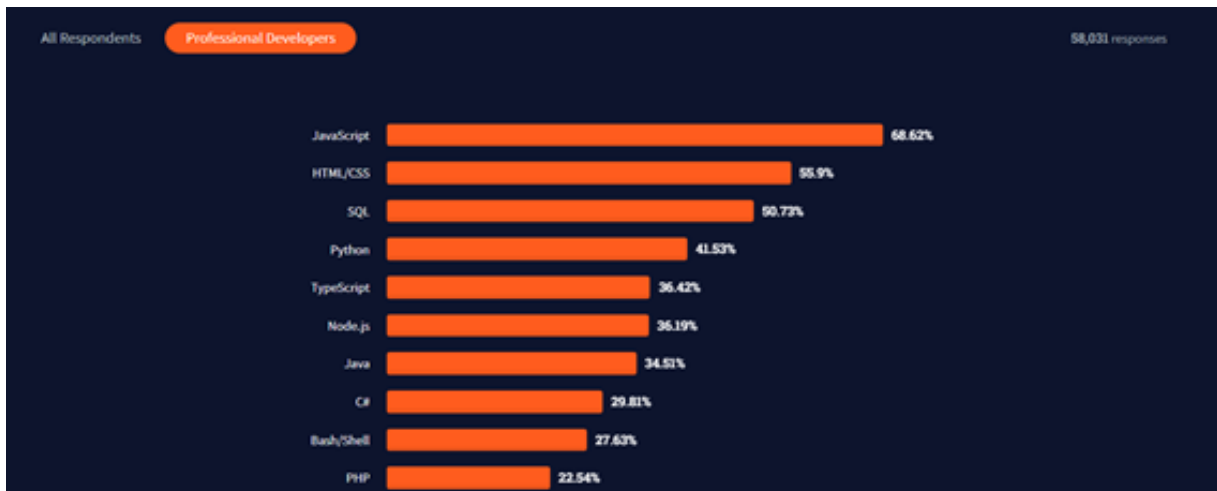


Figure 3.35: Poll from stackoverflow[19]

These are the two languages that are used in the project because ESED works with them.

4. Methodology

In order to accomplish the objectives that have been described earlier, the principles of the Agile Manifesto are taken into account. This means that there are meetings twice a month with both, the academic and the business tutor. The changes are welcome, and the top priority is to deliver value in the moment rather than a work in progress.

To follow these guidelines, the project is divided into different stages and with each stage having its own tasks:

1. Analysis and Investigation
 - (a) Investigate possible solutions
 - (b) Analyse which one of the possible solutions is the best for the project
2. Design and architecture
 - (a) Design the implementation
 - (b) Take into account the requirements
3. Implementation
 - (a) Develop the solution
4. Testing the implementation
 - (a) Test the requirements
 - (b) Test if the users can use the application without help
5. Evaluation of the results
 - (a) Check if the results align with the expected output
 - (b) Evaluate what has gone wrong
 - (c) Investigate ways to improve this last iteration
 - (d) Schedule the next iteration

Even though the stages and tasks are written in a sequence, this does not mean that they are followed just one time. They are a cycle of N iterations.

To make the evaluation of the project, it is important to notice that two different approaches are used, the first one in the academic part, and the second one in the business part:

1. **Academic:** The meetings take place two Mondays per month and, if it is possible, they are face-to-face. If it is not possible to be in the same room with the tutor, the meeting takes place in Zoom. Mail is used to be in contact.

The text editor platform used is Overleaf and it is where the tutor makes the corrections of the thesis. In this part of the project, the references are very important.

2. **Business:** There are meetings on Fridays twice a month. These meetings take place in Google Meet and have a duration of half an hour. Doubts and requests can be done in Google chat. The project is planned in the web application clickup. For this part of the project is very important to test if the implementations perform as expected.

5. Development

5.1 Definition of the functional and technological requirements

In order to be a satisfactory service, it has to accomplish different functional and technological requirements.

5.1.1 Functional Requirements

- The service must be available 24x7 online
 1. Check the service status continuously
- The service must be able to register and manage hosts
 1. Register host
 2. Check that the host is registered
 3. Remove host
 4. Check if the host is removed
- The service must be able to run a vulnerability analysis
 1. Define the attacked host
 2. Define the type of scan
 3. Check if the scan has started
- The service must be able to show the progress of an analysis
 1. Ask for the progress of a scan
 2. Return if the scan is in progress
 3. Return if the scan is finished

4. Return if the scan has failed
- The service must be able to generate and show a report of the analysis
 1. Ask for the results of a scan
 2. Return the results of the scan
 - The service must be able to perform multiple scans on the same host
 1. Define the attacked host
 2. Define the type of scan
 3. Check if the scan has started

5.1.2 Technological Requirements

- The service must be hosted online in a private machine
 1. The host can only be accessible to a specific user
 2. Check if the host functions as expected
- The service must be able to save more than one host
 1. Register one host
 2. Register a second host
 3. Check if both hosts exist
- The service must be able to make a vulnerability analysis
 1. Define a host
 2. Define the type of scan
 3. Check if the scan has started
- The service must be able to generate a report from an analysis
 1. Initialize a vulnerability analysis on a host

2. Check if it shows the results as a report
- The service must be able to store all the reports from one host
 1. Initialize a vulnerability analysis on a host
 2. Check if the report is stored in the database

5.2 Development Process

5.2.1 Design of the web application

The web application is divided in four parts: the front end, the back end, the scan server and the database.

