Configuration Manual

MSc Research Project
MSc in Cybersecurity

Muzammil Hussain
Student ID: x23218029

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

`

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Muzammil Hussain.....

Student ID: x23218029.....

Programme: MSc Cybersecurity..... **Year:** 2024.....

Module: Configuration Manual.....

Lecturer: Mr. Kamal Mahajan.....

Submission Due Date: 12-Dec-2024.....

Project Title: Evaluation of Open-Source Vulnerability Scanners for Web Applications and WordPress Websites

Word Count: .....1961..... **Page Count:** .....38.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....Muzammil Hussain.....

Date: .....11-Dec-2024.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

`

Configuration Manual

Muzammil Hussain
x23218029:

1 reNgine

reNgine is a comprehensive web application reconnaissance suite, developed to enhance the reconnaissance process for security professionals, penetration testers, and bug bounty hunters. This is a comprehensive online application reconnaissance suite intended to facilitate and optimize the reconnaissance process for security experts, penetration testers, and bug bounty hunters. ReNgine redefines the acquisition of essential information regarding target web applications with its highly customizable engines, data correlation capabilities, continuous monitoring, database-supported reconnaissance data, and an intuitive user interface. Conventional reconnaissance instruments frequently lack configurability and efficiency. reNgine rectifies these deficiencies and presents itself as a superior alternative to current commercial technologies. reNgine was developed to overcome the constraints of conventional reconnaissance tools and give a superior alternative, even exceeding certain commercial solutions. ReNgine is the optimal option for automating and augmenting information-gathering endeavours, whether you are a bug bounty hunter, penetration tester, or part of a corporate security team [1].

1.1 Workflow Diagram

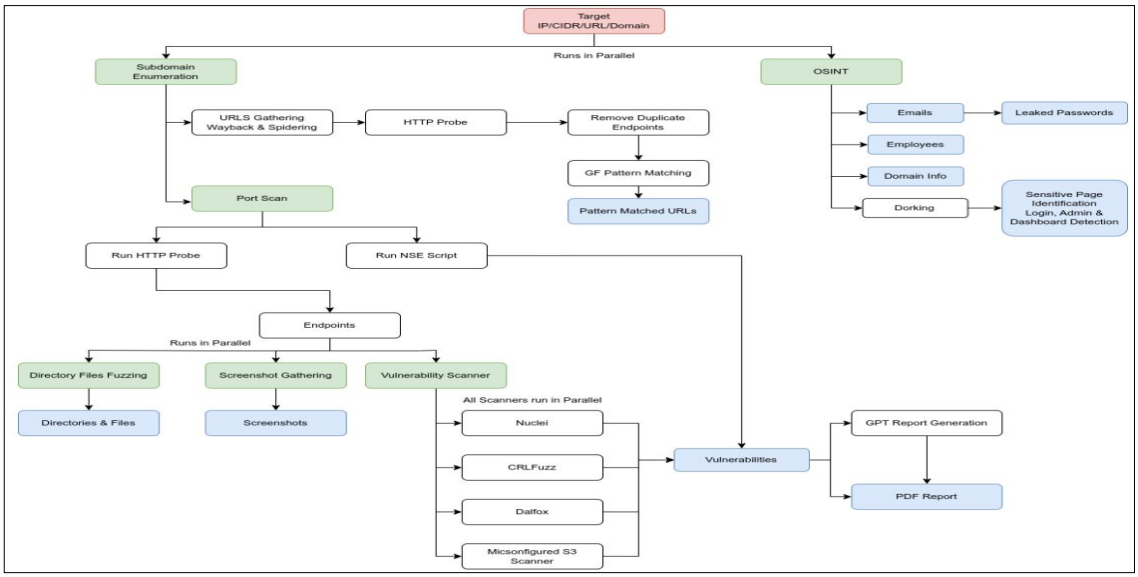


Figure 1: Workflow Diagram [2]

[1] reNgine Documentation
[2] https://yogeshojha.github.io/rengine/

`

1.2 Pre-requisites & Installation

reNgine employs several scripts and tools, which necessitate the installation of numerous dependencies such as Go and Python. To mitigate potential dependency conflicts, we choose to utilize Docker. Utilizing Docker will not only alleviate dependency complications but will also simplify the installation process. As a penetration tester, your emphasis should not be on resolving dependencies or installing necessary tools. After completing a minimal number of installation procedures, you will be prepared to operate reNgine. Before installing reNgine, ensure that Docker, Docker Compose, and Make should be installed.

1.2.1 Docker Installation

Update the Linux Operating system and packages

- Sudo apt install update
- Sudo apt install upgrade

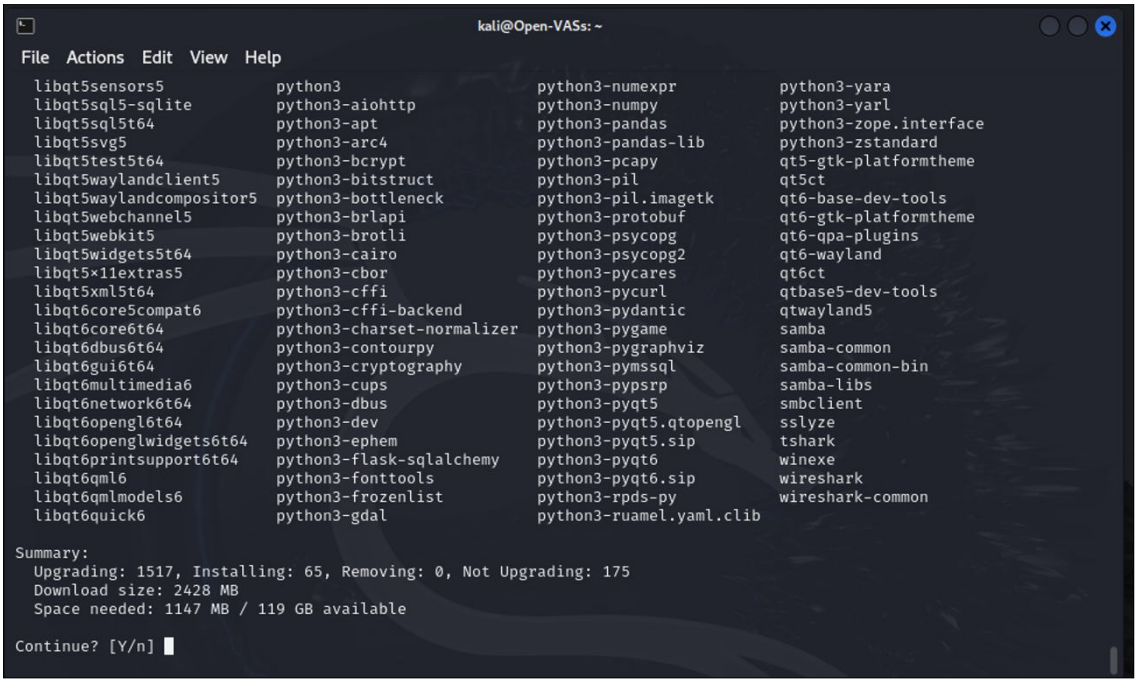


Figure 2 : Package Installation & Upgradation

Installation of Docker on Kali Linux

- sudo apt install -y docker.io
- sudo systemctl enable docker --now
- docker

`

Download Latest Stable version of Docker-Compose [3]

- sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
- sudo chmod +x /usr/local/bin/docker-compose
- sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
- docker-compose --version

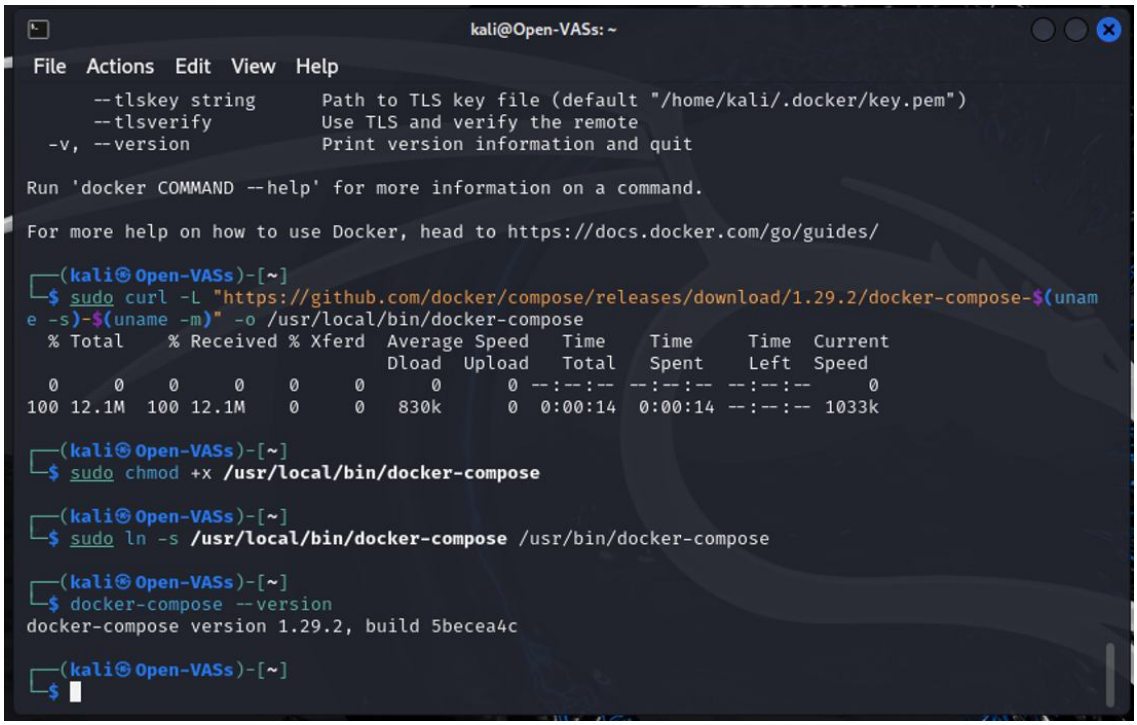


Figure 3 : Docker-Compose Installation

1.2.2 reNgine Installation

clone the repository

- sudo git clone https://github.com/yogeshojha/rengine
- cd reNgine

Concurrency Values

Edit the dotenv file using nano .env or vi .env or vim .env. Here is the ideal value for MIN_CONCURRENCY and MAX_CONCURRENCY depending on the number of RAM your machine has

- sudo nan .env

[3] Installing reNgine on Linux/Windows/Mac - reNgine

`

- **4GB**: `MAX_CONCURRENCY=10`
- **8GB**: `MAX_CONCURRENCY=30`
- **16GB**: `MAX_CONCURRENCY=50`

Figure 4 : Concurrency Values

Generating SSL Certificates

reNgine operates on HTTPS unless utilized for development purposes. Utilizing HTTPS is advisable. Certificates can be generated using.

- sudo make certs

Build reNgine

Utilize the subsequent command to construct the reNgine. The constructing procedure is protracted and anticipated to need considerable time.

- sudo make build

Run reNgine

Upon successful completion of the build phase, we are prepared to execute reNgine. This may be accomplished with the command below, and reNgine can now be viewed via https://127.0.0.1

- sudo make up

Registering an account

To access reNgine, you must establish a username and password. To register reNgine, execute the following command. You will then be required to provide some optional personal data, as well as a username and password. I strongly advise you to establish a robust password for reNgine.

- sudo make username

Checking logs

To observe the logs, execute the command.

