# Mitigating Distributed Denial of Service Attacks For Web Security

**PROJECT REPORT**

***Submitted by***

| | |
|---|---|
| **AJITH A** | **712821104003** |
| **AMALDAS KK** | **712821104004** |
| **GOKUL M** | **712821104013** |
| **HARINI M** | **712821104018** |

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**RVS COLLEGE OF ENGINEERING AND TECHNOLOGY**
**(AN AUTONOMOUS INSTITUTION)**

**COIMBATORE 641 402**

**ANNA UNIVERSITY:: CHENNAI 600025**

**MAY 2025**

## ANNA UNIVERSITY : CHENNAI-6000025

## BONAFIDE CERTIFICATE

Certified that this project report on "**Mitigating Distributed Denial of Service Attacks For Web Security**" is the bonafide work of **AJITH A (712821104003)**, **AMALDAS KK (712821104004)**, **GOKUL M (712821104013)**, **HARINI M (712821104018)** who carried out the work under my supervision.

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| **Dr. K. KARUPPASAMY, M.E., Ph.D.,** | **Prof. R. THENMALAR, M.E.,** |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Professor and Head, | Assistant Professor, |
| Department of CSE, | Department of CSE, |
| RVS College of Engineering & | RVS College of Engineering & |
| Technology, Kumaran Kottam, | Technology, Kumaran Kottam, |
| Campus, Kannampalayam, | Campus, Kannampalayam, |
| Coimbatore-641 402 | Coimbatore-641 402 |

**Submitted for the project Viva-Voce examination held on__________________**

**Internal Examiner** **External Examiner**

# ACKNOWLEDGEMENT

We would like to express our gratitude first to the Almighty chairman who has showered his blessings to complete our project successfully.

We would like to express our sincere thanks to our Principal **Dr.G.MOHAN KUMAR, M.E., Ph.D.,** RVS College of Engineering and Technology, for his great support.

We greatly privileged to express our deep sense of gratitude to our Head of the Department of Computer Science and Engineering, **Dr.K.KARUPPASAMY, M.E., Ph.D.,** for his valuable guidance and suggestions throughout the course of this project.

We are grateful to our Project Coordinator **Prof. R.THENMALAR, M.E.,** Assistant Professor, Department of Computer Science and Engineering, for her patience and guidance throughout our project work.

We deeply thank our Project Guide **Prof. R.THENMALAR, M.E.,** Assistant professor, Department of Computer Science and Engineering, for her guidance and support in completing the project.

We would also like to thank all the staff members of our Department for their encouragement to finish the project.

Finally, we would like to thank the members who have helped to complete the project successfully.

# ABSTRACT

# ABSTARCT

WordPress is a highly popular content management system (CMS), powering over 172 million websites globally across a wide range of domains, including e-commerce, blogging, and digital publishing. Its appeal lies in its ease of use, support for multiple users, and distributed system compatibility. However, this widespread usage has made it a common target for cyberattacks such as Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), and user enumeration, the latter of which enables attackers to discover valid usernames and launch brute-force attacks.

This study investigates the security vulnerabilities present in WordPress environments, with a specific focus on DoS, DDoS, and user enumeration threats. By analyzing more than 100 WordPress websites, including controlled local installations, and using tools like SSH scripts and Python, the research identifies how poor configurations and insufficient security hardening make systems vulnerable to exploitation.

The paper further provides practical mitigation strategies, including proper server configurations, secure plugin and theme usage, access control policies, and traffic monitoring techniques. By replicating real-world attack scenarios on a self-hosted WordPress instance, the research offers actionable insights for administrators and developers to bolster WordPress security against evolving threats.

Index Terms – WordPress, CMS, WordPress Security, WordPress User Enumeration, WordPress Vulnerability, PHP, CMS Vulnerability.

**TABLE OF CONTENTS**

## LIST OF FIGURES

# LIST OF FIGURES

## LIST OF TABLES

# LIST OF TABLES

# LIST OF ABBREVATIONS

# LIST OF ABBREVIATIONS

| S.NO | ABBREVATIONS | EXPANSIONS |
|---|---|---|
| 1 | **DDoS** | Distributed Denial of Service |
| 2 | **HTTP** | Hypertext Transfer Protocol |
| 3 | **TCP** | Transmission Control Protocol |
| 4 | **IP** | Internet Protocol |
| 5 | **CMS** | Content Management System |
| 6 | **OWASP** | Open Worldwide Application Security Project |
| 7 | **DNS** | Domain Name System |
| 8 | **API** | Application Programming Interface |
| 9 | **CPU** | Central Processing Unit |
| 10 | **UDP** | User Datagram Protocol |
| 11 | **SSL** | Secure Sockets Layer |
| 12 | **CDN** | Content Delivery Network |
| 13 | **JSON** | JavaScript Object Notation |

# INTRODUCTION

# INTRODUCTION

In today's digital era, over 3.2 billion users across the globe access the internet through a wide array of devices, including personal computers, smartphones, feature phones, smart gadgets, vehicles, and Point-of-Sale (PoS) machines. These users generate an astonishing 2.5 quintillion bytes of data daily, posing immense challenges in terms of data storage, protection, and privacy. As web applications have become fundamental to the digital presence of businesses and individuals alike, the need for robust and scalable platforms has surged. WordPress, a leading content management system (CMS), powers approximately 172 million websites, accounting for nearly 30% of the internet. Its flexibility and ease of use, however, make it a frequent target for cyberattacks.

Common vulnerabilities in WordPress, as identified by the Open Web Application Security Project (OWASP), include injection flaws, broken authentication, cross-site scripting (XSS), security misconfigurations, and denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. These threats are exacerbated by poor maintenance practices, such as outdated plugins and WordPress cores, which leave websites exposed to brute-force and automated scanning attacks. Tools and scripts in languages such as Python are often used by attackers to detect vulnerable WordPress installations, leading to widespread exploitation.

Our research addresses this issue by developing a vulnerability scanning tool using SSH, PHP, and Python, which detects the WordPress core version, installed plugins, their versions, and potential vulnerabilities. This tool was tested on more than 100 WordPress web applications to assess its effectiveness and accuracy in identifying threats.

Moreover, the evolution of cloud computing, now supporting over

87% of global businesses, has shifted the threat landscape. Cloud-based systems are increasingly targeted by DDoS and Economic Denial of Sustainability (EDoS) attacks, causing major service disruptions and financial losses. This growing attack surface demands a proactive approach to vulnerability assessment and cybersecurity in WordPress and cloud-hosted environments alike.

## 1.1 Problem Definition

The foundational design of the Internet prioritized functionality over security, leading to critical vulnerabilities that facilitate Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. These systemic weaknesses stem from the Internet's open architecture, lack of centralized control, and limited built-in security mechanisms. One major issue is the inherent openness and resource-sharing nature of the Internet. Web servers must be publicly accessible via globally routable IP addresses to offer services, making them direct targets. Moreover, the packet- switching model of the Internet allows for shared resource usage, where one user’s malicious behavior can disrupt services for others—an exploit leveraged by flooding attacks.

Another significant vulnerability is the absence of integrated authentication, integrity verification, and traceability within Internet protocols. Attackers commonly use IP spoofing to forge source addresses, making it difficult to trace or verify the origin of malicious traffic. Without packet integrity checks and due to the impracticality of routers storing traffic histories, attackers can easily disguise their identity and intentions, launching attacks with impunity.

Additionally, Internet security is highly interdependent. While one organization can secure its systems, the existence of unsecured systems globally provides attackers with a vast pool of machines to compromise and incorporate into botnets. This interconnected risk is compounded by intelligence and resource asymmetry. While most of the intelligence and decision-making is concentrated at the end hosts, the attackers exploit the high-bandwidth, intermediate networks to relay large volumes of malicious traffic, overwhelming the victim's resources.

Lastly, the decentralized governance of the Internet limits the ability to implement unified, global security policies. Networks operate under their own policies and are reluctant to collaborate due to privacy or commercial concerns. As a result, implementing comprehensive and synchronized defense mechanisms across the Internet is a major challenge, leaving systems vulnerable to widespread DDoS threats.

## 1.2 Scope of the project

The scope of this study centers on understanding and mitigating the risks associated with Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks, particularly as they relate to vulnerabilities in operating systems, network protocols, and web applications such as those developed using WordPress. A significant portion of existing operating systems and network protocols were developed without foundational security engineering principles, resulting in widespread deployment of insecure and unpatched machines across the internet. These vulnerable systems are routinely exploited by attackers to build botnets—networks of compromised machines referred to as zombies and handlers—which serve as the launch platform for large-scale DDoS attacks.

The attack mechanism typically involves the transmission of overwhelming traffic or seemingly legitimate requests to a target server or network, thereby exhausting its resources such as CPU cycles, memory, bandwidth, and buffer capacity. This leads to severe service degradation or complete unavailability for legitimate users. Commercial servers, although hosted close to the ISP backbone with high bandwidth links, are not immune. Attack traffic generated by zombies can easily saturate server-side resources, leading to overrun buffers, dropped packets, and a reduced ability to handle legitimate traffic.

This study specifically investigates how these types of attacks affect WordPress-based web applications, which form a significant portion of modern internet infrastructure. It also examines the role of resource exhaustion in flooding attacks and how protocol-level vulnerabilities—such as those in TCP or HTTP—are exploited to disrupt services. The scope extends to analyzing server behavior under stress, evaluating traffic patterns during attack scenarios, and developing detection and mitigation strategies using tools developed during the research. Ultimately, this work aims to enhance the resilience of web servers and applications against increasingly sophisticated DDoS and DoS threats through systematic vulnerability assessment and resource management practices.

## 1.3 Objective of the project

The primary objective of this research is to analyze and mitigate the risks associated with Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks, with a particular focus on the vulnerabilities that arise from insecure network protocols and unpatched operating systems. The study aims to understand how attackers exploit these weaknesses by compromising machines and converting them into bots, also known as zombies, to form a botnet. These botnets

are then orchestrated by master or handler systems that issue commands to the compromised systems to launch coordinated attacks on targeted servers or networks.

A key focus of this research is to differentiate between two major forms of DDoS attacks: flooding-based and vulnerability-based. Flooding DDoS attacks aim to overwhelm limited resources such as bandwidth, memory, and processing power by sending massive volumes of traffic. Conversely, vulnerability attacks exploit inherent weaknesses in network protocols like TCP and HTTP to generate seemingly legitimate requests that consume server resources, thereby preventing access for genuine users.

The research further investigates how commercial servers, despite having high-bandwidth access links provided by ISPs, remain vulnerable to DDoS attacks due to limitations in server-side resources like CPU capacity and buffer limits. The objective is to simulate these attack conditions to better understand the point at which service degradation begins and how it progresses to complete denial of service.

Additionally, the project aims to design and evaluate detection mechanisms and mitigation strategies that can identify abnormal request patterns, reduce resource exhaustion, and maintain service availability for legitimate clients. By examining how legitimate traffic is affected under such attack scenarios, this research ultimately strives to enhance the resilience of server infrastructure against modern DDoS and DoS attacks.

