

PROFESSIONAL TRAINING REPORT - II

entitled

WEBSITE SECURITY AUDIT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in cyber security

By

S. SANCHAI MANI [41614082]

R. NAVEEN NITHYA KALYAN [41614061]

RAMESH SRINIVASAN [41614076]

SRI RAM GANESH [41614094]



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Mr. S.SANCHAI MANI[41614082]** who carried out the project entitled "**WEBSITE SECURITY AUDIT**" under my supervision from January 2024 to May 2024.

Internal Guide

Dr. D. Anu Disney, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 7/5/2024

Internal Examiner

External Examiner

DECLARATION

I, **S.SANCHAI MANI [41614082]**, hereby declare that the Professional Training Report-II entitled "**WEBSITE SECURITY AUDIT**" done by me under the guidance of **Dr. D. Anu Disney, M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE: 7/5/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. D. Anu Disney, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



Elite

NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to

S SANCHAI MANI

for successfully completing the course

Cloud Computing and Distributed Systems

with a consolidated score of **69** %

Online Assignments	20.83/25	Proctored Exam	48/75
--------------------	----------	----------------	-------

Total number of candidates certified in this course: **3244**

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Jan-Mar 2024

(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur



Indian Institute of Technology Kanpur



Roll No: NPTEL24CS09S344106621

To verify the certificate



No. of credits recommended: 2 or 3

ABSTRACT

The aim of this project is to develop a website security audit, where cyber-attacks are becoming increasingly frequent and sophisticated, organizations must proactively assess vulnerabilities in their web applications and infrastructure. This research paper presents a comprehensive analysis of the significance of web security audits and penetration testing in bolstering website security to combat the rising tide of cyber threats. Through in-depth security audits and penetration testing, potential weaknesses can be identified and mitigated before they are exploited by malicious actors. The study explores various web security audit techniques, including vulnerability scanning and code review. It also highlights the benefits of penetration testing, such as simulated attacks and vulnerability exploitation. Regular security assessments are emphasized, along with practical recommendations for establishing a robust web security framework. The findings underscore the critical role of web security audits and penetration testing in safeguarding websites, protecting sensitive data, and maintaining the credibility and trustworthiness of online platforms. In this project, we'll develop a Python-based tool to perform a comprehensive security audit of websites. Our goal is to identify vulnerabilities, assess risks, and recommend security enhancements.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	
	1.1 Overview	1
2	LITERATURE SURVEY	
	2.1 Survey	3
3	REQUIREMENTS ANALYSIS	
	3.1 Objective of the Project	5
	3.2 Requirements	7
	3.2.1 Hardware Requirements	7
	3.2.2 Software Requirements	7
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	
	4.1 Proposed Methodology	10
	4.1.1 Ideation Map/System Architecture	11
	4.1.2 Various Stages	12
	4.1.3 Internal or Component Design Structure	13
	4.1.4 Working Principles	13
	4.2 Features	14
	4.2.1 Novelty of the Proposal	15
	4.2.2 Benefits	15
5	CONCLUSION	17
	REFERENCES	18
	APPENDIX	
	A. Research Paper	20
	B. Source Code	27
	C. Output	38

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1.1	Ideation flow	9
4.1.2	Design structure	11
4.1.3	Working Model	11
C.1	Security Headers	38
C.2	Start Vulnerability Auditing	38
C.3	Vulnerability Auditing	39
C.4	Vulnerability Auditing	39
C.5	Vulnerability Auditing	40
C.6	Summary of Website Auditing	40

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A website security audit involves a systematic approach to identifying vulnerabilities and security loopholes. It involves a full overview of the site's core files, server, and plugins, all of which are examined for both existing and potential problems. During this process, the website in question is scanned for common weaknesses, with the entire infrastructure examined in detail. They strive to detect a wide range of vulnerabilities and infections and call for a similarly wide range of strategies to reveal where and how your website may be at risk. We've created a detailed checklist that outlines this risk assessment process, so you can feel more confident in what, exactly, you're about to undergo with this project. The website security audit project seeks to comprehensively assess the security posture of an organization's online platform. Through a meticulous examination of the website's infrastructure, codebase, and configurations, the audit aims to identify and mitigate potential vulnerabilities and risks that could compromise the confidentiality, integrity, and availability of data. By conducting rigorous testing, including vulnerability scanning, penetration testing, and code review, the project aims to provide actionable insights and recommendations to bolster the website's defenses against cyber threats. Ultimately, Website security encompasses comprehensive measures and best practices to protect websites from unauthorized access, data breaches, and cyberattacks. Strong website security measures ensure that sensitive data, such as personal customer details and financial information, remain encrypted and inaccessible to malicious actors. This fosters trust, assuring users that their information is safe on your website. A secure website also upholds a brand's reputation by demonstrating a commitment to user privacy and protection. the goal is to enhance the overall security resilience of the website, safeguarding sensitive information and ensuring a trustworthy online experience for user

PROBLEM STATEMENT:

websites are increasingly targeted by cyber threats ranging from data breaches to malware injections, jeopardizing the integrity and confidentiality of sensitive information. Despite the growing awareness of cybersecurity risks, many organizations lack a comprehensive understanding of their website's vulnerabilities and are thus ill-prepared to mitigate potential threats effectively. This project aims to address this gap by conducting a thorough website security audit to identify and remediate vulnerabilities that could expose the organization to cyberattacks. By systematically assessing the website's infrastructure, codebase, and configurations, we aim to provide actionable recommendations to enhance its security posture and safeguard against potential threats, thereby ensuring the confidentiality, integrity, and availability of critical data.

INTRODUCTION TO WEBSITE SECURITY AUDIT:

A Website Security Audit is a systematic evaluation process designed to identify vulnerabilities and weaknesses within a website's infrastructure, codebase, and configurations. This introduction will provide a foundational understanding of Website Security Audits, outlining their importance, key components, methodologies, and the role they play in maintaining robust cybersecurity measures. website security audit cyberwar

The importance of Website Security Audits cannot be overstated. With cyber attacks becoming increasingly sophisticated and prevalent, businesses and individuals alike are at risk of data breaches, financial losses, and reputational damage. By proactively conducting security audits, organizations can detect vulnerabilities before they are exploited by malicious actors, thereby reducing the likelihood of security incidents and their associated consequences. Moreover, security audits are often required to comply with industry regulations and standards, demonstrating a commitment to data protection and privacy.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

A literature review or survey of a website security audit developed using Python encompasses an exploration of existing research, projects, and tools in the domain of website scanning and websecurity analysis. Currently available tools website security audit and more. Libraries that are used in website security audit scanning generally. Parallel scanning, Asynchronous Programming are used to minimize the use of network resources. website security audit is a network reconnaissance technique used to discover open, closed, or filtered ports on a target computer or network device. It involves sending website requests to various website to determine which are active and responding to the requests. website security audit serves several important purposes in the field of computer networking and security. injection attack system such web security audit role Website auditing is a fundamental practice in cybersecurity, plays a pivotal role in identifying potential vulnerabilities within networked systems. It involves the systematic exploration of network devices to uncover vulnerability, which serve as entry points for communication between devices. The significance of Website auditing lies in its ability to uncover weaknesses in network configurations, thereby enabling proactive measures to strengthen cybersecurity defenses. As the digital landscape continues to evolve, understanding the intricacies of Website auditing becomes increasingly critical for both offensive and defensive cybersecurity strategies. A web security audit using Linux typically involves assessing the security posture of a web application or website hosted on a Linux server. This involves identifying all the components of the web application, including web servers, databases, APIs, etc. Tools like Nmap, Nikto, and Dirb can be used for this purpose. Once the components are identified, vulnerability scanning tools such as OpenVAS or Nessus can be used to scan for known vulnerabilities in the software versions running on the server. Automated tools can miss certain vulnerabilities, so manual testing is also crucial. This involves inspecting the source code, testing input fields for vulnerabilities like SQL injection and XSS (Cross-Site Scripting), and assessing the security of server configurations. Testing the strength of user authentication mechanisms and ensuring that users can only access resources they're authorized to access. Ensuring that sensitive data is properly encrypted both in transit and at rest, and

assessing the effectiveness of access controls to prevent unauthorized access to sensitive data. Checking how the web application manages user sessions to prevent session hijacking or fixation attacks. Verifying that the server and web application are configured securely, including settings related to SSL/TLS, HTTP headers, and file permissions. Ensuring that proper logging mechanisms are in place to record security-related events, and setting up monitoring systems to detect and respond to security incidents in real-time. Finally, compiling the findings of the audit into a comprehensive report that outlines the vulnerabilities discovered, their potential impact, and recommendations for remediation.

review would begin with an introduction to the importance of web security audits in the context of the increasing prevalence of cyber threats and the critical role of web applications in today's digital landscape. It might delve into the historical evolution of web security auditing, highlighting key milestones, influential research papers, and notable incidents that have shaped the field. The literature review would cover various methodologies and techniques used in web security audits, ranging from automated vulnerability scanning tools to manual penetration testing approaches. It would discuss the strengths and limitations of each method and highlight emerging trends in the field. It would include case studies highlighting real-world examples of successful web security audits, as well as best practices for conducting audits, interpreting results, and implementing remediation measures.

literature review serves to provide a comprehensive overview of the state-of-the-art in web security auditing, synthesizing existing knowledge and identifying opportunities for further advancement in the field.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

The objective of this project is to create a website security audit using python. Here we are giving website as an input and the tool is going to find all the Open Ports for that particular website. This tool will be easy to use and faster with low false positives.