- sudo make logs

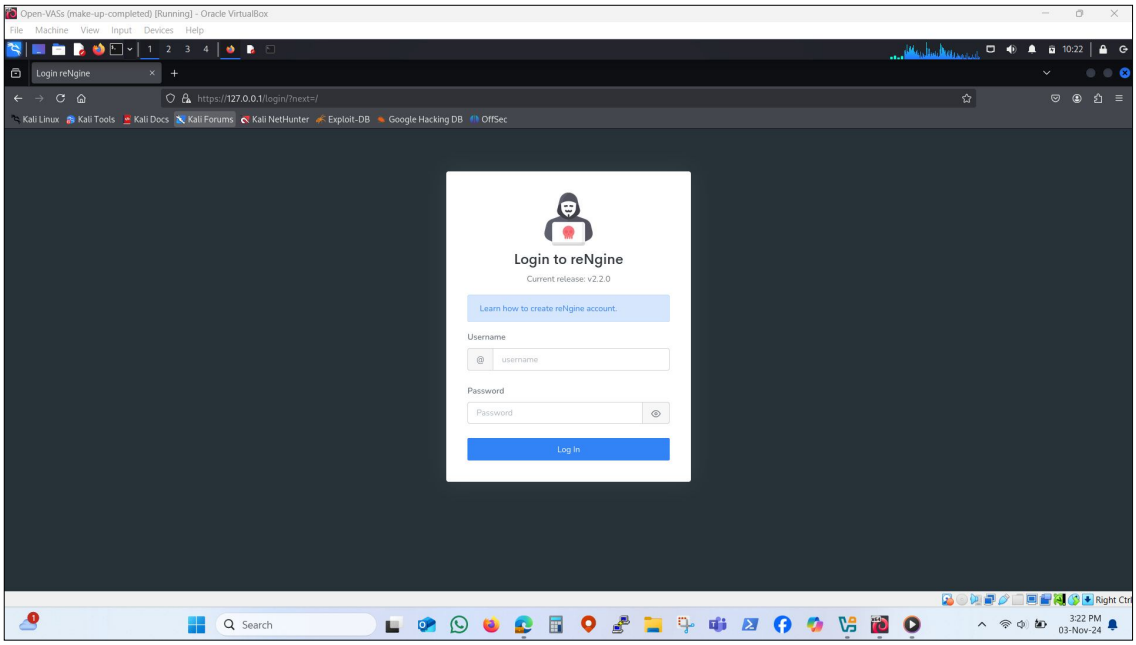


Figure 5 : reNgine Login Page

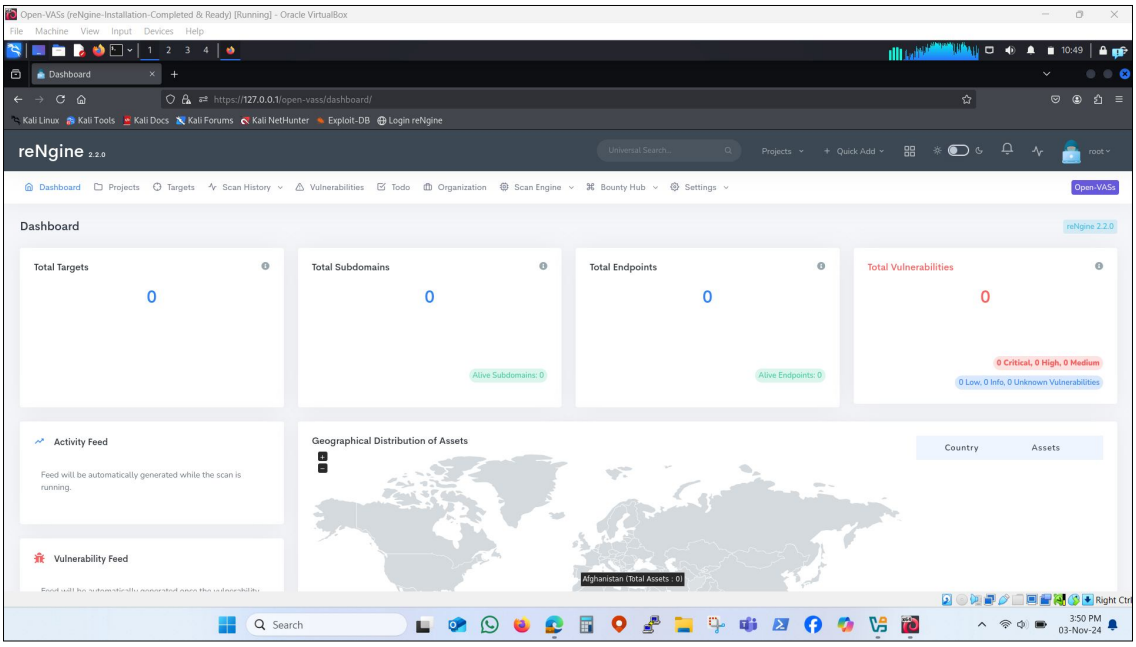


Figure 6 : reNgine Dashboard

1.3 Scanning Results

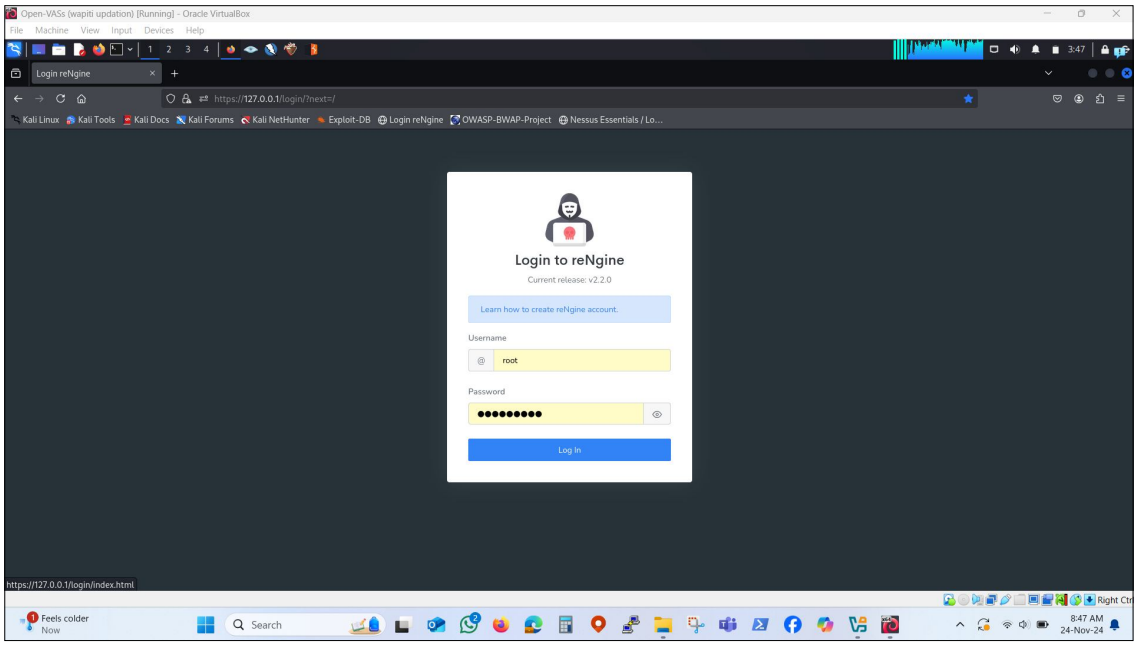


Figure 7 : reNgine Login Page

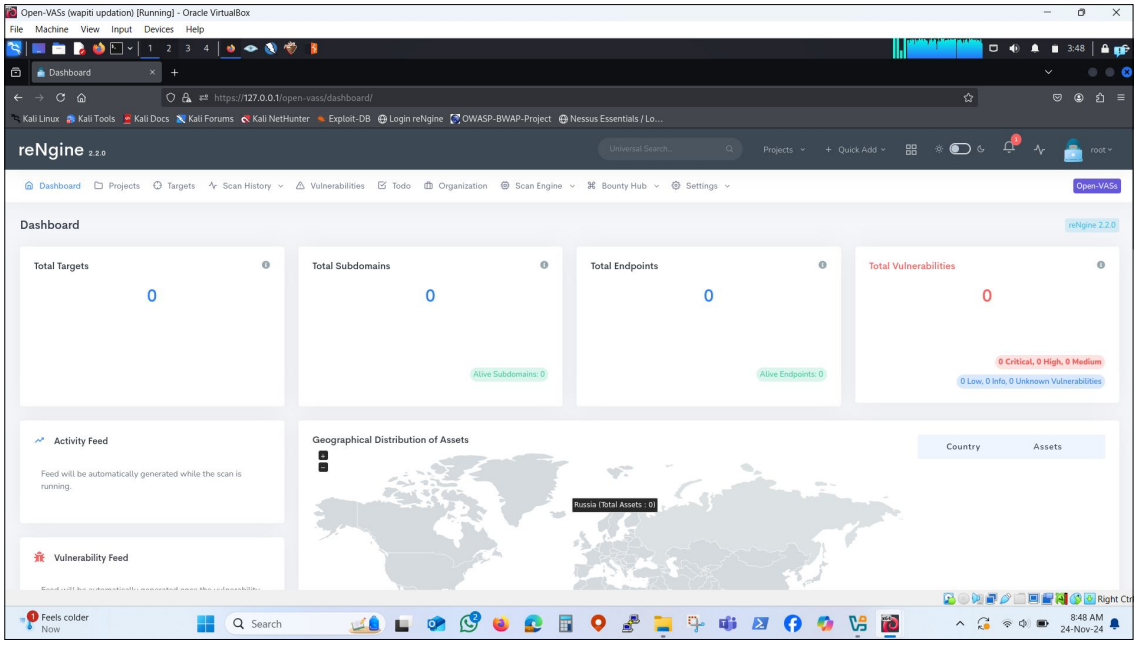


Figure 8 : reNgine Dashboard

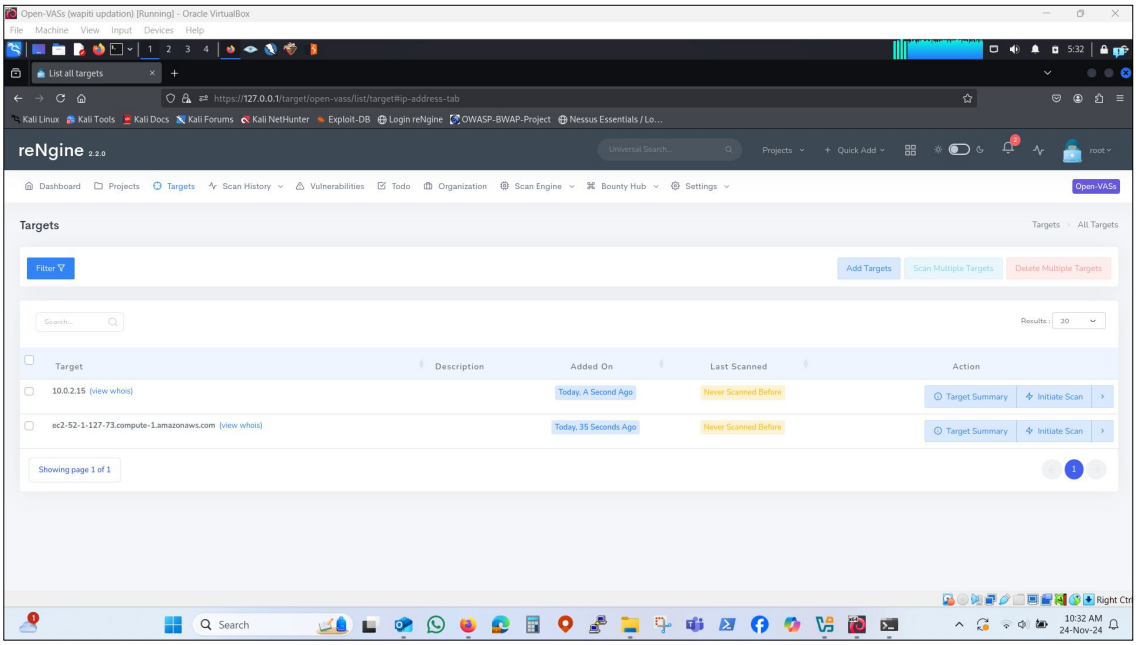


Figure 9 : Multiple Targets Added (OWASP-BWA & WordPress)