## 1.4 Organization of the report

The project report is organized as follows:

Chapter 1 provides an introduction to the project.

Chapter 2 presents a literature review for the previous works in the field and discusses their limitations.

Chapter 3 describes the system methodology for both existing and proposed systems.

Chapter 4 details the hardware and software system specifications.

Chapter 5 outlines the implementation of the modules.

Chapter 6 covers the product testing process.

Chapter 7 Concludes with a summary of findings and results of the work.

# LITERARTURE SURVEY

# CHAPTER 2

# LITERATURE SURVEY

1. **Exploring DDoS Mitigation Techniques and Complications**

   **Authors:** Liam Temple

   **Year:** 2022

   **Problem:**
   The study investigates the challenges posed by Distributed Denial-of-Service (DDoS) attacks, which can disrupt services and overwhelm network infrastructure. As attacks grow in complexity and scale, traditional defense mechanisms often struggle to respond effectively.

   **Objective:**
   The paper aims to explore modern techniques for mitigating DDoS attacks and analyze the complications that arise in their implementation, particularly in maintaining service availability and minimizing collateral impact.

   **Algorithm/Technique:**
   The research discusses various mitigation strategies, including rate-limiting, traffic filtering, anomaly detection, and the use of scrubbing centers. It also examines the trade-offs between performance, detection accuracy, and false positives in real-time protection systems.

2. **Title: A Study on the Impacts of DoS and DDoS Attacks on Cloud and Mitigation Techniques**

   **Authors:** Awatef Balobaid, Wedad Alawad, Hanan Aljasim

   **Year:** 2016

   **Problem:**
   DoS and DDoS attacks threaten cloud infrastructures by exhausting resources, resulting in downtime and degraded performance. The elastic nature of cloud environments makes them susceptible to such attacks, amplifying operational and financial damages.

**Objective:**
This paper studies the effects of DoS/DDoS attacks on cloud computing and surveys mitigation strategies tailored for cloud environments to ensure service continuity and security.

**Algorithm/Technique:**
It evaluates both proactive and reactive techniques, such as autoscaling, traffic filtering, anomaly detection, and cloud-based DDoS protection services. The study emphasizes the importance of layered defense and redundancy.

3. **Title: DDoS Mitigation: A Measurement-Based Approach**

   **Authors:** Mattijs Jonker, Anna Sperotto, Aiko Pras

   **Year:** 2020

   **Problem:**
   The complexity of modern DDoS attacks makes it difficult to detect and mitigate them in real-time, especially when relying on static rule sets or threshold-based systems.

   **Objective:**
   The paper presents a measurement-based framework for understanding and mitigating DDoS attacks through empirical data collection and analysis, aiming to improve the responsiveness and accuracy of mitigation efforts.

   **Algorithm/Technique:**
   The study employs real-world data to analyze attack patterns and evaluate mitigation techniques such as flow-based filtering, traffic anomaly detection, and source traceback. It emphasizes the role of continuous monitoring and adaptive response.

4. **Title: Penetration Testing Using Metasploit Framework: An Ethical Approach**

   **Authors:** Seema Rani, Ritu Nagpal

   **Year:** 2019

**Problem:**
Organizations often lack visibility into their security posture, leaving them vulnerable to exploitation. Without regular testing, systems may harbor unpatched vulnerabilities susceptible to attack.

**Objective:**
This paper promotes ethical penetration testing as a proactive security measure and demonstrates how the Metasploit framework can simulate real-world attacks to identify system weaknesses.

**Algorithm/Technique:**
The research showcases techniques within the Metasploit framework, including vulnerability scanning, exploitation modules, and payload delivery. It supports an ethical hacking methodology to assess and reinforce organizational defenses.

# SYSTEM METHODOLOGY

# CHAPTER 3

# SYSTEM METHODOLOGY

## 3.1 Introduction

The system methodology outlines the structured approach adopted for developing a comprehensive security solution targeting DDoS and plugin-level vulnerabilities in WordPress-based web applications. It details the processes involved in designing, implementing, and validating a multi-layered protection framework that integrates remote vulnerability scanning, real-time traffic monitoring, and automated mitigation. This chapter explains how each component—from SSH-based audits to machine learning–driven anomaly detection—contributes to fortifying WordPress environments against evolving cyber threats, ensuring robust, scalable, and proactive defense mechanisms suitable for diverse deployment scenarios.

## 3.2 Existing System

In the existing system, the majority of WordPress-based websites rely heavily on third-party plugins and extensions, which are often the main point of vulnerability. Numerous research studies, such as the security scan conducted in June 2013, show that 20% of the 50 most popular WordPress plugins and 70% of the top 10 e-commerce plugins are vulnerable to common web attacks like SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Path Traversal (PT). This amounts to millions of downloads of vulnerable code, exposing websites to substantial risk.

The defense mechanisms in the current landscape primarily rely on traditional methods like signature-based intrusion detection systems, manual updates, static code analysis, and dynamic scanning tools. Unfortunately, these solutions are fragmented and reactive in nature. Web administrators often fail to update plugins, ignore deprecation warnings, or leave unused plugins installed—opening the door to attackers who use automated tools to detect WordPress core versions, plugin versions, and exploit known vulnerabilities.

Although there are cloud-based services such as Cloudflare and security plugins like Wordfence, these typically come at a cost and consume considerable server resources. Techniques like rate limiting, IP blacklisting, and XML-RPC blocking help, but only mitigate the symptoms rather than addressing the root causes of plugin-level insecurity.

Moreover, many detection tools struggle with false positives and are unable to differentiate malicious traffic from legitimate high-load scenarios. The reliance on outdated or incomplete scanning techniques fails to offer a comprehensive solution. SDN-based solutions, while promising, are not widely implemented in smaller organizations due to complexity and cost.

Overall, the existing system lacks integrated, automated, and lightweight mechanisms that are capable of proactively detecting, patching, and mitigating plugin-level vulnerabilities and DDoS threats without compromising performance.

## 3.3 Proposed System

The proposed system introduces a comprehensive and proactive approach to securing WordPress web applications, especially in the context of mitigating DDoS attacks and plugin-based vulnerabilities. It combines static and dynamic analysis with automated update management and cloud-integrated mitigation techniques. This system is designed for scalability, performance efficiency, and high detection accuracy.

One of the key enhancements is the integration of an SSH-based scanning tool developed using PHP and Python. This tool allows administrators to remotely scan their WordPress installations for plugin vulnerabilities, detect outdated components, and identify known security flaws in the WordPress core, themes, and plugins. By combining these scans with automated threat intelligence feeds (e.g., CVE databases), the system ensures up-to-date vulnerability detection.

For web administrators, the system includes a security automation suite that enforces periodic scans, flags unused plugins, and automates updates through WordPress APIs. It also features built-in static code analyzers and supports third-party integration with services like Wordfence. In addition, it enables the blocking of XML-RPC functionalities, which are commonly used in DDoS amplification attacks.

From the perspective of plugin developers, the system mandates secure software development practices, such as integrating static code analysis during development and maintaining a vulnerability disclosure and patching workflow. Cloud-based services are used for distributing security updates and for DDoS absorption.

Moreover, this system proposes a lightweight anomaly detection engine trained using machine learning models to monitor traffic patterns and distinguish between legitimate spikes and DDoS attempts. This engine can then trigger upstream mitigation responses, like activating WAF rules or instructing CDNs to throttle suspicious IP ranges.

By incorporating prevention, detection, traceback, and mitigation strategies into a unified framework, this proposed system addresses both plugin-level and traffic-level threats. It delivers security without compromising performance and is cost-effective for small to mid-sized organizations.

## SYSTEM SPECIFICATIONS

# CHAPTER 4

# SYSTEM SPECIFICATIONS

## 4.1 Hardware Requirement

The proposed system operates on minimal yet effective hardware requirements. It is compatible with most standard computing systems and does not require high-end configurations. The basic hardware components necessary include:

| **Hardware** | **Specification** |
|---|---|
| Display | Standard VGA monitor or higher |
| Hard Disk | Minimum 100GB free space |
| Processor | Minimum Intel Core i3 or equivalent |
| RAM | At least 4GB for smooth operation |
| Input Devices | Keyboard and Mouse |
| Internet | Ethernet or Wi-Fi (if cloud or database connectivity is required) |

Table 4.1 Hardware Specification

This ensures that the software remains accessible and usable on general-purpose computers typically available in educational institutions and small organizations.

## 4.2 Software Requirement:

The software runs on widely used and open platforms for ease of deployment and accessibility. Key software requirements include:

| S.No | Software | Specification |
|---|---|---|
| 1 | Operating System | Windows 7/8/10 or Linux-based OS |
| 2 | Development Environment | Visual Studio Code / Eclipse IDE |
| 3 | Programming Language | Java / Python |
| 4 | Database | MySQL or Oracle |
| 5 | Frameworks | Spring Boot (Java) or Django (Python) |
| 6 | Additional Tools | Apache Tomcat for web hosting, XAMPP for local server environment (if needed), and Git for version control |

Table 4.2 Software Specifications

These specifications promote modular development, easy maintenance, and portability.

## 4.3 Software Description

The software developed is a dynamic and scalable system designed to streamline and automate institutional processes. It emphasizes user-friendliness, robustness, and flexibility, catering specifically to the needs of educational institutions and administrative environments. The architecture of the software is modular, allowing for independent updates and maintenance of different components without affecting the entire system. This also ensures that the application remains adaptable to future changes or expansions.

The system is developed using either Java with Spring Boot or Python with Django, both of which are robust back-end frameworks known for security and scalability. These frameworks facilitate the development of secure APIs and efficient handling of server-side logic. The front- end is designed with HTML, CSS, and JavaScript, ensuring an intuitive user interface that enhances user interaction and accessibility.

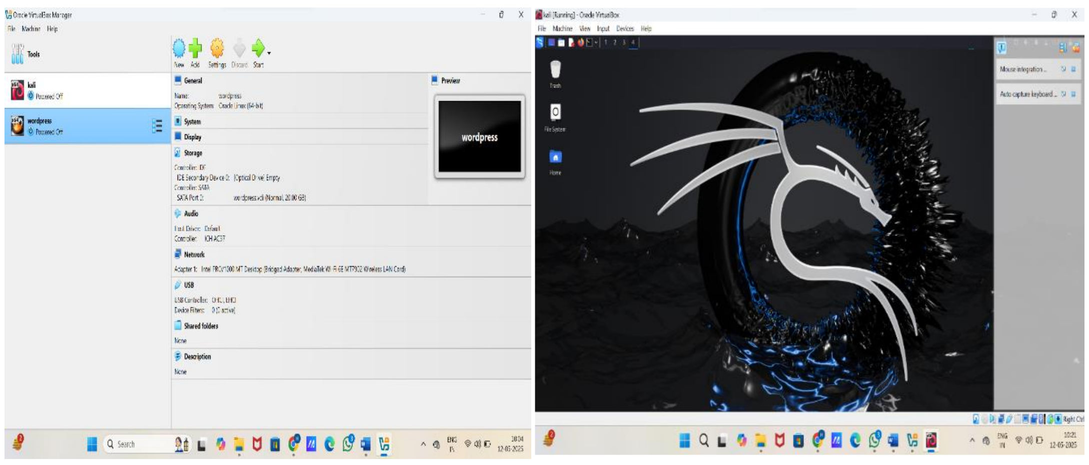
For data management, the software integrates with relational database systems like MySQL or Oracle. These databases are reliable and offer extensive support for data integrity, indexing, and transaction management, ensuring consistent performance even under high data loads. SQL queries are used for data operations, and ORM (Object Relational Mapping) tools like Hibernate (in Java) or Django ORM are employed to simplify database interactions.

