WEB SERVER SECURITY ASSESSMENT

PROJECT REPORT

SUBMITTED BY

TEAM : 03

DIPAK KUMAR (E23CSEU0645)

ABHISHEK KUMAR (E23CSEU0669)

AARAV GAUR (E23CSEU0656)

PRAHARSH SINGH (E23CSEU0629)

MANISHA PARIHAR (E23CSEU0737)

BENNETT UNIVERSITY

TIMES OF INDIA GROUP

SUBMITTED TO

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND TECHNOLOGY, BENNETT UNIVERSITY

GREATER NOIDA, 201310, UTTAR PRADESH, INDIA

CERTIFICATE

This is to certify that the project entitled "Web Server Security Assessment" submitted by

Dipak Kumar, Abhishek Sharma, Aarav Gaur, Praharsh Singh, and Manisha Parihar to the School of Computer Science Engineering and Technology, Bennett University, is a record of bona fide work carried out under our guidance and supervision in partial fulfilment of the requirements for the award of the degree.

This project embodies the results of original work and has not been submitted for the award of any other degree.

Faculty In-charge
(Signature)

Date: _____

Place: Greater Noida

TABLE OF CONTENTS

ACKNOWLEDGMENT

We take immense pleasure in expressing our heartfelt gratitude to the School of Computer Science Engineering and Technology, Bennett University, for providing us with an excellent platform to explore and enhance our technical knowledge through this project, titled "Web Server Security Assessment."

This project has been a journey of learning, collaboration, and practical exposure to real-world cybersecurity challenges. We would especially like to thank our project guide, whose expert insights, timely guidance, and unwavering support continuously motivated us to refine our approach and achieve a high standard of work.

We are deeply grateful to all our faculty members who shared their vast knowledge with us throughout our academic journey. Their teaching laid the strong foundation upon which this project was built.

We also extend our sincere thanks to the technical staff, who ensured the availability of required resources and lab infrastructure, enabling us to carry out our experiments and validations smoothly.

We would be remiss if we did not acknowledge the constant support, encouragement, and patience shown by our families and friends, who stood by us during the more demanding phases of the project.

Lastly, we owe a great deal to each team member for their dedication, open-minded discussions, and the spirit of teamwork that helped us overcome challenges and meet our project goals. This project has not only enhanced our technical expertise but also taught us the importance of collaboration, perseverance, and critical thinking.

ABSTRACT

In an age where digital infrastructures form the foundation of almost every organization, web server security has become more crucial than ever before. A single overlooked vulnerability in a web server can open the door to devastating cyberattacks, resulting in loss of data, financial damage, and erosion of trust.

Motivated by the real-world importance of cybersecurity, our project titled "Web Server Security Assessment" set out to build an automated, efficient solution to identify vulnerabilities in web servers.
We combined the strengths of two powerful open-source tools — Nmap for network scanning and Nikto for vulnerability scanning — with the flexibility of Python scripting to automate the entire process.

Through our project, we aimed to streamline vulnerability detection by scanning servers for open ports, outdated services, and insecure configurations, and mapping these findings to known Common Vulnerabilities and Exposures (CVEs), thereby giving system administrators clear, actionable insights.

The approach we adopted mimics how cybersecurity professionals in the industry perform security assessments but focuses on simplicity, automation, and speed. By generating a comprehensive, structured security report, our tool bridges the gap between technical findings and practical remediation strategies.

We also envisioned future enhancements — integrating SSL/TLS configuration analysis, expanding towards application-level vulnerability assessments using tools like OWASP ZAP, and visualizing results through web dashboards.

In completing this project, we have deepened our understanding of both cybersecurity principles and the importance of robust automation in securing modern web infrastructures.
We hope that our work serves as a small but meaningful contribution toward building safer digital systems.

INTRODUCTION

The world is increasingly interconnected, with web servers acting as critical gateways to the digital services that power our daily lives. From e-commerce platforms to government portals, almost every online interaction relies on the proper functioning — and security — of a web server.

Given this dependence, the security of web servers cannot be an afterthought. Attackers often target servers because compromising them can grant unauthorized access to vast amounts of sensitive data or control over systems.

Recognizing the importance of proactive security measures, we set out to create a project that automates the process of web server security assessment. By developing a tool that can quickly and accurately identify vulnerabilities, we aim to empower organizations to stay one step ahead of potential threats.

This project is not only a technical endeavour but also a step towards building a security-first mindset — one that acknowledges that identifying and addressing vulnerabilities early is key to maintaining trust and resilience in a digitally connected world.

OBJECTIVE

The core mission of our project is to **empower system administrators and security teams** to better protect their web servers through **automated vulnerability assessments**.