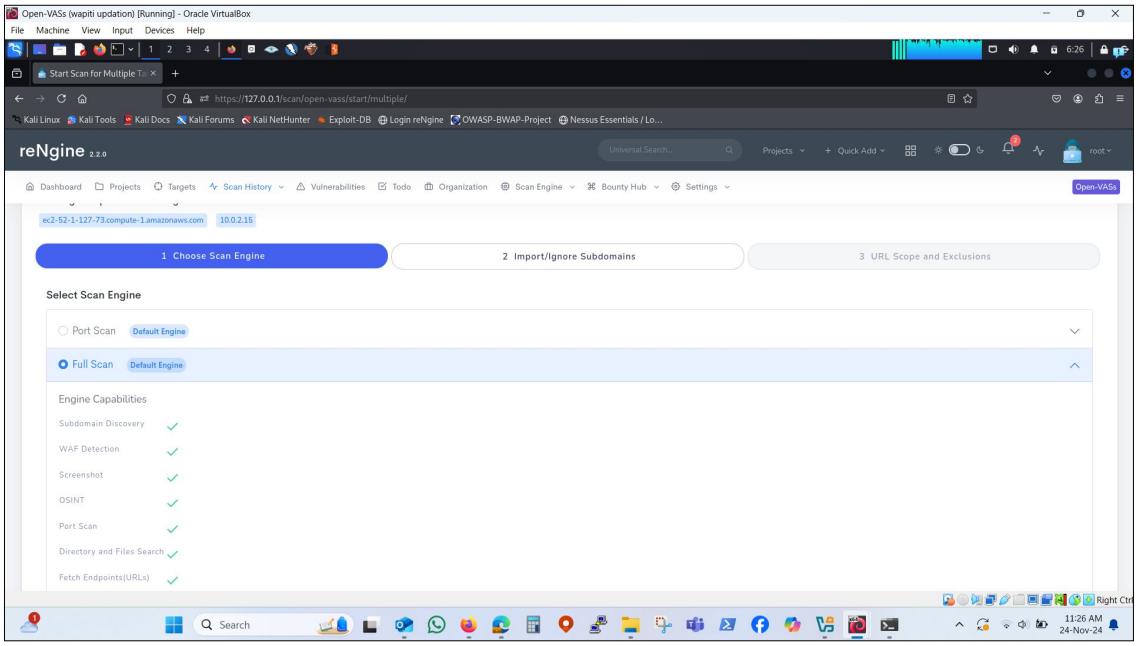


Figure 10 : Selected reNgine Full Scan Engine

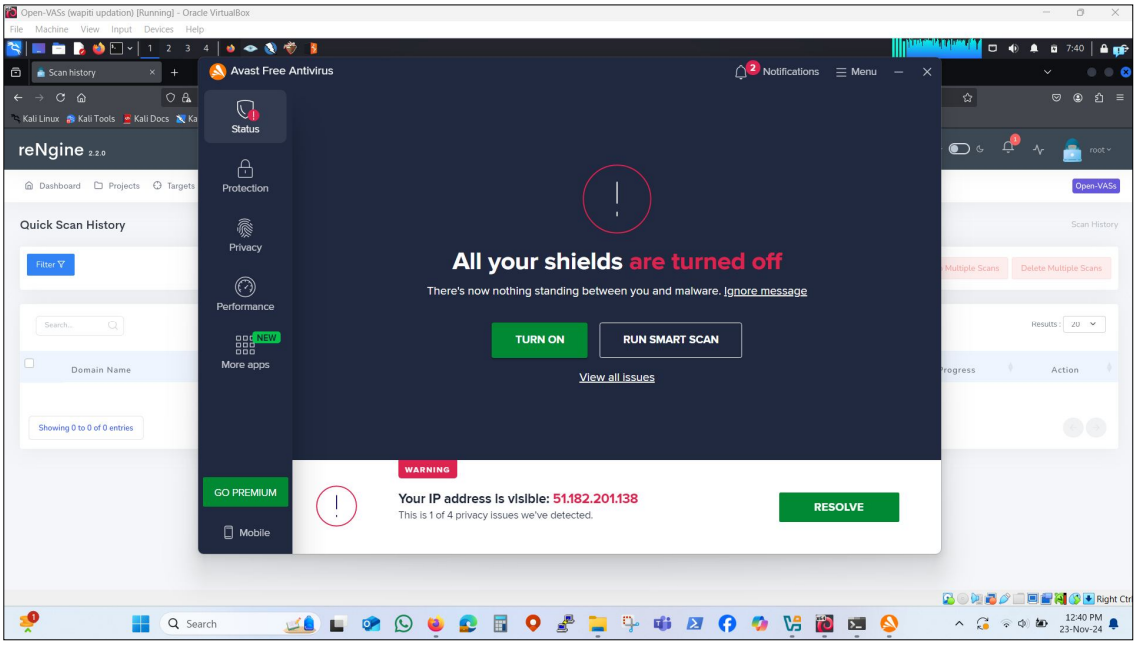


Figure 11 : Turning Off Avast Antivirus

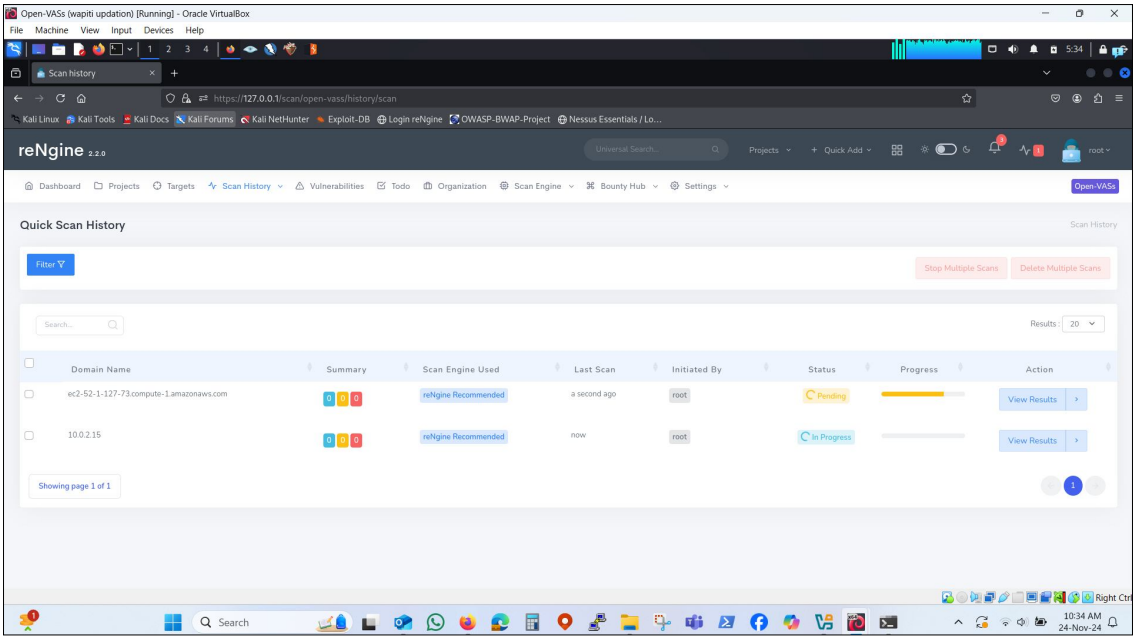


Figure 12 : Multiple Scanning In Progress

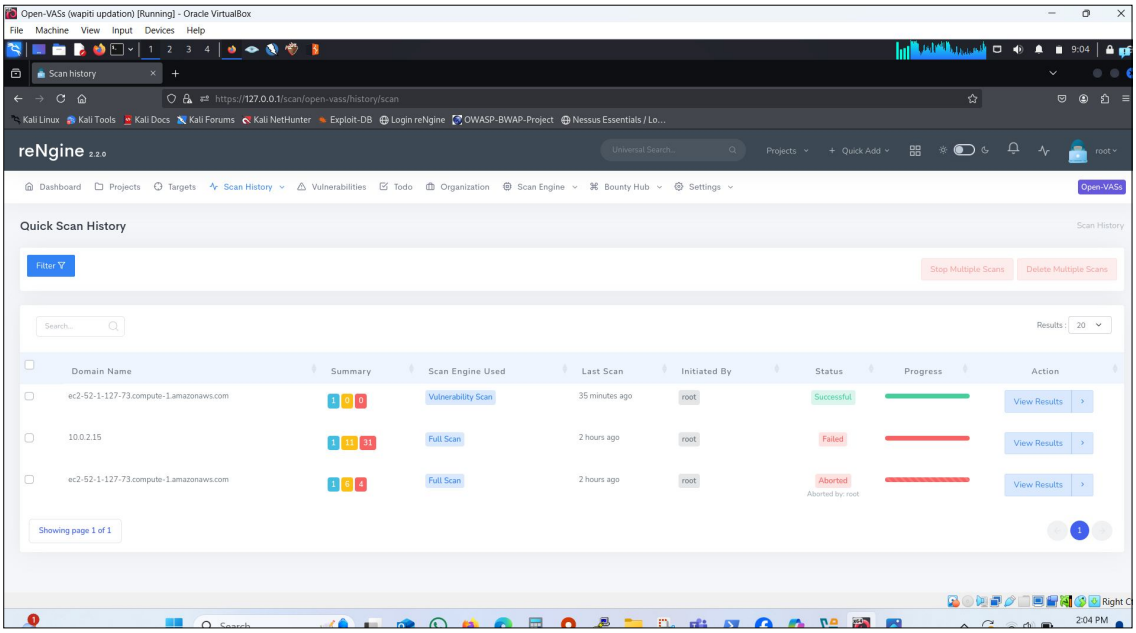


Figure 13 : Scanning Completed

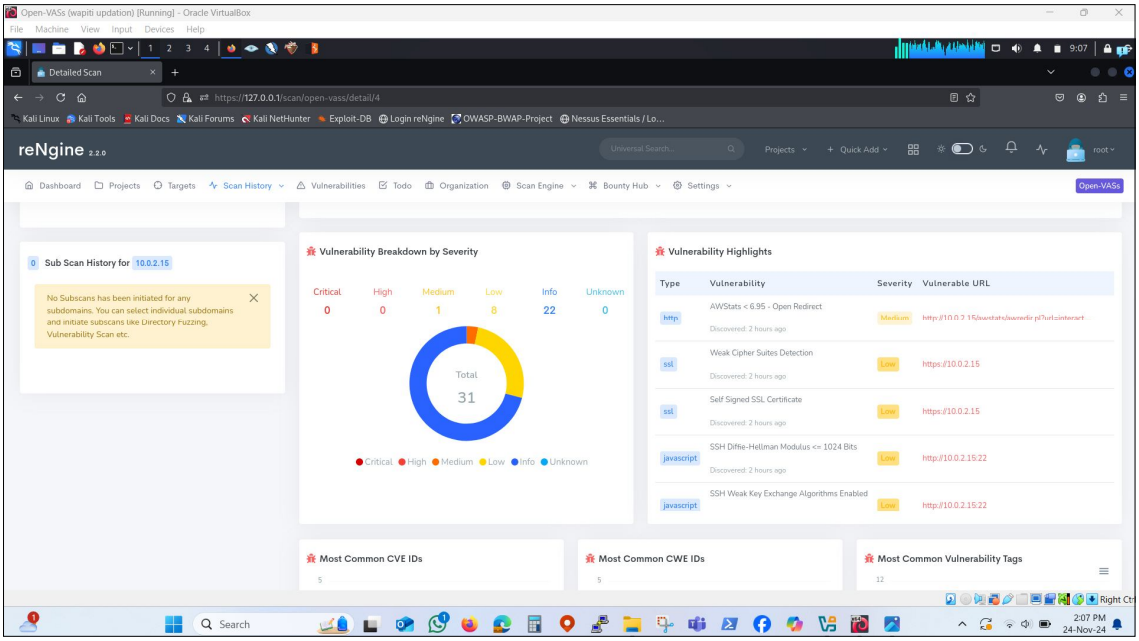


Figure 14 : Vulnerabilities Breakdown in OWASP-BWA

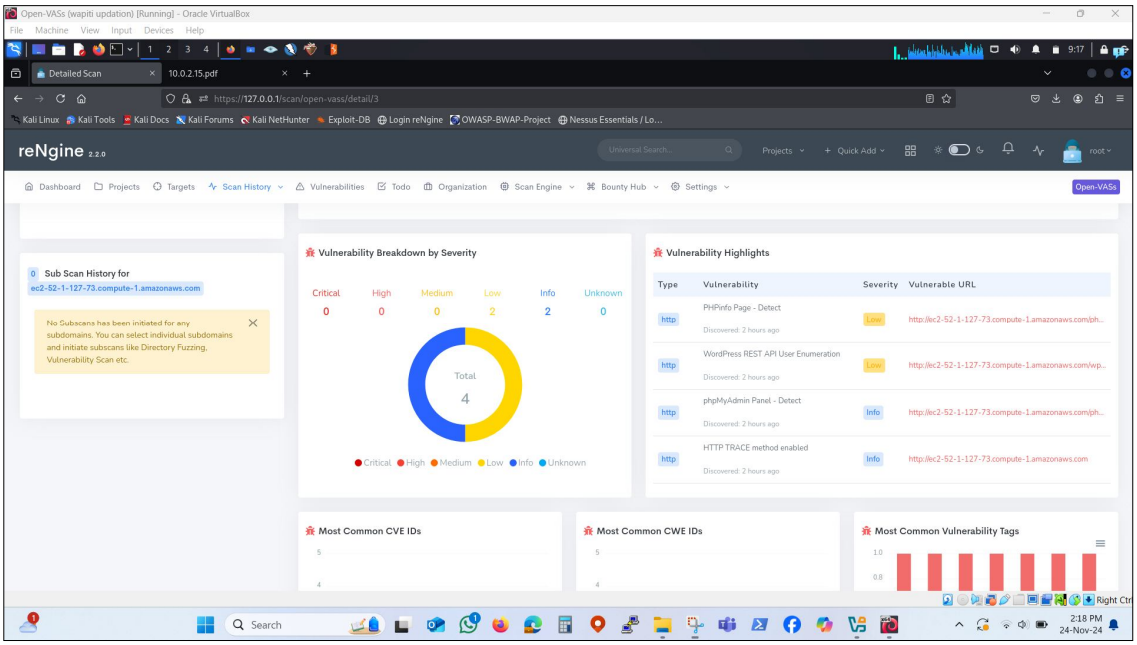


Figure 15 : Vulnerabilities Breakdown in WordPress

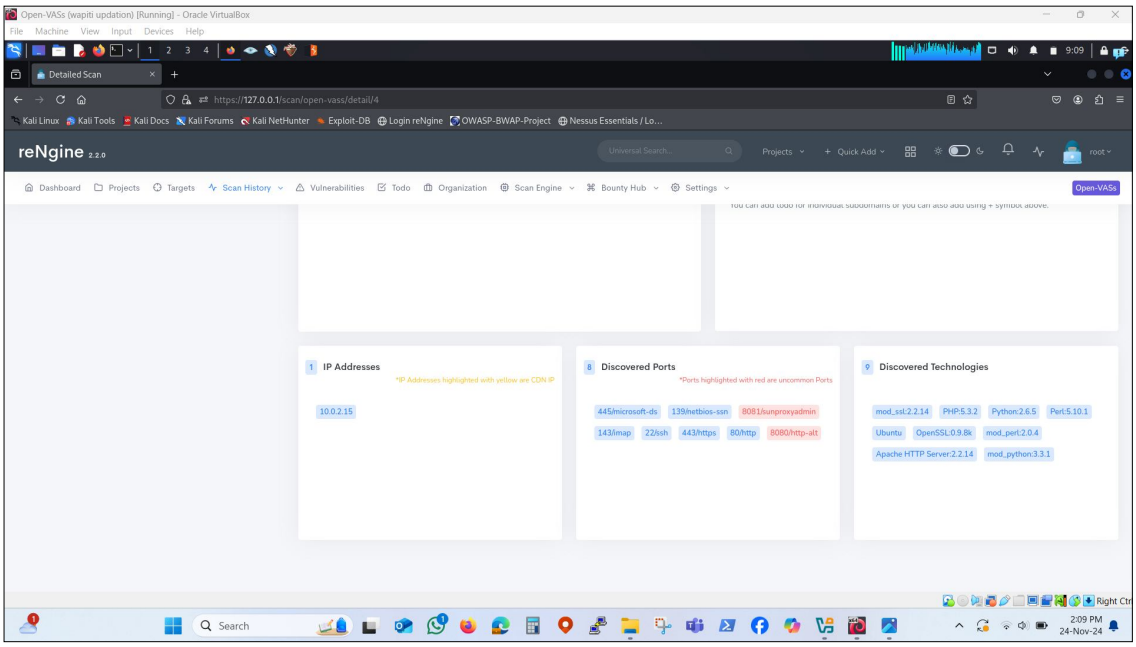


Figure 16 : Data Correlation

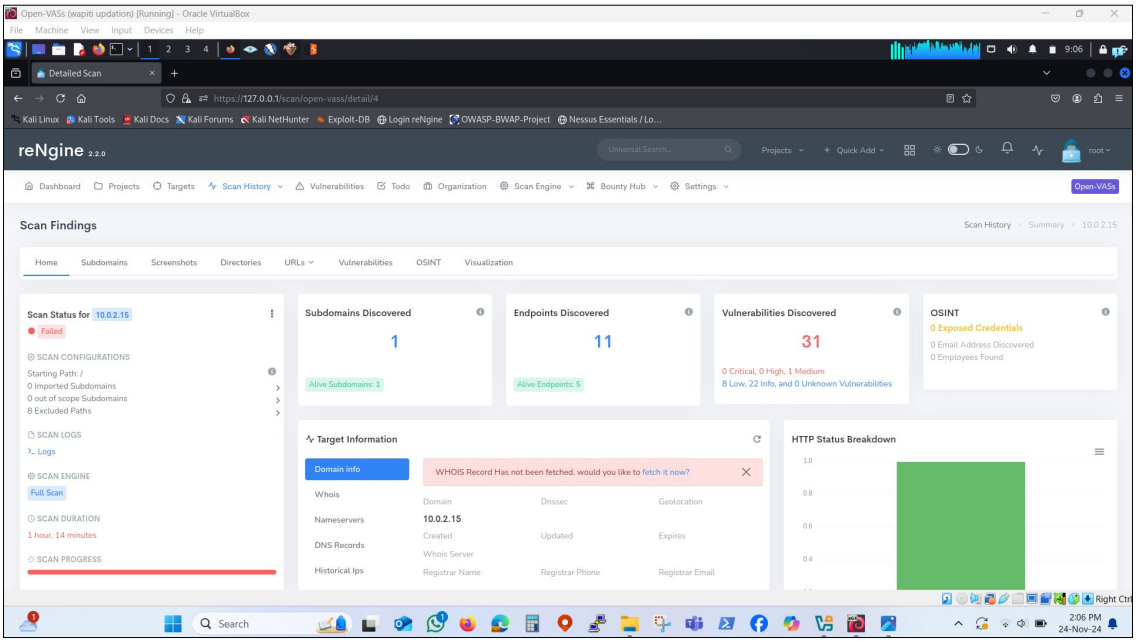


Figure 17 : Vulnerabilities Found in OWASP-BWA

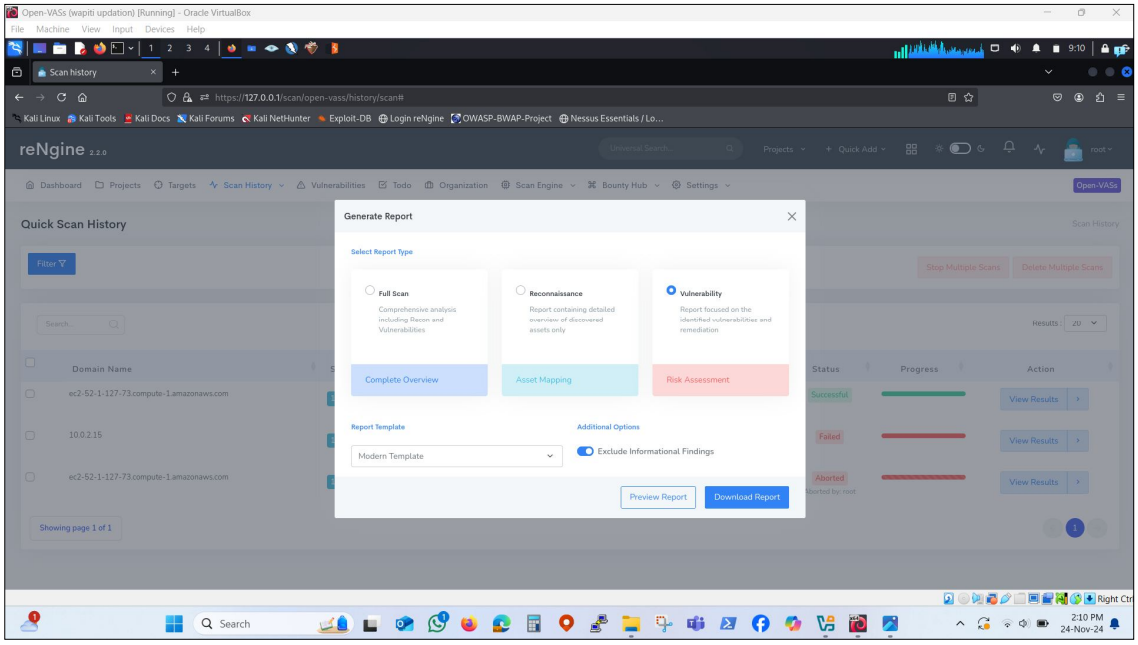


Figure 18 : Report Generation of Vulnerabilities

`

2 Nessus

Nessus is recognized for its extensive plugin repository. These plugins are dynamically and automatically integrated into the program to enhance its scanning efficiency and minimize the time needed to evaluate, investigate, and rectify vulnerabilities. Plugins can be tailored to establish evaluations unique to an organization's application environment. Nessus has a function known as Predictive Prioritization, which use algorithms to classify vulnerabilities based on their severity, assisting IT teams in identifying the most pressing risks to resolve. Each vulnerability is assigned a Vulnerability Priority Rating (VPR) on a scale from 0 to 10, with 10 indicating the highest risk, to assess its severity as critical, high, medium, or low. IT teams may utilize pre-existing rules and templates to swiftly identify vulnerabilities and comprehend the threat landscape [4]. Several researchers highlighted its characteristics and use i.e. (Abdullah, 2020) and (Sllame, Tomia and Rahuma, 2024)
A notable feature of Nessus is Live Results, which does intelligent vulnerability assessments in offline mode with each plugin update. It eliminates the necessity of doing a scan to verify a vulnerability, hence streamlining the process of assessing, prioritizing, and remediating security concerns. Nessus has the capability to generate customizable reports in several forms, including Hypertext Markup Language, comma-separated values, and Nessus Extensible Markup Language. Reports may be tailored and refined based on the most pertinent information, including vulnerability kinds, vulnerabilities categorized by host, and vulnerabilities classified by client, among others. A significant attribute is Grouped View. Nessus consolidates analogous concerns or categories of vulnerabilities into a single thread, facilitating more efficient vulnerability examinations and prioritization. Simultaneously, the Nessus packet capture functionality allows teams to efficiently diagnose and resolve scanning problems. This approach reduces disruptions and ensures uninterrupted protection for the company IT environment.

2.1 Vulnerability Priority Rating (VPR)

Vulnerability scores and categories

SCORE RANGE	SEVERITY CATEGORY
0.0	None
0.1–3.9	Low
4.0–6.9	Medium
7.0–8.9	High
9.0–10.0	Critical

Figure 19 : Vulnerability Priority Rating (VPR) [5]

[4] What is the Nessus vulnerability scanning platform? | Definition from TechTarget

[5] What is the Nessus vulnerability scanning platform? | Definition from TechTarget

2.2 Workflow Diagram

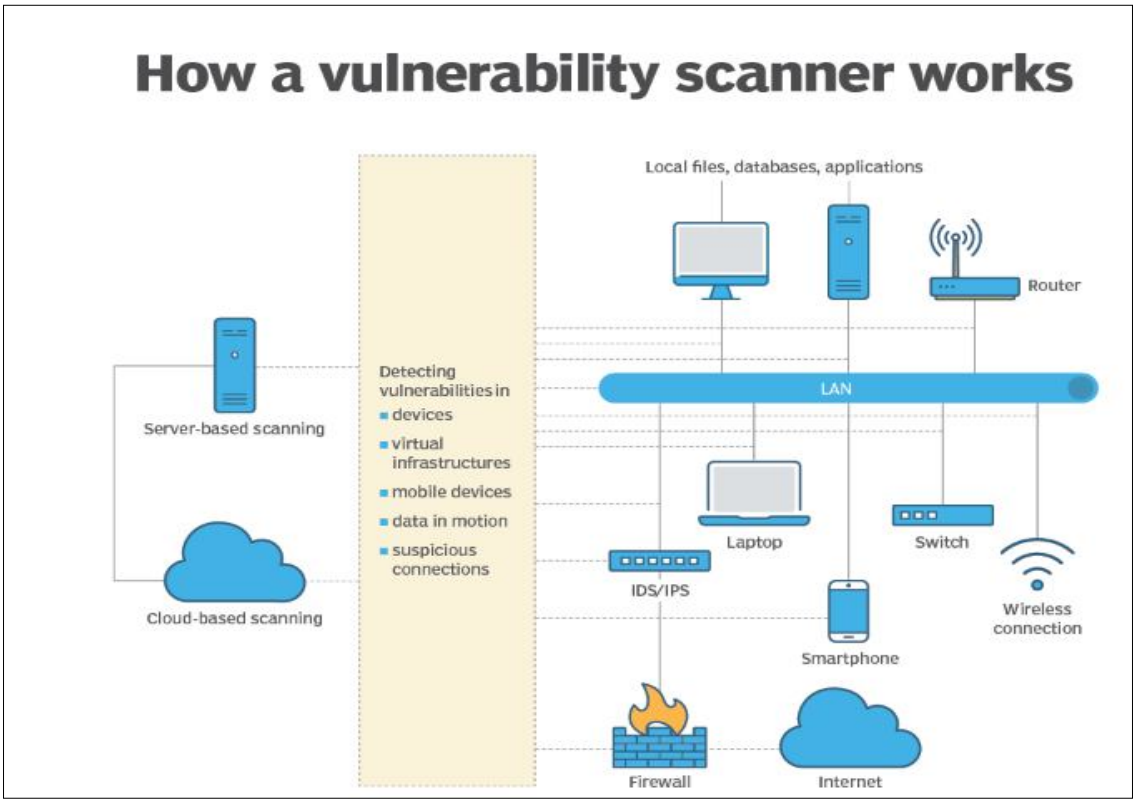


Figure 20 : Workflow Diagram [6]

2.3 Download and Installation

I downloaded the Nessus vulnerability scanning tool from its official download link [7]. I selected version 10.8.3 for Linux-Debian-amd64. I got registration by email and password and got the activation code for community version through registered email that will use later to log in first time to the Nessus web interface.

- sudo dpkg -i Nessus-10.8.3-debian10_amd64.deb
- sudo start nessusd.service
- https://127.0.0.1:8834

After installation, Nessus took 1 hour to update the plugins.

[6] What is the Nessus vulnerability scanning platform? | Definition from TechTarget
[7] Download Tenable Nessus | Tenable®