- **Identify web security:** Determine which website are open and accessible on networked devices. This provides insight into the attack surface and potential entry points for attackers.
- **Detect Services and Applications:** Determine the services and applications running on the open ports. Understanding the services and versions helps assess the security posture and potential vulnerabilities of the target systems.
- **Assess Vulnerabilities:** Identify security vulnerabilities and weaknesses present in the services and applications discovered during website scanning. Vulnerability scanning helps evaluate the risk posed by these vulnerabilities and prioritize remediation efforts.
- **Improve Security Posture:** Enhance the overall security posture of the network by proactively identifying and mitigating security vulnerabilities before they can be exploited by attackers. By conducting regular scanning activities, organizations can reduce the likelihood of successful cyber-attacks and data breaches.
- **Promote Security Awareness:** Raise awareness among stakeholders about the importance of network security and the role of website scanning and vulnerability scanning in maintaining a secure environment. Educating users and staff about security best practices helps reinforce a culture of cybersecurity awareness within the organization.
- **Enhance Risk Management:** Support risk management efforts by identifying and quantifying security risks associated with open website and vulnerabilities. This

enables organizations to make informed decisions about risk mitigation strategies and resource allocation.

- Support Incident Response: Provide valuable information for incident response and security incident investigations. website scanning and vulnerability scanning data can help security teams identify Indicators of Compromise (IOCs) and understand the scope and impact of security incidents.

3.2 REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

The hardware requirements of a computer refer to the necessary physical components needed for it to function optimally. These components typically include a central processing unit CPU which acts as the brain of the computer, managing tasks and executing instructions; random-access memory RAM, which provides temporary storage for data and programs currently in use by the CPU; a storage device, such as a hard disk drive HDD or solid-state drive SSD, for long-term storage of data and programs; a motherboard, which houses and connects the various components of the computer, providing communication between them; a power supply unit PSU, which supplies electrical power to the computer's components; a graphics processing unit GPU, responsible for rendering graphics and images on the computer's display; and various input/output I/O devices, including a keyboard, mouse, monitor, and other peripherals for user interaction. The specific hardware requirements of a computer can vary depending on factors such as the intended use case gaming, multimedia editing, business productivity, the operating system being used, and budget constraints. Generally, a balance must be struck between the capabilities of the hardware components and the demands of the tasks the computer will be performing to ensure smooth and efficient operation.

3.2.2 SOFTWARE REQUIREMENTS

LINUX

Linux is a widely-used open-source operating system kernel that forms the foundation for various Linux distributions, such as Ubuntu, Fedora, and Debian. Developed by Linus Torvalds in the early 1990s, Linux is known for its stability, security, and flexibility. It operates on a Unix-like architecture, offering multi-user capabilities, networking support, and a robust filesystem. One of its key features is its modularity, allowing users to customize and tailor their systems to their specific needs. Linux has gained popularity not only in traditional computing environments like servers and desktops but also in embedded systems, smartphones, and supercomputers due to its versatility and adaptability. Additionally, its open-source nature fosters a vibrant community of developers who contribute to its ongoing development and support.

PYTHON

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python has gained widespread popularity across various domains, including web development, data science, artificial intelligence, and automation. Its syntax emphasizes code readability and expressiveness, making it an ideal language for beginners and experienced developers alike. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, offering developers the flexibility to choose the approach that best suits their needs. One of Python's strengths lies in its extensive standard library, which provides a rich set of modules and tools for tasks ranging from file I/O to network programming. Moreover, Python's active community continually develops and maintains a vast ecosystem of third-party libraries and frameworks, further expanding its capabilities. With its user-friendly syntax, powerful features, and strong community support, Python has become one of the most widely-used programming languages for building a diverse range of applications.

VIRTUAL MACHINE

A virtual machine VM is a software-based emulation of a physical computer that operates within another computer's environment. It enables the creation and execution of multiple independent computing instances, each with its own operating system OS and applications, on a single physical machine. Virtualization technology, which underpins VMs, allows for the efficient sharing of hardware resources such as CPU, memory, storage, and network interfaces among multiple VMs. This enables greater flexibility, scalability, and resource utilization in computing environments. VMs are commonly used for various purposes, including server consolidation, software testing and development, application sandboxing, and legacy application support. They provide isolation between VMs, enhancing security and stability by containing potential issues within individual instances. Additionally, VMs can be easily migrated or replicated across different physical hosts, facilitating workload management, disaster recovery, and high availability strategies. Popular virtualization platforms such as VMware, Microsoft Hyper-V, and open-source solutions like KVM and VirtualBox offer tools and features for creating, managing, and deploying virtual machines efficiently.

GITHUB

GitHub is a web-based platform and service that facilitates collaboration among developers for software development projects. It offers a distributed version control system based on Git, a popular open-source version control system created by Linus Torvalds. GitHub provides developers with tools for hosting, managing, and sharing code repositories, enabling seamless collaboration across teams and individuals. Users can create repositories to store their code, track changes made by collaborators, and manage project workflows using features such as issue tracking, pull requests, and project boards. GitHub also offers social networking features, allowing users to follow other developers, explore open-source projects, and contribute to community-driven initiatives. It serves as a hub for software development communities, fostering knowledge sharing, code reuse, and innovation. Moreover, GitHub provides integration with various development tools

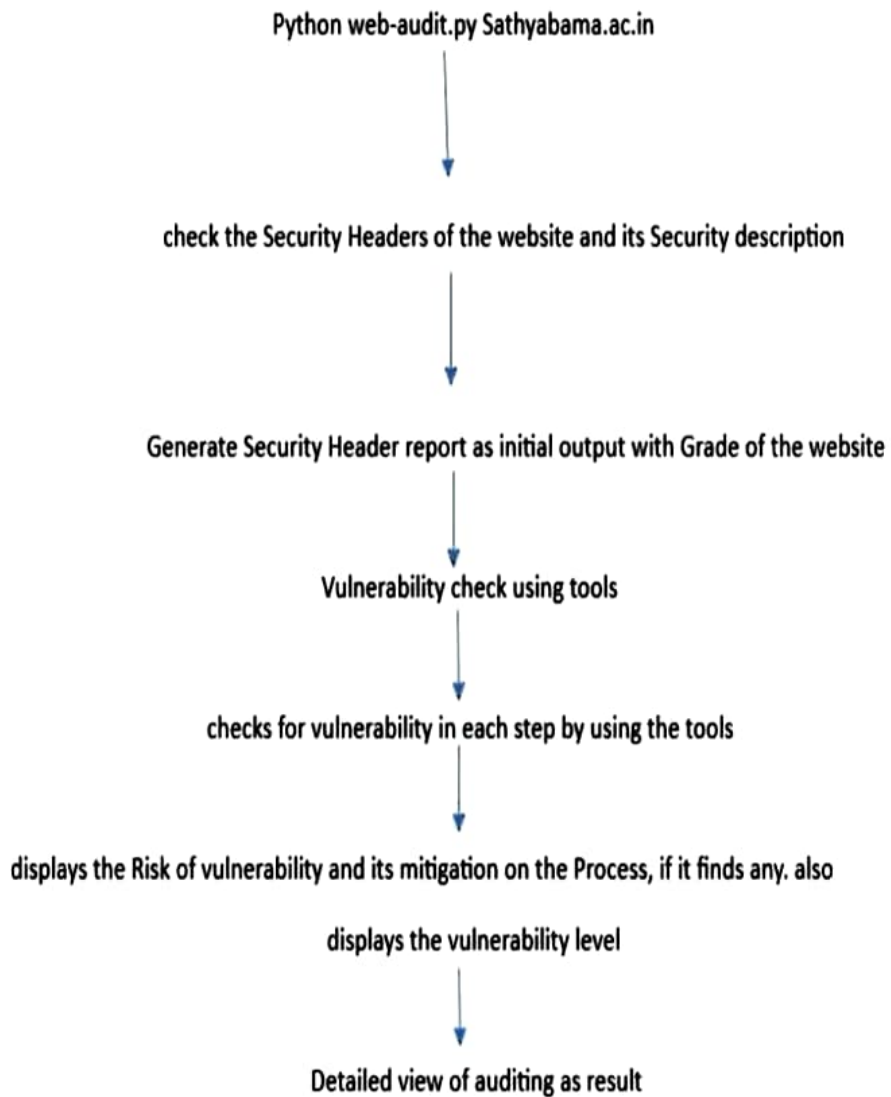
CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