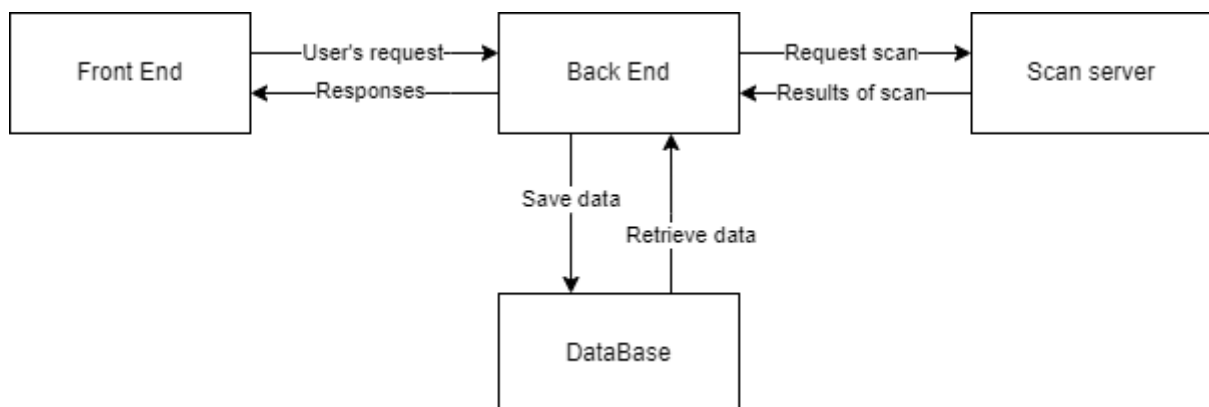


Figure 5.1: Diagram of the application

Front End

This part of the application is the visible part for the final user. From this part, the user is able to:

- Register to the application

- Login
- Do a scan
- See the results of the scan

Back End

All the users' requests are processed in this part of the application. It is in charge of:

- Register a host
- Send a scan request to the scan server
- Process the results of the scan
- Save the results of the scan in the database

Scan server

The scan server is where all the attack tools are used. The back-end sends a request with all the parameters needed to create a scan, this server processes it and sends the results in a raw JSON. This server is able to use different types of tools. Nessus, Metasploit, and Sqlmap could not enter due to lack of time. There are five tools available:

- Nmap
- OWASP ZAP
- Traceroute
- TLSSLed
- Joomscan
- WPScan

Database

This part is in charge of saving all the information relative to:

- Hosts' information
- Failed scans
- Results of the scans
- Raw data of the scans

5.2.2 Code Tools Integration

All the tools used for this project are commonly used in a manual way, the user introduces the commands necessary to perform an action and the tool returns something, generally text, that the user can understand. The objective for this part of the project is to introduce the commands and interpret the data with a script. This allows to automatize and control the scans from the back-end.

To be able to automatize the different tools, there must be a first phase of communication where it is defined what the tools must do, and a second phase of gathering the results. There are tools that the first and second phase is performed with a simple API provided by the same tool, but there are others that must be executed with commands, and the results gathered also from the command prompt.

In the next subsections, it is detailed the process followed for each of the different tools available now in the project.

Nmap

Nmap does not have an API to make scans and gather results, but some people have made libraries to integrate Nmap with Python. The library used for this project is called `python-nmap` [20] and it allows the user to perform scans with simple python commands

and receive the result in an array.

To create a scan with this library it is necessary to import the library, invoke the method `PortScanner()` and invoke the method `scan()` (Figure 5.2).

```
>>> import nmap
>>> nm = nmap.PortScanner()
>>> nm.scan('127.0.0.1', '22-443')
```

Figure 5.2: Import and usage of python-nmap. Source [20]

The scan method can accept arguments like the IP of the host, the ports to be scanned, and the arguments to be used among others. This function executes a Nmap command and waits for the results. When Nmap has ended the scan, the results can be retrieved from an array that python-nmap creates (Figure 5.3).

```
>>> for host in nm.all_hosts():
>>>     print('-----')
>>>     print('Host : %s (%s)' % (host, nm[host].hostname()))
>>>     print('State : %s' % nm[host].state())
>>>     for proto in nm[host].all_protocols():
>>>         print('-----')
>>>         print('Protocol : %s' % proto)
>>>
>>>         lport = nm[host][proto].keys()
>>>         lport.sort()
>>>         for port in lport:
>>>             print ('port : %s\tstate : %s' % (port, nm[host][proto][port]['state']))
>>>
-----
Host : 127.0.0.1 (localhost)
State : up
-----
Protocol : tcp
port : 22    state : open
port : 25    state : open
port : 80    state : open
port : 111   state : open
port : 443   state : open
```

Figure 5.3: Example of result of a scan. Source [20]

In the actual state of the project, this script is able to create a scan with the IP, the ports, and the arguments of a Nmap normal scan and return the results in a JSON response.