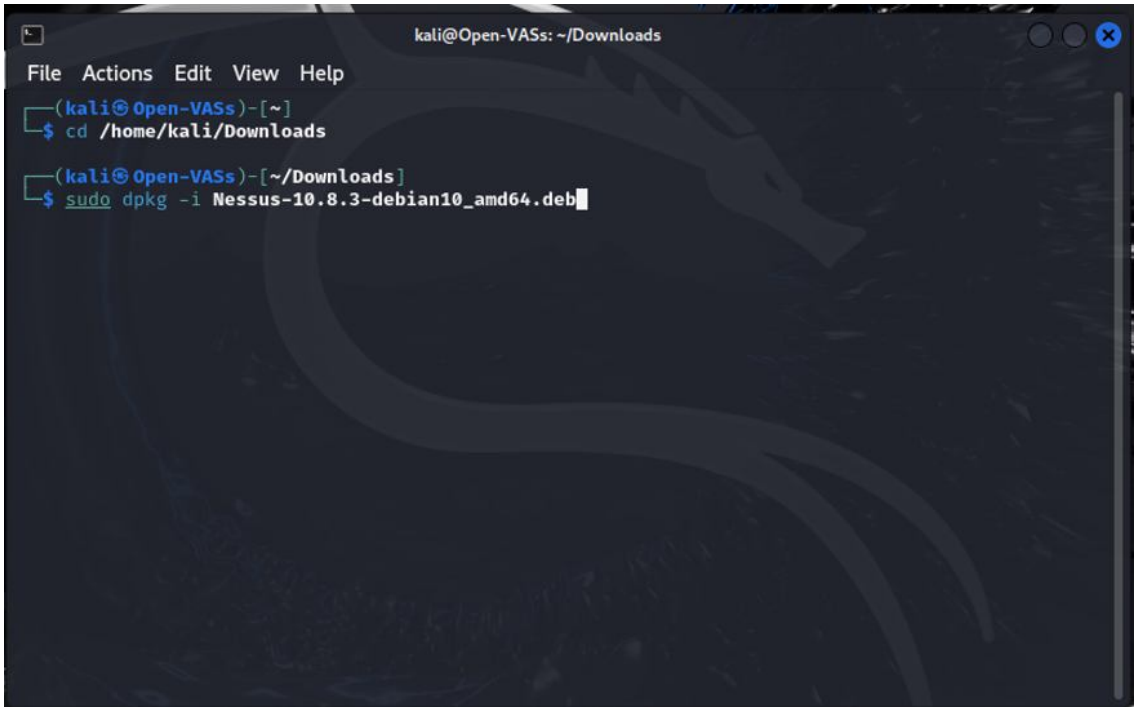


Figure 21 : Nessus Installation

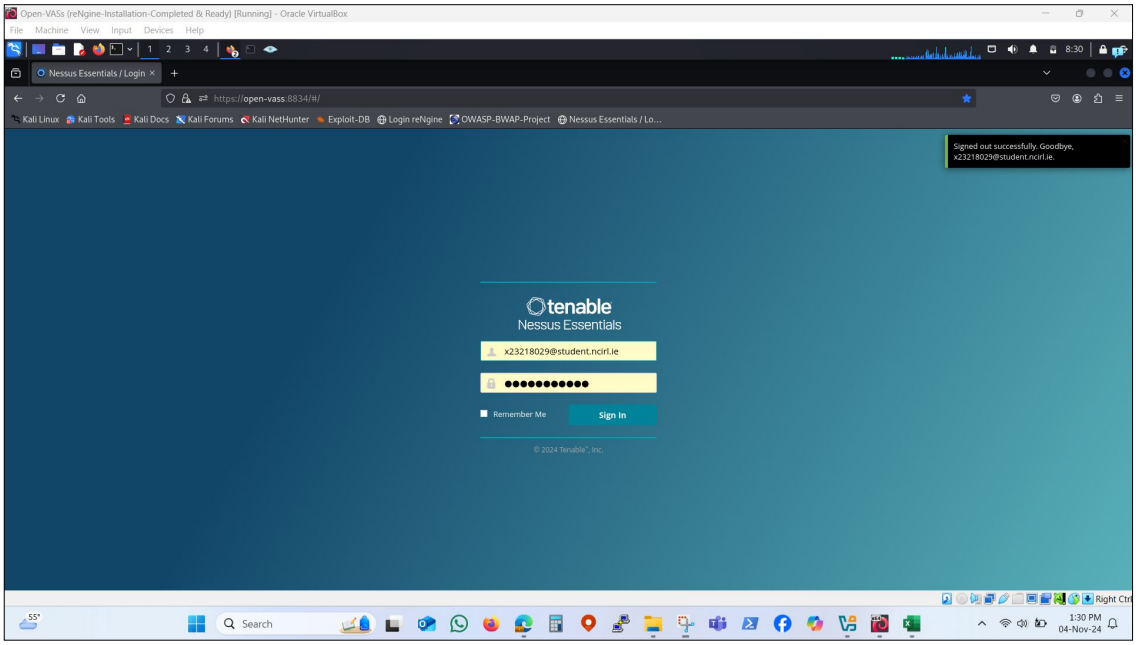


Figure 22 : Login Page

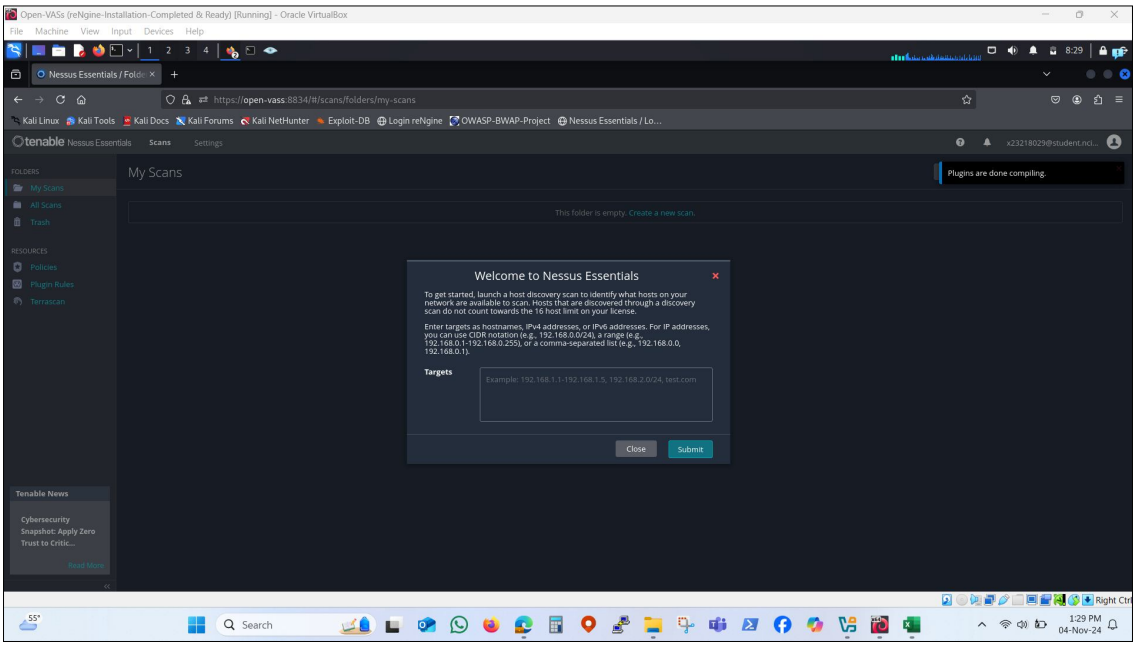


Figure 23 : Nessus Dashboard

2.4 Scanning Results

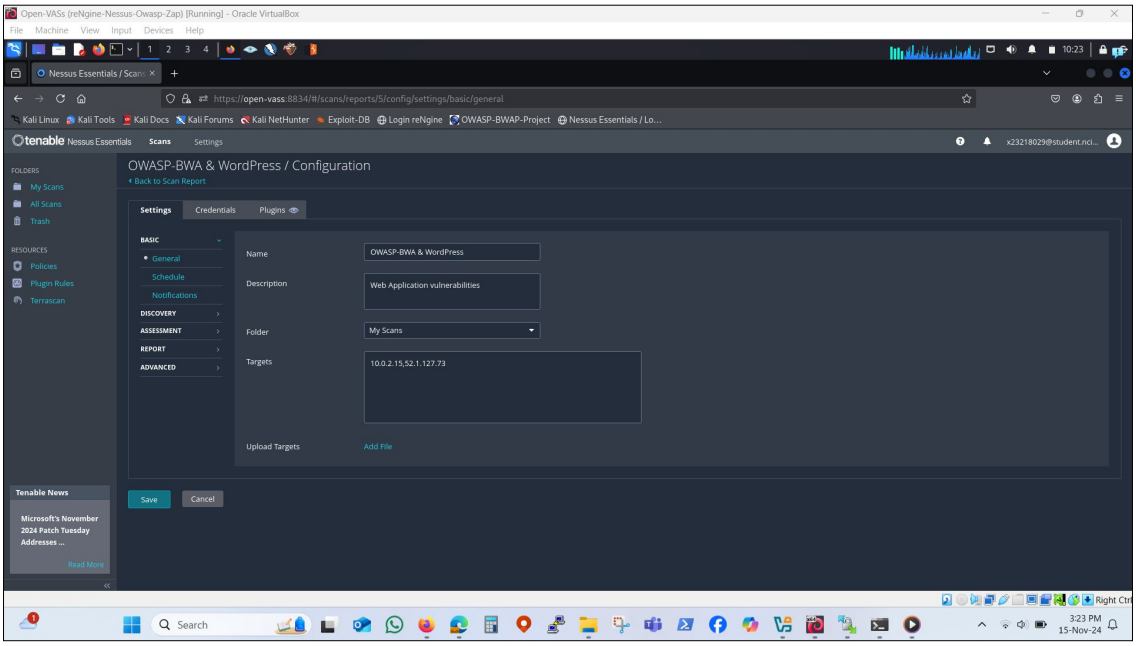


Figure 24 : Target configuration

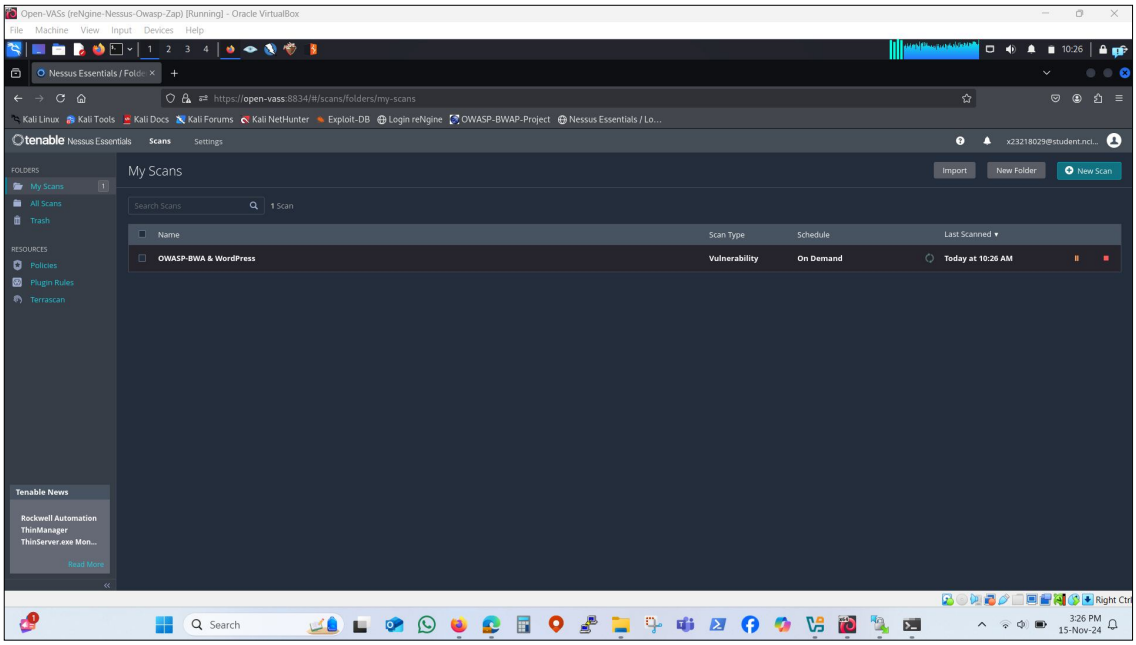


Figure 25 : Scanning in Progress

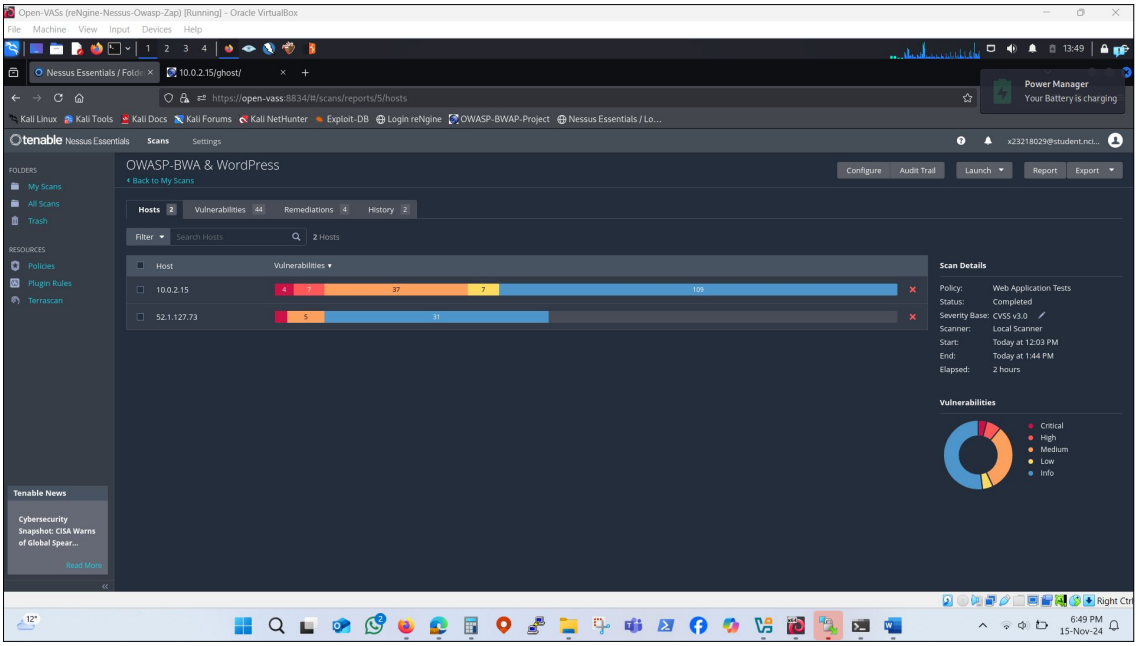


Figure 26 : Scanning Complete (OWASP-BWA & WordPress)

3 OWASP ZAP

OWASP ZAP (ZAP) is among the most widely utilized free security tools globally and is diligently maintained by several worldwide volunteers. It can assist in identifying security flaws in online applications. It is an excellent resource for both seasoned penetration testers and novices. ZAP can analyze the web application and identify vulnerabilities associated with the OWASP Top 10. Its design accommodates individuals with varying levels of penetration testing knowledge, making it suitable for our team, who were novices in this field [8]. ZAP is a complimentary open-source program that is simple to configure and utilize. The broader community provides extensive online assistance through the ZAP blog and several articles to facilitate the setup and utilization of the program. ZAP is a cross-platform application, compatible with Windows, Linux, and Mac OS. Many researchers tested this tool which is still top priority i.e. (Samgir *et al.*, 2024)

3.1 Download and Installation

I downloaded OWASP ZP 2.15.0 from its official download link [9]. Then I changed its mode settings for installation. Then I started setup file and completed the installation.

- chmod +x ZAP_2_15_0_unix.sh
- ./ ZAP_2_15_0_unix.sh

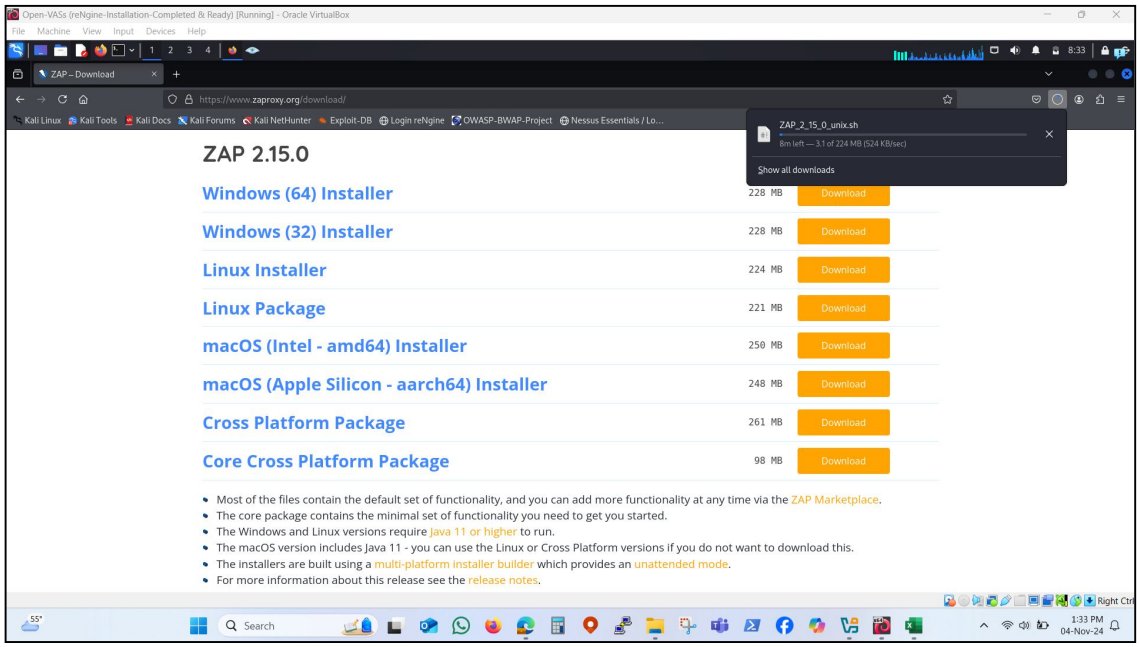


Figure 27 : Official Downloading Page [10]

[8] Using OWASP ZAP to find web app security vulnerabilities - Triad article
[9] ZAP – Download
[10] ZAP – Download

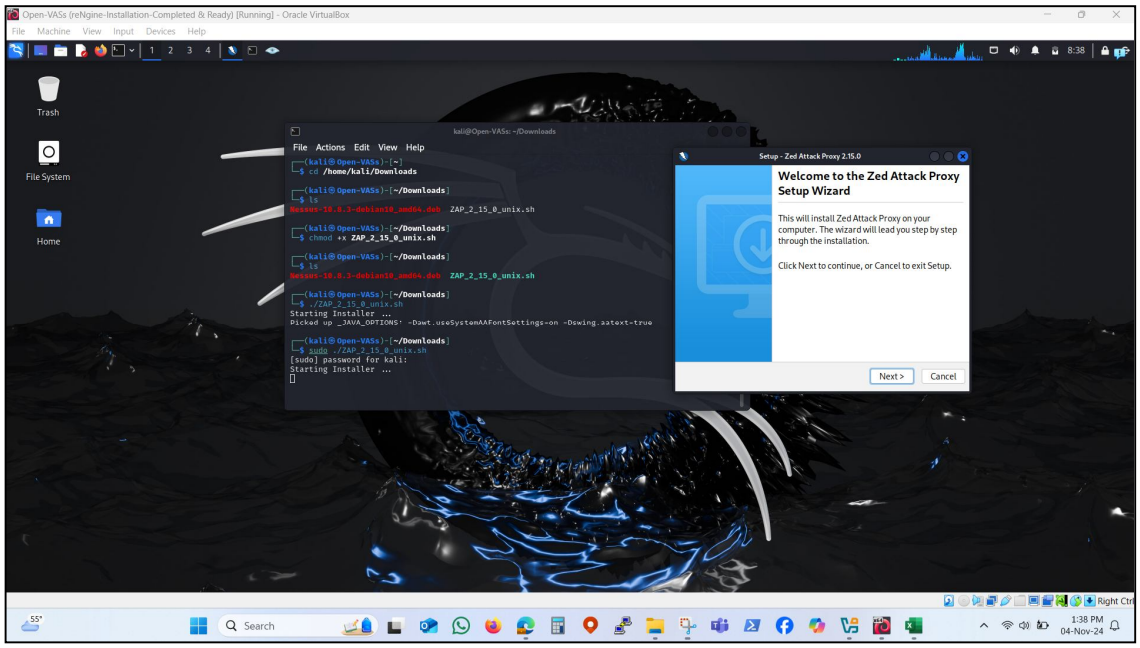


Figure 28 : Installation

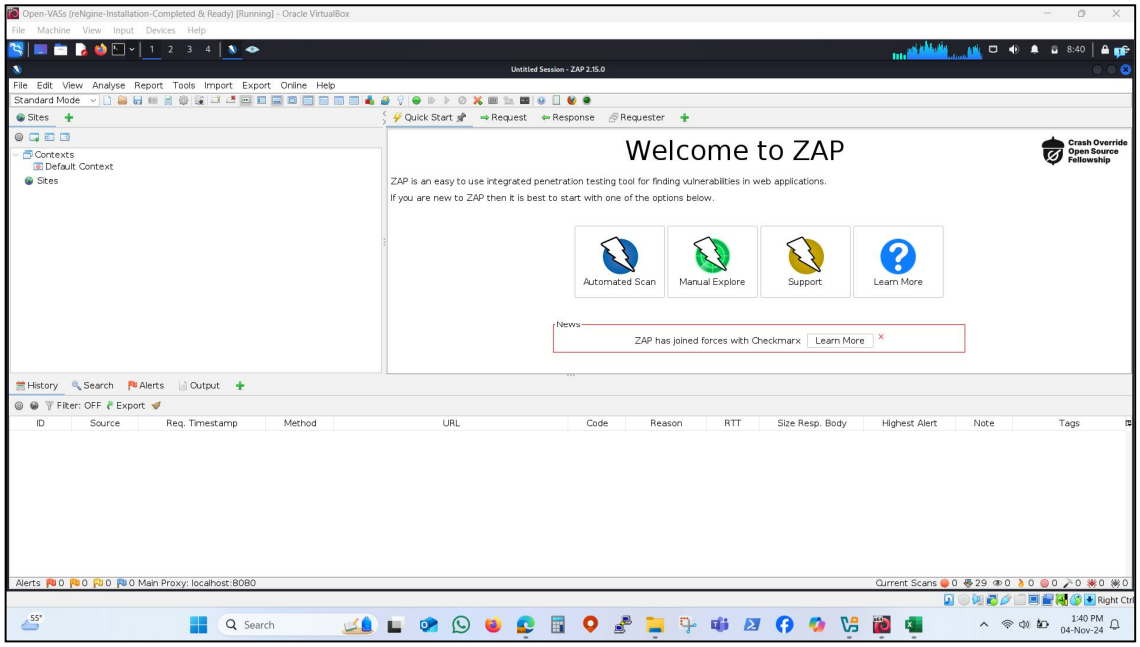


Figure 29 : OWASP Dashboard

3.2 Scanning Results

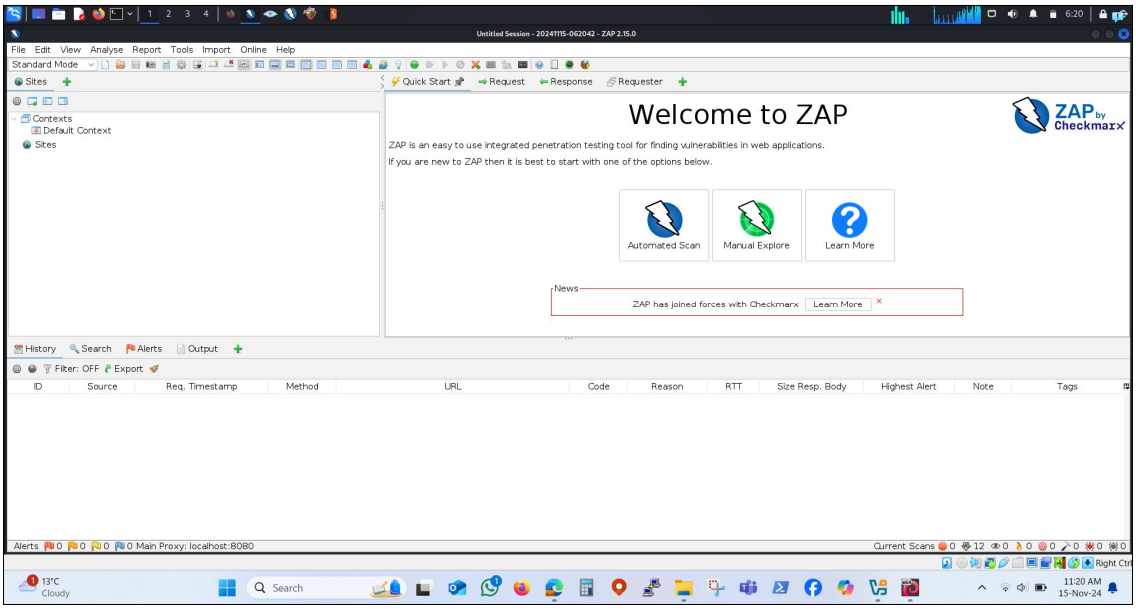


Figure 30 : OWASP-ZAP Main Page

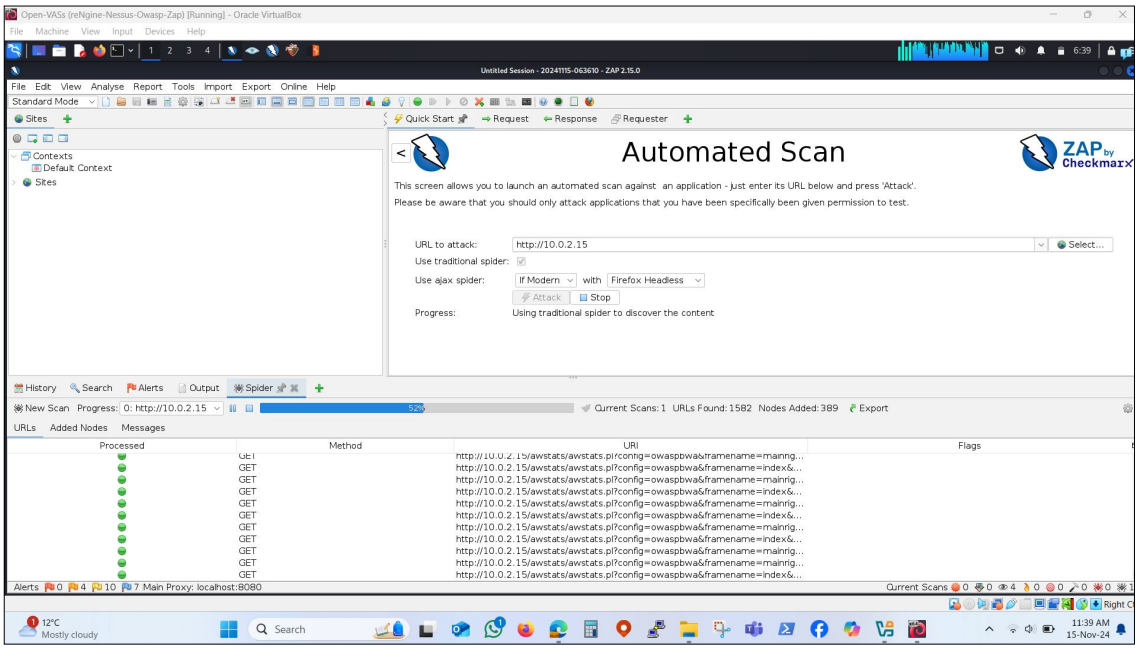


Figure 31 : Active Scanning to Target System (OWASP-BWA)