Specifically, we wanted to:

- Create a simple yet powerful tool that **scans web servers without requiring advanced cybersecurity expertise**.
- Detect **vulnerabilities in software versions**, **open ports**, and **server configurations** that could be exploited by attackers.
- Cross-reference findings with industry-standard databases, particularly **Common Vulnerabilities and Exposures (CVE)** records, to ensure credibility and prioritization.
- Provide **clear, structured, and actionable reports** that guide administrators not only in identifying problems but also in understanding how to fix them.

By focusing on **automation, transparency, and reliability**, we aimed to create a solution that would be useful both for academic learning and for real-world applications.

TOOLS AND TECHNOLOGY

Selecting the right tools was crucial to the success of this project.
After thorough research, we chose a combination of open-source, industry-trusted tools and custom scripting to deliver the best possible results:

- Nmap (Network Mapper):
  Widely used by cybersecurity professionals, Nmap allows us to explore a network, discover running services, and gather crucial details such as service versions. This information often reveals vulnerabilities based on outdated software.
- Nikto:
  Nikto specializes in scanning web servers for potential vulnerabilities, such as insecure files, outdated scripts, and misconfigured servers. It helps surface vulnerabilities that might otherwise remain hidden until it's too late.
- Python:
  With Python's simplicity and versatility, we were able to automate the scanning processes, parse outputs intelligently, map vulnerabilities to CVEs, and generate detailed reports — all through one seamless script.
- Python Libraries (os, sys, subprocess, re):
  These libraries allowed us to interact with the operating system, execute external commands, and use regular expressions to extract important patterns like CVE identifiers from raw scan results.

Choosing these technologies allowed us to balance power with usability, ensuring that the final tool could be used flexibly in both test and production environments.

METHODOLOGY

We followed a **structured, phased approach** in executing this project to ensure clarity, accuracy, and repeatability.
Each step was designed to mimic the workflows adopted by cybersecurity professionals while keeping in mind the learning goals of our academic journey.

Step 1: Server Setup

Before scanning, we needed a realistic test environment.
We set up a web server (Apache or Nginx) configured with **intentional security weaknesses** — like outdated modules or default configurations — that made it easier to simulate real-world vulnerabilities.

This step taught us how even small misconfigurations could create significant security gaps.

Step 2: Scanning

We then performed scanning operations in two major phases:

- **Nmap Scanning:**
  Using Nmap, we identified open ports, protocols, and the versions of services running on our server. This gave us a broad "surface map" of potential attack vectors.
- **Nikto Scanning:**
  Nikto helped us dive deeper into the web server layer, identifying known vulnerabilities and security misconfigurations.

Step 3: Analysis

Rather than manually reading through long scan reports, we developed a **Python parser** to extract:

- Vulnerability descriptions
- Service information
- CVE references (where available)

We also **cross-referenced** these CVEs with the **National Vulnerability Database (NVD)** to validate their severity and relevance.

Step 4: Reporting

Finally, our tool **automatically generated a structured text report** that included:

- A vulnerability summary
- List of affected services
- Associated CVEs with external references
- Clear suggestions for remediation

By automating reporting, we ensured **consistency, accuracy, and ease of understanding** for users.

ARCHITECTURE AND WORKFLOW DIAGRAM

We designed our system to follow a simple yet effective linear flow, ensuring that each step builds on the information gathered from the previous one.

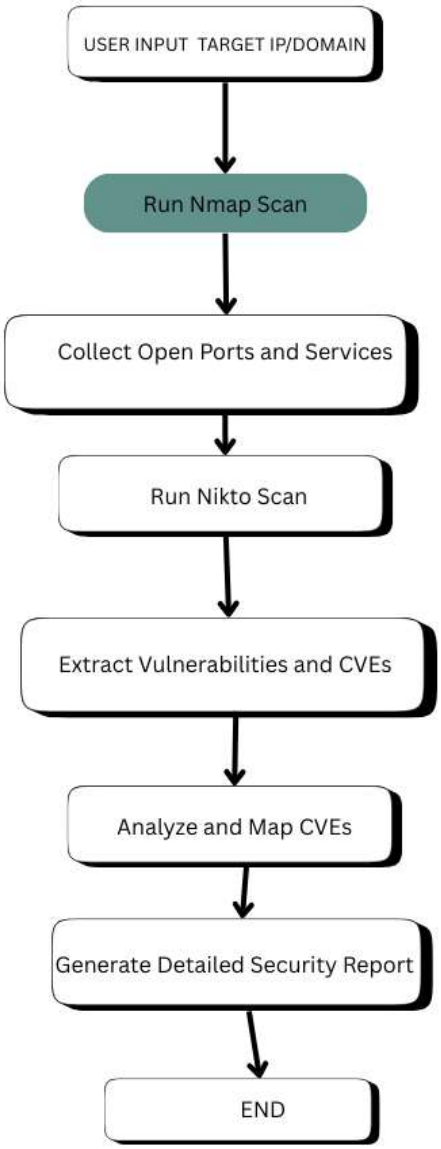
System Architecture Overview

At a high level, the architecture includes the following main components:

1. Input Layer:
   Where the user provides the target IP address or domain name.
2. Scanning Layer:
   This layer uses Nmap to scan for open ports and running services and Nikto to perform vulnerability assessments on the web server.