OWASP ZAP

The OWASP ZAP software has an option to enable an API mode [21] Figure 5.4. With this API it is possible to ask the same things as if it was used manually. It also has a library for Python [22] that allows to make the API calls from it.

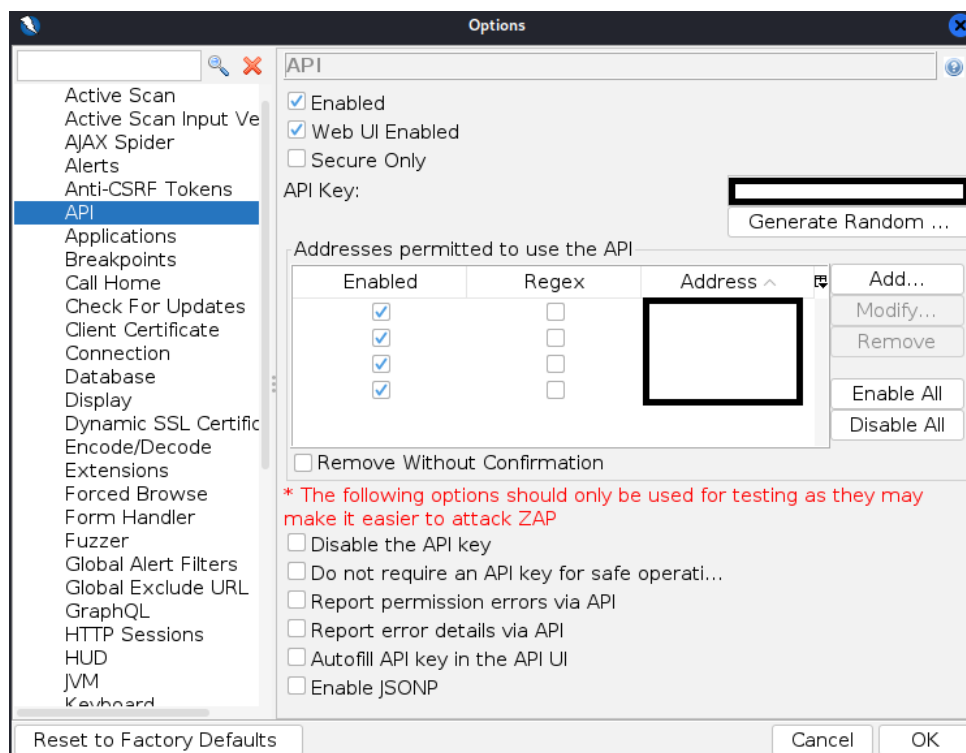


Figure 5.4: Window to activate the API

The possible requests that can be done to the scan server can perform spider attacks to index a targeted web application and perform active scans to discover possible vulnerabilities.

Traceroute

This tool was implemented in the last place and asked for ESED. It is capable to know the route that the user follows to reach the destination address. For this tool, there are no APIs, libraries, or saves the result in a .txt. To gather the information that this tool gives, it has been necessary to analyze the output of the machine. The subprocess module [23] can save the output when the `capture_output` is set to `True`. In the Figure 5.5 we can see an example of the execution of this tool.

```
process = subprocess.run(['traceroute', '-I', url], capture_output=True)
output = process.stdout.decode("utf-8").splitlines()
```

Figure 5.5: Example of capturing the output of traceroute

TLSSLed

This tool does not have any API or parser to gather the output. TLSSLed creates a folder with the results of the scan in plain text. To gather the results, a script of Python deletes the previous scan and then processes the results which are saved as JSON data. To know which one is the file that contains the output, first, the output is captured with the subprocess modules, as seen earlier, and then the script searches the file and the path (Figure 5.6).

```
directory = ""
for x in output:
    if "    [.] Output directory:" in x:
        directory = x.replace("    [.] Output directory: ", "")
        directory = directory.replace(" ...", "")
        break

output_file = "sslscan" + directory.replace("TLSSLed_1.3", "") + ".log"
path = "./" + directory + "/" + output_file
```

Figure 5.6: The script finds where is the output and saves the path to use it later

Joomscan

Joomscan functions similar to TLSSLed. It creates an output that is saved in a folder. To convert the output to a JSON format, a Python script analyses the different rows of the output.txt. The output of the scan is delimited by marks that make it easier to process (Figure 5.7).

```
[+] FireWall Detector
[++] Firewall not detected

[+] Detecting Joomla Version
[++] Joomla 4.0.5

[+] Core Joomla Vulnerability
[++] Target Joomla core is not vulnerable

[+] Checking Directory Listing
[++] directory has directory listing :
http://192.168.56.1/joomla/administrator/components
http://192.168.56.1/joomla/administrator/modules
http://192.168.56.1/joomla/administrator/templates
http://192.168.56.1/joomla/images/banners
```

Figure 5.7: The [+] signs define what are they checking, and the [++] signs are the results of that check

WPScan

To this tool there is a library called `wpscan_out_parse`[24]. This library can be used as a client or as a Python library. This project uses it as a library. This parser returns the results as a JSON, it shows the info, warnings, and alert messages (Figure 5.8).

```
{
  'infos': [],
  'warnings': [],
  'alerts': [],
  'summary': {
    'table': None,
    'line': 'WPScan result summary: alerts={}, warnings={}, infos={}, error={}'
  },
  'error': None
}
```

Figure 5.8: Structure that returns the `wpscan_out_parse` library

5.2.3 Code Front-end

This part of the project was abandoned because ESED does not need the front-end to use the service. They use the service with the API. The developed part uses HTML, CSS, JavaScript and Vue. It consists of different views:

A welcome view (Figure 5.9), where the user can select if he wants to go to the login page or the register page.