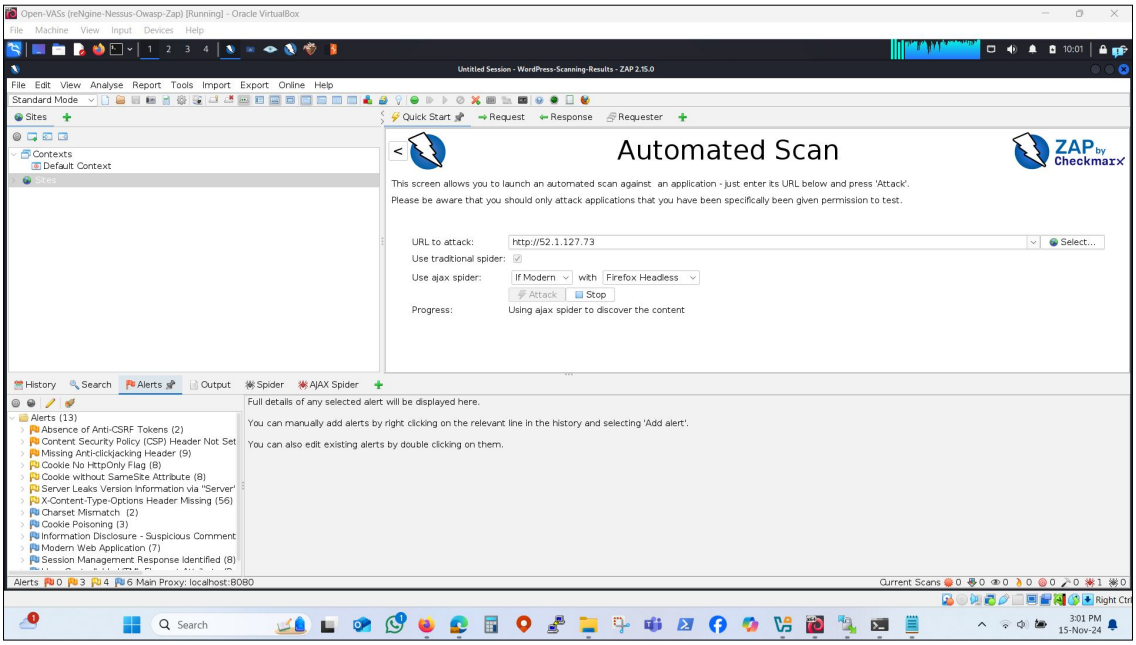


Figure 32 : Active Scanning to Target System (WordPress)

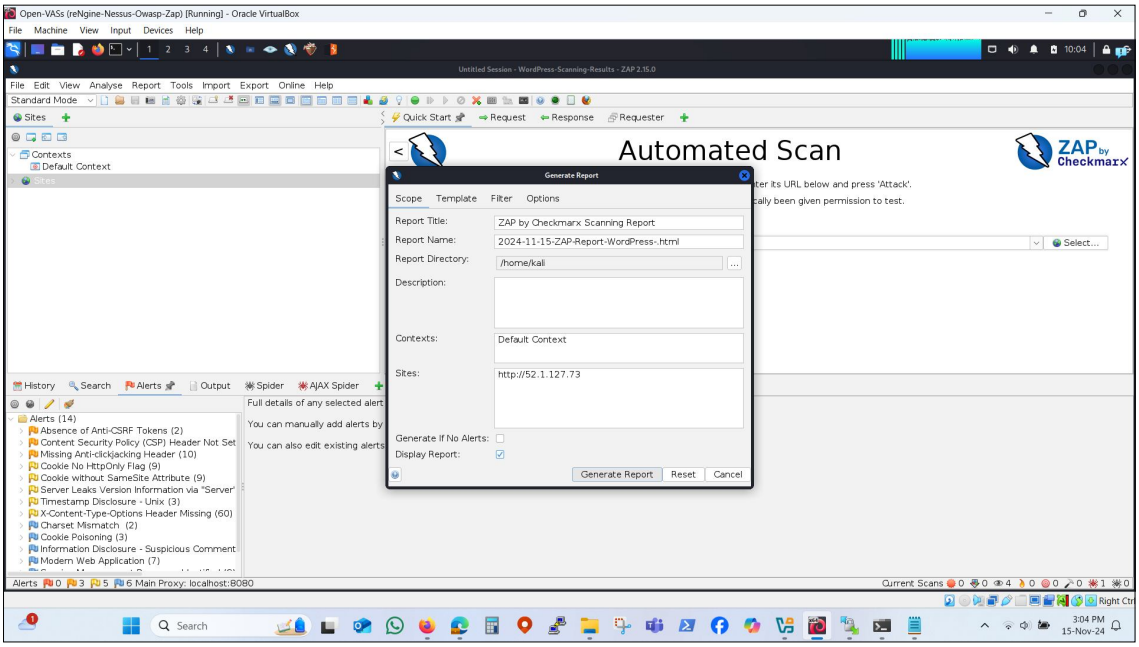


Figure 33 : Report Generation

4 Wapiti

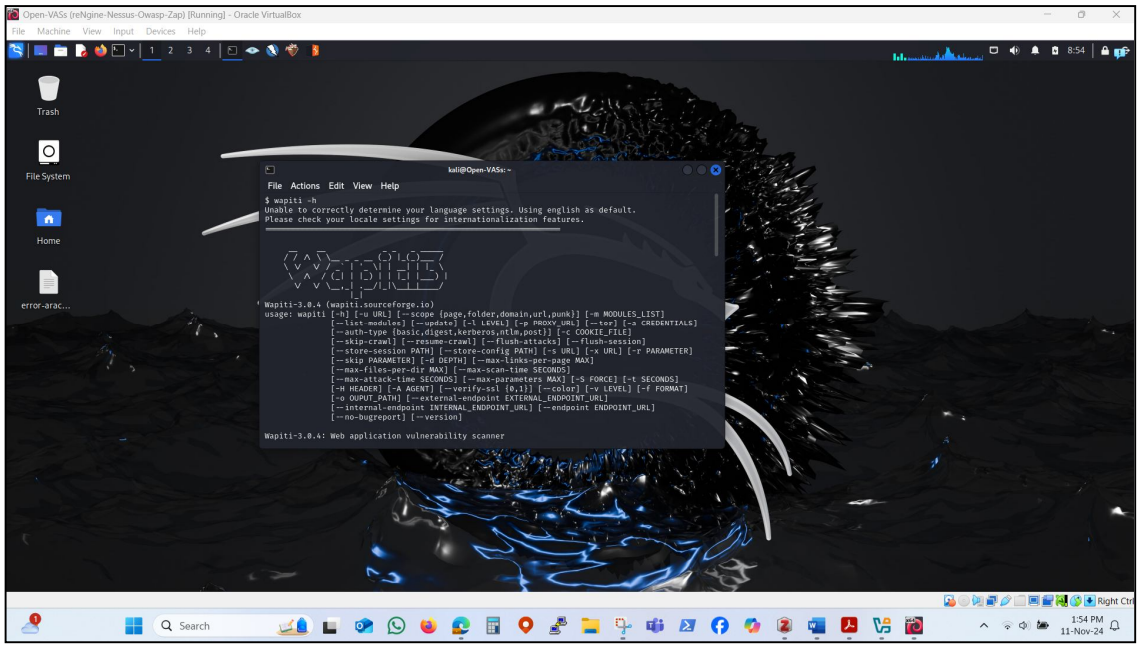


Figure 34 : Wapiti Dashboard

While performing the scanning through wapiti, I found that it is not compatible with the latest Python 3.12 version, and I tried best to downgrade the version but not successful. So, I installed windows subsystem for Linux (WSL) on the host computer and installed the wapiti. Then I have performed the scanning of both target systems.

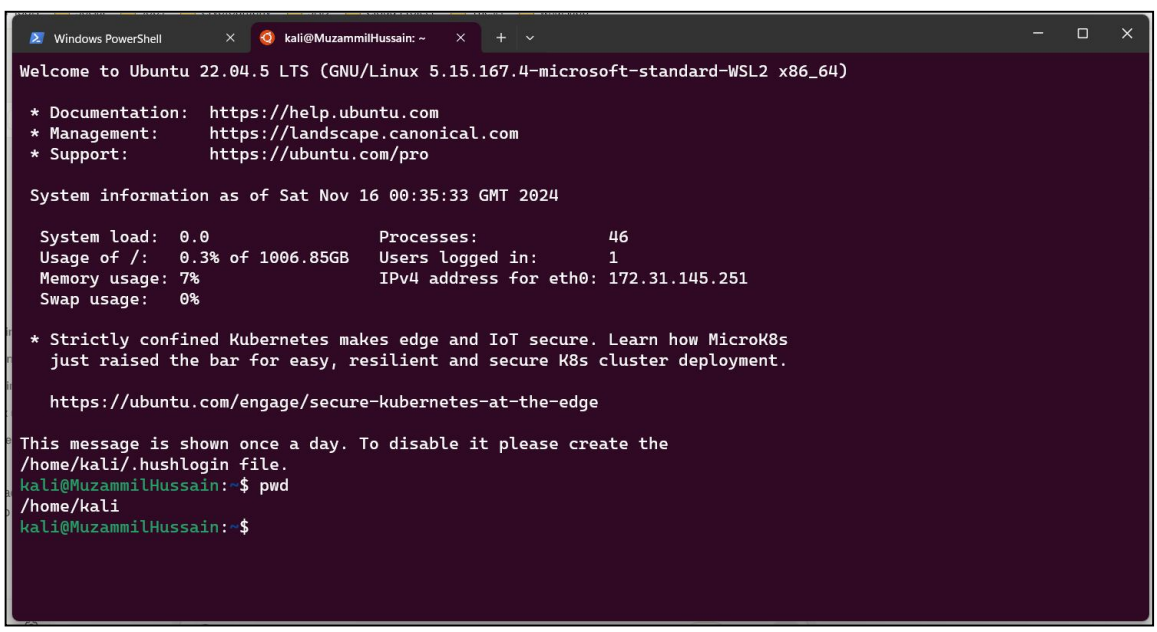


Figure 35 : Home Page of Ubuntu 22.04.5 LTS

4.1 Scanning Results

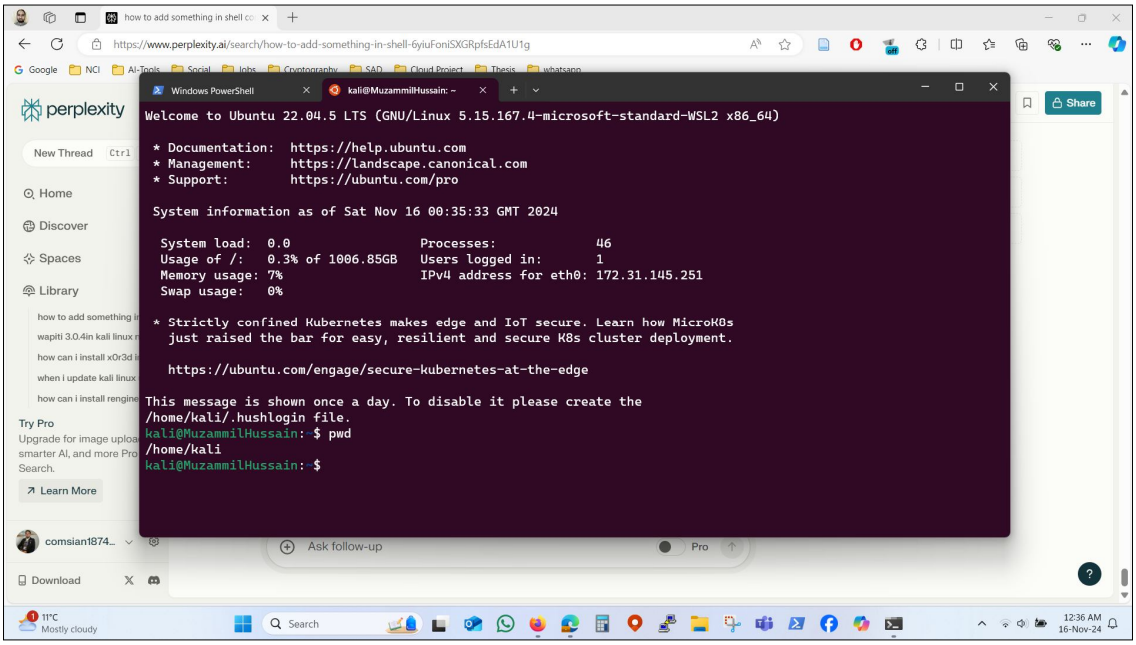


Figure 36 : Home Page of Ubuntu 22.04.5 LTS

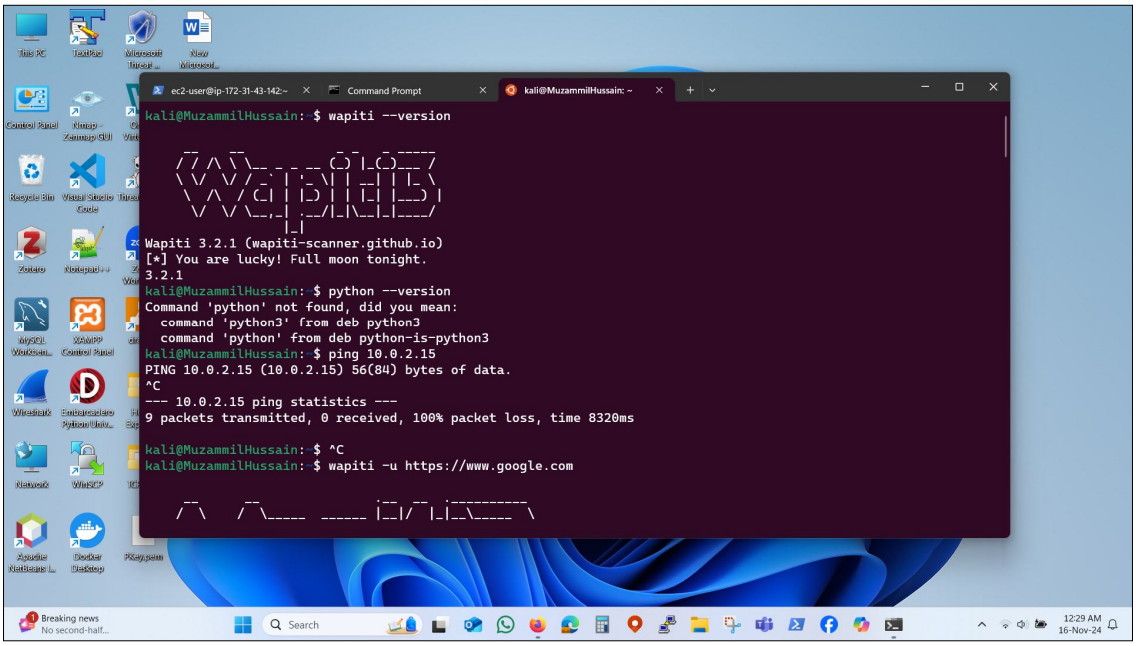


Figure 37 : Wapiti-Using-WSL

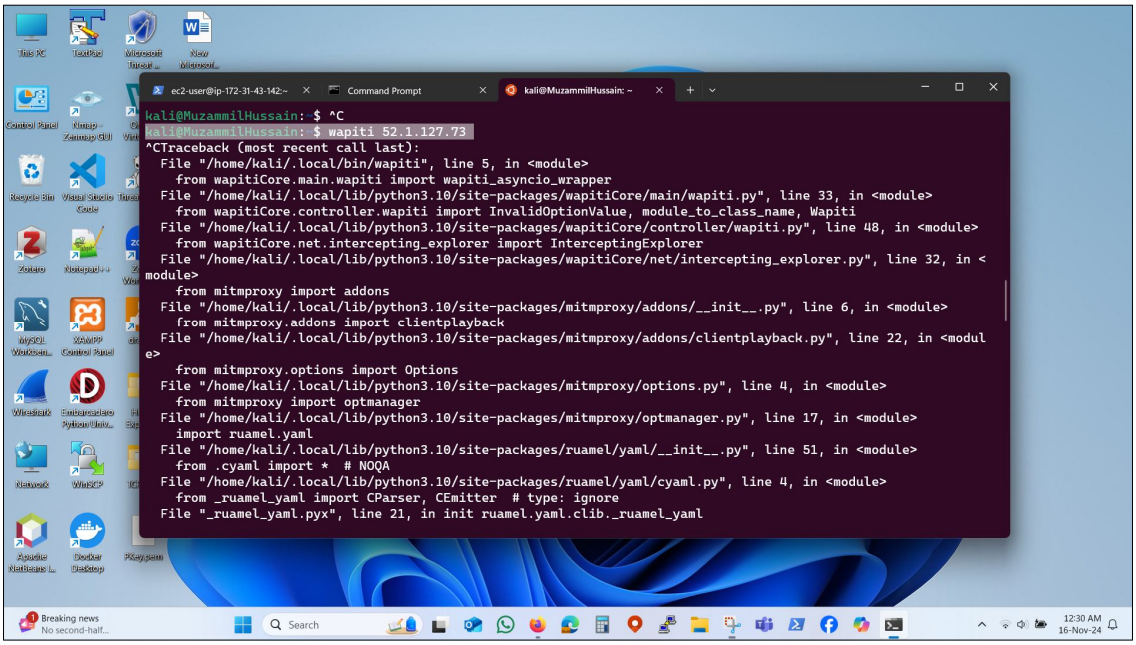


Figure 38 : Wapiti Scanning WordPress (52.1.127.73)

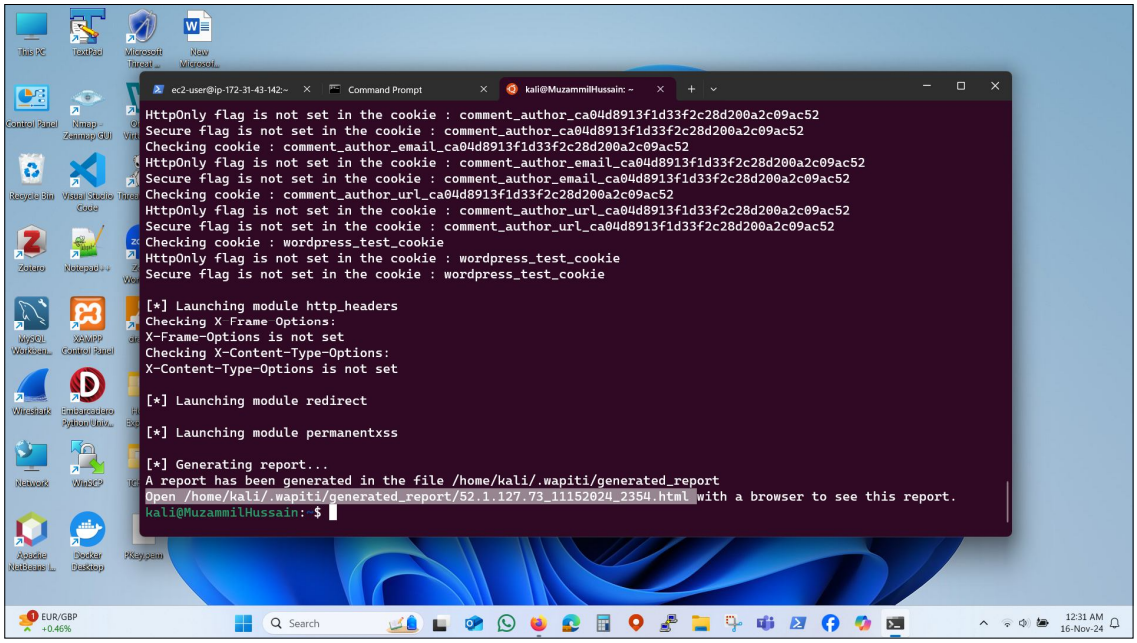


Figure 39 : Wapiti Scanning WordPress (52.1.127.73)-Report Generation

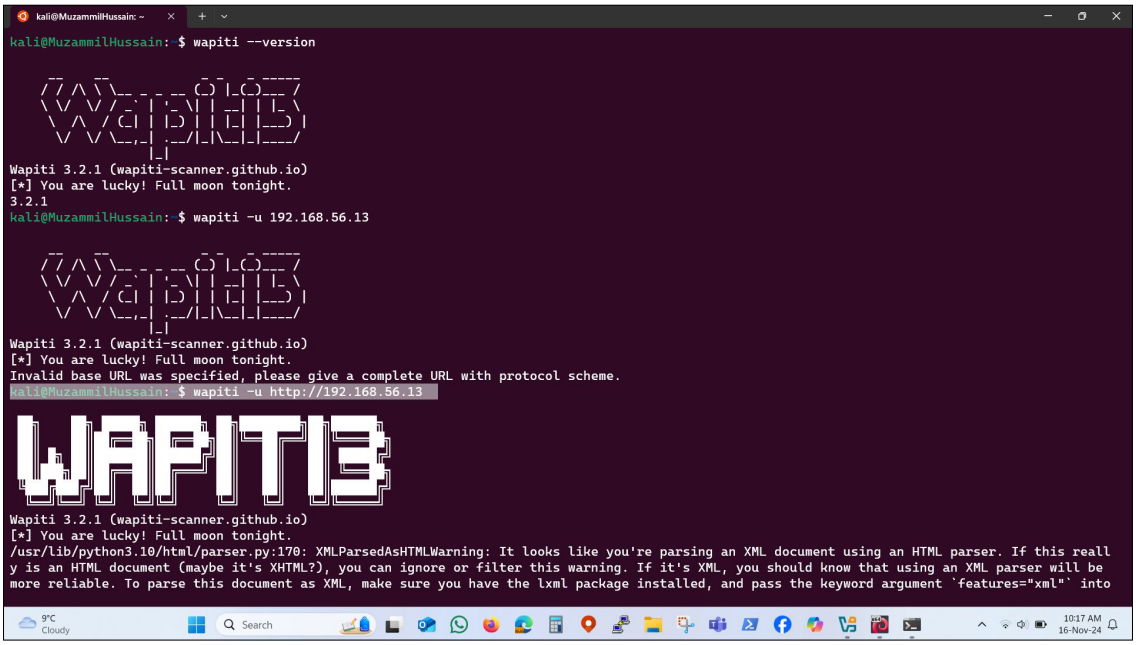


Figure 40 : Wapiti Scanning OWASP-BWA (192.168.56.13)