The Proposed methodology for developing a website security audit tool involves a systematic approach to ensure its effectiveness and efficiency. First, we will understand the projects goals and requirements. That includes defining user input, customization options, and expected results. Now we will be choosing the programming languages, libraries and framework based on the project requirements and development. As it is python-based so here we will create a user-friendly command-line interface using a suitable python framework. we define our commands, parameters, and options. Decide on the overall architecture of the website . Create data structure to store scan results and relevant information. Implement the core scanning algorithm based on user inputs, and their specific methods. Design the reporting mechanism to present scan results to the user. Decide the format of the output we can easily save the output. If the user wants to stop the auditing, then it can be easily done by pressing control + x. We made sure that your tool complies with legal and ethical standards for network scanning and security assessments. Even this tool with various operating system like operating Windows, Linux, Mac OS to ensure cross-platform. By following this methodology, you can develop website security audit using python tool that is user-friendly, efficient, and adheres to ethical and legal standards. Maintain a strong focus on user needs to ensure the tools long term success.

4.1.1 Ideation Map/System Architecture



4.1.1 Ideation flow

4.1.2 VARIOUS STAGES

- 1)** Firstly we have written a code in python using code editor that will perform website security audit.
- 2)** Secondly we have to get a target website to perform the by entering website securityaudit it into the tool.
- 3)** We have given a loop for checking ports on the target website one by one for checkingif its open or not.
- 4)** To avoid the program from getting crashed when the website is error we have to givethat website is not responding in the given time then the website is closed and pass it and check other website.
- 5)** As soon as the website security audit is complete all the website security audit are displayed.
- 6)** Check the vulnerability in each step uing tools like nmap,dnsrecon, wafwoof,uniscan etc.
- 7)** Display the risk of vulnerability and its mitigation of the process it find and display thevulnerability level.

4.1.3 Internal or Component design structure



Fig:4.1.3 Design Structure

4.1.4 Working Model

```

[~]vassant@sathiyabana: ~/project
$ cat security-header.txt

Domain: https://www.sathiyabana.ac.in

Used security headers:
1 Content-Type-Options: Prevents MIME type sniffing.
2 Frame-Options: Protects against clickjacking.

Missing security headers:
1 Strict-Transport-Security: Enforces secure (HTTP over SSL/TLS) connections to the server.
2 Cross-Site-Scripting-Protection: Helps prevent cross-site scripting (XSS) and other code injection attacks.
3 X-XSS-Protection: Protects against XSS attacks.
4 Referrer-Policy: Controls how much referrer information should be included with requests.
5 Content-Security-Policy: Controls which browser features can be used by the site.

The Grade of website is : F
-----
```

Fig:4.1.4 Working Model

4.2 FEATURES

TOOLS FEATURE

Nmap is a powerful network scanning tool used for discovering hosts and services on a computer network, thus creating a "map" of the network.

dnsrecon is a DNS reconnaissance tool that can discover DNS servers, perform zone transfers, enumerate subdomains, gather DNS records, and more.

WAFW00F is a tool used to identify and fingerprint Web Application Firewalls (WAFs) by analyzing HTTP responses.

Uniscan is a web vulnerability scanner that can detect various vulnerabilities in web applications, including SQL injection, XSS, LFI, RFI, etc.

SSLyze is a Python tool used to analyze SSL/TLS configurations of servers. It can detect vulnerabilities, misconfigurations, and provide detailed information about SSL/TLS certificates.

LBD is a tool used to detect and fingerprint load balancers by sending requests to target servers and analyzing their responses.

Nikto is a web server vulnerability scanner that checks for known vulnerabilities, misconfigurations, and common security issues in web servers.

Dnswalk is a DNS debugger tool used to perform zone transfers, DNS lookups, and verify DNS records.

Plecost is a WordPress vulnerability scanner and fingerprinting tool that analyzeWordPress installations for vulnerabilities and misconfigurations.

4.2.1 Novelty of the proposal

We have import threading in python is the implementation of a multithreaded website security auditing tool that allows user to scan either a specific website or all website.

Multithreaded website security auditing:- Our tool leverages python threading module to perform website security auditing on multiple ports concurrently.

User-Friendly:- The tool takes user input to determine the target and the scanning mode. It prompts the user to enter the target website security auditing and whether they want to scan a specific website or all website.

Flexible website Scanning:- Depending on the users choice the tool either scans a specific website.

Error Handling:- Our tool includes error handling by setting a timeout for the website security auditing cannot be established.

Thread Management:- The tool creates and manages threads dynamically based on the scanning mode chosen by the user. This enables efficient resource utilization and allow the tool to take full advantage of the available CPU cores.

4.2.2 Benefits

Website Security Assessment: website scanning is a fundamental technique for assessing the security posture of a website. By scanning for open website, administrators can identify potential vulnerabilities and weaknesses in their systems and take proactive measures to address them before they are exploited by attackers.

Risk Mitigation: Identifying website security through scanning helps organizations understand their exposure to potential threats. By knowing which website are open and potentially vulnerable, security teams can prioritize security measures and allocate resources effectively to mitigate risks.

Troubleshooting and Debugging: websites scanning can be used as a diagnostic tool to troubleshoot network connectivity issues and identify misconfigurations or service disruptions. By scanning website for administrators can pinpoint the source of problems and implement appropriate solutions.

CHAPTER 5

CONCLUSION

In conclusion website security audit provide a brief overview of the vulnerabilities and security issues identified during the audit, categorized by severity and potential impact. Highlight any critical or high-risk vulnerabilities that require immediate attention

Evaluate the overall security posture of the website based on the findings of the audit. Discuss strengths and weaknesses in the website's security controls, infrastructure, configurations, and codebase

Discuss the potential impact of the identified vulnerabilities on the organization's operations, reputation, and compliance obligations. Emphasize the importance of addressing security issues to mitigate risks and safeguard sensitive

DataProvide actionable recommendations for remediation to address the identified vulnerabilities and enhance the website's security posture.

Website Security Audits are essential for proactively identifying and mitigating security risks, protecting valuable assets, and maintaining regulatory compliance. By conducting regular audits and implementing recommended security measures, organizations can enhance their cybersecurity posture and build trust with their users and stakeholders.

REFERENCES

1. Nmap is a powerful network scanning tool used for discovering hosts and services on a computer network, thus creating a "map" of the network. [Nmap Official Website](<https://nmap.org/>).
2. dnsrecon is a DNS reconnaissance tool that can discover DNS servers, perform zone transfers, enumerate subdomains, gather DNS records, and more. [dnsrecon GitHub Repository](<https://github.com/darkoperator/dnsrecon>)
3. WAFW00F is a tool used to identify and fingerprint Web Application Firewalls (WAFs) by analyzing HTTP responses. [WAFW00F GitHub Repository](<https://github.com/EnableSecurity/wafw00f>)
4. Uniscan is a web vulnerability scanner that can detect various vulnerabilities in web applications, including SQL injection, XSS, LFI, RFI, etc. [Uniscan GitHub Repository](<https://github.com/poedumb/uniscan>)
5. SSLyze is a Python tool used to analyze SSL/TLS configurations of servers. It can detect vulnerabilities, misconfigurations, and provide detailed information about SSL/TLS certificates.
Reference: [SSLyze GitHub Repository](<https://github.com/nabla-c0d3/sslyze>)
6. Fierce is a DNS reconnaissance tool used to discover non-contiguous IP space, DNS servers, and perform DNS enumeration of subdomains. [Fierce GitHub Repository](<https://github.com/mschwager/fierce>)
7. LBD is a tool used to detect and fingerprint load balancers by sending requests to target servers and analyzing their responses.
Reference: [LBD GitHub Repository](<https://github.com/stamparm/ip-tool>)
8. TheHarvester is a tool for gathering information about email addresses, subdomains, virtual hosts, and open ports from public sources.