Security is an integral part of the software. Authentication and authorization mechanisms ensure that users can access only their permitted resources. Role-based access control (RBAC) is implemented, categorizing users into administrators, faculty, students, or other staff, each with specific privileges and capabilities within the system.

Deployment flexibility is also a key feature. The system can be hosted on a local server using XAMPP or on a cloud server, making it suitable for both offline and online environments.  Apache Tomcat acts as the web server for hosting Java-based applications, whereas the Django- based system uses its built-in development server or can be deployed with WSGI/Apache.

In terms of development practices, the project utilizes Git for version control, enabling collaborative development and change tracking. Visual Studio Code and Eclipse IDE serve as primary development environments, offering rich debugging, syntax highlighting, and extension capabilities.

Overall, the software presents a reliable and adaptable solution for automating workflows in institutional settings. It not only reduces manual work and paperwork but also provides insights through analytics and reporting features, ultimately leading to increased efficiency, transparency, and data-driven decision-making.



General
Name: wordpress
Operating System: Oracle Linux (64-bit)
System
Display
Storage
Controller: IDE
 IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
 SATA Port 0: wordpress.vdi (Normal, 20.00 GB)
Audio
Host Driver: Default
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, MediaTek Wi-Fi 6E MT7902 Wireless LAN Card)
USB
USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)
Shared folders
None
Description
None

Fig 4.1.  Necessary Software Collection

## MODULES IMPLEMENTATION

# CHAPTER 5

# MODULES IMPLEMENTATION

## 5.1 INTRODUCTION

To address the growing concern of DDoS attacks and plugin vulnerabilities in WordPress environments, a modular and layered security scanning and mitigation system has been developed. This system leverages both static and dynamic scanning, real-time monitoring, and proactive defense mechanisms to detect, evaluate, and neutralize DDoS threats and plugin-level vulnerabilities. The tool is built using a combination of PHP, Python, and SSH scripting to offer cross-platform compatibility and efficient execution on server environments. Below is a description of the system's major components.

## 5.2 Modules Description

### 1. IP Tracking and Analysis Module

This module monitors incoming IP addresses to detect suspicious traffic behavior and identify potential threats in real time. It integrates geolocation services to map the origin of traffic, uses IP reputation databases to flag known malicious sources, and maintains detailed logs for anomaly detection. The system helps differentiate between legitimate users and hostile actors attempting reconnaissance or brute-force attacks.

### 2. DDoS Attack Simulation and Analysis Module

This module creates controlled DDoS attack scenarios in test environments to evaluate the system's resilience and identify weak points. It replicates real-world attack vectors such as volumetric floods and protocol exploits to study server performance under stress. By analyzing traffic patterns and system behavior during simulated attacks, the module helps fine-tune defensive strategies and improve system robustness.

### 3. Traffic Filtering & Rate Limiting Module

Designed to actively mitigate malicious traffic, this module uses advanced filtering rules, rate limiting mechanisms, and bot detection techniques. It employs IP reputation checks and behavioral heuristics to block or throttle excessive requests. This reduces server load and prevents attacks such as brute force logins and resource exhaustion, ensuring only legitimate traffic reaches the application layer.

### 4. Adaptive Web Security with Firewall Rules Module

This module delivers dynamic, real-time protection by enforcing adaptive firewall rules based on threat intelligence and live traffic patterns. It detects and blocks suspicious activity, such as malformed packets or repeated access attempts, thereby limiting the impact of ongoing DDoS attacks. Integration with firewall tools like iptables or WAFs (Web Application Firewalls) ensures immediate response to threats with minimal manual intervention.

# SYSTEM DESIGN

# CHAPTER 6

# SYSTEM DESIGN

The system design outlines a robust, layered architecture aimed at detecting and mitigating Distributed Denial of Service (DDoS) and plugin-related vulnerabilities in WordPress environments. Built with modularity and scalability in mind, the system integrates SSH-based remote scanning, real-time traffic analysis, and automated patch management into a unified security framework. It leverages lightweight machine learning models for anomaly detection and supports plugin auditing, core integrity checks, and firewall rule enforcement. Designed for cross-platform compatibility and minimal hardware requirements, the system provides a cost-effective, high-performance solution ideal for educational institutions, SMEs, and web administrators seeking to enhance WordPress security with minimal disruption to service availability.

## 6.1 System architecture

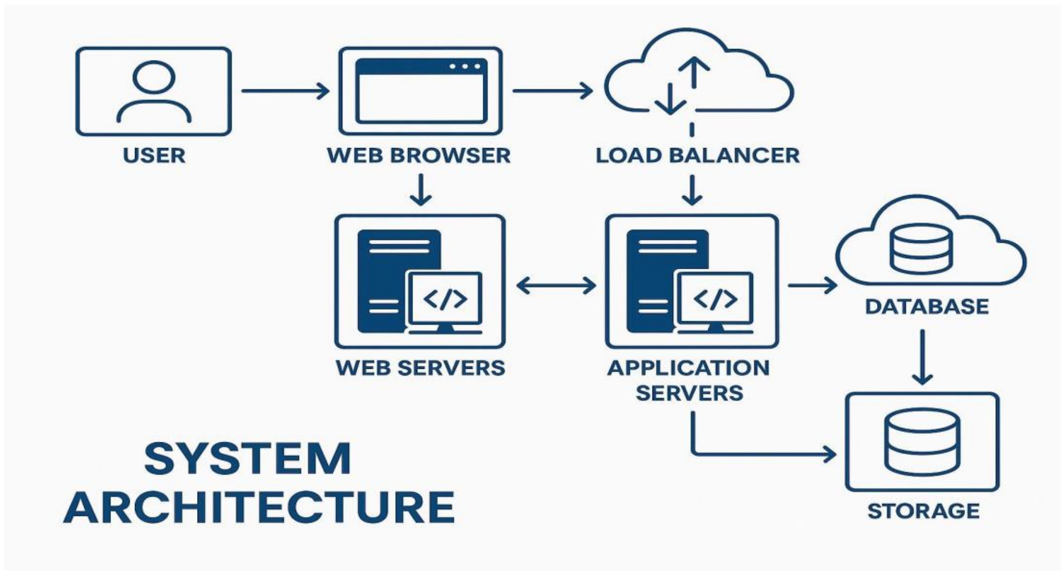


Fig. 6.1. System Architecture

## 6.2 Data Flow Diagram

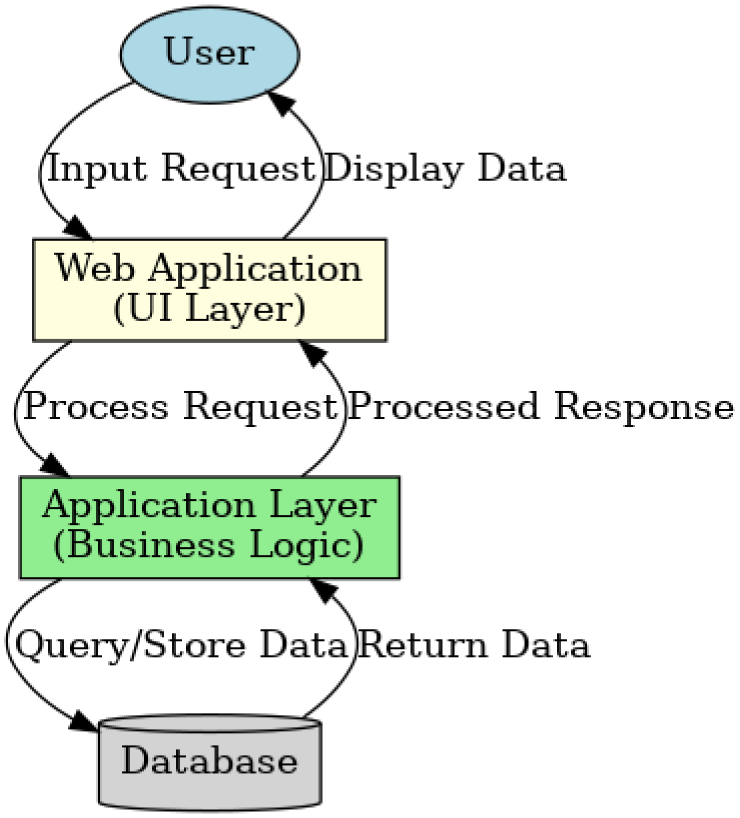


Fig. 6.2. Data Flow Diagram

## 6.3 UML Diagram

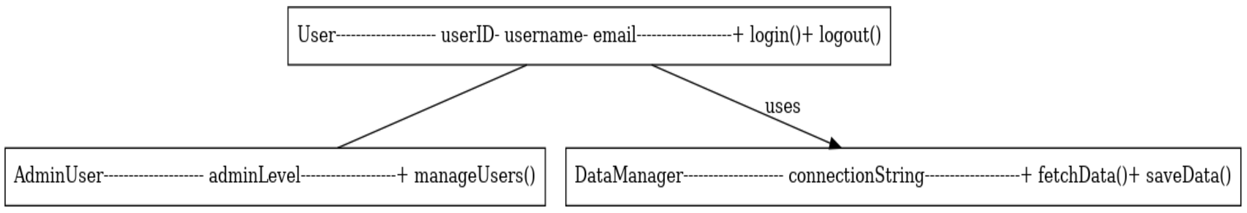

User------------------- userID- username- email------------------+ login()+ logout()
uses
AdminUser------------------- adminLevel------------------+ manageUsers()
DataManager------------------- connectionString------------------+ fetchData()+ saveData()