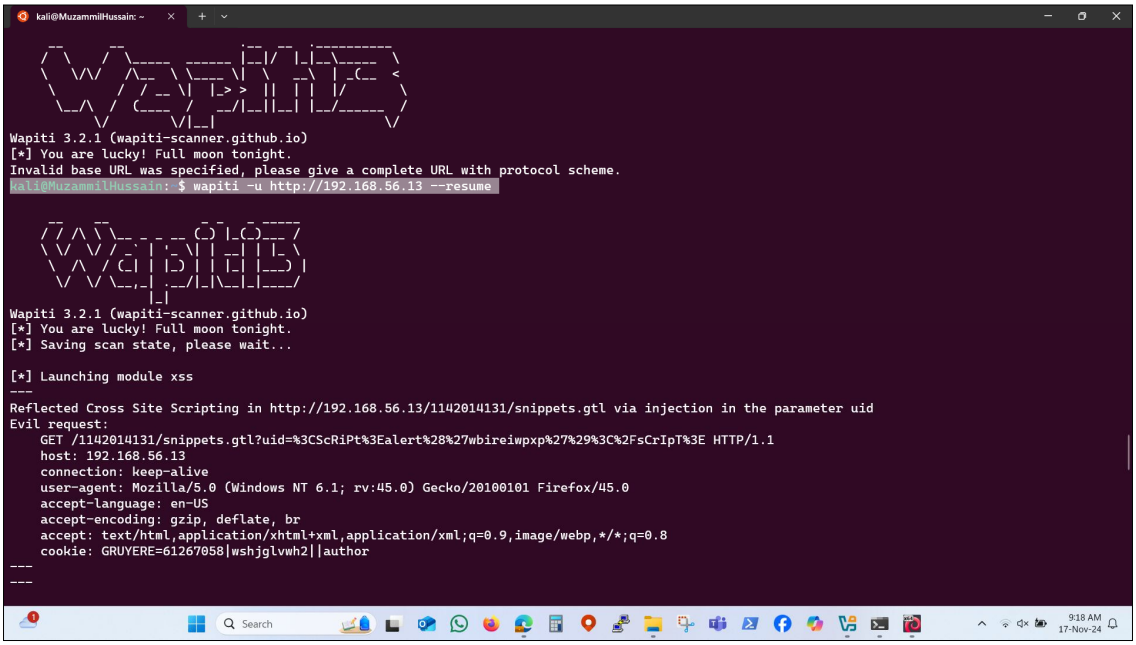


Figure 41 : Wapiti Scanning OWASP-BWA (192.168.56.13)-resume scan

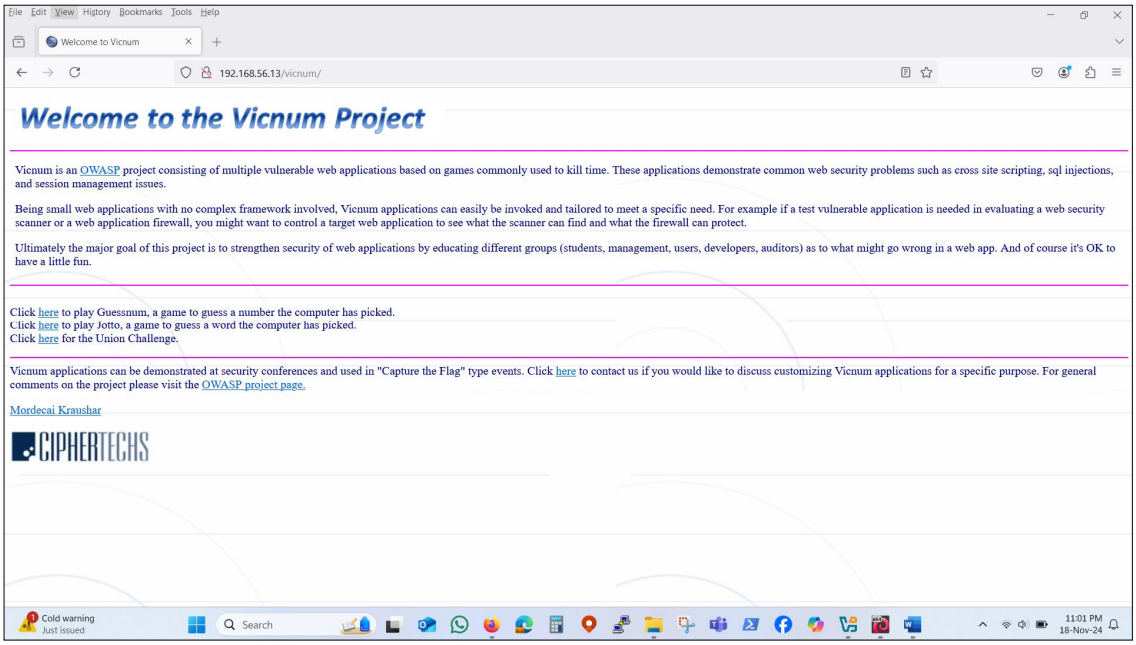


Figure 42 : Random Application Selected (vicnum)

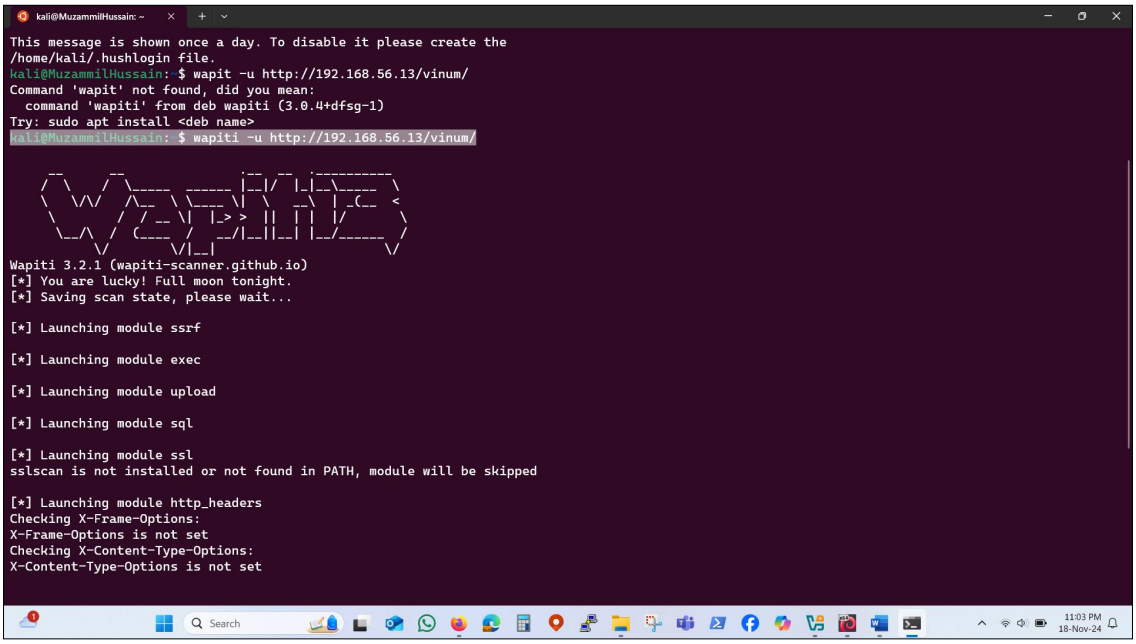


Figure 43 : wapiti scanning (vicnum)

5 Burp Suite

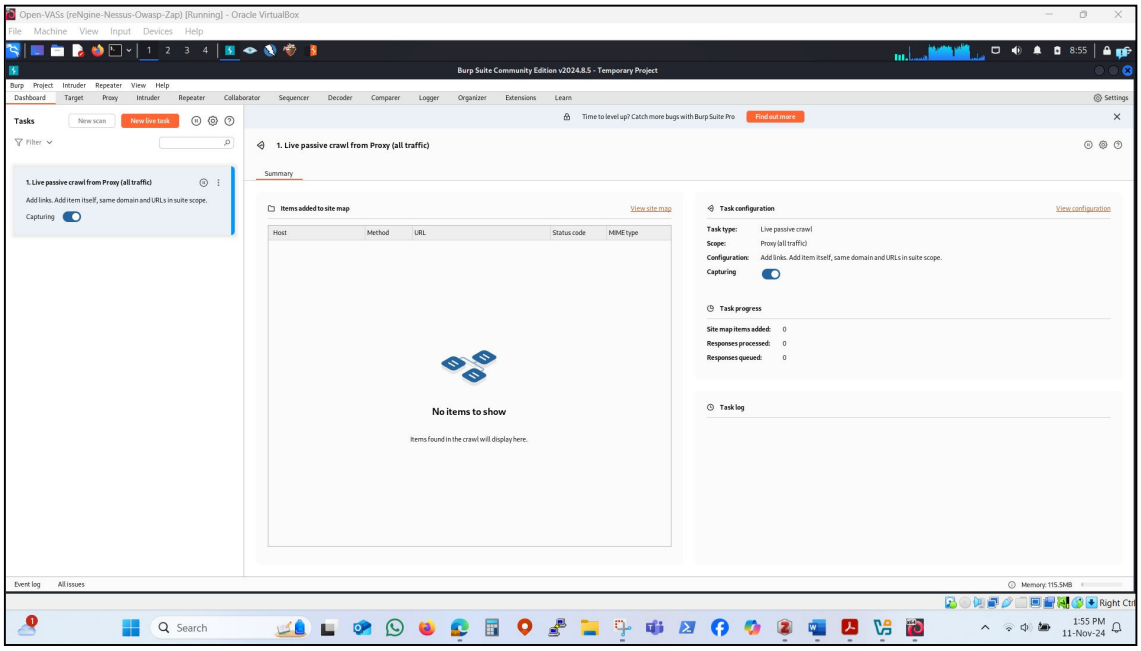


Figure 44 : Burp Suite Dashboard

6 Cloud-Base Application

To test the selected tools on cloud-based application, I decided to install WordPress website on AWS-EC2 instance. type, specifically t2.micro, for its cost-effectiveness, and subsequently chose the Amazon Machine Image (AMI) of Amazon Linux 2 because to its stability and support. We established storage and configured security groups to permit HTTP on port 80, HTTPS on port 443, and SSH on port 22, after the instance information setup, network verification, and activation of auto-assign Public IP. Consequently, upon obtaining the SSH key pair following the instance initiation, I established the connection to the instance utilizing an SSH client. I set the Apache HTTP Server to initiate at startup and deployed it concurrently with the deployment of this software on the AMI EC2 machine. Subsequently, I installed MySQL (MariaDB), protected it, initiated it, and configured it for bootability. Subsequently, I initiated the installation of PHP and configured it for compatibility with Apache. I generated the PHP info file and accessed it using a web browser to verify functionality. The last steps in the deployment procedure were building the requisite databases and transferring the application files to the designated directory inside the LAMP installation, namely the Apache document root.

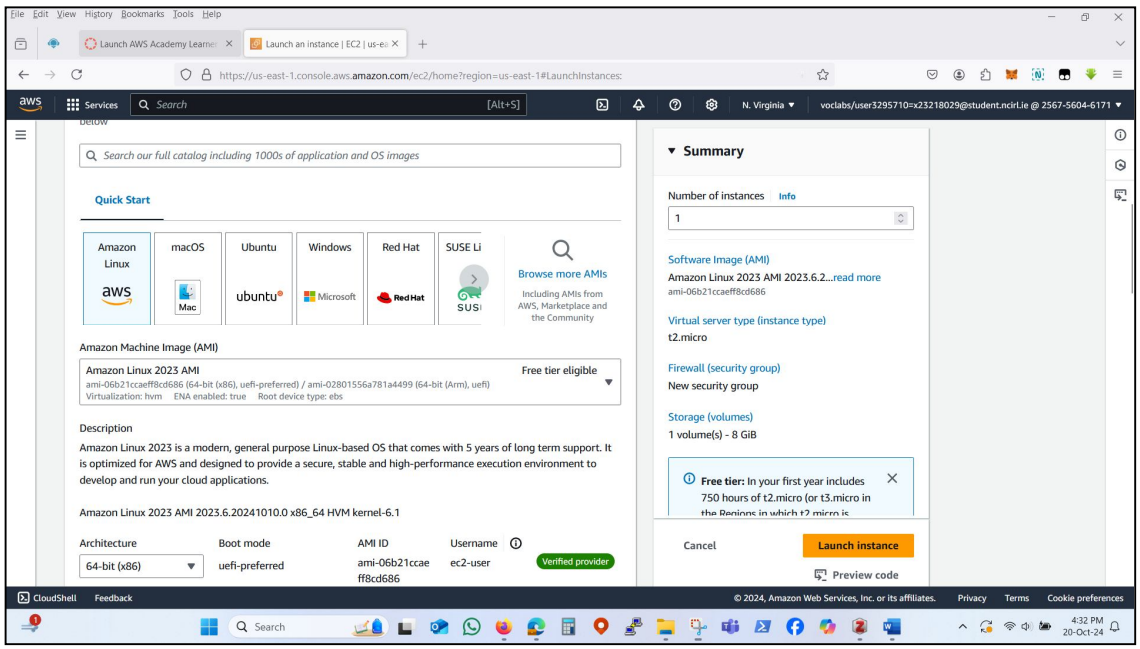


Figure 45 : Selection of Amazon Linux 2023 AMI

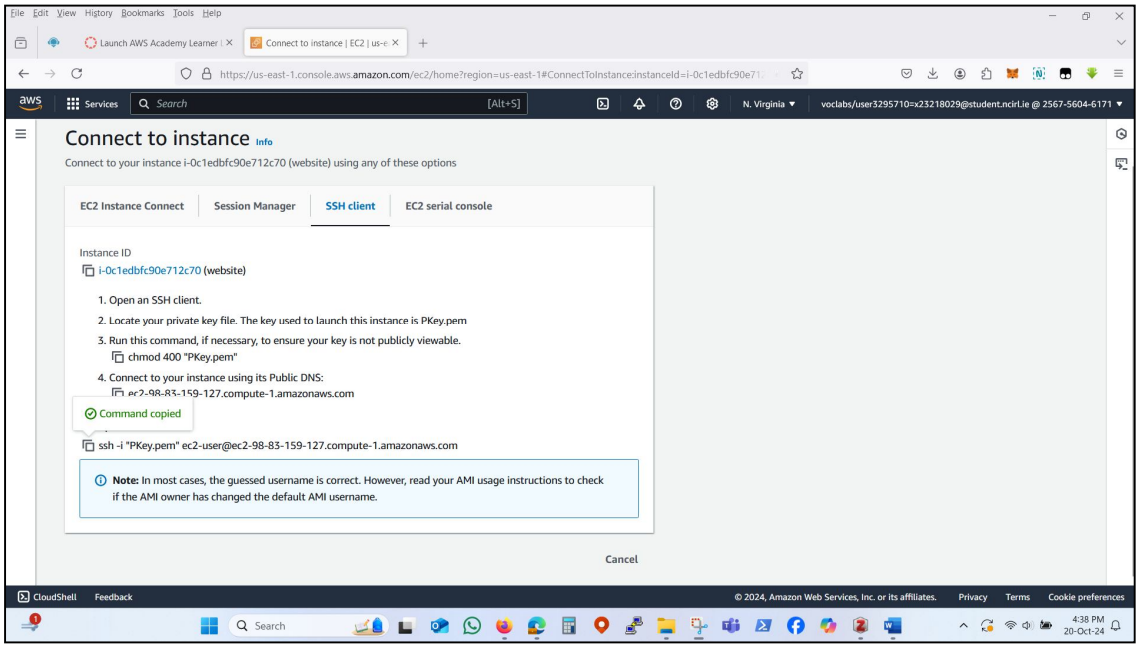


Figure 46 : Connected Through SSH Using Paired Key

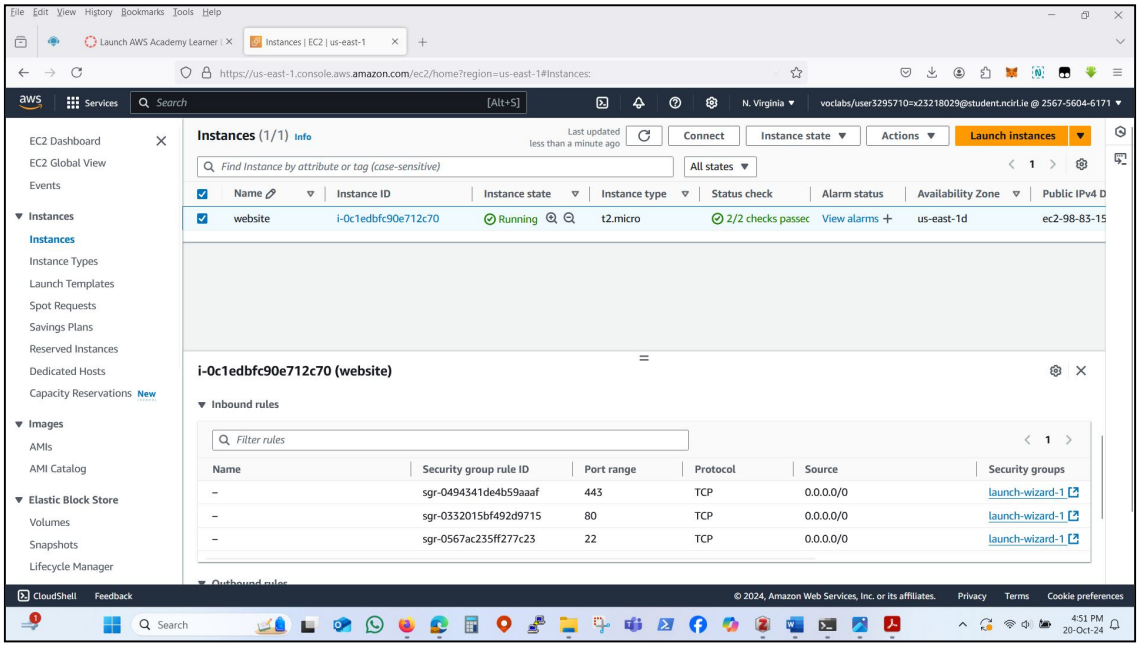
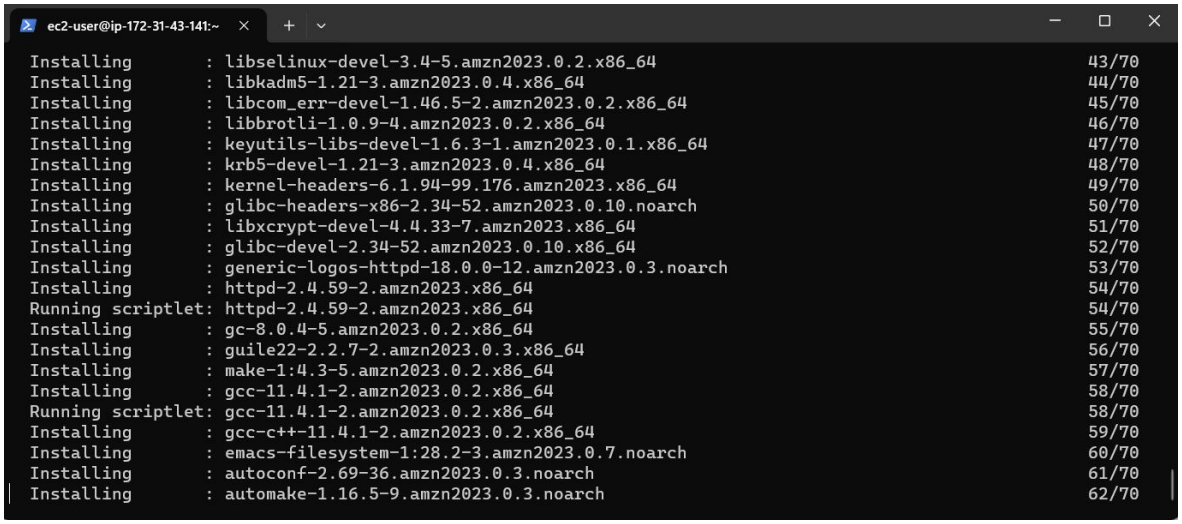


Figure 47 : AWS EC2 Dashboard

6.1 WordPress Installation

I deployed the LAMP (Linux, Apache, MySQL, PHP) stack on our EC2 instance and subsequently installed WordPress on the LAMP stack. This infrastructure is essential for hosting web applications and is finalized by the installation of the LAMP stack (Linux, Apache, MySQL, and PHP). After the installation of the LAMP stack, I downloaded WordPress, extracted it, and proceeded with the installation. This procedure encompassed the establishment of a database, downloading WordPress, unpacking and configuring it, and installing security keys in the wp-config.php file.