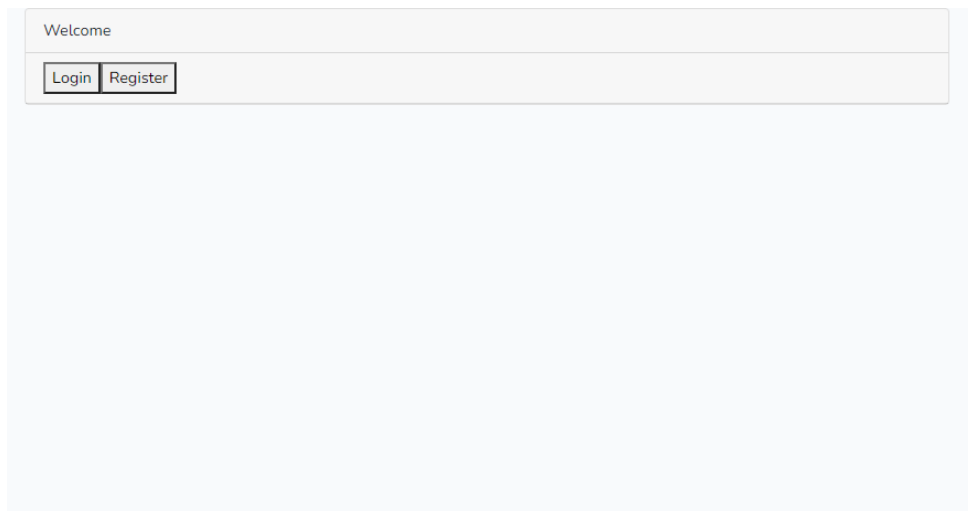


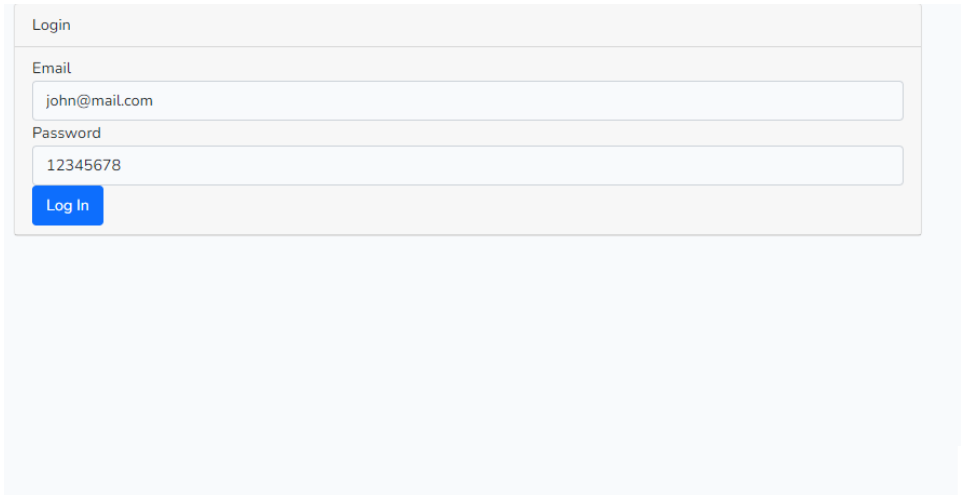
Figure 5.9: Welcome page

A register page (Figure 5.10), to register the personal information of the user.



Figure 5.10: Register page

The login page (Figure 5.11), the normal entrance point of a known user.

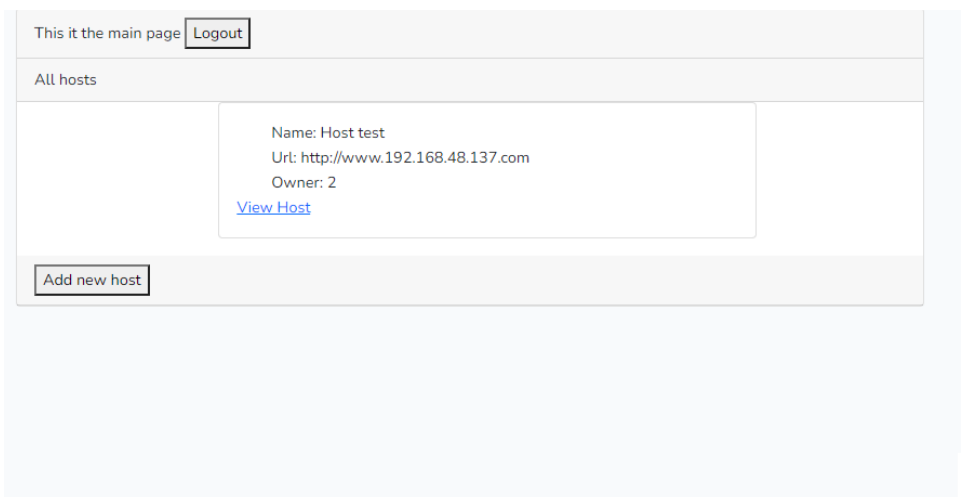


The screenshot shows a login form with the following elements:

- Login** header
- Email** field containing the text "john@mail.com"
- Password** field containing the text "12345678"
- Log In** button

Figure 5.11: Login page

The main page (Figure 5.12) where the user can find all the hosts created by him.