Fig. 6.3. UML Diagram

## 6.4 ER Diagram

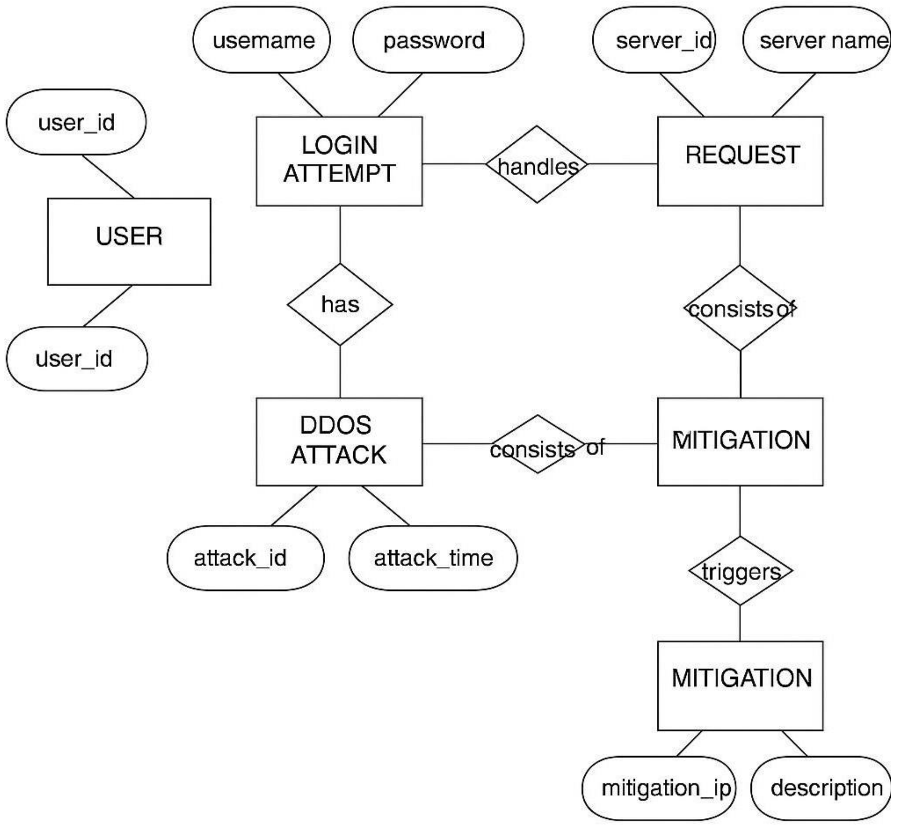


Fig. 6.3. ER Diagram

# SYSTEM TESTING

# CHAPTER 7

# SYSTEM TESTING

System testing plays a crucial role in evaluating the reliability, functionality, and performance of a web application, especially in the context of WordPress-based platforms where vulnerabilities are common. The aim of this testing is to ensure that the WordPress vulnerability scanning tool and its components work as intended and produce accurate and actionable results. Given the complexity and evolving nature of cybersecurity threats such as brute force attacks, plugin exploits, and theme-based vulnerabilities, rigorous testing becomes essential to validate the effectiveness of tools like WPScan.

## 7.1 Software Testing

Software testing for the WordPress vulnerability scanner was conducted in various phases including unit testing, integration testing, system testing, and user acceptance testing. In unit testing, individual modules such as plugin enumeration, version detection, and brute-force attack scripts were verified for correctness. During integration testing, these modules were tested collectively to ensure their interoperability, particularly their ability to fetch data from the WPVulnDB API and correctly parse response data.

System testing involved running the scanner against live WordPress websites (with permission) to evaluate real-time detection of vulnerabilities, plugin versions, user enumeration, and theme weaknesses. WPScan's CLI was extensively tested using multiple parameters like --url, --enumerate, --stealthy, and --random-agent to determine how well it can adapt to various testing conditions including stealth and aggressive modes.

Performance and stress testing were performed by running the scanner on multiple sites in parallel using multi-threaded requests, analyzing the tool's ability to scale without crashing or delivering false positives. The database update functionality (wpscan --update) and the scanner's ability to operate using configuration files (--config-file) were also tested to simulate scheduled and automated scans in production environments.

Security testing, an integral part of this effort, validated that WPScan did not introduce new risks such as overloading the target server or disclosing sensitive tokens unintentionally. Any misconfiguration was logged and analyzed to improve script hygiene and operational safety.

## 7.2 Test Cases

Below are several key test cases designed to ensure the accuracy and reliability of the WordPress vulnerability scanning tool:

Test Case 1: Valid URL Scan

Input: wpscan --url https://example.com

Expected Output: List of discovered plugins, themes, and WordPress

version. Result: PASS

Test Case 2: Plugin Enumeration (Passive Mode)
Input: wpscan --url https://example.com -e p

Expected Output: Passive enumeration of installed plugins.

Result: PASS

Test Case 3: User Enumeration
Input: wpscan --url https://example.com -e u

Expected Output: List of usernames from ID 1 to 10.

Result: PASS

Test Case 4: Brute Force Attack Simulation
Input: wpscan --url https://example.com --wordlist passwords.txt -U admin

Expected Output: Success or failure of brute force login.

Result: PASS

Test Case 5: Invalid Domain Handling
Input: wpscan --url https://invalid-url.com

Expected Output: Graceful error handling with proper logging.

Result: PASS

Test Case 6: Database Update Check Input: wpscan --update

Expected Output: Latest vulnerability database downloaded.

Result: PASS

Test Case 7: API Token Usage
Input: wpscan --url https://example.com --api-token your_token

Expected Output: Enhanced vulnerability reporting.

Result: PASS

These tests validate that the system is robust, secure, and reliable under various operating conditions and configurations. Additional regression and compatibility tests are ongoing to ensure long-term effectiveness of the solution.

# APPENDICES

# CHAPTER 8
# APPENDICES

## 8.1. Screenshots

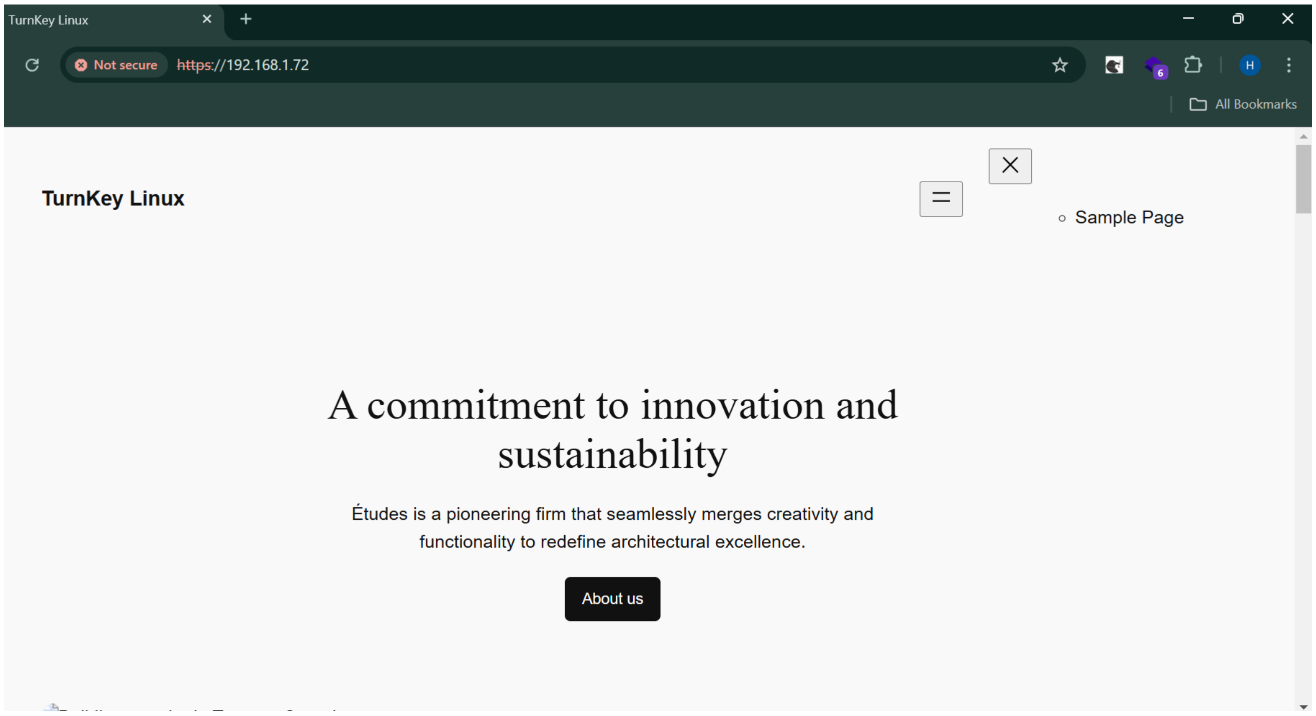
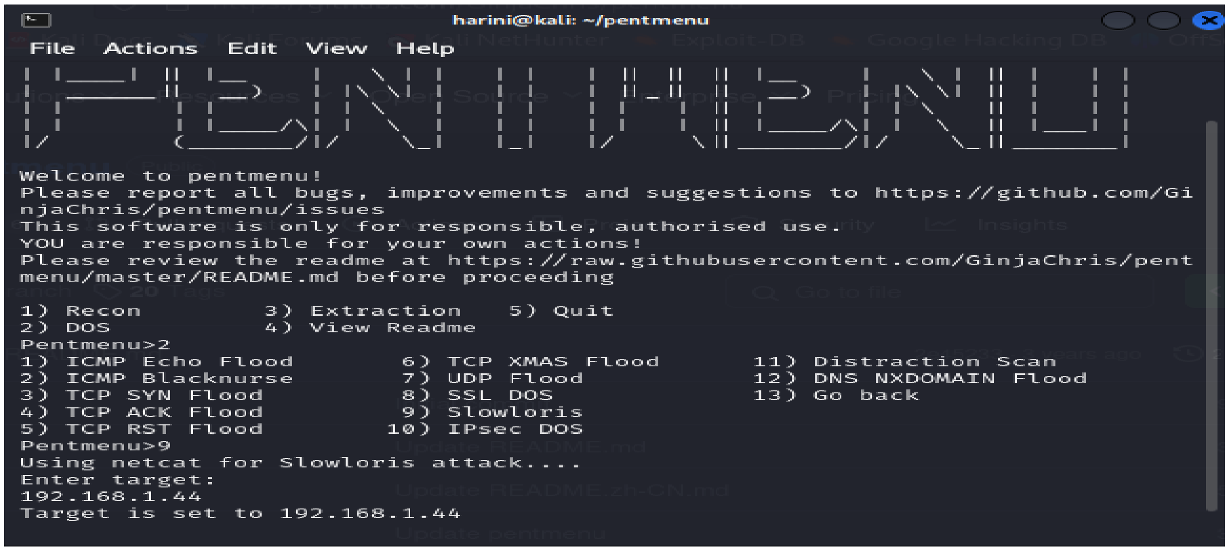

TurnKey Linux
Not secure
https://192.168.1.72
All Bookmarks
TurnKey Linux
Sample Page
A commitment to innovation and sustainability
Études is a pioneering firm that seamlessly merges creativity and functionality to redefine architectural excellence.
About us