WordPress necessitates the establishment and configuration of a MySQL database. A distinct MySQL user for WordPress was established to enhance security. I obtained the most recent version of WordPress from the official website and downloaded its zip package to my PC using the wget program. Upon completion of the download, the WordPress zip package was extracted and relocated to the root/blog directory of the Apache server. Upon completing the WordPress setup, I initiated the installation process by accessing our EC2 machine's Elastic IP address via a browser, which was configured to connect just to the WordPress database.



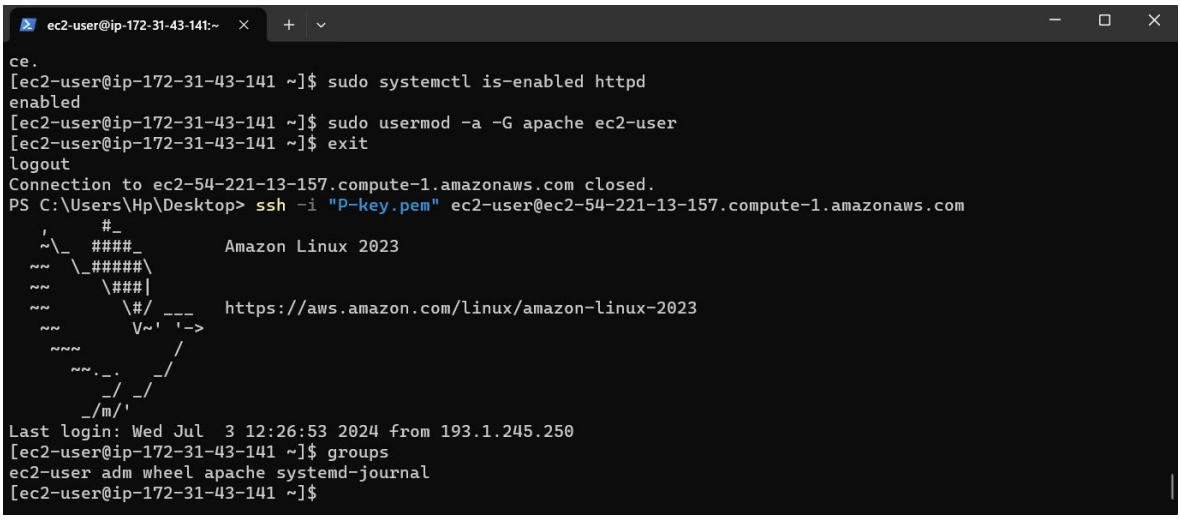
```
ec2-user@ip-172-31-43-141:~
  Installing       : libselinux-devel-3.4-5.amzn2023.0.2.x86_64                          43/70
  Installing       : libkadm5-1.21-3.amzn2023.0.4.x86_64                                 44/70
  Installing       : libcom_err-devel-1.46.5-2.amzn2023.0.2.x86_64                       45/70
  Installing       : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                               46/70
  Installing       : keyutils-libs-devel-1.6.3-1.amzn2023.0.1.x86_64                     47/70
  Installing       : krb5-devel-1.21-3.amzn2023.0.4.x86_64                               48/70
  Installing       : kernel-headers-6.1.94-99.176.amzn2023.x86_64                        49/70
  Installing       : glibc-headers-x86-2.34-52.amzn2023.0.10.noarch                      50/70
  Installing       : libxcrypt-devel-4.4.33-7.amzn2023.x86_64                            51/70
  Installing       : glibc-devel-2.34-52.amzn2023.0.10.x86_64                            52/70
  Installing       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                   53/70
  Installing       : httpd-2.4.59-2.amzn2023.x86_64                                      54/70
  Running scriptlet: httpd-2.4.59-2.amzn2023.x86_64                                      54/70
  Installing       : gc-8.0.4-5.amzn2023.0.2.x86_64                                      55/70
  Installing       : guile22-2.2.7-2.amzn2023.0.3.x86_64                                 56/70
  Installing       : make-1:4.3-5.amzn2023.0.2.x86_64                                    57/70
  Installing       : gcc-11.4.1-2.amzn2023.0.2.x86_64                                    58/70
  Running scriptlet: gcc-11.4.1-2.amzn2023.0.2.x86_64                                    58/70
  Installing       : gcc-c++-11.4.1-2.amzn2023.0.2.x86_64                                59/70
  Installing       : emacs-filesystem-1:28.2-3.amzn2023.0.7.noarch                       60/70
  Installing       : autoconf-2.69-36.amzn2023.0.3.noarch                                61/70
  Installing       : automake-1.16.5-9.amzn2023.0.3.noarch                               62/70
```

Figure 48 : Installation of Apache web server

54.221.13.157

It works!

Figure 49 : Testing webserver using DNS



```
ec2-user@ip-172-31-43-141:~
ce.
[ec2-user@ip-172-31-43-141 ~]$ sudo systemctl is-enabled httpd
enabled
[ec2-user@ip-172-31-43-141 ~]$ sudo usermod -a -G apache ec2-user
[ec2-user@ip-172-31-43-141 ~]$ exit
logout
Connection to ec2-54-221-13-157.compute-1.amazonaws.com closed.
PS C:\Users\Hp\Desktop> ssh -i "P-key.pem" ec2-user@ec2-54-221-13-157.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Jul  3 12:26:53 2024 from 193.1.245.250
[ec2-user@ip-172-31-43-141 ~]$ groups
ec2-user adm wheel apache systemd-journal
[ec2-user@ip-172-31-43-141 ~]$
```

Figure 50 : Add EC2 User to Apache Terminal

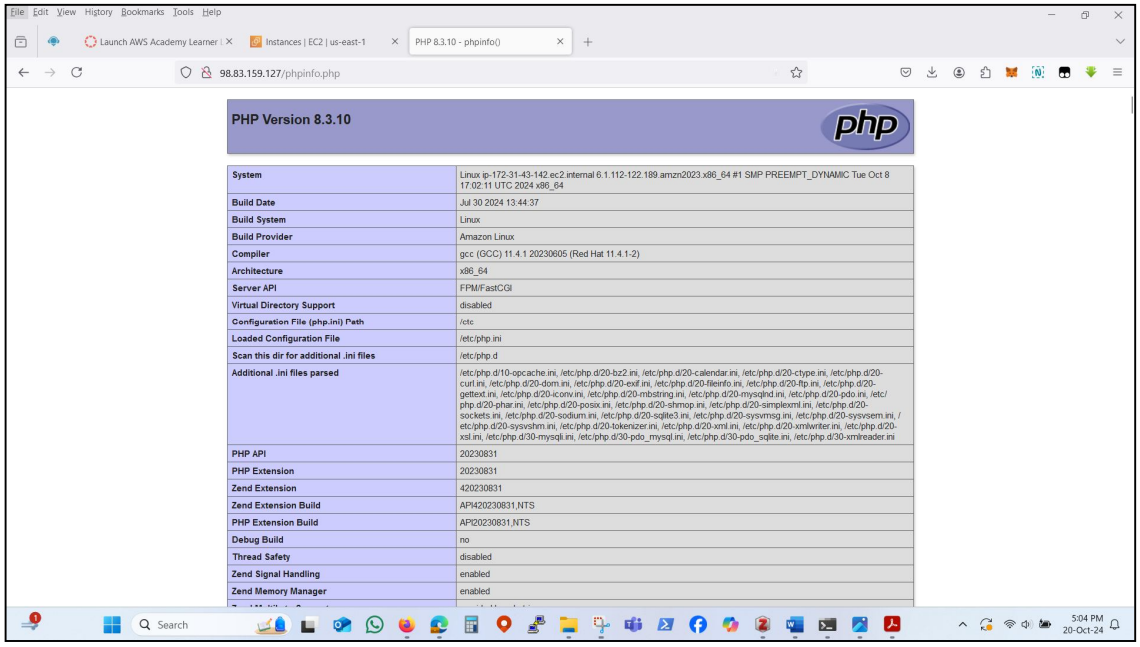


Figure 51 : Testing of Lamp Server

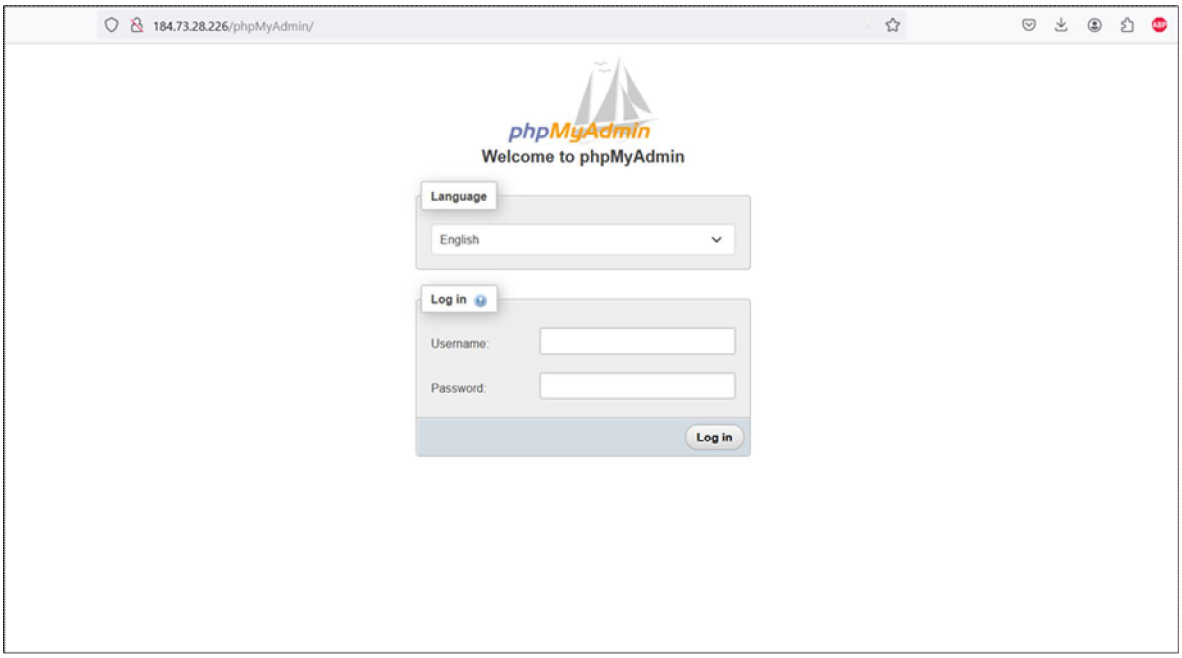


Figure 52 : phpMyAdmin Login Page

```
ec2-user@ip-172-31-34-163:~
--2024-07-05 11:00:46--  https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24696391 (24M) [application/octet-stream]
Saving to: ‘latest.tar.gz’

latest.tar.gz              100%[===================================>]  23.55M  46.0MB/s    in 0.5s

2024-07-05 11:00:47 (46.0 MB/s) - ‘latest.tar.gz’ saved [24696391/24696391]

[ec2-user@ip-172-31-34-163 ~]$ tar -xzf latest.tar.gz
[ec2-user@ip-172-31-34-163 ~]$ sudo systemctl start mariadb httpd
[ec2-user@ip-172-31-34-163 ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 25
Server version: 10.5.23-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]>
```

Figure 53 : Start Database server

```
ec2-user@ip-172-31-34-163:~
 server version for the right syntax to use near 'Allied$$2024' at line 1
MariaDB [(none)]> CREATE USER x23218029 @'localhost' IDENTIFIED BY Allied$$2024;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB
 server version for the right syntax to use near 'Allied$$2024' at line 1
MariaDB [(none)]> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';Ctrl-C -- e
xit!
Aborted
[ec2-user@ip-172-31-34-163 ~]$ sudo systemctl start mariadb httpd
[ec2-user@ip-172-31-34-163 ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 26
Server version: 10.5.23-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE USER 'x23218029'@'localhost' IDENTIFIED BY Allied$$2024;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB
 server version for the right syntax to use near 'Allied$$2024' at line 1
MariaDB [(none)]> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'Allied$@2024';
ERROR 1396 (HY000): Operation CREATE USER failed for 'wordpress-user'@'localhost'
MariaDB [(none)]> CREATE USER 'x23218029'@'localhost' IDENTIFIED BY 'Allied$@2024';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
```

Figure 54 : Creating MySQL Username and Password

```
ec2-user@ip-172-31-34-163:~
 server version for the right syntax to use near 'Allied$$2024' at line 1
MariaDB [(none)]> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';Ctrl-C -- e
xit!
Aborted
[ec2-user@ip-172-31-34-163 ~]$ sudo systemctl start mariadb httpd
[ec2-user@ip-172-31-34-163 ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 26
Server version: 10.5.23-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE USER 'x23218029'@'localhost' IDENTIFIED BY Allied$$2024;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB
 server version for the right syntax to use near 'Allied$$2024' at line 1
MariaDB [(none)]> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'Allied$@2024';
ERROR 1396 (HY000): Operation CREATE USER failed for 'wordpress-user'@'localhost'
MariaDB [(none)]> CREATE USER 'x23218029'@'localhost' IDENTIFIED BY 'Allied$@2024';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> CREATE DATABASE `wordpress-db`;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]>
```

Figure 55 : Creating Database 'WordPress dB'

```
ec2-user@ip-172-31-34-163:~
  GNU nano 5.8                          wordpress/wp-config.php
 * Change these to different unique phrases! You can generate these using
 * the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}.
 *
 * You can change these at any point in time to invalidate all existing cookies.
 * This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define( 'AUTH_KEY',         ']n*v2%69#GC,E[gbP,$06?&ogZIWy+;):38~DM{sp/o9=A7e,&oEbD:Nppz-L=x}' );
define( 'SECURE_AUTH_KEY',  'cBS {*m;zjX dPo!.k`%$=Gd=_.tC*uG=t^7[`+ 2uu3-5?xbu!TBZM<Bp~[_EG}' );
define( 'LOGGED_IN_KEY',    '?^J?OA+-.-i`UT=xfOUWmNIcZDMrSJ+3U<2++%)=3pCNEEn=Xn-x~!Zkd`6^&^(N' );
define( 'NONCE_KEY',        'c]k`u-7:?1~[p~+xMXADf18mlW}XdJDaeMj)BX-I6rZw*dY-%<w;sC$rrS,B^wEC' );
define( 'AUTH_SALT',        'n(=zM3t~X+-Em<KWn5jbqmp|;(lOS!?UJe5A9p~-qj-16a8P+2a^|0E#0Nut4-fb' );
define( 'SECURE_AUTH_SALT', 'MO|[>ff|V*N$+2il|q^PxHz^|vZ.+iX2H `eu#fM9H7{4sTw+jk-C9~t>BWH|fMy' );
define( 'LOGGED_IN_SALT',   'B04hzr);JwIBmeV~RkIp@G}U$wUV%I^p7qdje$=B^I7xVqy7*G*B[ ?mIP`n_RZu' );
define( 'NONCE_SALT',       'E)-XF`wc+tm]]|$9u*@|#h7n0i4GwpvX 4Tie1fny+uu;8OU#1giE-N$zI|bO|-4' );

/**#@-*/

/**
 * WordPress database table prefix.
 *
 * You can have multiple installations in one database if you give each

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

Figure 56 : Changing Values of key and salt

```
ec2-user@ip-172-31-34-163:~
  GNU nano 5.8                    wordpress/wp-config.php                    Modified
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress-db' );

/** Database username */
define( 'DB_USER', 'x23218029' );

/** Database password */
define( 'DB_PASSWORD', 'Allied$@2024' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo
```

Figure 57 : Editing the WordPress config file