[TheHarvester GitHub Repository](https://github.com/laramies/theHarvester)

9. Nikto is a web server vulnerability scanner that checks for known vulnerabilities, misconfigurations, and common security issues in web servers.

Reference: [Nikto GitHub Repository](https://github.com/sullo/nikto)

10. Dnswalk is a DNS debugger tool used to perform zone transfers, DNS lookups, and verify DNS records.

[Dnswalk GitHub Repository](https://github.com/davepeck/dnswalk)

11. Davtest is a tool used to test WebDAV-enabled servers for vulnerabilities and misconfigurations. [Davtest GitHub Repository](https://github.com/cldrn/davtest)

12. Dirb is a web content scanner and directory brute-forcing tool that discovers hidden files and directories on web servers.

[Dirb GitHub Repository](https://github.com/v0re/dirb)

13. Wapiti is a web application vulnerability scanner that can detect various security vulnerabilities in web applications, including SQL injection, XSS, LFI, RFI, etc.

[Wapiti GitHub Repository](https://github.com/IFGHou/Wapiti)

14. XSSer is an automated tool for finding and exploiting XSS vulnerabilities in web applications.

[XSSer GitHub Repository](https://github.com/epsylon/xsser)

15. Dmitry is an information-gathering tool that retrieves detailed information about hosts, emails, subdomains, and more.

[Dmitry GitHub Repository](https://github.com/jaygreig86/dmitry)

APPENDIX

A.Research Paper

Website Security Audit

Naveen Nithya Kalyan,
Ramesh Srinivasan,
Sanchai Mani,
Sri Ram Ganesh
Department of Computer Science and Engineering,
Sathyabama Institute of Science and Technology,
Chennai, Tamil Nadu, India

Abstract

A website security audit is a comprehensive evaluation of a website's entire ecosystem aimed at uncovering vulnerabilities susceptible to attacker exploitation. This in-depth examination includes critical components like the core system, extensions, server environment, Secure Sockets Layer (SSL) connections, themes, and plugins¹. Essentially, it involves scanning your website and its server for existing or potential weaknesses that hackers can exploit. The audit covers your website's entire infrastructure, from its core software to extensions, themes, server settings, SSL connections, and configurations. Once all loopholes and gaps have been identified, the next step is to conduct penetration tests (pentests). In these tests, security teams simulate hacking attacks against your application, targeting the vulnerabilities detected earlier to assess the associated risks¹. The purpose of website security audits is to proactively look

for discrepancies in your website's architecture and eliminate them before malicious hackers can exploit them. Regular security auditing is crucial because hackers constantly challenge website safety using various techniques, and relying solely on basic practices is not sufficient¹. So, stay vigilant and keep your website secure!

1.Introduction

In today's digital landscape, where websites serve as the storefronts for businesses and gateways for countless interactions, ensuring their security is paramount. A website security audit is a comprehensive examination of a website's infrastructure, codebase, and configurations to identify vulnerabilities, weaknesses, and potential threats that could compromise its integrity, confidentiality, and availability. The importance of website security cannot be overstated, as cyber threats continue to evolve in sophistication and frequency. From data breaches to malware infections

and denial-of-service attacks, the consequences of a compromised website can be severe, ranging from financial loss to damage to reputation and loss of customer trust.

2. A website security audit typically involves:

- **Assessment of Infrastructure:** This includes examining the hosting environment, server configurations, network architecture, and other foundational components to identify potential entry points for attackers.

- **Code Review:** Analyzing the website's source code for security vulnerabilities such as injection flaws (e.g., SQL injection, XSS), insecure authentication mechanisms, and improper error handling.

- **Configuration Review:** Evaluating the configuration settings of web servers, databases, content management systems (CMS), and other software components to ensure they adhere to security best practices and are hardened against common attacks.

- **Risk Assessment:** Identifying and prioritizing security risks based on their likelihood and potential impact, allowing organizations to allocate resources effectively to mitigate the most critical threats.

- **Compliance Check:** Ensuring that the

website complies with relevant security standards and regulations, such as the Payment Card Industry Data Security Standard (PCI DSS), General Data Protection Regulation (GDPR), and others applicable to its industry and geographical location.

- **Security Testing:** Conducting vulnerability scanning, penetration testing, and other security assessments to simulate real-world attack scenarios and validate the effectiveness of existing security controls.

- **Documentation and Reporting:** Documenting findings, recommendations, and remediation steps in a comprehensive report that stakeholders can use to understand the current state of website security and prioritize mitigation efforts.

By conducting regular website security audits, organizations can proactively identify and address security weaknesses before they can be exploited by malicious actors. This not only helps protect sensitive data and assets but also fosters trust among customers, partners, and other stakeholders who rely on the website for various purposes.

In conclusion, a website security audit is an essential practice for any organization that values the security and integrity of its online presence. By investing in proactive security measures and staying vigilant against

emerging threats, organizations can minimize the risk of cyber incidents and maintain a strong defense posture in an increasingly hostile digital environment.

3.Related Works

Website security audit-related works encompass a wide range of tasks and responsibilities aimed at assessing, improving, and maintaining the security posture of a website. Here are some common activities and roles associated with website security audits:

- Security Analyst/Consultant:** Security analysts or consultants specialize in evaluating the security of websites and providing recommendations for improvement. They conduct comprehensive audits, identify vulnerabilities, and offer guidance on mitigating risks.

- Penetration Testing:** Penetration testers, also known as ethical hackers, simulate real-world cyberattacks to identify weaknesses in a website's defenses. They may perform various techniques, such as network penetration testing, web application testing, and social engineering assessments.

- Vulnerability Scanning:** Automated vulnerability scanning tools are used to scan websites for known security vulnerabilities, misconfigurations, and outdated software

versions. Security professionals analyze the scan results to prioritize and address identified issues.

- Code Review:** Software developers or security specialists review the source code of web applications to identify security flaws, such as injection vulnerabilities, insecure authentication mechanisms, and inadequate input validation.

- Compliance Auditing:** Compliance auditors ensure that websites adhere to industry standards and regulatory requirements, such as PCI DSS, GDPR, HIPAA, and others. They assess whether the website's security controls align with the specified compliance criteria.

- Risk Assessment:** Risk assessors evaluate the likelihood and potential impact of security threats and vulnerabilities to determine the level of risk associated with the website. This helps organizations prioritize security measures and allocate resources effectively.

- Security Awareness Training:** Educating website administrators, developers, and other stakeholders about security best practices and common threats is crucial for maintaining a secure environment. Security awareness training programs help raise awareness and promote a security-conscious culture within organizations.

- Incident Response Planning: Developing and implementing incident response plans prepares organizations to effectively respond to security incidents and minimize their impact. This includes procedures for detecting, containing, and mitigating security breaches on websites.

- Security Policy Development: Creating and enforcing security policies and procedures establishes guidelines for maintaining the security of websites. These policies cover areas such as access control, data protection, incident reporting, and security incident response.

- Continuous Monitoring: Implementing tools and processes for continuous monitoring allows organizations to detect and respond to security threats in real-time. This includes monitoring website traffic, logs, and security alerts for suspicious activities.

Overall, website security audit-related works involve a combination of technical assessments, compliance evaluations, risk management activities, and security awareness initiatives to safeguard websites against cyber threats and ensure their resilience in the face of evolving security challenges.

4.Requirements

Linux tools :

nmap, dnsrecon, wafw00f, uniscan, sslyze, fierce, lbd, theharvester, amass, nikto, dnswalk, davtest, dirb, wapiti, xsser, dmitry, dnsenum, dnsmap, golismero, wpscan, plecost

Python tools :

sys, argparse, subprocess, os, time, random, threading, re, random, urllib.parse, asyncio, aiohttp, termcolor.

5. PROPOSED METHODOLOGY

The Proposed methodology for developing a website security audit tool involves a systematic approach to ensure its effectiveness and efficiency. First, we will understand the projects goals and requirements. That includes defining user input, customization options, and expected results. Now we will be choosing the programming languages, libraries and framework based on the project requirements and development. As it is python-based so here we will create a user-friendly command-line interface using a suitable python framework. we define our commands, parameters, and options. Decide on the overall architecture of the website . Create data structure to store scan results and relevant information. Implement the core scanning algorithm based on user

inputs, and their specific methods. Design the reporting mechanism to present scan results to the user. Decide the format of the output we can easily save the output. If the user wants to stop the auditing, then it can be easily done by pressing control + x. We made sure that your tool complies with legal and ethical standards for network scanning and security assessments. Even this tool with various operating system like (Windows, Linux, Mac OS) to ensure cross-platform. By following this methodology, you can develop website security audit using python tool that is user-friendly, efficient, and adheres to ethical and legal standards. Maintain a strong focus on user needs to ensure the tools long term success.