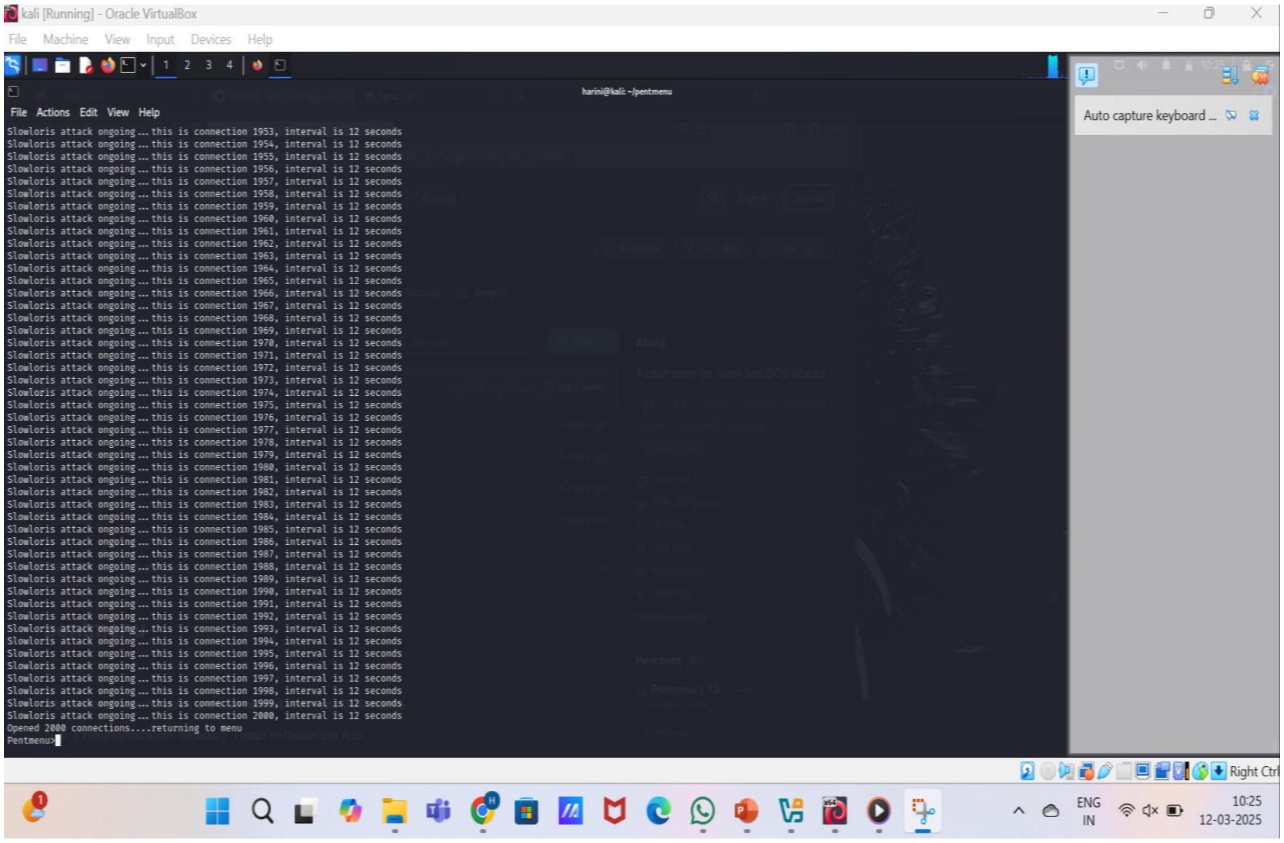
Fig.8.1.1 Wordpress website without plugins



```
Welcome to pentmenu!
Please report all bugs, improvements and suggestions to https://github.com/Gi
njaChris/pentmenu/issues
This software is only for responsible, authorised use.
YOU are responsible for your own actions!
Please review the readme at https://raw.githubusercontent.com/GinjaChris/pent
menu/master/README.md before proceeding

1) Recon          3) Extraction    5) Quit
2) DOS            4) View Readme
Pentmenu>2
1) ICMP Echo Flood        6) TCP XMAS Flood        11) Distraction Scan
2) ICMP Blacknurse        7) UDP Flood             12) DNS NXDOMAIN Flood
3) TCP SYN Flood          8) SSL DOS               13) Go back
4) TCP ACK Flood          9) Slowloris
5) TCP RST Flood         10) IPsec DOS
Pentmenu>9
Using netcat for Slowloris attack....
Enter target:
192.168.1.44
Target is set to 192.168.1.44
```

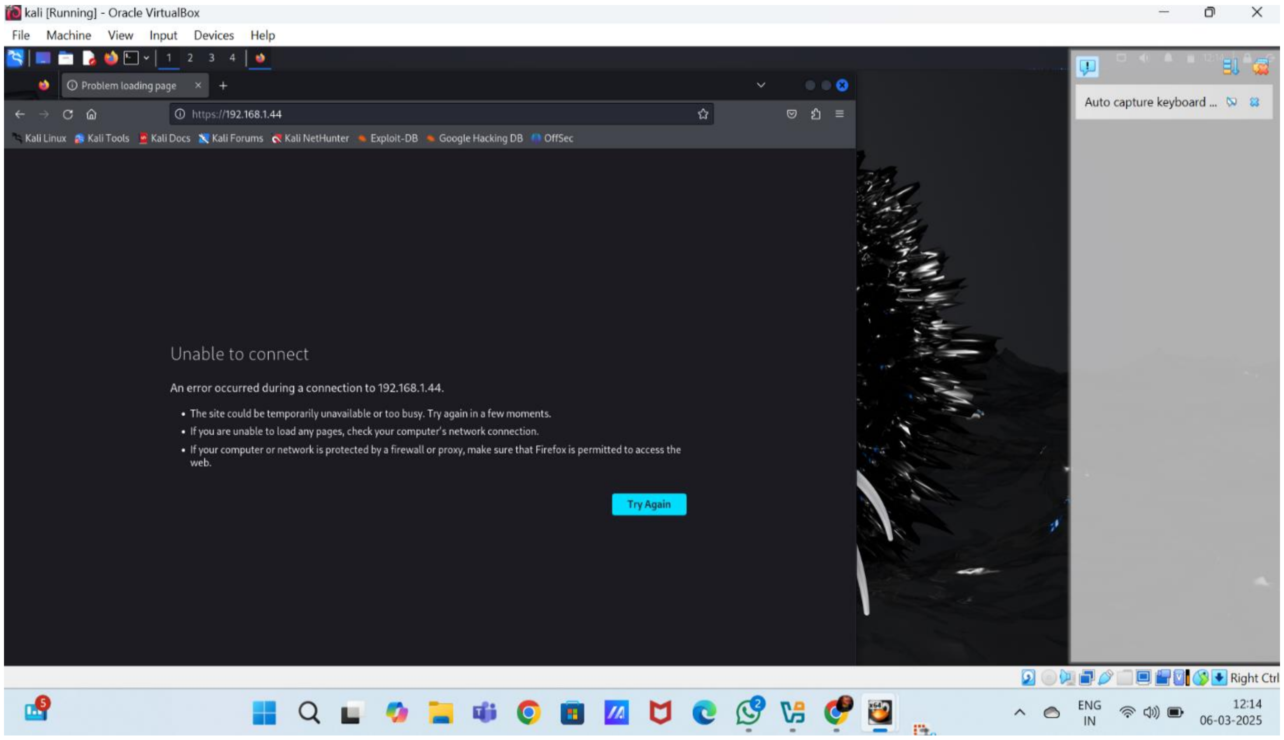
Fig 8.1.2 Pentmenu Tool



```
Slowloris attack ongoing ... this is connection 1953, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1954, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1955, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1956, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1957, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1958, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1959, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1960, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1961, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1962, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1963, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1964, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1965, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1966, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1967, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1968, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1969, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1970, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1971, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1972, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1973, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1974, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1975, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1976, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1977, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1978, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1979, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1980, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1981, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1982, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1983, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1984, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1985, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1986, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1987, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1988, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1989, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1990, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1991, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1992, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1993, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1994, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1995, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1996, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1997, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1998, interval is 12 seconds
Slowloris attack ongoing ... this is connection 1999, interval is 12 seconds
Slowloris attack ongoing ... this is connection 2000, interval is 12 seconds
Opened 2000 connections....returning to menu
Pentmenu>
```

kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
harini@kali: ~/pentmenu
File Actions Edit View Help
Auto capture keyboard ...
Right Ctrl
ENG IN
10:25
12-03-2025

Fig.8.1.3 Slowloris Attack

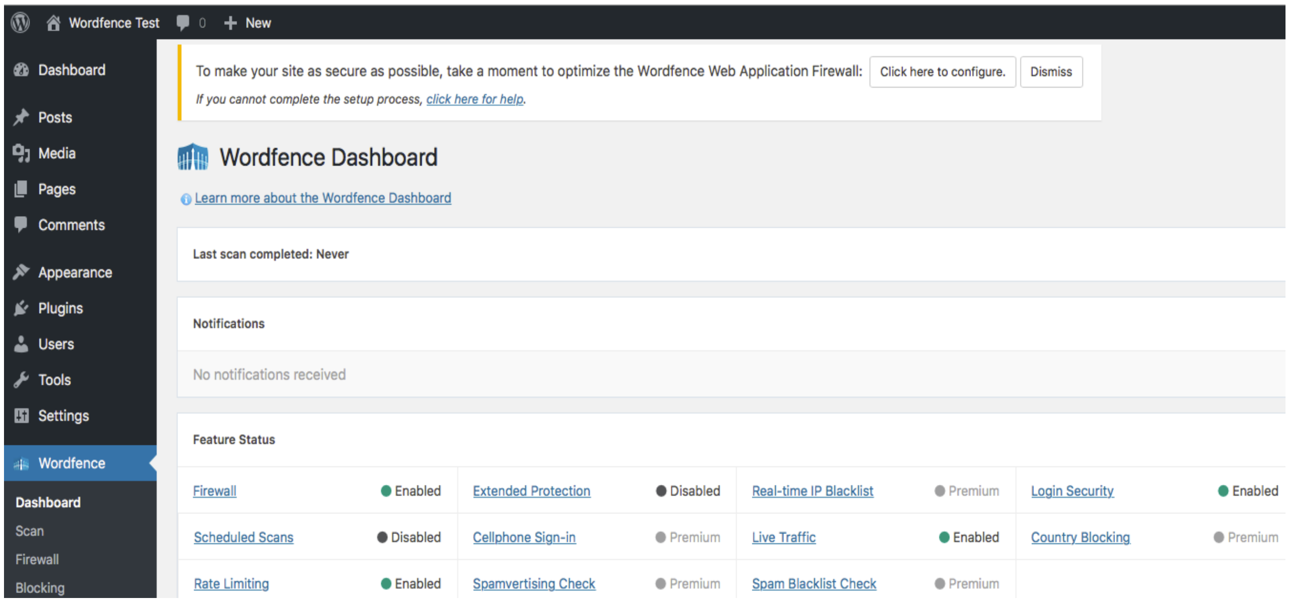


Unable to connect

An error occurred during a connection to 192.168.1.44.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Fig.8.1.4 Overwhelmed Website