```
ec2-user@ip-172-31-34-163:~
=================================================================================================
 Package                        Architecture  Version                          Repository      Size
=================================================================================================
Installing:
 php8.3-gd                      x86_64        8.3.7-1.amzn2023.0.1             amazonlinux     43 k
Installing dependencies:
 cairo                          x86_64        1.17.6-2.amzn2023.0.1            amazonlinux    684 k
 fontconfig                     x86_64        2.13.94-2.amzn2023.0.2           amazonlinux    273 k
 fonts-filesystem               noarch        1:2.0.5-12.amzn2023.0.2          amazonlinux    9.5 k
 freetype                       x86_64        2.13.0-2.amzn2023.0.1            amazonlinux    422 k
 gd                             x86_64        2.3.3-5.amzn2023.0.3             amazonlinux    139 k
 google-noto-fonts-common       noarch        20201206-2.amzn2023.0.2          amazonlinux     15 k
 google-noto-sans-vf-fonts      noarch        20201206-2.amzn2023.0.2          amazonlinux    492 k
 graphite2                      x86_64        1.3.14-7.amzn2023.0.2            amazonlinux     97 k
 harfbuzz                       x86_64        7.0.0-2.amzn2023.0.1             amazonlinux    868 k
 jbigkit-libs                   x86_64        2.1-21.amzn2023.0.2              amazonlinux     54 k
 langpacks-core-font-en         noarch        3.0-21.amzn2023.0.4              amazonlinux     10 k
 libX11                         x86_64        1.7.2-3.amzn2023.0.4             amazonlinux    657 k
 libX11-common                  noarch        1.7.2-3.amzn2023.0.4             amazonlinux    152 k
 libXau                         x86_64        1.0.9-6.amzn2023.0.2             amazonlinux     31 k
 libXext                        x86_64        1.3.4-6.amzn2023.0.2             amazonlinux     41 k
 libXpm                         x86_64        3.5.15-2.amzn2023.0.3            amazonlinux     65 k
 libXrender                     x86_64        0.9.10-14.amzn2023.0.2           amazonlinux     28 k
 libjpeg-turbo                  x86_64        2.1.4-2.amzn2023.0.5             amazonlinux    190 k
 libpng                         x86_64        2:1.6.37-10.amzn2023.0.6         amazonlinux    128 k
 libtiff                        x86_64        4.4.0-4.amzn2023.0.18            amazonlinux    213 k
 libwebp                        x86_64        1.2.4-1.amzn2023.0.6             amazonlinux    341 k
```

Figure 58 : PHP Graphics Drawing Library

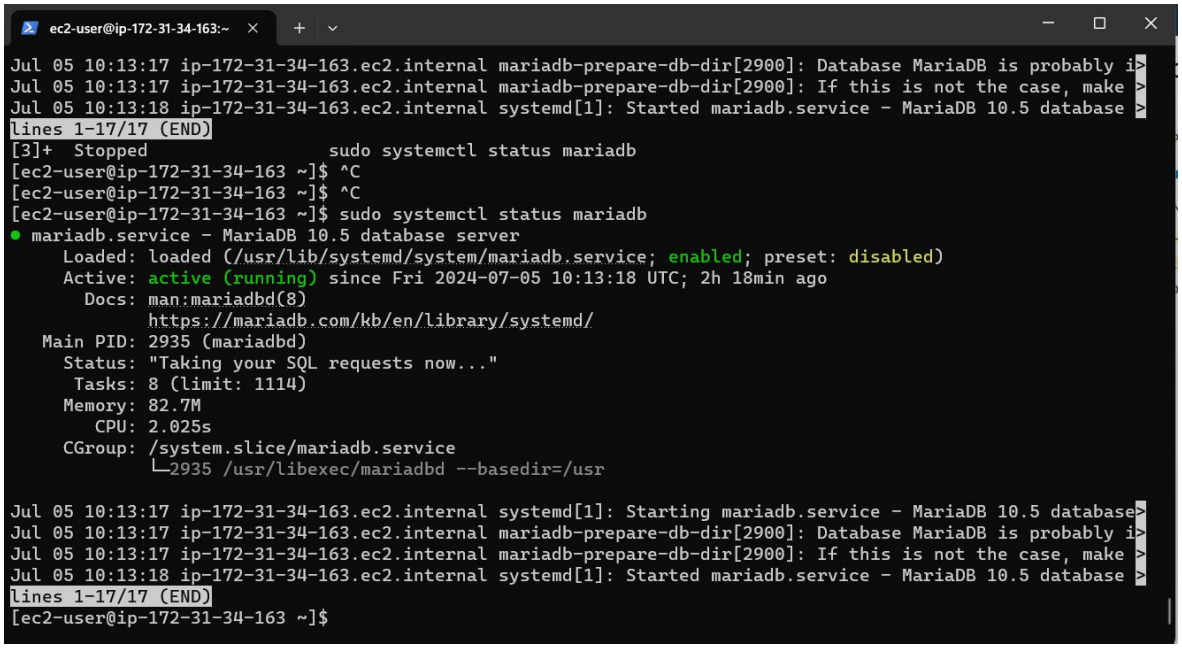


Figure 59 : Verification of Database Server is running

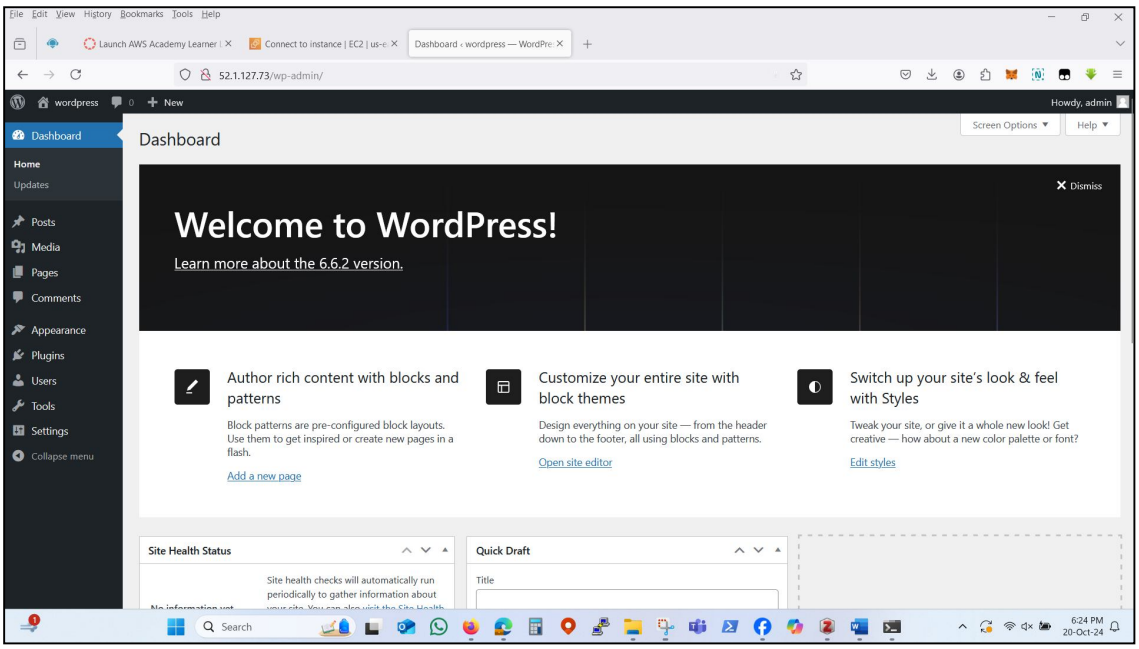


Figure 60 : WordPress Main Page

6.2 Scanning Results

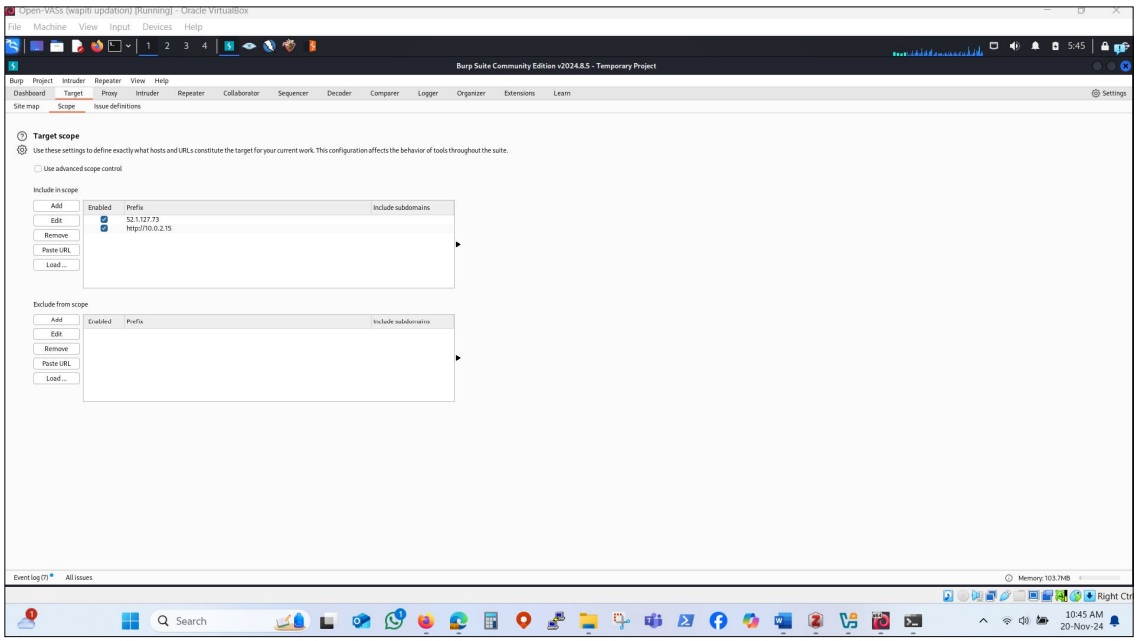


Figure 61 : Add Target IPs to scope

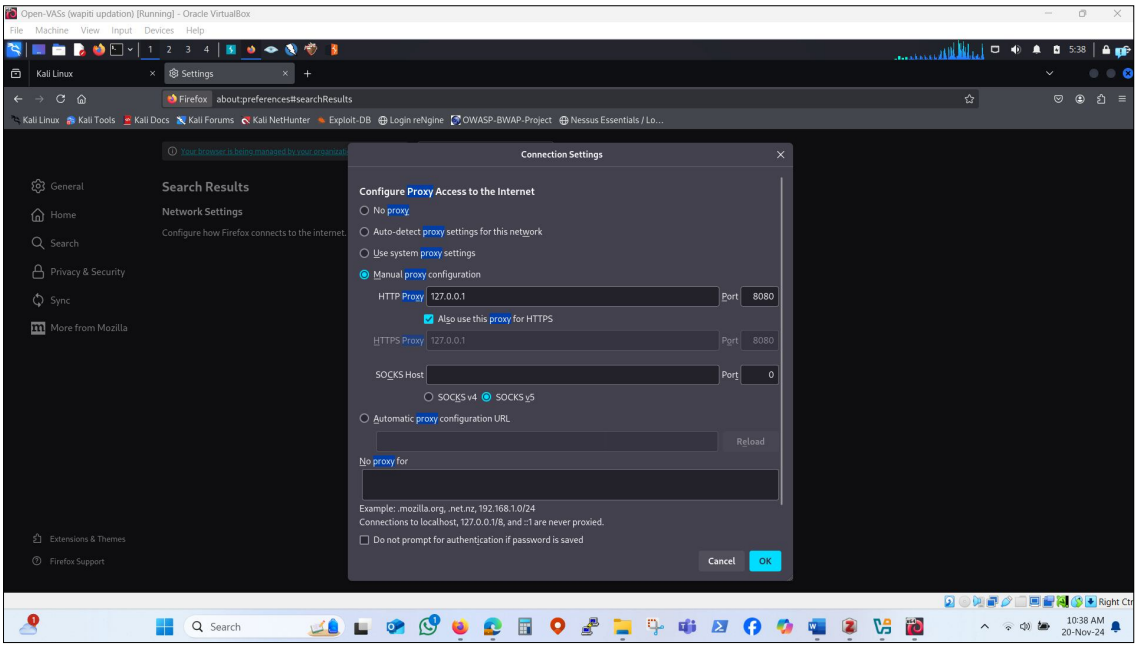


Figure 62 : Browser Proxy Settings

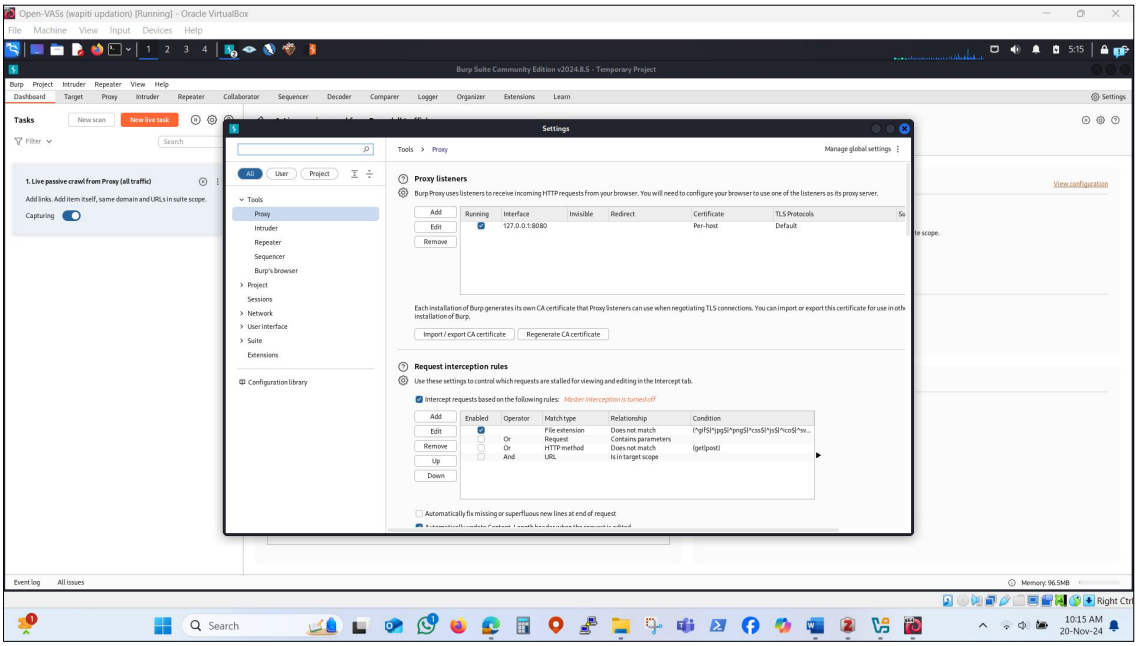


Figure 63 : Checking localhost and Proxy settings

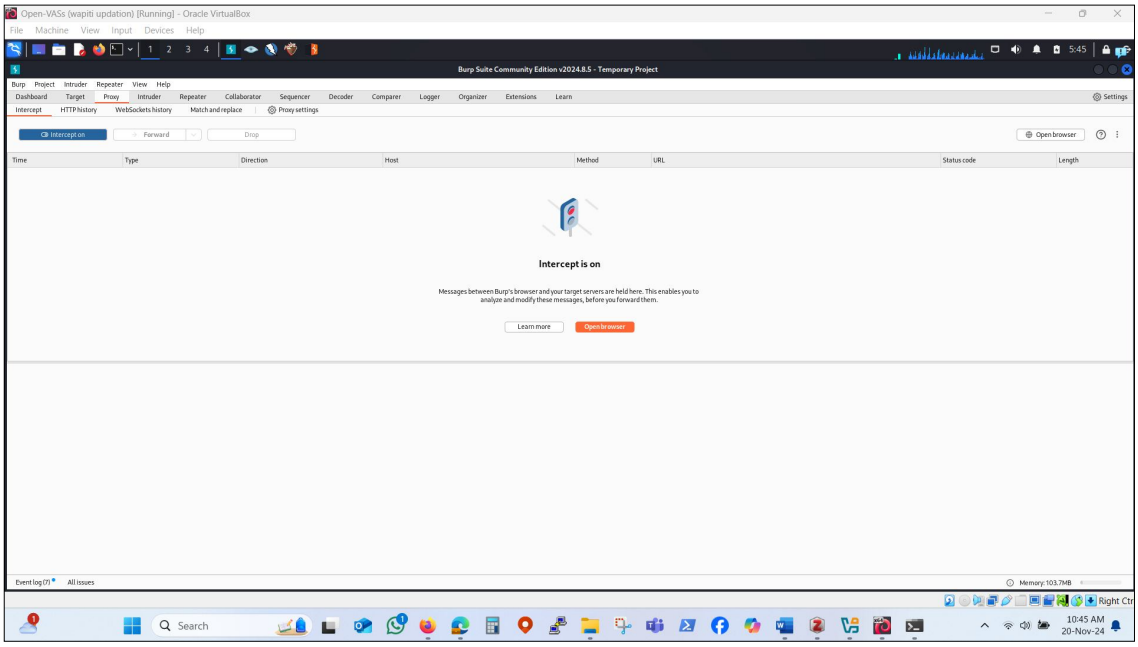


Figure 64 : Intercept Turning On

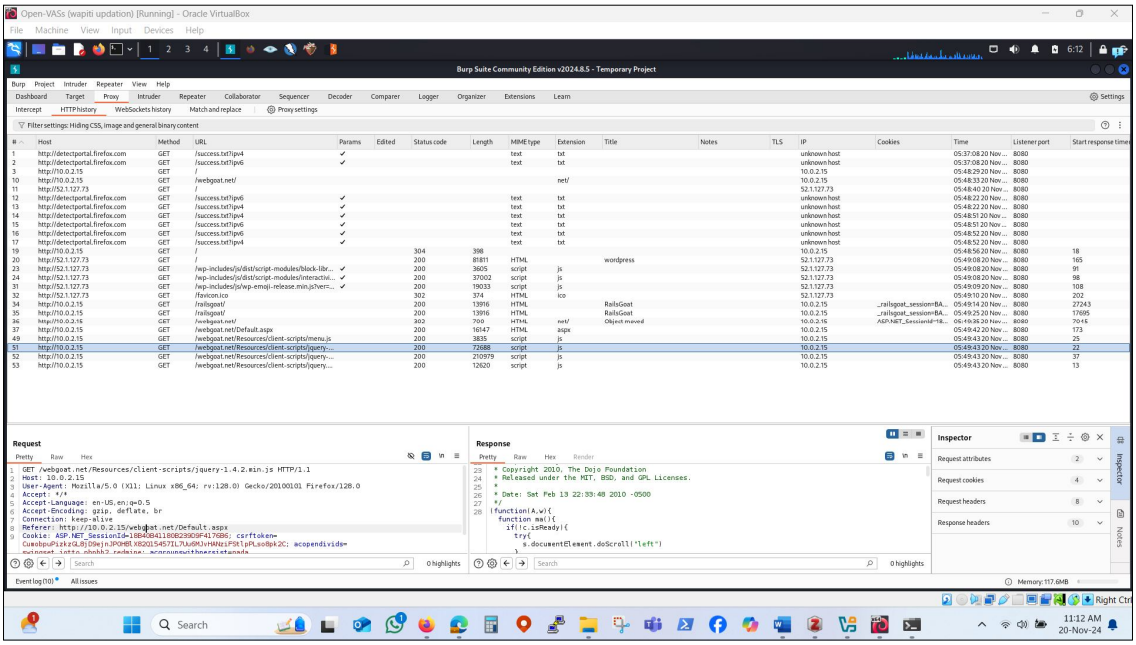


Figure 65 : Manual Testing

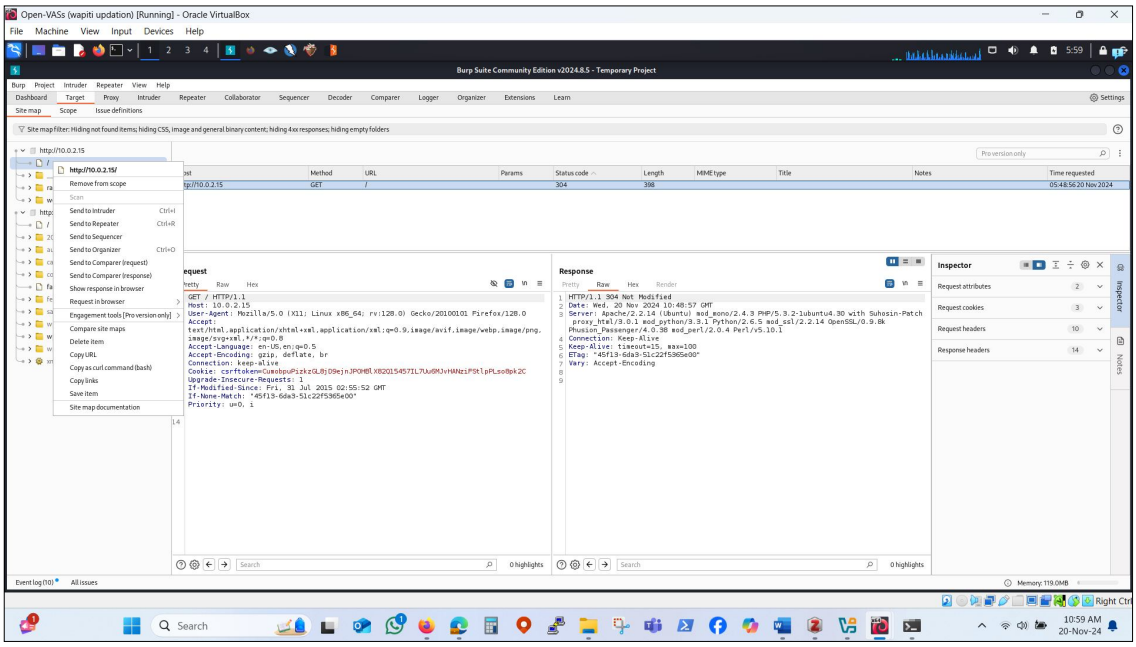


Figure 66 : Scan Option is not visible (community edition)

`

References

References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at https://libguides.ncirl.ie/referencing
You can use a reference management system such as Zotero or Mendeley to cite in MS Word.

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* 27(5): 1310–1333.

Feng, G. and Buyya, R. (2016). Maximum revenue-oriented resource allocation in cloud, *IJGUC* 7(1): 12–21.

Gomes, D. G., Calheiros, R. N. and Tolosana-Calasanz, R. (2015). Introduction to the special issue on cloud computing: Recent developments and challenging issues, *Computers & Electrical Engineering* 42: 31–32.

Kune, R., Konugurthi, P., Agarwal, A., Rao, C. R. and Buyya, R. (2016). The anatomy of big data computing, *Softw., Pract. Exper.* 46(1): 79–105.