5.1 Various Stages

- Firstly, we have written a code in python using code editor that will perform website security audit.
- Secondly, we have to get a target website to perform the by entering website security audit it into the tool.
- We have given a loop for checking ports on the target website one by one for checking if its open or not.
- To avoid the program from getting crashed when the website is error we have to give that website is not responding in the given time then the website is closed and pass it

and check other website.

- As soon as the website security audit is complete all the website security audit are displayed

6. Conclusion

- In conclusion website security audit Provide a brief overview of the vulnerabilities and security issues identified during the audit, categorized by severity and potential impact. Highlight any critical or high-risk vulnerabilities that require immediate attention

- Evaluate the overall security posture of the website based on the findings of the audit. Discuss strengths and weaknesses in the website's security controls and the infrastructure, configurations, and codebase.

- Discuss the potential impact of the identified vulnerabilities on the organization's operations, reputation, and compliance obligations. Emphasize the importance of addressing security issues to mitigate risks and safeguard sensitive

- Data Provide actionable recommendations for remediation to address the identified vulnerabilities and enhance the website's security posture.

Reference

1. Nmap is a powerful network scanning tool

used for discovering hosts and services on a computer network, thus creating a "map" of the network.[Nmap Official Website](<https://nmap.org/>)

2.dnsrecon is a DNS reconnaissance tool that can discover DNS servers, perform zone transfers, enumerate subdomains, gather DNS records, and more.[dnsreconGitHubRepository](<https://github.com/darkoperator/dnsrecon>)

3.WAFW00F is a tool used to identify and fingerprint Web Application Firewalls (WAFs) by analyzing HTTP responses.[WAFW00F GitHub Repository](<https://github.com/EnableSecurity/wafw00f>)

4.Uniscan is a web vulnerability scanner that can detect various vulnerabilities in web applications, including SQL injection, XSS, LFI, RFI, etc.[Uniscan GitHub Repository](<https://github.com/poedumb/uniscan>)

5.SSLyze is a Python tool used to analyze SSL/TLS configurations of servers. It can detect vulnerabilities, misconfigurations, and provide detailed information about SSL/TLS certificates.Reference: [SSLyze GitHub Repository](<https://github.com/nabla-c0d3/sslyze>)

6.Fierce is a DNS reconnaissance tool used to discover non-contiguous IP space, DNS servers, and perform DNS enumeration of subdomains.[Fierce GitHub Repository](<https://github.com/mschwager/fierce>)

7.LBD is a tool used to detect and fingerprint load balancers by sending requests to target servers and analyzing their responses.Reference:[LBD GitHub Repository](<https://github.com/stamparm/ip-tool>)

8.TheHarvester is a tool for gathering information about email addresses, subdomains, virtual hosts, and open ports from public sources.[TheHarvester GitHub Repository](<https://github.com/laramies/theHarvester>)

9.Amass is a DNS enumeration tool that discovers subdomains by scraping data sources, performing brute force subdomain enumeration, and more.Reference: [Amass GitHub Repository](<https://github.com/OWASP/Amass>)

10.Nikto is a web server vulnerability scanner that checks for known vulnerabilities, misconfigurations, and common security issues in web

servers.Reference: [Nikto GitHub Repository](<https://github.com/sullo/nikto>)

11.Dnswalk is a DNS debugger tool used to perform zone transfers, DNS lookups, and verify DNS records.[Dnswalk GitHub Repository](<https://github.com/davepeck/dnswalk>)

12.Davtest is a tool used to test WebDAV-enabled servers for vulnerabilities and misconfigurations.[Davtest GitHub Repository](<https://github.com/cldrn/davtest>)

13.Dirb is a web content scanner and directory brute-forcing tool that discovers hidden files and directories on web servers.[DirbGitHub Repository](<https://github.com/v0re/dirb>)

14.Wapiti is a web application vulnerability scanner that can detect various security vulnerabilities in web applications, including SQL injection, XSS, LFI, RFI, etc.[Wapiti GitHub Repository](<https://github.com/IFGHou/Wapiti>)

15.XSSer is an automated tool for finding and exploiting XSS vulnerabilities in web applications.[XSSer GitHub Repository](<https://github.com/epsylon/xsser>)

16.Dmitry is an information-gathering tool

that retrieves detailed information about hosts, emails, subdomains, and more.[Dmitry GitHub Repository](<https://github.com/jaygreig86/dmitry>)

17. Dnsenum is a DNS enumeration tool used to gather information about DNS servers, subdomains, and zone transfers.[Dnsenum GitHub Repository](<https://github.com/fwaeytens/dnsenum>)

18.Dnsmap is a passive DNS network mapper that performs zone transfers and gathers DNS information from domain names.[Dnsmap GitHub Repository](<https://github.com/makefu/dnsmap>)

19.Golismero is a web application and network security scanner that performs vulnerability scanning, information gathering, and reporting.[Golismo GitHub Repository](<https://github.com/golismero/golismero>)

20.Wpscan is a WordPress vulnerability scanner that checks for known vulnerabilities, misconfigurations, and security issues in WordPress installations.Reference:GitHubRepository](<https://github.com/wpscanteam/wpscan>)

B.Source Code

1.security_header.py

```
import asyncio import aiohttp
from termcolor import colored
from urllib.parse import urlparse

SECURITY_HEADERS = {
    "Strict-Transport-Security": "Enforces secure (HTTP over SSL/TLS) connections to the server.",
    "Content-Security-Policy": "Helps prevent cross-site scripting (XSS) and other code injection attacks.",
    "X-Content-Type-Options": "Prevents MIME type sniffing.", "X-Frame-Options": "Protects against clickjacking.",
    "X-XSS-Protection": "Protects against XSS attacks.",
    "Referrer-Policy": "Controls how much referrer information should be included with requests.",
    "Permissions-Policy": "Controls which browser features can be used by the site.",
}

class SecurityHeaderAnalyzer:
    def __init__(self, threshold=0.8): self.threshold = threshold
    async def analyze_security_headers_async(self, url, session): try:
        async with session.get(url) as response: headers = response.headers used_headers = {}
        missing_headers = {}
        for header, description in SECURITY_HEADERS.items(): if header in headers:
            used_headers[header] = description else:
                missing_headers[header] = description return used_headers, missing_headers
        except aiohttp.ClientError as e: print(f"Error analyzing {url}: {e}") return {}, {}
    async def process_url_async(self, url, session, output_file=None): parsed_url = urlparse(url)
    if not parsed_url.scheme:
        url = "https://" + url
    try:
        used_headers, missing_headers = await self.analyze_security_headers_async(url, session)

        with open(output_file, 'a') as f: f.write(f"\nDomain: {url}\n\n") f.write(f"\nUsed security headers:\n")
```

```

self.write_headers_to_file(used_headers, "green", f)
f.write("\nMissing security headers:\n") self.write_headers_to_file(missing_headers, "red", f)
grade = self.get_grade(used_headers) f.write(f"\n\nThe Grade of website is : {grade}\n")
f.write("    \n")
except Exception as e:
print(f"Error processing {url}: {e}")
def write_headers_to_file(self, headers, color, file_obj): for header, description in
headers.items():
file_obj.write(f"{colored(header, color)}: {description}\n")
def get_grade(self, used_headers): num_headers = len(SEcurity_HEADERS)
num_used_headers = len(used_headers)
if num_used_headers == num_headers: return "A+"
elif num_used_headers >= num_headers * self.threshold: return "A"
elif num_used_headers >= num_headers * 0.8: return "B"
elif num_used_headers >= num_headers * 0.7: return "C"
elif num_used_headers >= num_headers * 0.6: return "D"
else:
return "F"
async def analyze_urls_async(urls, output_file=None, threshold=0.8): async with
aiohttp.ClientSession() as session:
analyzer = SecurityHeaderAnalyzer(threshold)
tasks = [analyzer.process_url_async(url.strip(), session, output_file) for url in urls]
await asyncio.gather(*tasks)
def analyze_urls_from_file(file_path, output_file=None, threshold=0.8): with open(file_path) as
file:
urls = [line.strip() for line in file] asyncio.run(analyze_urls_async(urls, output_file, threshold))
def analyze_urls_from_input(url, output_file=None, threshold=0.8): urls = [url]
asyncio.run(analyze_urls_async(urls, output_file, threshold))

# Example usage:
# analyze_urls_from_input("https://example.com", "output.txt", 0.8)