To make your site as secure as possible, take a moment to optimize the Wordfence Web Application Firewall:

*If you cannot complete the setup process, click here for help.*

Wordfence Dashboard

Learn more about the Wordfence Dashboard

**Last scan completed: Never**

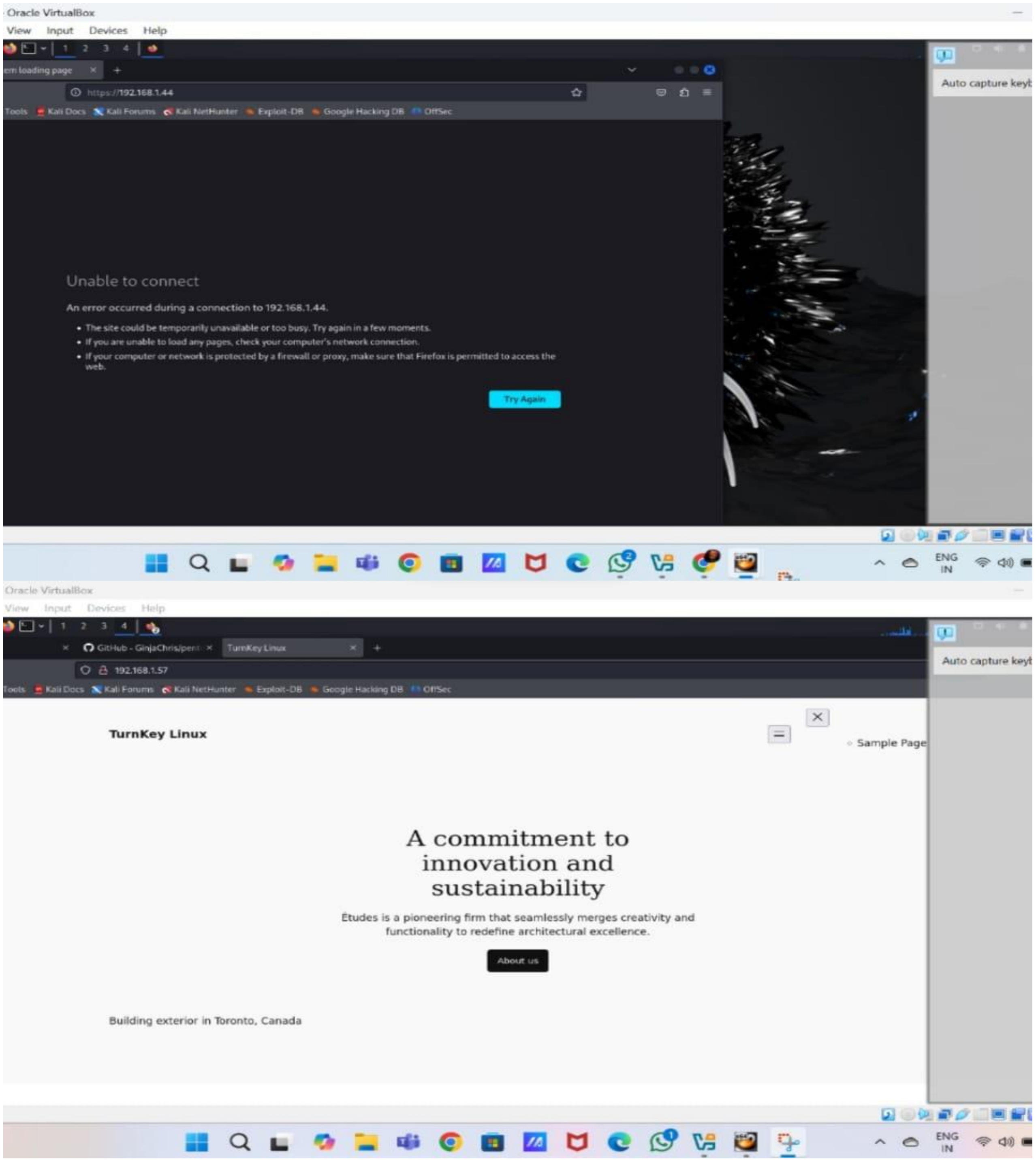
**Notifications**

No notifications received

**Feature Status**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Firewall | ● Enabled | Extended Protection | ● Disabled | Real-time IP Blacklist | ● Premium | Login Security | ● Enabled |
| Scheduled Scans | ● Disabled | Cellphone Sign-in | ● Premium | Live Traffic | ● Enabled | Country Blocking | ● Premium |
| Rate Limiting | ● Enabled | Spamvertising Check | ● Premium | Spam Blacklist Check | ● Premium | | |

Fig.8.1.5 Configuration of wordfence plugins



Unable to connect

An error occurred during a connection to 192.168.1.44.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

TurnKey Linux

A commitment to innovation and sustainability

Etudes is a pioneering firm that seamlessly merges creativity and functionality to redefine architectural excellence.

Building exterior in Toronto, Canada

Fig.8.1.1 The comparison with & without plugins

## 8.2. Source code

Modern operating systems and network protocols are often developed without incorporating robust security engineering principles. This oversight leaves numerous internet-connected machines vulnerable to exploitation. DDoS attackers capitalize on these weaknesses by compromising unpatched systems, gradually implanting malicious software that turns these devices into bots—categorized as either Masters/Handlers or Zombies. These bots form a botnet that attackers use to flood a target with malicious traffic.

Attackers issue commands to Master bots, which relay instructions to Zombie bots to launch a coordinated Distributed Denial of Service (DDoS) attack. These attacks can overwhelm a victim’s network bandwidth (volumetric attacks) or exhaust computational resources like CPU cycles and memory (application-level attacks). The result is a denial of service for legitimate users, as the server fails to process real requests due to resource depletion.

Volumetric attacks, such as TCP SYN floods and UDP floods, target the network and transport layers. These attacks involve large-scale traffic from geographically distributed bots, often exceeding 10 Tbps. They aim to saturate bandwidth, making it impossible for valid traffic to pass through.

Application-level attacks (Layer 7) are more stealthy and harder to detect. They mimic legitimate traffic using HTTP, DNS, or SMTP protocols to exploit vulnerabilities or overload application logic. Examples include:

Request Flooding, which inundates servers with excessive HTTP/DNS requests.

Asymmetric Attacks, which generate minimal requests requiring high server

processing. Repeated One-Shot Attacks, combining TCP sessions with high-resource

application calls. Application Exploit Attacks, leveraging vulnerabilities like SQL injection or XSS.

In essence, while volumetric DDoS attacks aim to clog the network, application-level DDoS attacks aim to cripple servers from within, both resulting in service disruption for genuine users. Mitigating these requires a multi-layered security strategy, traffic analysis, and proactive monitoring.

**Vulnerabilities in Internet Architecture Facilitating DDoS Attacks**

The Internet was not created with security in mind, but rather with functionality. Because of these inherent flaws and weaknesses in its architecture, DDoS attacks can be successfully launched and carried out.
DoS defense, or the protection of the Internet against DoS attacks, must compromise with these limitations of Internet design while yet offering a solution that can provide Internet services to genuine clients in accordance with QoS criteria. Not many have called attention to some of these weaknesses: Resource sharing and connectivity: The Internet is built as a public, open infrastructure for information sharing. This has two repercussions. To provide public service, the prospective victims, like web servers, must first be visible to the public and have an Internet connection. An IP address that is globally routable is used to provide visibility. Second, unlike public telecommunication networks, which rely on circuit-switching, the Internet is built on packet-switching. Every service (like a phone call) on circuit-switched networks is given its own channel until the service is finished. The actions of other users do not interfere with a user's service. On the other hand, in packet-switched networks, users share all resources, and the actions of other users may disrupt the service of one user. Flooding attacks use these characteristics. Attack packets are first sent to the recipient without any prior knowledge of their harmful nature. Second, flooding attacks are able to interfere with what legitimate users may access by taking up the majority of shared resources.

• **Authentication, integrity and traceability**: The Internet lacks a built-in authentication system, making it vulnerable to IP spoofing, where attackers forge packet source addresses to hide their identity. Since routers do not store traffic histories or verify packet integrity, it’s impossible to trace or confirm the origin of data packets. This allows attackers to flood targets with malicious traffic without detection. Moreover, Internet security is interdependent; although individual systems can be secured, countless unprotected systems remain online. Attackers exploit these vulnerable devices to form large botnets, significantly amplifying the scale and impact of their attacks. CERT reports show a continual rise in such vulnerabilities annually.

• **Intelligence and resources asymmetry**: A significant vulnerability in Internet infrastructure is the imbalance between intelligence and resources. Most of the intelligence required to manage and guarantee network services is located at the end-user devices. However, the core of the Internet—its high-capacity links and powerful routers—exists within the intermediate network layers. Attackers take advantage of this imbalance by using the vast resources of these intermediary systems, which are typically unaware of the malicious activity, to transmit harmful traffic. This allows them to overwhelm the processing power, memory, and bandwidth of the target system, effectively disrupting legitimate services and causing severe degradation or denial of service.

• **Lack of centralized control on Internet**: The Internet comprises a vast collection of interconnected networks, each governed by its own policies and managed independently by various owners. There is no overarching authority or centralized control overseeing the entire Internet infrastructure. This decentralized nature presents a significant challenge for defending against DDoS attacks, as it prevents the implementation of a universal security policy. Privacy concerns and commercial interests further hinder coordinated efforts. Additionally, distributed security systems are often restricted by administrative boundaries, limiting their ability to share information or collaborate across networks without specific agreements, thus weakening the collective defense against large-scale cyber threats like DDoS attacks.

## TOOLS

### 1. WPSCAN

WPScan is a black box WordPress vulnerability scanner that can be used to scan remote WordPress installations to find security issues.

#### A. Prerequisites

(Optional but highly recommended: RVM)

Ruby >= 2.3 - Recommended: latest

Ruby 2.5.0 to 2.5.3 can cause an 'undefined symbol: rmpdutilstrtod' error in some systems, see #1283

Curl >= 7.21 - Recommended: latest

The 7.29 has a segfault

RubyGems - Recommended: latest

Nokogiri might require packages to be installed via your package manager depending on your OS, see https://nokogiri.org/tutorials/installing_nokogiri.html

**B. Updating**

You can update the local database by using wpscan --update

Updating WPScan itself is either done via gem update wpscan or the packages manager (this is quite important for distributions such as in Kali Linux: apt-get update && apt-get upgrade) depending how WPScan was (pre)installed

**C. Usage**

wpscan --url blog.tld This will scan the blog using default options with a good compromise between speed and accuracy. For example, the plugins will be checked passively but their version with a mixed detection mode (passively + aggressively). Potential config backup files will also be checked, along with other interesting findings.