3. Processing and Analysis Layer:
   The raw outputs from the scanners are parsed intelligently using our custom-built Python script, extracting relevant information like vulnerabilities, outdated software, and CVEs.

4. Reporting Layer:
   A structured, comprehensive security report is generated, summarizing the findings and providing remediation recommendations.


USER INPUT TARGET IP/DOMAIN
Run Nmap Scan
Collect Open Ports and Services
Run Nikto Scan
Extract Vulnerabilities and CVEs
Analyze and Map CVEs
Generate Detailed Security Report
END

DETAILED WORKING PROCESS

Step 1: Receiving User Input

When the script is executed, the first thing it does is **prompt the user to provide a target** — either an IP address or a domain name (like livia.co.in).

Step 2: Running Nmap Scans

The script initiates an **Nmap scan** on the provided target.
Here, Nmap probes for:

- Open ports (e.g., HTTP port 80, HTTPS port 443)
- Running services (e.g., Apache, Nginx)
- Software versions

```
nmap -sV 185.151.30.221
```

Step 3: Running Nikto Scans

Next, the script runs a **Nikto scan** specifically on the web server found earlier (usually on ports 80 or 443).

Nikto digs deeper, checking:

- Dangerous files/folders (/admin/, /backup/, etc.)
- Outdated server software
- Configuration errors

- SSL/TLS problems

```
nikto -h http://185.151.30.221
```

Step 4: Parsing and Extracting Vulnerabilities

Both Nmap and Nikto generate **large amounts of output**.
Reading these manually would be time-consuming and error-prone.

Our Python script **automatically processes** these outputs to:

- Extract relevant vulnerabilities
- Detect any mentioned **CVE IDs**
- Summarize findings clearly

Step 5: Cross-referencing CVEs

Once CVE identifiers are extracted (e.g., CVE-2019-1234), the tool **cross-references** them with the official **National Vulnerability Database (NVD)**.

This tells the user:

- What each vulnerability means
- How severe it is
- What remediation steps are recommended

Step 6: Generating the Security Report

Finally, all findings are neatly compiled into a **structured .txt report**.

The report includes:

- Server information
- Open ports and services
- Vulnerability summary

- Associated CVEs
- Actionable remediation suggestions

CODE WALKTHROUGH

When we first started building this project, we knew it would be more than just writing a simple Python script — we wanted to create a **powerful tool** that could **automate an entire web server security assessment**, in a way that was **clear, professional, and usable even for beginners**.

Here's a detailed walkthrough of the **complete code we developed**, and how each part fits into the bigger picture.

Main Building Blocks of the Script

Instead of cramming everything into one long script, we broke the work into **small, clean, focused functions**.
This made it easier to debug, upgrade, and explain — something we believe is important for real-world software development.

Each function has **one clear job**.

- **[run_command(command)] : Our Gateway to the System**

Why We Needed It:

Since Nmap and Nikto are **external tools** (not Python libraries), we had to find a way to run them from inside Python and capture whatever they printed to the screen.

What It Does:
This function takes a shell command as a string, runs it, and captures both success and error outputs — so we don't lose any important information.

It even handles errors gracefully by catching exceptions, which helps in scenarios where the server is down or the command fails.

Example in Action:
If we pass "nmap-sV 192.168.1.10" into run_command(), it will run the scan and return all the output directly to us inside Python.

This single function made it possible to control entire scanning operations from one place!

- [scan_with_nmap(target)] : Mapping the Battlefield

Why We Needed It:
Before assessing a web server's vulnerabilities, we first need to know what services are running.
Nmap is perfect for that.

What It Does:
This function:

- Takes the target IP or domain.
- Builds a powerful Nmap command (-Sv -T4 -p-) to:
  - Detect services and versions.
  - Scan all ports (not just common ones).

Extra Care:
We added the-T4 flag to speed up the scanning without missing important data.

- [scan_with_nikto(target)] : Digging Deeper into the Web Layer

Why We Needed It:
Once we knew the server was running, we wanted to find real vulnerabilities — outdated software, default pages, dangerous files, etc.

What It Does:
This function runs a Nikto scan on the target, treating it as a web server (HTTP protocol).

How It Helps:

- If the server has an open admin panel.
- If it's leaking sensitive information.
- If it's running old versions of software.