```

4.2.web-audit.py

```
import sys import argparse
```

```

import subprocess import os
import time import random import threading import re
import random
from urllib.parse import urlsplit from security_header import *
CURSOR_UP_ONE = '\x1b[1A' ERASE_LINE = '\x1b[2K'
intervals = ( ('h', 3600),
('m', 60),
('s', 1),
)
def display_time(seconds, granularity=3): result = []
seconds = seconds + 1
for name, count in intervals: value = seconds // count if value:
seconds -= value * count result.append("{}{}".format(value, name))
return ' '.join(result[:granularity])
def terminal_size(): try:
rows, columns = subprocess.check_output(['stty', 'size']).split() return int(columns)
except subprocess.CalledProcessError as e: return int(20)
def url_maker(url):
if not re.match(r'http(s?)\:', url): url = 'http://' + url
parsed = urlsplit(url) host = parsed.netloc
if host.startswith('www.'): host = host[4:]
return host
def check_internet():
os.system('ping -c1 github.com > rs_net 2>&1') if "0% packet loss" in open('rs_net').read():
val = 1 else:
val = 0
os.system('rm rs_net > /dev/null 2>&1') return val

# Initializing the color module class class bcolors:
HEADER = '\033[95m' OKBLUE = '\033[94m' OKGREEN = '\033[92m' WARNING = '\033[93m'
BADFAIL = '\033[91m' ENDC = '\033[0m' BOLD = '\033[1m' UNDERLINE = '\033[4m'
BG_ERR_TXT = '\033[41m' BG_HEAD_TXT = '\033[100m' BG_ENDL_TXT = '\033[46m'
BG_CRIT_TXT = '\033[45m' BG_HIGH_TXT = '\033[41m' BG_MED_TXT = '\033[43m'
BG_LOW_TXT = '\033[44m' BG_INFO_TXT = '\033[42m'
BG_SCAN_TXT_START = '\x1b[6;30;42m' BG_SCAN_TXT_END = '\x1b[0m'

```

```

# Classifies the Vulnerability's Severity def vul_info(val):
result = ""
if val == 'c':
result = bcolors.BG_CRIT_TXT+" critical "+bcolors.ENDC elif val == 'h':
result = bcolors.BG_HIGH_TXT+" high "+bcolors.ENDC elif val == 'm':
result = bcolors.BG_MED_TXT+" medium "+bcolors.ENDC
elif val == 'l':
result = bcolors.BG_LOW_TXT+" low "+bcolors.ENDC else:
result = bcolors.BG_INFO_TXT+" info "+bcolors.ENDC return result

# Legends
proc_high = bcolors.BADFAIL + "●" + bcolors.ENDC proc_med = bcolors.WARNING + "●" +
bcolors.ENDC proc_low = bcolors.OKGREEN + "●" + bcolors.ENDC
# Links the vulnerability with threat level and remediation database def
vul_remed_info(v1,v2,v3):
print(bcolors.BOLD+"Vulnerability Threat Level"+bcolors.ENDC) print("\t"+vul_info(v2)+
"+bcolors.WARNING+str(tool_resp[v1][0])+bcolors.ENDC) print(bcolors.BOLD+"Vulnerability
Definition"+bcolors.ENDC) print("\t"+bcolors.BADFAIL+str(tools_fix[v3-1][1])+bcolors.ENDC)
print(bcolors.BOLD+"Vulnerability Remediation"+bcolors.ENDC)
print("\t"+bcolors.OKGREEN+str(tools_fix[v3-1][2])+bcolors.ENDC)

# Help
def helper():
print(bcolors.OKBLUE+"Information:"+bcolors.ENDC) print(" ")
print("\t./web-audit.py example.com: Scans the domain example.com.") print("\t./web-audit.py
example.com --skip dmitry --skip theHarvester: Skip the 'dmitry' and 'theHarvester' tests.")
print("\t./web-audit.py example.com --nospinner: Disable the idle loader/spinner.")
print("\t./web-audit.py --help : Displays this help context.")
print(bcolors.OKBLUE+"Interactive:"+bcolors.ENDC)
print(" ")
print("\tCtrl+C: Skips current test.") print("\tCtrl+Z: Quits web-audit")
print(bcolors.OKBLUE+"Legends:"+bcolors.ENDC) print(" ")
print("\t["+proc_high+"]: Scan process may take longer times (not predictable).")
print("\t["+proc_med+"]: Scan process may take less than 10 minutes.") print("\t["+proc_low+]:
Scan process may take less than a minute or two.") print(bcolors.OKBLUE+"Vulnerability

```

```

Information:"+bcolors.ENDC)
print(" ")
print("\t"+vul_info('c')+": Requires immediate attention as it may lead to compromise or service
unavailability.")
print("\t"+vul_info('h')+": May not lead to an immediate compromise, but there are
considerable chances for probability.")
print("\t"+vul_info('m')+": Attacker may correlate multiple vulnerabilities of this type to launch a
sophisticated attack.")
print("\t"+vul_info('l')+": Not a serious issue, but it is recommended to tend to the finding.")
print("\t"+vul_info('i')+": Not classified as a vulnerability, simply an useful
informational alert to be considered.\n")
# Clears Line def clear():
sys.stdout.write("\033[F") sys.stdout.write("\033[K")
# Logo def logo():
print(bcolors.WARNING) logo_ascii = """
Pheonix Web Scanner
"""

print(logo_ascii) print(bcolors.ENDC)
class Spinner: busy = False delay = 0.005
    @staticmethod
    def spinning_cursor(): while 1:
        #for cursor in '|/-\\V': yield cursor #←↑↓→ #for cursor in '←↑↓→': yield cursor
        #for cursor in '....scanning...please..wait.': yield cursor
        for cursor in ' ': yield cursor
    def init (self, delay=None):
        self.spinner_generator = self.spinning_cursor() if delay and float(delay): self.delay = delay
        self.disabled = False
    def spinner_task(self): inc = 0
    try:
        while self.busy:
            if not self.disabled: x =
bcolors.BG_SCAN_TXT_START+next(self.spinner_generator)+bcolors.BG_SCAN_TXT_END
            inc = inc + 1 print(x,end="")
            if inc>random.uniform(0,terminal_size()): print(end="\r") bcolors.BG_SCAN_TXT_START =
'\x1b[6;30;'+str(round(random.uniform(40,47)))+ 'm' inc = 0

```

```

sys.stdout.flush()
time.sleep(self.delay) if not self.disabled:
sys.stdout.flush()
except (KeyboardInterrupt, SystemExit):
print("\n\t" + bcolors.BG_ERR_TXT + "received a series of Ctrl+C hits.
Quitting..." + bcolors.ENDC) sys.exit(1)
def start(self): self.busy = True try:
threading.Thread(target=self.spinner_task).start() except Exception as e:
print("\n")
def stop(self): try:
self.busy = False time.sleep(self.delay)
except (KeyboardInterrupt, SystemExit):
print("\n\t" + bcolors.BG_ERR_TXT + "received a series of Ctrl+C hits.
Quitting..." + bcolors.ENDC) sys.exit(1)
# End of loader/spinner class
# Instantiating the spinner/loader class spinner = Spinn
# Tool Set tools_precheck = [
["wapiti"], ["whatweb"], ["nmap"], ["golismero"], ["host"], ["wget"],
["uniscan"], ["wafw00f"], ["dirb"], ["davtest"], ["theHarvester"], ["xsser"],
["dnsrecon"], ["fierce"], ["dnswalk"], ["whois"], ["sslyze"], ["lbd"], ["golismero"],
["dnsenum"], ["dmitry"], ["davtest"], ["nikto"], ["dnsmap"], ["amass"]
]
def get_parser():
parser = argparse.ArgumentParser(add_help=False) parser.add_argument('-h', '--help',
action='store_true',
help='Show help message and exit.') parser.add_argument('-u', '--update', action='store_true',
help='Update tool.')
parser.add_argument('-s', '--skip', action='append', default=[],
help='Skip some tools', choices=[t[0] for t in tools_precheck]) parser.add_argument('-n', '--
nospinner', action='store_true',
help='Disable the idle loader/spinner.') parser.add_argument('target', nargs='?',
metavar='URL', help='URL to scan.',
default="", type=str) return parser
# Shuffling Scan Order (starts)
scan_shuffle = list(zip(tool_names, tool_cmd, tool_resp, tool_status))