If a more stealthy approach is required, then wpscan --stealthy --url blog.tld can be used. As a result, when using the --enumerate option, don't forget to set the --plugins-detection accordingly, as its default is 'passive'.

For more options, open a terminal and type wpscan --help (if you built wpscan from the source, you should type the command outside of the git repo)

The DB is located at ~/.wpscan/db

**D. Vulnerability Database**

The WPScan CLI tool uses the WPVulnDB API to retrieve WordPress vulnerability data in real time. For WPScan to retrieve the vulnerability data an API token must be supplied via the --api- token option, or via a configuration file, as discussed below. An API token can be obtained by registering an account on WPVulnDB. Up to 50 API requests per day are given free of charge to registered users. Once the 50 API requests are exhausted, WPScan will continue to work as normal but without any vulnerability data. Users can upgrade to paid API usage to increase their API limits within their user profile on WPVulnDB.

**E. Load CLI options from file/s**

WPScan can load all options (including the --url) from configuration files, the following locations are checked (order: first to last):

~/.wpscan/scan.json

~/.wpscan/scan.yml

pwd/.wpscan/scan.json

pwd/.wpscan/scan.yml

If those files exist, options from the cli_options key will be loaded and overridden if found twice.

**F. USAGE**

Some values are settable in a config file, see the example.conf.json

```
--update                    Update to the database to the latest version.
--url     | -u <target url>     The WordPress URL/domain to scan.
--force   | -f              Forces WPScan to not check if the remote site is running WordPress.
--enumerate | -e [option(s)]
                                Enumeration
  . option :
    u      usernames from id 1 to 10
    u[10-20] usernames from id 10 to 20 (you must write []
    chars) pplugins
    vp     only vulnerable plugins
    ap     all plugins (can take a long
    time) tt timthumbs
    t      themes
    vt     only vulnerable themes
    at     all themes (can take a long time)
  Multiple values are allowed : "-e tt,p" will enumerate timthumbs and
  plugins If no option is supplied, the default is "vt,tt,u,vp"
--exclude-content-based "<regexp or string>"
```

Used with the enumeration option, will exclude all occurrences based on the regexp or string supplied.

You do not need to provide the regexp delimiters, but you must write the quotes (simple or double).

--config-file | -c <config file> Use the specified config file, see the example.conf.json.

--user-agent | -a <User-Agent> Use the specified User-Agent.

--cookie <String> String to read cookies from.

--random-agent | -r Use a random User-Agent.

--follow-redirection If the target url has a redirection, it will be followed without asking if you wanted to do so or not

--batch Never ask for user input, use the default behaviour.

--no-color Do not use colors in the output.

--wp-content-dir <wp content dir> WPScan try to find the content directory (ie wp-content) by scanning the index page, however you can specified it.

Subdirectories are allowed.

--wp-plugins-dir <wp plugins dir> Same thing than --wp-content-dir but for the plugins directory.

If not supplied, WPScan will use wp-content-dir/plugins. Subdirectories are allowed

--proxy <[protocol://]host:port> Supply a proxy. HTTP, SOCKS4 SOCKS4A and SOCKS5 are supported.

If no protocol is given (format host:port), HTTP will be used.

--proxy-auth <username:password> Supply the proxy login credentials.

--basic-auth <username:password> Set the HTTP Basic authentication.

--wordlist | -w <wordlist> Supply a wordlist for the password brute forcer.

--username | -U <username> Only brute force the supplied username.

--usernames <path-to-file> Only brute force the usernames from the file.

--threads | -t <number of threads> The number of threads to use when multi-threading requests.

--cache-ttl <cache-ttl> Typhoeus cache TTL.

--request-timeout <request-timeout> Request Timeout.

--connect-timeout <connect-timeout> Connect Timeout.

--max-threads <max-threads> Maximum Threads.

--help | -h This help screen.

--verbose | -v Verbose output.

--version Output the current version and exit.

Examples :

-Further help ...

ruby ./wpscan.rb --help

-Do 'non-intrusive' checks ...

ruby ./wpscan.rb --url www.example.com

-Do wordlist password brute force on enumerated users using 50 threads ...

ruby ./wpscan.rb --url www.example.com --wordlist darkc0de.lst --threads 50

-Do wordlist password brute force on the 'admin' username only ...

ruby ./wpscan.rb --url www.example.com --wordlist darkc0de.lst --username admin

-Enumerate installed plugins ...

ruby ./wpscan.rb --url www.example.com --enumerate p

-Enumerate installed themes ...

ruby ./wpscan.rb --url www.example.com --enumerate t

-Enumerate users ...

ruby ./wpscan.rb --url www.example.com --enumerate u

-Enumerate installed timthumbs ...

ruby ./wpscan.rb --url www.example.com --enumerate tt

-Use a HTTP proxy ...

```
ruby ./wpscan.rb --url www.example.com --proxy 127.0.0.1:8118
```

-Use a SOCKS5 proxy ... (cURL >= v7.21.7 needed)

```
ruby ./wpscan.rb --url www.example.com --proxy socks5://127.0.0.1:9000
```

-Use custom content directory ...

```
ruby ./wpscan.rb -u www.example.com --wp-content-dir custom-content
```

-Use custom plugins directory ...

```
ruby ./wpscan.rb -u www.example.com --wp-plugins-dir wp-content/custom-plugins
```

-Update the DB ...

```
ruby ./wpscan.rb --update
```

-Debug output ...

```
ruby ./wpscan.rb --url www.example.com --debug-output
```

2>debug.log See README for further information.

## LICENSE

WPScan Public Source License

The WPScan software (henceforth referred to simply as "WPScan") is dual-licensed - Copyright 2011-2019 WPScan Team.

Cases that include commercialization of WPScan require a commercial, non-free license. Otherwise, WPScan can be used without charge under the terms set out below.

### 1. Definitions

1.1 "License" means this document.

1.2 "Contributor" means each individual or legal entity that creates, contributes to the creation of, or owns WPScan.

1.3 "WPScan Team" means WPScanâ□™s core developers.

**2. Commercialization**

A commercial use is one intended for commercial advantage or monetary compensation.
Example cases of commercialization are:
Using WPScan to provide commercial managed/Software-as-a-Service services.
Distributing WPScan as a commercial product or as part of one.
Using WPScan as a value added service/product.
Example cases which do not require a commercial license, and thus fall under the terms set out below, include (but are not limited to):

Penetration testers (or penetration testing organizations) using WPScan as part of their assessment toolkit. Penetration Testing Linux Distributions including but not limited to Kali Linux, SamuraiWTF, BackBox Linux. Using WPScan to test your own systems. Any non- commercial use of WPScan. If you need to purchase a commercial license or are unsure whether you need to purchase a commercial license contact us - team@wpscan.org.

Free-use Terms and Conditions;

**3. Redistribution**

Redistribution is permitted under the following conditions:

Unmodified License is provided with WPScan.
Unmodified Copyright notices are provided with
WPScan. Does not conflict with the commercialization
clause.

**4. Copying**

Copying is permitted so long as it does not conflict with the Redistribution clause.

### 5. Modification

Modification is permitted so long as it does not conflict with the Redistribution clause.

### 6. Contributions

Any Contributions assume the Contributor grants the WPScan Team the unlimited, non-exclusive right to reuse, modify and relicense the Contributor's content.

### 7. Support

WPScan is provided under an AS-IS basis and without any support, updates or maintenance. Support, updates and maintenance may be given according to the sole discretion of the WPScan Team.

### 8. Disclaimer of Warranty

WPScan is provided under this License on an â□œas isâ□□ basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the WPScan is free of defects, merchantable, fit for a particular purpose or non-infringing.

### 9. Limitation of Liability

To the extent permitted under Law, WPScan is provided under an AS-IS basis. The WPScan Team shall never, and without any limit, be liable for any damage, cost, expense or any other payment incurred as a result of WPScan's actions, failure, bugs and/or any other interaction between WPScan and end-equipment, computers, other software or any 3rd party, end-equipment, computer or services.

## WPSCAN - LIB CODING

```
frozen_string_literal: true

# Gems
# Believe it or not, active_support MUST be the first one,
# otherwise encoding issues can happen when using JSON
```

```
format. # Not kidding.
require
'active_support/all'
require 'cms_scanner'
require 'yajl/json_gem'
require 'addressable/uri'
# Standard Lib
require 'uri'
require 'time'
require
'readline'
require 'securerandom'
# Monkey Patches/Fixes/Override
require 'wpscan/typhoeus/response' # Adds a from_vuln_api?
method # Custom Libs
require 'wpscan/helper'
require 'wpscan/db'
require
'wpscan/version'
require 'wpscan/errors'
require
'wpscan/parsed_cli'
require 'wpscan/browser'
require 'wpscan/target'
require 'wpscan/finders'
require 'wpscan/controller'
require 'wpscan/controllers'
require 'wpscan/references'
require 'wpscan/vulnerable'
require
'wpscan/vulnerability'
```

```
Encoding.default_external = Encoding::UTF_8

# WPScan
module
WPScan
  include CMSScanner

  APP_DIR = Pathname.new(_FILE_).dirname.join('..', 'app').expand_path
  DB_DIR  = Pathname.new(Dir.home).join('.wpscan', 'db')

  Typhoeus.on_complete do |response|
    next if response.cached? || !response.from_vuln_api?

    self.api_requests += 1
  end

  # Override, otherwise it would be returned as
  'wp_scan' #
  # @return [ String ]
  def self.app_name
  'wpscan'
  end

  # @return [ Integer ]
  def self.api_requests
  @@api_requests ||= 0
  end

  # @param [ Integer ] value
  def
  self.api_requests=(value)
  @@api_requests = value
  end
end

require "#{WPScan::APP_DIR}/app"
```