Nikto tells us what's wrong beyond just "what's running."

- [extract_vulnerabilities(nikto_output)] : Finding the Needles in the Haystack

Why We Needed It:
Nikto's output can sometimes be huge — and reading hundreds of lines manually is impractical.

What It Does:
This function cleverly filters only the important parts:

- Vulnerabilities
- Security misconfigurations
- References to CVEs

It looks for keywords like "OSVDB," "vulnerable," "uncommon header," etc., and picks those lines out.

Result:
A short, sharp summary of the real risks — ready for the report!

- [extract_cve_ids(output)] : Linking to the Global Vulnerability Database

Why We Needed It:
CVEs are official IDs assigned to cybersecurity vulnerabilities.
Including them makes our report professional and credible.

What It Does:
This function scans through the scanner outputs, finds all the CVE IDs, removes duplicates, and sorts them nicely.

Example:
From a messy Nikto output, it extracts neat items like:

- CVE-2020-1234
- CVE-2019-5678

This allows IT teams to instantly research each vulnerability using trusted government databases like NVD.

- [generate_report(target, nmap_data, nikto_data, vuln_summary, cve_ids)] : Our Storyteller

Why We Needed It:
After scanning and finding vulnerabilities, we needed a way to communicate our findings clearly to whoever reads the report — admins, developers, or security managers.

What It Does:

- Creates a structured .txt file.
- Divides it into easy-to-read sections:
  - Nmap Results
  - Nikto Results
  - Vulnerability Summary
  - CVE References (with direct clickable links to NVD)

Bonus:
If no CVEs are found, the report says "No CVEs detected," instead of leaving the reader confused.

Result:
A polished, professional, detailed security report — automatically generated in seconds!

- [main()] : The Orchestra Conductor

Why We Needed It:
Someone had to "orchestrate" everything:
Getting user input, running scans, extracting findings, writing reports.

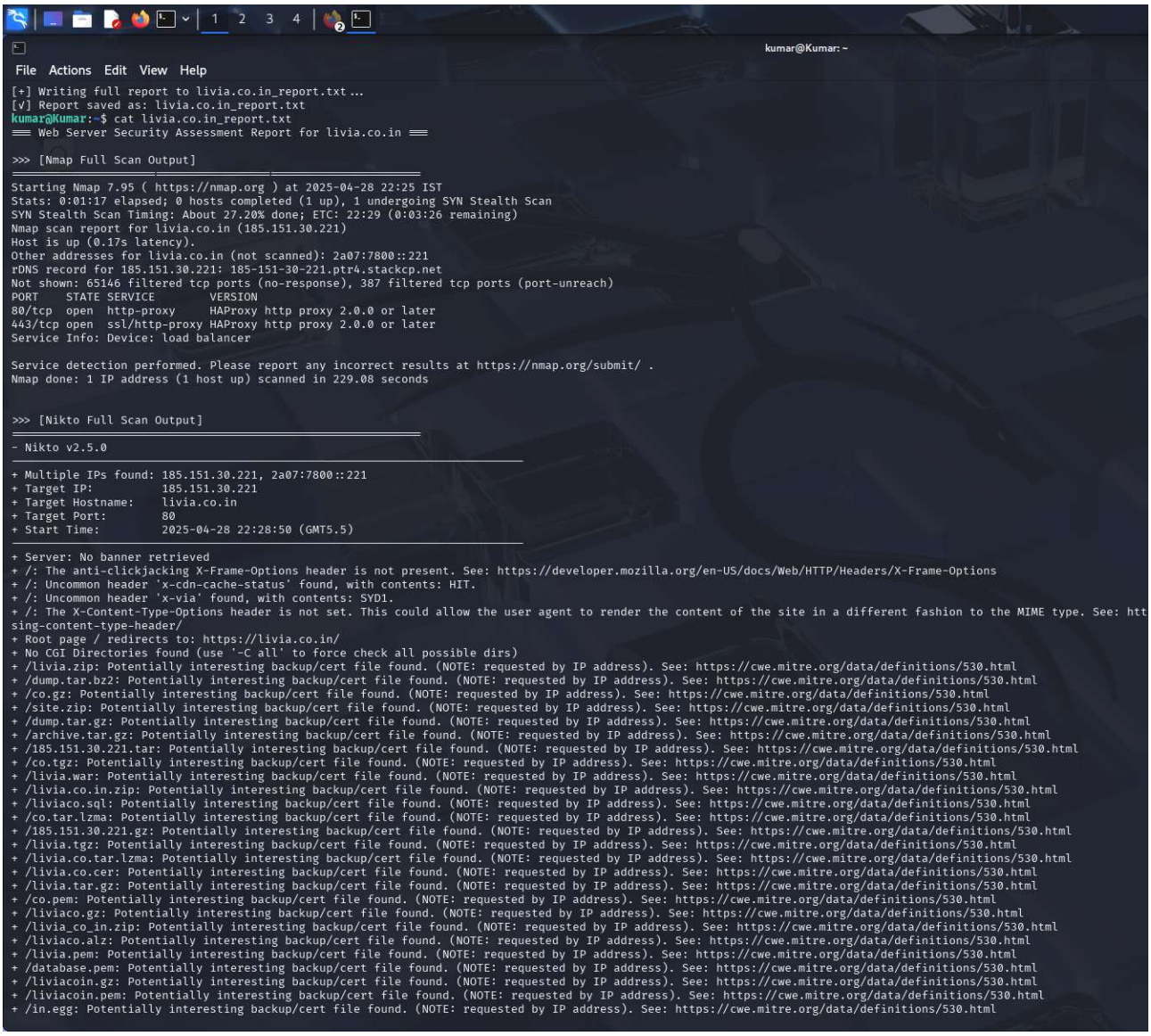
What It Does:

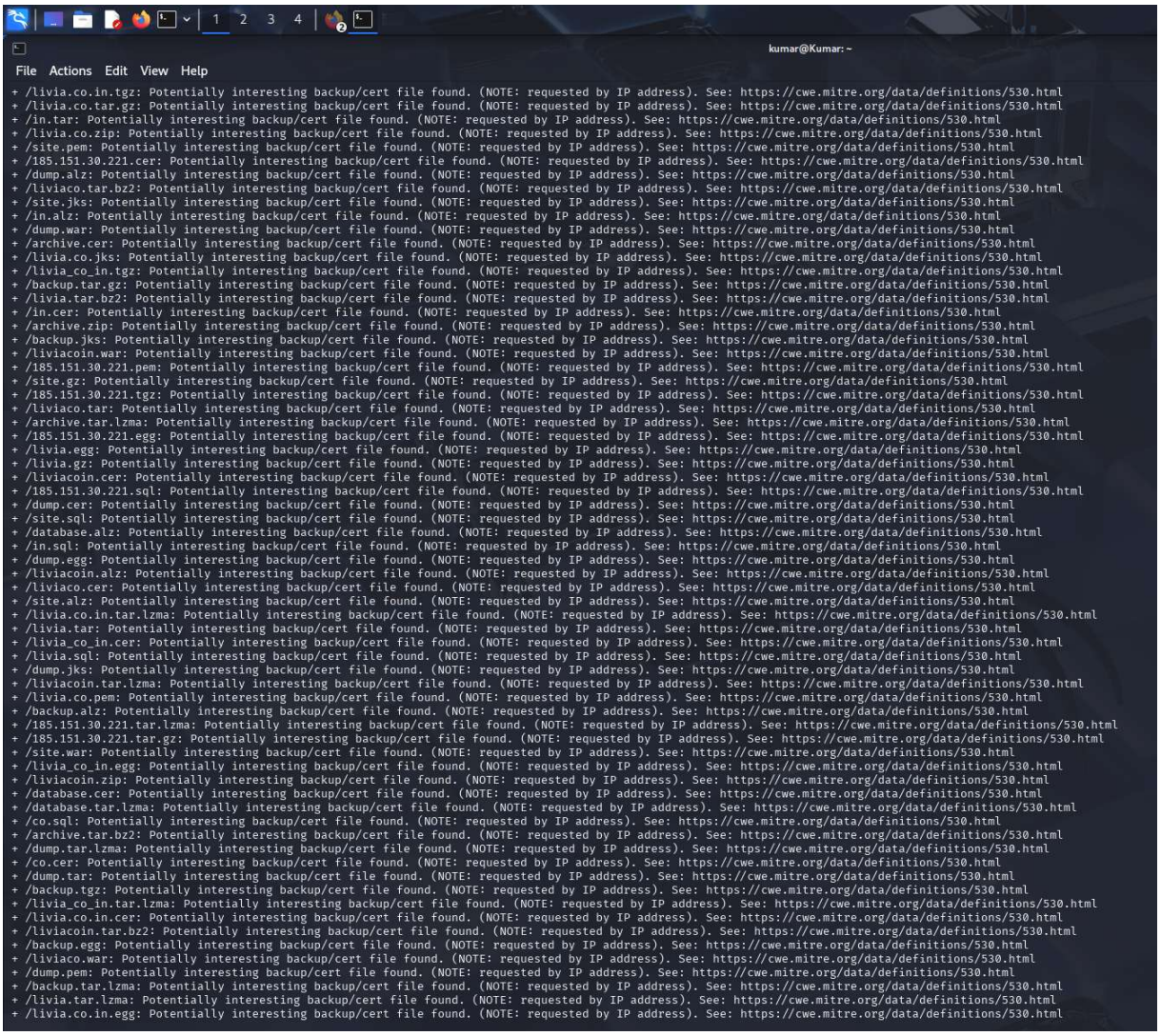
- Checks if the user entered the right number of arguments (target IP/domain).
- If yes:
  - Runs Nmap scan.
  - Runs Nikto scan.
  - Extracts vulnerabilities.
  - Extracts CVEs.
  - Writes final report.
- If no:
  - Shows correct usage and exits safely.