```

```

random.shuffle(scan_shuffle)
tool_names, tool_cmd, tool_resp, tool_status = zip(*scan_shuffle)
tool_checks = (len(tool_names) + len(tool_resp) + len(tool_status)) / 3 # Cross verification
incase, breaks.
tool_checks = round(tool_checks) # Shuffling Scan Order (ends)
# Tool Head Pointer: (can be increased but certain tools will be skipped) tool = 0
# Run Test runTest = 1
# For accessing list/dictionary elements arg1 = 0
arg2 = 1
arg3 = 2
arg4 = 3
arg5 = 4
arg6 = 5
# Detected Vulnerabilities [will be dynamically populated] rs_vul_list = list()
rs_vul_num = 0
rs_vul = 0
# Total Time Elapsed rs_total_elapsed = 0
# Tool Pre Checker rs_avail_tools = 0
# Checks Skipped rs_skipped_checks = 0
if len(sys.argv) == 1: logo()
helper() sys.exit(1)
args_namespace = get_parser().parse_args()
analyze_urls_from_input(args_namespace.target, "security-header.txt", 0.8) if
args_namespace.nospinner:
spinner.disabled = True
if args_namespace.help or (not args_namespace.update \ and not args_namespace.target):
logo() helper()
elif args_namespace.target:
target = url_maker(args_namespace.target) #target = args_namespace.target
os.system('rm /tmp/rapidscan* > /dev/null 2>&1') # Clearing previous scan files
os.system('clear')
os.system('setterm -cursor off') logo()
print(bcolors.BG_HEAD_TXT+"[ Checking Available Security Scanning Tools Phase... Initiated.
]"+bcolors.ENDC)
unavail_tools_names = list()

```

```

while (rs_avail_tools < len(tools_precheck)):
    precmd = str(tools_precheck[rs_avail_tools][arg1]) try:
    p = subprocess.Popen([precmd], stdin=subprocess.PIPE, stdout=subprocess.PIPE,
    stderr=subprocess.PIPE, shell=True)
    output, err = p.communicate() val = output + err
    except:
    print("\t"+bcolors.BG_ERR_TXT+"web-audit was terminated abruptly..." +bcolors.ENDC)
    sys.exit(1)
    # If the tool is not found or it's part of the --skip argument(s), disabling it if b"not found" in val or
    tools_precheck[rs_avail_tools][arg1] in args_namespace.skip :
    if b"not found" in val:
    print("\t"+bcolors.OKBLUE+tools_precheck[rs_avail_tools][arg1]+bcolors.ENDC
    +bcolors.BADFAIL+"...unavailable." +bcolors.ENDC)
    elif tools_precheck[rs_avail_tools][arg1] in args_namespace.skip :
    print("\t"+bcolors.OKBLUE+tools_precheck[rs_avail_tools][arg1]+bcolors.ENDC
    +bcolors.BADFAIL+"...skipped." +bcolors.ENDC)
    for scanner_index, scanner_val in enumerate(tool_names): if scanner_val[2] ==
    tools_precheck[rs_avail_tools][arg1]:
    scanner_val[3] = 0 # disabling scanner as it's not available.
    unavail_tools_names.append(tools_precheck[rs_avail_tools][arg1])
    else:
    print("\t"+bcolors.OKBLUE+tools_precheck[rs_avail_tools][arg1]+bcolors.ENDC
    +bcolors.OKGREEN+"...available." +bcolors.ENDC) rs_avail_tools = rs_avail_tools + 1
    clear()
    unavail_tools_names = list(set(unavail_tools_names)) if len(unavail_tools_names) == 0:
    print("\t"+bcolors.OKGREEN+"All Scanning Tools are available. Complete vulnerability checks
    will be performed by web-audit." +bcolors.ENDC)
    else:
    print("\t"+bcolors.WARNING+"Some of these tools
    "+bcolors.BADFAIL+str(unavail_tools_names)+bcolors.ENDC+bcolors.WARNIN G+" are
    unavailable or will be skipped. web-audit will still perform the rest of the tests. Install these tools
    to fully utilize the functionality of web- audit." +bcolors.ENDC)
    print(bcolors.BG_ENDL_TXT+"[ Checking Available Security Scanning Tools Phase...
    Completed. ]"+bcolors.ENDC)
    print("\n")

```

```

print(bcolors.BG_HEAD_TXT+"[ Preliminary Scan Phase Initiated... Loaded
"+str(tool_checks)+" vulnerability checks. ]"+bcolors.ENDC)
#while (tool < 1):
while(tool < len(tool_names)): print("[ "+tool_status[tool][arg3]+tool_status[tool][arg4]+"
Deploying "+str(tool+1)+"/"+str(tool_checks)+" |
"+bcolors.OKBLUE+tool_names[tool][arg2]+bcolors.ENDC,)
if tool_names[tool][arg4] == 0: print(bcolors.WARNING+"\nScanning Tool Unavailable.
Skipping
Test...\n"+bcolors.ENDC)
rs_skipped_checks = rs_skipped_checks + 1 tool = tool + 1
continue try:
spinner.start() except Exception as e:
print("\n")
scan_start = time.time()
temp_file = "/tmp/rapidscan_temp_"+tool_names[tool][arg1]
cmd = tool_cmd[tool][arg1]+target+tool_cmd[tool][arg2]+" > "+temp_file+" 2>&1"
try:
subprocess.check_output(cmd, shell=True) except KeyboardInterrupt:
runTest = 0 except:
runTest = 1 if runTest == 1:
spinner.stop() scan_stop = time.time()
elapsed = scan_stop - scan_start rs_total_elapsed = rs_total_elapsed + elapsed
#print(bcolors.OKBLUE+"\b...Completed in
"+display_time(int(elapsed))+bcolors.ENDC+"\n") sys.stdout.write(ERASE_LINE)
print(bcolors.OKBLUE+"\nScan Completed in
"+display_time(int(elapsed))+bcolors.ENDC, end='\r', flush=True) print("\n")
#clear()
rs_tool_output_file = open(temp_file).read() if tool_status[tool][arg2] == 0:
if tool_status[tool][arg1].lower() in rs_tool_output_file.lower():
#print "\t"+ vul_info(tool_resp[tool][arg2]) + bcolors.BADFAIL + " "+ tool_resp[tool][arg1] +
bcolors.ENDC
vul_remed_info(tool,tool_resp[tool][arg2],tool_resp[tool][arg3])
rs_vul_list.append(tool_names[tool][arg1]+""+tool_names[tool][arg2])
else:
if any(i in rs_tool_output_file for i in tool_status[tool][arg6]): m = 1 # This does nothing.

```

```

else:
#print "\t"+ vul_info(tool_resp[tool][arg2]) + bcolors.BADFAIL +" "+ tool_resp[tool][arg1] +
bcolors.ENDC
vul_remed_info(tool,tool_resp[tool][arg2],tool_resp[tool][arg3])
rs_vul_list.append(tool_names[tool][arg1]+""+tool_names[tool][arg2])
else:
runTest = 1 spinner.stop() scan_stop = time.time()
elapsed = scan_stop - scan_start rs_total_elapsed = rs_total_elapsed + elapsed
#sys.stdout.write(CURSОР_UP_ONE) sys.stdout.write(ERASE_LINE)
#print("-" * terminal_size(), end='\r', flush=True) print(bcolors.OKBLUE+"\nScan Interrupted in
"+display_time(int(elapsed))+bcolors.ENDC, end='\r', flush=True) print("\n"+bcolors.WARNING
+ "\tTest Skipped. Performing Next. Press
Ctrl+Z to Quit.\n" + bcolors.ENDC) rs_skipped_checks = rs_skipped_checks + 1
tool=tool+1
print(bcolors.BG_ENDL_TXT+"[ Preliminary Scan Phase Completed.
]" +bcolors.ENDC) print("\n")
# Report & Documentation Phase
date = subprocess.Popen(["date", "+%Y-%m-
%d"],stdout=subprocess.PIPE).stdout.read()[:-1].decode("utf-8") debuglog = "web-
audit.dbg.%s.%s" % (target, date)
vulreport = "web-audit.vul.%s.%s" % (target, date) print(bcolors.BG_HEAD_TXT+"[ Report
Generation Phase Initiated.
]" +bcolors.ENDC) if len(rs_vul_list)==0:
print("\t"+bcolors.OKGREEN+"No Vulnerabilities Detected."+bcolors.ENDC) else:
with open(vulreport, "a") as report: while(rs_vul < len(rs_vul_list)):
vuln_info = rs_vul_list[rs_vul].split("") report.write(vuln_info[arg2]) report.write("\n \n\n")
temp_report_name = "/tmp/rapidscan_temp_"+vuln_info[arg1] with open(temp_report_name,
'r') as temp_report:
data = temp_report.read() report.write(data) report.write("\n\n")
temp_report.close() rs_vul = rs_vul + 1

print("\tComplete Vulnerability Report for "+bcolors.OKBLUE+target+bcolors.ENDC+" named
"+bcolors.OKGREEN+vulreport+bcolors.ENDC+" is available under the same directory
resides.")
report.close()