The screenshot shows a main page with the following elements:

- This is the main page** header with a **Logout** button
- All hosts** section containing a list of hosts. One host is visible with the following details:
 - Name: Host test
 - Url: <http://www.192.168.48.137.com>
 - Owner: 2
 - [View Host](#) link
- Add new host** button

Figure 5.12: Main page

A host creation page (Figure 5.13), to create all the hosts of the user.

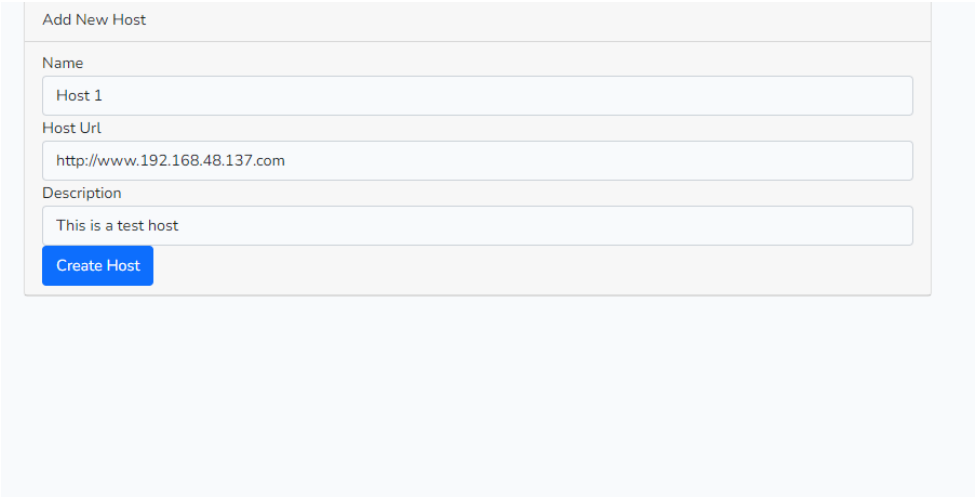


Figure 5.13: Create host page

A detailed view of a host (Figure 5.14), to see all the information about it.

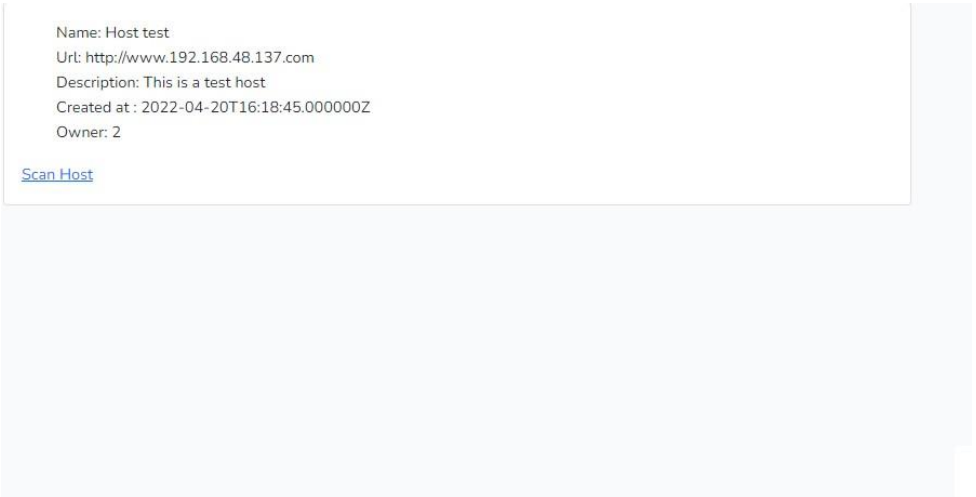


Figure 5.14: Host in detail

And the scan host page (Figure 5.15), where the user can decide to make a fast scan, a normal scan or a complete scan.