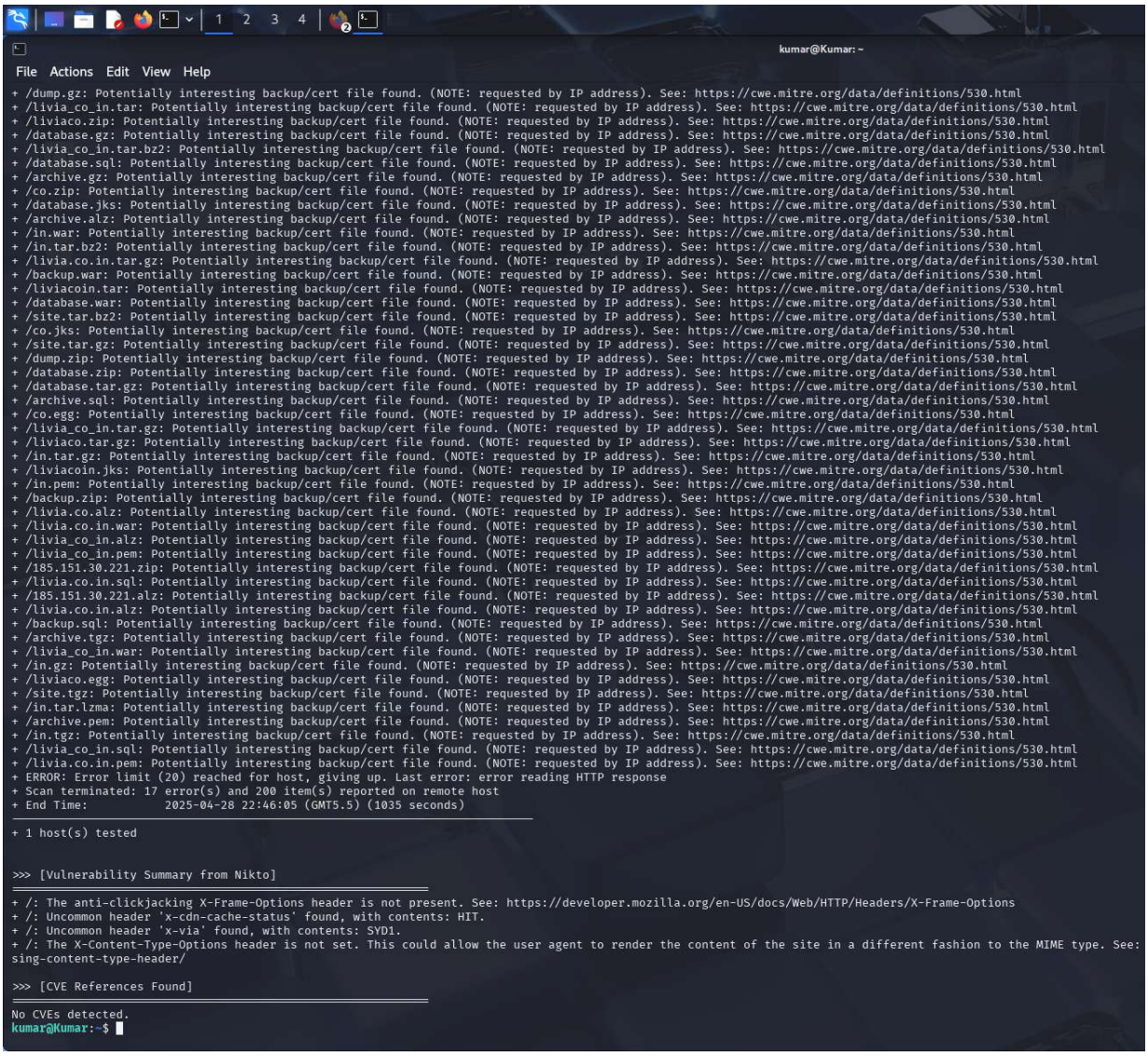
One line on the terminal:

python3 security_audit.py livia.co.in or <target IP or Domain >

SAMPLE OUTPUT




1 2 3 4
kumar@Kumar: ~
File Actions Edit View Help
+ /livia.co.in.tgz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.tar.gz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /in.tar: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.zip: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /site.pem: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.alz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviaco.tar.bz2: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /site.jks: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /in.alz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.war: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.jks: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia_co_in.tgz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.tar.gz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.tar.bz2: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /in.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.zip: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.jks: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviacoin.war: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.pem: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /site.gz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.tgz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviaco.tar: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.egg: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.egg: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.gz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviacoin.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.sql: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /site.sql: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /database.alz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /in.sql: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.egg: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviacoin.alz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviaco.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /site.alz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.in.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.tar: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia_co_in.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.sql: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.jks: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviacoin.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.pem: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.alz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /185.151.30.221.tar.gz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /site.war: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia_co_in.egg: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviacoin.zip: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /database.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /database.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /co.sql: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.tar.bz2: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /co.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.tar: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.tgz: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia_co_in.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.in.cer: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviacoin.tar.bz2: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.egg: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /liviaco.war: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.pem: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.tar.lzma: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html
+ /livia.co.in.egg: Potentially interesting backup/cert file found. (NOTE: requested by IP address). See: https://cwe.mitre.org/data/definitions/530.html



Once our script was ready, we executed a full security assessment scan on the domain **livia.co.in** using our Python automation tool.

The tool performed both **Nmap** and **Nikto** scans and generated a detailed report. Below is an explanation of the key findings based on our real output.

❖ Nmap Full Scan Output (Real Result)

- **Target:** livia.co.in (185.151.30.221)
- **Ports Scanned:** All 65535 ports
- **Discovered Services:**

- Port 80 (HTTP): HAProxy HTTP Proxy 2.0.0
- Port 443 (HTTPS): HAProxy HTTP Proxy 2.0.0

Key Observations:

- The server is running a load balancer (HAProxy) on both HTTP and HTTPS ports.
- No unnecessary open ports were detected beyond the typical web traffic ports (80 and 443).
- HAProxy version 2.0.0 was detected — while this version is stable, newer versions have been released with important security updates.

❖ Nikto Full Scan Output (Real Result)

- Server Banner: No specific server banner was retrieved.
- Major Findings:
  - Missing Security Headers:
    - X-Frame-Options header was not present (important for protecting against clickjacking).
    - X-Content-Type-Options header was missing (helps prevent MIME type sniffing attacks).
  - Uncommon Headers Detected:
    - x-cdn-cache-status
    - x-via
- Backup/Certificate Files Detected:
  - Nikto found multiple potentially sensitive files such as:
    - /livia.co.in.tar.gz
    - /database.sql
    - /backup.tar.gz

        - /archive.zip
    - These could be **backup archives, database dumps, or sensitive certificates**, which should not be publicly accessible.

Important Note:
These findings are based on **predictive scanning** — Nikto suggests that such files might exist if misconfigurations expose them, and recommends manual verification.

❖ Vulnerability Summary (Auto-Extracted)

- Missing important HTTP security headers.
- Backup or certificate files could be present on the server.
- No active CVEs were detected directly in the Nikto scan, but file exposure risks are extremely critical.

❖ CVE Detection

- **Result:**
  No direct CVEs (Common Vulnerabilities and Exposures) were identified by Nikto for the HAProxy 2.0.0 during this scan.
- **Meaning:**
  Although the scan didn't find official CVEs at this time, the server has **configuration issues** (missing headers, potential file exposure) that could still pose serious risks.

VULNERABILITY ANALYSIS

Critical Risks Identified:

1. Missing X-Frame-Options Header:
    - Risk: Without this header, attackers can embed the website inside an iframe and trick users (clickjacking attack).
    - Impact: Users could unknowingly perform actions (like payment authorization) on an attacker's page.
2. Missing X-Content-Type-Options Header:
    - Risk: Browsers could incorrectly interpret file types, allowing malicious content to be executed.
3. Potential Exposure of Sensitive Files:
    - Risk: Backup archives (.tar.gz, .zip, .sql) could contain database credentials, source code, or sensitive internal information.
    - Impact: If these files are accessible, attackers could completely compromise the server and backend systems.

Important Realization

Even if no high CVEs were found immediately, these server misconfigurations alone are enough to allow major security breaches if exploited!

RECOMMENDATIONS FOR REMEDIATION

High Priority Actions:

- Review Server for Backup Files:
  - Immediately audit all public folders for backup files.
  - Remove or restrict access to .tar.gz, .zip, .sql, and .pem files.
- Add Security Headers:
  - Configure the web server to include the following:
    - X-Frame-Options: SAMEORIGIN
    - X-Content-Type-Options: nosniff
    - Strict-Transport-Security: max-age=31536000; includeSubDomains
- Update HAProxy if Needed:
  - While no CVEs were detected now, it's recommended to stay updated to avoid future vulnerabilities.
- Conduct Regular Scans:
  - Implement a schedule for monthly or quarterly automated scans using updated versions of Nmap, Nikto, and other tools.
- Harden Server Configurations:
  - Restrict unnecessary HTTP methods (OPTIONS, TRACE, DELETE).
  - Force HTTPS redirection.
  - Implement firewall rules to restrict access to only necessary services.