```

```

# Writing all scan files output into RS-Debug-ScanLog for debugging purposes. for file_index,
file_name in enumerate(tool_names):
with open(debuglog, "a") as report: try:
with open("/tmp/rapidscan_temp_"+file_name[arg1], 'r') as temp_report: data =
temp_report.read()
report.write(file_name[arg2]) report.write("\n\n")
report.write(data) report.write("\n\n")
temp_report.close() except:
break report.close()
print("\tTotal Number of Vulnerability Checks :
"+bcolors.BOLD+bcolors.OKGREEN+str(len(tool_names))+bcolors.ENDC)
print("\tTotal Number of Vulnerability Checks Skipped:
"+bcolors.BOLD+bcolors.WARNING+str(rs_skipped_checks)+bcolors.ENDC)
print("\tTotal Number of Vulnerabilities Detected :
"+bcolors.BOLD+bcolors.BADFAIL+str(len(rs_vul_list))+bcolors.ENDC)
print("\tTotal Time Elapsed for the Scan :
"+bcolors.BOLD+bcolors.OKBLUE+display_time(int(rs_total_elapsed))+bcolors.ENDC)
print("\n")
print("\tFor Debugging Purposes, You can view the complete output generated by
all the tools named "+bcolors.OKBLUE+debuglog+bcolors.ENDC+" under the same
directory.")
print(bcolors.BG_ENDL_TXT+"[ Report Generation Phase Completed.
]"+bcolors.ENDC)
os.system('setterm -cursor on')
os.system('rm /tmp/rapidscan_te* > /dev/null 2>&1')

```

```

[vasanth@Sathyabama] (~/.project)
$ cat security-header.txt

Domain: https://www.sathyabama.ac.in

Used security headers:
X-Content-Type-Options: Prevents MIME type sniffing.
X-Frame-Options: Protects against clickjacking.

Missing security headers:
Strict-Transport-Security: Enforces secure (HTTP over SSL/TLS) connections to the server.
Content-Security-Policy: Helps prevent cross-site scripting (XSS) and other code injection attacks.
X-XSS-Protection: Protects against XSS attacks.
Referrer-Policy: Controls how much referrer information should be included with requests.
Permissions-Policy: Controls which browser features can be used by the site.

The Grade of website is : F
-----

```



```

[* + 15s] Deploying 6/80 | Nmap - Checks for Slowloris Denial of Service
Task Completed at 15s

[* + 15s] Deploying 7/80 | curl - Checks for Denial of Service
Scanning Tool Unavailable - Skipping Test

[* + 45s] Deploying 8/80 | Nmap - Checks for Slowloris Denial of Service Vulnerability
Task Completed at 45s

Vulnerability Threat Level
Critical Vulnerable to Slowloris Denial of Service.
Vulnerability Definition
This attack works by opening multiple simultaneous connections to the web server and it keeps them alive as long as possible by continuously sending
small HTTP requests which never get a completion. They easily pass through IDS by opening partial requests.
Vulnerability Remediation
If you are using Apache Module mod_proxy would help. For other setup you can find more detailed remediation in this resource: https://www.wisecoders.com/blog/2015/05/05/http-dos-attacks-mitigate-apache-http-server/
[* + 9s] Deploying 9/80 | Nmap - Checks for Denial of Service
Task Completed at 9s

[* + 28s] Deploying 10/80 | Nmap - Checks if the host is affected by SYN flood attack
Task Completed at 28s

[* + 3s] Deploying 11/80 | Nmap - Checks for HTTP, SSL, TLS and other vulnerabilities
Task Completed at 3s

[* + 15s] Deploying 12/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 15s

[* + 15s] Deploying 13/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 15s

```

Fig C.3 Vulnerability auditing

```

[* + 15s] Deploying 14/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 15s

[* + 10s] Deploying 17/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 10s

[* + 10s] Deploying 18/80 | DNSMap - Checks subdomains
Task Completed at 10s

Vulnerability Threat Level
Critical Found Subdomains with DNSMap
Vulnerability Definition
Attackers may gather more information from subdomains relating to the target domain. Attackers may also find other services like the subdomains and
use the domain for the purpose. There are many attacks for the attackers find in subdomains. If the attack surface gets larger with more sub
domains, the attack surface gets larger.
Vulnerability Remediation
It is sometimes wise to block subdomains like development, staging to the outside world, as it gives more information to the attackers about the
internal complex naming practices also help in reducing the attack surface as attackers find hard to perform subdomain brute forcing through continuous
and regularity.
[* + 15s] Deploying 19/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 15s

[* + 20s] Deploying 20/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 20s

[* + 3s] Deploying 21/80 | The Harvester - Checks for sensitive data using a generic search
Task Completed at 3s

[* + 10s] Deploying 22/80 | Nikto - Checks for Apache specific vulnerabilities
Task Completed at 10s

[* + 15s] Deploying 23/80 | Nmap - Checks for DoS (Denial of Service)
Task Completed at 15s

```

Fig C.4 Vulnerability auditing

```

Scan Completed in 14s

Vulnerability Threat Level
Low Some ports are open. Perform a Full-scan manually.
Vulnerability Definition
Open ports give attackers a hint to the exposed services. Attackers try to retrieve banner information through the ports and understand what type of
services are host is running.
Vulnerability Remediation
It is recommended to close the ports of unused services and use a firewall to filter the ports wherever necessary. This resource may give more insight
https://www.cyberciti.ch/stackexchange.com/a/165781/6127
[* - 28s] Deploying 25/88 | Checks for SMB service over 445
Scan Completed in 14s

[* - 38s] Deploying 26/88 | New: Heartbleed | Checks only for Heartbleed vulnerability.
Scan Completed in 14s

[* - 75s] Deploying 27/88 | Fierce Subdomain Spiker | Brute Force Subdomain Discovery.
Scan Completed in 2s 22s

Vulnerability Threat Level
Low Found Subdomains with Fierce.
Vulnerability Definition
Attackers may gather more information from subdomains relating to the parent domain. Attackers may even find other services from the subdomains and
try to learn the architecture of the target. There are even chances for the attacker to find vulnerabilities as the attack surface gets larger with more sub
domains discovered.
Vulnerability Remediation
It is sometimes wise to block sub domains like development staging to the outside world, as it gives more information to the attacker about the tar
get stack. Complex naming practices also help in reducing the attack surface as attackers find hard to perform subdomain bruteforcing through dictionaries and
wordlists.
[* - 25s] Deploying 28/88 | Wdirc | Checks for Administrator's Contact Information.
Scan Completed in 14s

Vulnerability Threat Level
Low Whois Information Publicly Available
Vulnerability Definition
The owner's identity of the domain is publicly available through whois. While it is not able to identify an attacker, it is still a security risk. An attacker may use these information to
gather information that may lead to more attacks and a person whose name is used in a domain is more likely to be attacked.

```

Fig C.5 vulnerability auditing

```

[* - 15s] Deploying 88/88 | AFMass | Brute's Domain for Subdomains.
Scan Completed in 1s 36s

Vulnerability Threat Level
Low Found Subdomains with AFMass
Vulnerability Definition
Attackers may gather more information from subdomains relating to the parent domain. Attackers may even find other services from the subdomains and
try to learn the architecture of the target. There are even chances for the attacker to find vulnerabilities as the attack surface gets larger with more sub
domains discovered.
Vulnerability Remediation
It is sometimes wise to block sub domains like development staging to the outside world, as it gives more information to the attacker about the tar
get stack. Complex naming practices also help in reducing the attack surface as attackers find hard to perform subdomain bruteforcing through dictionaries and
wordlists.
Report Generation Phase Completed

Report Generation Phase Initiated
Complete Vulnerability Report for sathya@kali:rs-vul sathya@kali in 2024-04-29 is available under the same directory resides.
Total Number of Vulnerability Checks 89
Total Number of Vulnerability Checks Skipped 10
Total Number of Vulnerabilities Detected 11
Total Time Elapsed for the Scan : 2h 53s

For Debugging Purposes, You can view the complete output generated by all the tools named rs-vul sathya@kali in 2024-04-29 under the same director
Report Generation Phase Completed

sathya@kali:~$ cd /project

```

Fig C.8 Summary of website auditing