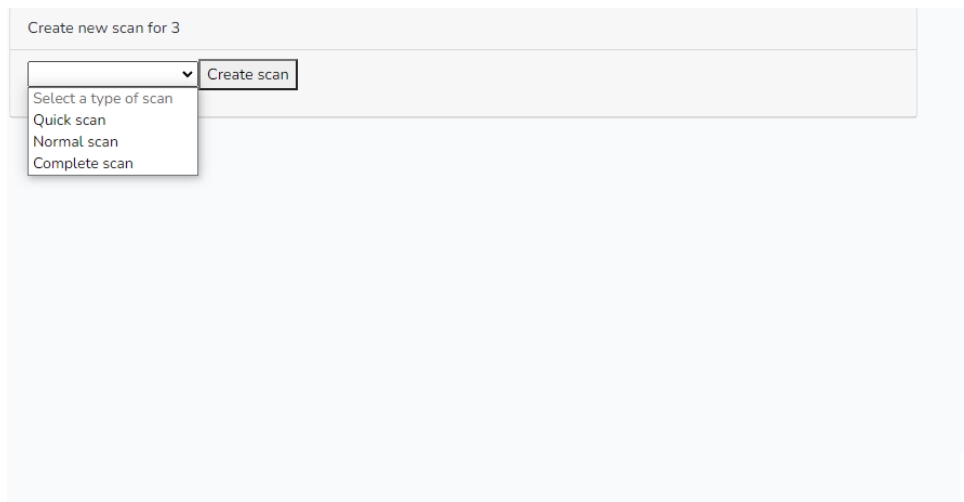


Figure 5.15: Create scan page

The user does not know the details of how this scan is performed, he can just see the results.

The result of the scan is returned in JSON, but it is explained in the next section.

5.2.4 Code Back-end

The back-end is programmed with PHP and the help of the Laravel framework. It is the core of the application and is where all the requests are processed. The process that takes place when a scan is requested follows the next path (Figure5.16):

1. The user sends a scan request
2. The scan launcher uses one of the three types of scanner classes to start the scan
3. The scan launcher informs that the scan has started successfully
4. The scanner class creates the jobs required to make the scan

5. Each job is executed and saves the result to the database
6. Once all the jobs have finished, the results of each one of them is retrieved and analysed giving to each scan a score that is saved in the database

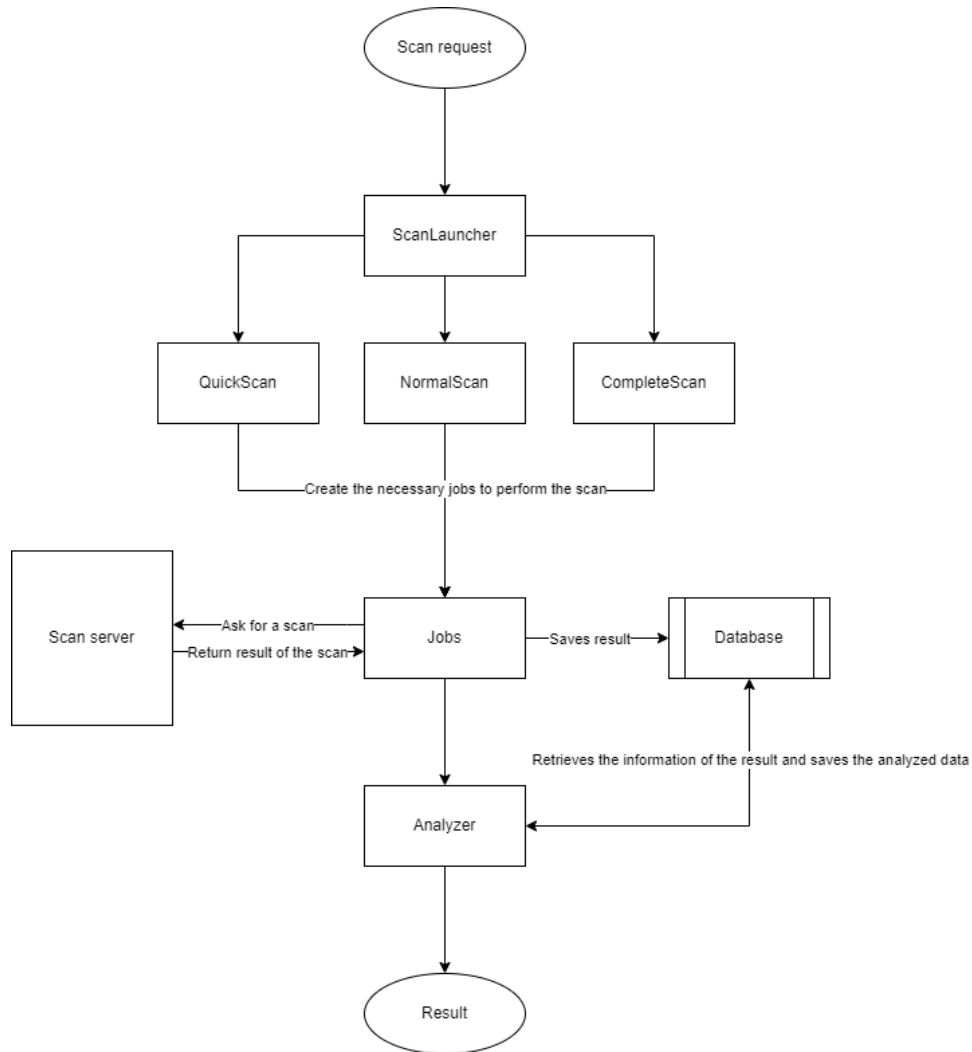


Figure 5.16: Diagram of the scan back end

API

The scan request is done with an API. The endpoints of the API are:

- **/api/targets**
 - method: GET
 - description: Returns a list of all the hosts
 - returns:
- **/api/targets**
 - method: POST
 - description: Adds a new target
 - body:
 - * url : String – mandatory
- **/api/targets/{target_id}**
 - method: DELETE
 - description: Deletes the target of the parameter target_id
 - parameter:
 - * target_id : Integer – mandatory
- **/api/scans**
 - method: POST
 - description: Starts an scan
 - body:
 - * target_id : Integer – mandatory
 - * type : {"fast", "normal", "complete"} – mandatory
- **/api/scans/{scan_id}**