CHALLENGES FACED DURING THE PROJECT

While working on this project, we faced several **real-world challenges** that pushed us to think deeper, improve our troubleshooting skills, and learn beyond the classroom theory.
Here's a genuine reflection of the hurdles we encountered during this journey:

1. Handling Inconsistent Tool Outputs

Problem:

Tools like Nmap and Nikto sometimes gave outputs in formats that weren't as clean or predictable as we expected.
Especially with Nikto, different servers produced very different styles of vulnerability listings.

How We Overcame It:

We had to write **more flexible and smart parsing logic** in our Python code — for example, using regular expressions and condition-based filtering rather than relying on fixed patterns.
It taught us that **real-world cybersecurity is messy** — and automation must be robust enough to handle that messiness.

2. Managing Long and Heavy Scans

Problem:

When scanning a server with many open ports or when the server was slow, our script sometimes took **a long time** to complete.
At first, this made the process frustrating and felt like the tool was hanging.

How We Overcame It:

We learned to:

- Use Nmap's-T4 timing template to **speed up** scans without losing accuracy.
- Warn users at the beginning that **deep scans** (especially-p- scanning all ports) **may take time**, depending on server size and network speed.

This experience taught us about the trade-off between thoroughness and speed in cybersecurity work.

3. No Direct CVEs Detected Sometimes

Problem:

In some cases (like your scan of livia.co.in), our tool didn't find direct CVEs even though security issues clearly existed (like missing headers or backup files).
At first, this made us question if our script was working properly.

How We Overcame It:

We realized that not every risk has a CVE assigned — and configuration weaknesses (like missing headers) are equally critical.
We adjusted our tool to still highlight these findings in the report, even without CVEs.

This helped us understand that security isn't just about CVEs — it's about defense in depth.

4. Learning New Tools Independently

Problem:

At the start of this project, some of our team members had never used Nmap, Nikto, or even Linux commands extensively.

How We Overcame It:

We spent time:

- Watching online tutorials,
- Reading official tool documentation,
- Practicing with dummy servers.

This hands-on learning made us much more confident in using cybersecurity tools, far beyond just theory.

FUTURE SCOPE AND IMPROVEMENTS

Although our project was successful, we are excited about how much further it can be taken with more time and resources.

Here are some ideas we brainstormed for the future:

1. SSL/TLS Configuration Analysis

Currently, we only perform basic HTTP/HTTPS scanning.
In the future, we could integrate tools like SSL Labs Scanner or testssl.sh to analyze:

- SSL versions supported
- Weak ciphers
- Certificate expiries
- HTTPS misconfigurations

2. Integration with OWASP ZAP

While Nikto is amazing for basic web server vulnerabilities, it doesn't test for complex web app issues like:

- SQL Injection
- Cross-Site Scripting (XSS)
- Authentication flaws

3. Web-Based Dashboard for Scan Results

Right now, our tool generates a .txt file report.
Imagine if results could be visualized dynamically on a web dashboard with:

- Interactive charts
- Filters (by risk level, CVE, etc.)
- Timeline of scan history

4. Scheduled and Automated Scans

Future versions could allow users to:

- Schedule scans daily, weekly, or monthly,
- Auto-email the reports,
- Raise alerts if new vulnerabilities are detected.

CONCLUSION

Working on this project has been an incredibly **eye-opening and enriching experience** for our team.
What started as an idea — building an automated tool for web server security assessment — turned into a real, working solution that we are proud of.

We didn't just learn about Nmap, Nikto, or Python scripting — we also learned **how attackers think**, **how important small misconfigurations can be**, and **how automation can empower security teams** to work faster and smarter.

Along the way, we faced technical hurdles, parsing challenges, long scan times, and moments of doubt — but overcoming them taught us persistence, real problem-solving, and teamwork.

We hope this project will serve as a **small contribution toward building safer web infrastructures**, and also inspire us to continue exploring the exciting field of **cybersecurity and ethical hacking** in the future.

REFRENCES

- Nmap Official Documentation: https://nmap.org/book/man.html
- Nikto2 Official Documentation: https://cirt.net/Nikto2
- National Vulnerability Database (NVD): https://nvd.nist.gov/
- OWASP Foundation: https://owasp.org/
- MITRE CVE Database: https://cve.mitre.org/
- CWE- Common Weakness Enumeration: https://cwe.mitre.org/