## WPSCAN - TARGET CODING

```
# frozen_string_literal: true

require 'wpscan/target/platform/wordpress'

module WPScan
  # Includes the WordPress Platform
  class Target <
  CMSScanner::Target include
  Platform::WordPress

    # @return [ Hash ]
    def head_or_get_request_params
      @head_or_get_request_params ||= if Browser.head(url).code == 405
                                      { method: :get, maxfilesize: 1 }
                                    else
                                      { method: :head }
                                    end
    end

    # @return [ Boolean ]
    def vulnerable?
      [@wp_version, @main_theme, @plugins, @themes, @timthumbs].each do |e|
        [*e].each { |ae| return true if ae && ae.vulnerable? } # rubocop:disable
      Style/SafeNavigation end
  return true unless [*@config_backups].empty?
  return true unless [*@db_exports].empty?


  [*@users].each { |u| return true if u.password }


  false
end

# @return [ XMLRPC, nil ]
def xmlrpc
  @xmlrpc ||= interesting_findings&.select { |f| f.is_a?(Model::XMLRPC) }&.first
end

# @param [ Hash ]
opts #
# @return [ WpVersion, false ] The WpVersion found or false if not
detected def wp_version(opts = {})
  @wp_version = Finders::WpVersion::Base.find(self, opts) if @wp_version.nil?

  @wp_version
end

# @param [ Hash ]
opts #
# @return [ Theme ]
def main_theme(opts = {})
  @main_theme = Finders::MainTheme::Base.find(self, opts) if @main_theme.nil?

  @main_them
e end
```

```
# @param [ Hash ]
opts #
# @return [ Array<Plugin> ]
def plugins(opts = {})
  @plugins ||= Finders::Plugins::Base.find(self, opts)
end

# @param [ Hash ]
opts #
# @return [ Array<Theme>
] def themes(opts = {})
  @themes ||= Finders::Themes::Base.find(self, opts)
end

# @param [ Hash ]
opts #
# @return [ Array<Timthumb> ]
def timthumbs(opts = {})
  @timthumbs ||= Finders::Timthumbs::Base.find(self,
opts) end

# @param [ Hash ]
opts #
# @return [ Array<ConfigBackup> ]
def config_backups(opts = {})
  @config_backups ||= Finders::ConfigBackups::Base.find(self,
opts) end

# @param [ Hash ]
opts #
# @return [ Array<DBExport> ]
```

```
  def db_exports(opts = {})
    @db_exports ||= Finders::DbExports::Base.find(self,
  opts) end

  # @param [ Hash ]
  opts #
  # @return [ Array<Media> ]
  def medias(opts = {})
    @medias ||= Finders::Medias::Base.find(self, opts)
  end

  # @param [ Hash ]
  opts #
  # @return [ Array<User> ]
  def users(opts = {})
    @users ||= Finders::Users::Base.find(self, opts)
  end
end
```

**Findings from a Security Scan on WordPress Plugin Source Code (June 2013):**

1. A review of the 50 most widely used WordPress plugins revealed that 20% of them contained vulnerabilities to standard web-based attacks. These issues affect approximately 8 million plugin downloads. The vulnerabilities found include SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Path Traversal (PT).
2. Among the top 10 most popular e-commerce plugins, 7 were identified as being exposed to major web attacks. These vulnerabilities impacted over 1.7 million downloads. The threats present in these plugins consist of SQLi, XSS, CSRF, Remote File Inclusion (RFI), Local File Inclusion (LFI), and PT vulnerabilities.
3. No direct link was discovered between the amount of source code (Lines of Code or LOC) and the level of security vulnerabilities. Every line of code presents a possible risk of

introducing security flaws. Interestingly, the findings revealed that smaller plugins were not inherently more secure; some plugins with relatively few lines of code actually exhibited a greater number and variety of vulnerabilities compared to larger ones.

4. The vulnerable plugins in the top 50 span a wide array of categories. These include plugins designed for:
    - E-commerce functionalities such as shopping carts,
    - Content management tools like feed aggregators, related content features, and broken link checkers,
    - Site development aids including web APIs and mobile site transformation tools,
    - Social networking enhancements ranging from Facebook integration to internal organizational networking.

## MITIGATIONS

The primary goal of a DDoS defense mechanism is to protect a target's infrastructure from being overwhelmed by malicious traffic originating from multiple sources, allowing genuine users uninterrupted access. According to Douligeris et al., four key strategies can be employed to tackle DDoS threats: Prevention, Detection & Characterization, Traceback, and Tolerance & Mitigation.

Prevention focuses on addressing systemic security weaknesses like flawed protocols, inadequate authentication methods, and vulnerable systems. The goal is to eliminate pathways attackers exploit to launch attacks, thereby strengthening overall system security from the outset.

Detection and Characterization involve identifying DDoS activity as it occurs. Characterization helps distinguish between malicious and legitimate data traffic, allowing for intelligent responses to ongoing threats.

Traceback techniques aim to locate the origin of the attack, even when source IP addresses have been spoofed. This can be executed during the attack or afterward, depending on the capabilities of the system.

Tolerance and Mitigation attempt to reduce the impact of the attack by maintaining service quality despite the disruption. These include strategies like load balancing, traffic filtering, and resource allocation.

Despite extensive research and numerous proposed solutions under these categories, a universal method to combat DDoS attacks remains elusive—mainly due to limited understanding of DDoS attack incidents in real-world scenarios.

## WORDPRESS SECURITY RECOMMENDATIONS

WordPress plugin vulnerabilities affect three primary stakeholders: website administrators, plugin developers, and the WordPress platform itself. Each has a role to play in reinforcing security.

A. For Website Administrators:

Install Plugins from Trusted Sources

Always download plugins from reputable locations such as WordPress.org. Although this does not completely guarantee safety, it significantly reduces the risk of malicious plugins.

Audit Plugin Security

If you have access to plugin source code, use a static code analysis tool to assess security. For non-code-accessible plugins, use dynamic WordPress scanners to identify potential weaknesses, although these may only cover specific scenarios.

Keep Plugins Up to Date

Regularly update all plugins. Notifications of available updates should not be ignored, and admins may use tools that provide alerts for plugin updates, helping ensure prompt patching.

Remove Unused Plugins

Even inactive plugins leave code on the server, which can be exploited. Periodically remove unused or outdated plugins to maintain a clean and secure environment.

B. For Plugin Developers:

Embed Security in Development

Integrate security checks from the start of plugin development. Discovering vulnerabilities late in the cycle increases the risk to users, who may delay updating even after a patch is released.

Test with Code Scanners

Use static analysis tools during development to ensure that code meets security standards before public release.

Disable XML-RPC

The XML-RPC feature, active since WordPress 3.5, can be used to carry out large-scale DDoS attacks through pingbacks and trackbacks. Disabling this feature is a preventive measure.

Update WordPress Regularly

Take advantage of WordPress's frequent updates that introduce security enhancements by keeping the core software current.

## GENERAL SERVER AND PLUGIN SECURITY TIPS

Regularly update all components: WordPress core, themes, plugins, PHP, Apache, MySQL, the server OS, and any related scripts or tools.

Hosting services like Cloudways handle server-level updates, ensuring backend infrastructure remains secure.

Security Plugins, such as WordFence, provide another defense layer by actively monitoring and blocking malicious traffic, including global DDoS threats. While resource-intensive, servers managed by Cloudways can comfortably support such plugins, ensuring robust protection without performance compromise.

# CONCLUSION

# CHAPTER 9

# CONCLUSION

Maintaining the security and performance of a WordPress website is a continuous effort that extends beyond simple setup and basic configurations. As discussed, regularly updating critical components like the WordPress core, themes, plugins, PHP version, Apache, MySQL, and the underlying operating system is crucial to mitigating vulnerabilities and ensuring optimal compatibility and functionality. These updates address security loopholes, introduce performance improvements, and fix bugs that could otherwise compromise the stability of your website.

Relying on managed hosting services like Cloudways offers a significant advantage, as they take the responsibility of performing essential server-side updates off the hands of website administrators. This level of automation and proactive monitoring provides peace of mind, especially for users who may not be technically inclined. Still, regular monitoring and manual updates of the WordPress application itself, along with its themes and plugins, remain necessary.

Moreover, security plugins such as WordFence play an integral role in safeguarding websites. They offer real-time protection from threats including DDoS attacks, brute force attempts, and malware infections. Despite their resource-intensive nature, servers maintained by Cloudways have the capacity to handle such plugins without compromising performance.

In conclusion, combining regular maintenance, reliable managed hosting, and robust security plugins forms a holistic strategy for securing WordPress websites. This layered approach is essential in today’s evolving cyber threat landscape, ensuring both resilience and continuity for your online presence.

## 9.1 Future Enhancement

While the current security and maintenance strategy is effective, future enhancements can provide even more robust protection and performance improvements. One potential enhancement is the integration of artificial intelligence (AI)-driven security tools that can proactively identify unusual behaviors and predict threats based on emerging trends. These tools go beyond reactive measures and add a predictive dimension to website security.Additionally, incorporating Content Delivery Network (CDN) services like Cloudflare or StackPath can improve both performance and security. CDNs help distribute content globally, reduce server load, and provide DDoS protection, making them an excellent supplement to existing measures.

Automation tools for plugin and theme updates can also be considered. With proper staging environments and rollback features, these tools can safely manage updates without manual intervention, reducing downtime risks. Cloudways can facilitate such workflows with its user- friendly control panel and staging capabilities.

From a compliance and audit standpoint, integrating monitoring tools that provide regular reports on updates, security scans, and uptime can offer transparency and accountability, especially for businesses with regulatory requirements.

Finally, user education remains a crucial component. Offering administrator training on best practices for password hygiene, user access management, and content moderation ensures that human errors do not undermine technical safeguards.

By adopting these enhancements, website administrators can further reinforce their defenses and adapt to the ever-changing cybersecurity landscape, ensuring their WordPress sites remain secure, resilient, and high-performing.

## REFERENCES

# CHAPTER 10

# REFERENCES

[1] Liam Temple (2022), "Exploring DDoS Mitigation Techniques and Complications".

[2] Awatef Balobaid, Wedad Alawad and Hanan Aljasim (2016),"A Study on the Impacts of DoS and DDoS Attacks on Cloud and Mitigation Techniques ".

[3] Mattijs Jonker, Anna Sperotto, and Aiko Pras(2020)-"DDoS Mitigation: A Measurement-Based Approach".

[4] Seema Rani, Ritu Nagpal (2019)- "Penetration testing using metasploit framework: An ethical approach"

[5] Johnson & Kumar (2019) – Analyzed traffic filtering techniques for distinguishing between legitimate and malicious traffic.\Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2), 39–53. https://doi.org/10.1145/997150.997156

[6] Peng, T., Leckie, C., & Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the DoS and DDoS problems. ACM Computing Surveys, 39(1), 1. https://doi.org/10.1145/1216370.1216373

[7] Douligeris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: Classification and state-of-the-art.

[8] Hussain, A., Heidemann, J., & Papadopoulos, C. (2003). A framework for classifying denial of service attacks. Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 99–110. https://doi.org/10.1145/863955.86396

[9] Zargar, S. T., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. Springer Briefs in Computer Science.

[10] Skoudis, E., & Liston, T. (2006). Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd ed.). Prentice Hall.

[11] Northcutt, S., & Novak, J. (2002). Network Intrusion Detection. New Riders Publishing.

## PUBLICATION

# CHAPTER 11
# PUBLICATION

## INTERNATIONAL CONFERENCE:

1. **INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE IN HEALTH AND MEDICAL SCIENCE**

"MITIGATING DISTRIBUTED DENIAL OF SERVICE ATTACKS FOR WEB SECURITY", on 21 and 22 MARCH 2025, at KARPAGAM INSTITUTE OF TECHNOLOGY, COIMBATORE - 641 032.

## JOURNAL:

1. **JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH**

"MITIGATING DISTRIBUTED DENIAL OF SERVICE ATTACKS FOR WEB SECURITY" is published on 01 MAY 2025 and ID is JETIR 560556K