- method: GET
- description: Returns the scan with the same scan_id
- parameter:
 - * scan_id : Integer – mandatory
- **/api/scans/{scan_id}/results**
 - method: GET
 - description: Returns the results of the scan with the same scan_id
 - body:
 - * scan_id : Integer – mandatory

The results of the scans are returned in an array. Each element of the array is another array that contains a title, description, remediation, and the severity of the problem (Figure 5.17).

```
{
  "title": "TLS Fallback",
  "description": "When TLS Fallback SCSV is not active, malicious agents can use older versions of ssl to exploit vulnerabilities.",
  "remediation": "Activate TLS Fallback SCSV.",
  "severity": 3
}
```

Figure 5.17: Example of result

5.2.5 Scan Server

The scan server is a Kali machine that contains the tools and scripts needed to perform the scans. The communication with the back-end is done thanks to an API programmed in Python that has seven endpoints:

- **/nmap**
 - method: POST

- body:
 - * url : String – mandatory
 - * ports : Integer – optional
 - * parameters : String – optional

- **/zap/spider**

- method: POST
- body:
 - * url : String – mandatory
 - * maxchildren : Integer – optional
 - * recurse : Boolean – optional
 - * contextname : String – optional
 - * subtreeonly : String – optional

- **/zap/scan**

- method: POST
- body:
 - * url : String – mandatory
 - * recurse : Boolean – optional
 - * inscopeonly : Boolean – optional
 - * scanpolicyname : String – optional
 - * method : String – optional
 - * postdata : String – optional
 - * contextid : String – optional

- **/tlssled**

- method: POST
- body:
 - * url : String – mandatory

- * port : Integer – mandatory
- **/joomscan**
 - method: POST
 - body:
 - * url : String – mandatory
- **/wpscan**
 - method: POST
 - body:
 - * url : String – mandatory
- **traceroute**
 - method: POST
 - body:
 - * url : String – mandatory

Each of the different endpoints calls to a function that executes the tool script and returns the result as a JSON.

At the beginning of the project, this Kali was a virtual machine with a GUI and some tools that were not required. Eshed requested to change the Kali machine from a virtual image to a Docker container. This change has allowed us to just select the tools that we need for the scans and has also allowed us to gain in performance.

5.2.6 Database

The database (Figure 5.18) is a MySQL database. It is generated automatically and controlled with Laravel. To create a new table, it is necessary to create first a model class that contains the attributes. Once the model class is created, it should appear a create migration file that contains the definition of the database.

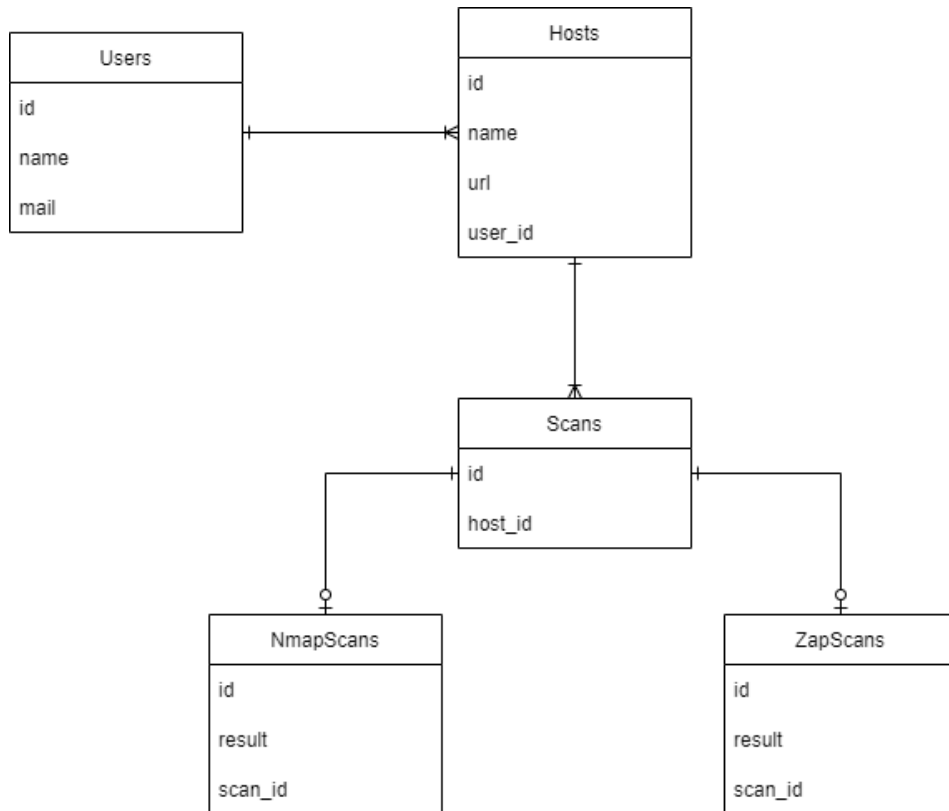


Figure 5.18: Diagram of the database

At the moment there is no need to use a non-relational database because the volume of data is small, but in the future, it may be required to save part of the data in a MongoDB database.

6. Conclusions

In this final degree project, we have developed a service to detect vulnerabilities in public hosts. The service is now running on an Ubuntu machine and is being used by ESED as another provider. From the technical part, the project fulfils the goals defined at the start of this document.

Even though it has not been possible to implement all the tools and features that we would like, the project has a solid base that allows us to add more tools and features with ease. To implement or change any of the tools, there is a predefined way that helps the developer to focus on the details of the tool, instead of worrying about how to implement it into the system.

6.1 Difficulties

The first difficulty that we have encountered in this project has been the need of learning how to use the different tools for scanning vulnerabilities. Each one of the tools works in a completely different way. To implement them into this service, we have had to learn to use new specific Python libraries, and specific APIs or extract the results from the output in plain text.

The second major difficulty has been learning to use PHP from scratch and using Laravel. PHP is a language that we have not seen in the degree, and even though it is similar to others, it has some peculiarities which make it unique. On the other hand, Laravel is a framework to work with PHP that has a specific way of working. Once the developer is used to working with Laravel, it is fast and efficient, but during the process of learning we have found some situations that seemed logical, but Laravel would not permit.

The last difficulty that we have struggled with it has been Docker. At the start of the

project, we did not know that we were going to use this technology, but at the start of May, ESED proposed to use this technology to host the scan server. It has not been one of the most difficult parts of the project, but it has forced us to move the schedule and adapt to the new situation.

6.2 Future investigations

This project will probably continue to develop with ESED. Some of the improvements that they could do are:

- Add more scanning tools to contrast the information.
- Add different tools to check other types of vulnerabilities.
- Change some of the tools like OWASP ZAP for others that do not have such a big impact on the performance of the machine.
- Write more detailed descriptions of the results that the different scans give.
- Adapt the code to use more than one scan server at the same time.

These are the future investigations that we propose from an academic perspective, but this project is still a service that will be used by ESED. In the end, they will know which are the lacking parts of the project, and which are the improvements that the project needs.

6.3 Personal reflections

This project has allowed me to investigate and learn about the cybersecurity field, which we have not seen in the degree. It also has given me the opportunity to apply and practice concepts seen in class like APIs, different frameworks, and virtualization among others.

As seen earlier in the future improvements section, there are still things that could be done to continue with this project. But time is finite and there is always something that can be improved. The overall feeling of the project is that expectations have been met, and it has been a productive investment of time.

Personally, I have enjoyed making this final degree project. I think it has been an opportunity to learn a lot about cybersecurity, virtualization with Docker, web development, and meet new people that work in the same field that I want to work in the future.

7. Bibliography

- [1] What is cybersecurity? - cisco. <https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html>.
- [2] Descargue la evaluación de vulnerabilidades nessus — tenable®. <https://es-la.tenable.com/products/nessus>.
- [3] Invicti – web application security for enterprise. <https://www.invicti.com/>.
- [4] Intruder – an effortless vulnerability scanner. <https://www.intruder.io/>.
- [5] What is the free plan? – probely help center. <https://help.probely.com/en/articles/1974121-what-is-the-free-plan>.
- [6] Owasp zap. <https://www.zaproxy.org/>.
- [7] Nmap: the network mapper - free security scanner. <https://nmap.org/>.
- [8] Tcp syn (stealth) scan (-ss) – nmap network scanning. <https://nmap.org/book/synscan.html>.
- [9] Metasploit – penetration testing software, pen testing security – metasploit. <https://www.metasploit.com/>.
- [10] sqlmap: automatic sql injection and database takeover tool. <https://sqlmap.org/>.
- [11] Wpscan wordpress security scanner. <https://wpscan.com/wordpress-security-scanner>.
- [12] Owasp/joomscan: Owasp joomla vulnerability scanner project. <https://github.com/OWASP/joomscan>.
- [13] tlssled – kali linux tools. <https://www.kali.org/tools/tlssled/>.
- [14] State of js 2020: Front-end frameworks. <https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/>.

- [15] Vue vs react: What is the best javascript framework in 2022? — codica. <https://www.codica.com/blog/react-vs-vue/>,.
- [16] What is a back end system? - definition from techopedia. <https://www.techopedia.com/definition/1405/back-end-system>.
- [17] What is python? - definition from techopedia. <https://www.techopedia.com/definition/3533/python>.
- [18] Laravel - the php framework for web artisans. <https://laravel.com/>.
- [19] Stack overflow developer survey 2021. <https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies>.
- [20] python-nmap · pypi. <https://pypi.org/project/python-nmap/>.
- [21] Introduction – api reference. <https://www.zaproxy.org/docs/api/#introduction>.
- [22] python-owasp-zap-v2.4 · pypi. <https://pypi.org/project/python-owasp-zap-v2.4/>.
- [23] subprocess – subprocess management – python 3.10.5 documentation. <https://docs.python.org/3/library/subprocess.html>.
- [24] tristanlatr/wpscan_out_parse: Python parser for wpscan output files (json and cli). https://github.com/tristanlatr/wpscan_out_parse.