**FILE INCLUSION ATTACKS**

Exploiting Vulnerabilities and Implementing Defenses

**(Web Application Security)**

**CDAC, Noida**

**CYBER GYAN VIRTUAL INTERNSHIP PROGRAM**

**Submitted By:**

VYSHNAV S

Project Trainee, (May-June) 2024

**BONAFIDE CERTIFICATE**

This is to certify that this project report entitled File Inclusion Attacks submitted to CDAC Noida, is a Bonafede record of work done by Vyshnav s under my supervision from May to June 2024.

**Declaration by Author(s)**

This is to declare that this report has been written by me/us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be plagiarized, I/we are shall take full responsibility for it.

Name of Author:Vyshnav S

**TABLE OF CONTENTS**

**1.Introduction**................................................... 1

**2.Acknowledgement** ................................................... 6

**3.File Inclusion Attacks** .............................................. 7

3.1 **Problem Statement** ......................................... 7

3.2 **Learning Objectives** ........................................ 7

3.3 **Approach** .................................................... 8

**4.Implementation** ................................................... 9

4.1 **Infrastructure Setup** ...................................... 9

4.1.1 **Virtual Environment** ................................... 10

4.1.2 **Tools Used** .............................................. 10

4.2 **Identification of Vulnerabilities** .......................... 11

4.2.1 **Automated Scanning with OWASP ZAP** ........... 11

4.2.2 **Manual Testing with Burp Suite** ..................... 11

4.3 **Exploitation of Vulnerabilities** ............................ 11

4.3.1 **Local File Inclusion (LFI)** .............................. 11

4.3.2 **Remote File Inclusion (RFI)** .......................... 13

4.4 **Implementation of Defensive Measures** .............. 15

4.4.1 **Input Validation** ........................................ 15

4.4.2 **Secure Coding Practices** ............................... 15

4.4.3 **Web Application Firewall (WAF) Configuration** .. 15

**5.Conclusion and Recommendations** ........................ 16

5.1 **Conclusion** .................................................. 16

5.2 **Recommendations** ......................................... 16

**6.List of References** ................................................ 18

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to everyone who has contributed to the successful completion of this project on "File Inclusion Attacks: Exploiting Vulnerabilities and Implementing Defenses."

First and foremost, I am deeply grateful to my project supervisor, Mr. Prateek Saraswat, for their invaluable guidance, support, and encouragement throughout the duration of this project. Their expertise and insights have been instrumental in shaping the direction and outcomes of this research.

I also wish to thank CDAC, Noida for providing the resources and environment necessary for conducting this study. The access to necessary tools and infrastructure has been crucial for the practical implementation and testing phases of this project.

I extend my heartfelt thanks to the broader web security community, including organizations like OWASP, whose resources, tools, and guidelines have been invaluable for both the research and implementation phases of this project.

Last but not least, I am deeply appreciative of my family and friends for their unwavering support and encouragement, which has been a constant source of strength and inspiration.

Thank you all for your support and contributions.

Sincerely,

Vyshnav s

24/07/24

CDAC, Noida

**FILE INCLUSION ATTACKS**

(Exploiting Vulnerabilities and Implementing Defenses)

**3.1 PROBLEM STATEMENT:**

File inclusion vulnerabilities pose significant threats to web applications, allowing attackers to manipulate file inclusion mechanisms like Local File Inclusion (LFI) and Remote File Inclusion (RFI) to execute unauthorized code or access sensitive information. Despite advancements in web security, these vulnerabilities persist due to inadequate input validation and insecure coding practices. This project aims to investigate and understand file inclusion vulnerabilities in-depth, analyze their exploitation techniques, and develop effective defensive strategies to mitigate these risks effectively.

**3.2 Learning Objectives:**

**Investigate File Inclusion Vulnerabilities:**

* Research and analyze different types of file inclusion vulnerabilities, including LFI and RFI, to understand their mechanics and potential impact on web applications.

**Explore Exploitation Techniques:**

* Delve into various methods and techniques used by attackers to exploit file inclusion vulnerabilities, demonstrating their exploitation through practical scenarios and examples.

**Develop Defensive Strategies:**

* Propose and implement defensive measures to mitigate file inclusion vulnerabilities, such as input validation techniques, secure coding practices, and the use of web application firewalls (WAFs).

**Evaluate and Test Defenses:**

* Evaluate the effectiveness of implemented defensive strategies through testing and validation against simulated file inclusion attacks.

**Document and Disseminate Findings:**

* Compile a detailed report documenting research findings, analysis of exploitation techniques, and effectiveness of defensive strategies.
* Provide recommendations and best practices for developers and security professionals to enhance web application security against file inclusion vulnerabilities.

**3.3APPROACH:**

#### Tools and Technologies Used

* **OWASP ZAP: Web application security scanner for detecting file inclusion vulnerabilities.**
* **Burp Suite:** Comprehensive platform for security testing, used for intercepting and modifying HTTP requests.
* **Curl:** Command-line tool for sending HTTP requests to test LFI and RFI exploits.
* **Netcat:** Networking utility for setting up listeners for capturing data from exploited servers.
* **Metasploit Framework:** Penetration testing framework with modules for LFI and RFI exploitation.
* **DirBuster/Dirsearch:** Tools for discovering potentially exploitable directories and files.
* **Custom Scripts:** Python, Perl, or Bash scripts for automating exploitation of file inclusion vulnerabilities.

#### Infrastructure Created

* **Local Development Server:** Hosts a web application with intentional file inclusion vulnerabilities.
* **Testing Machine:** Equipped with OWASP ZAP, Burp Suite, and other tools for conducting vulnerability scans and exploit attempts.
* **Web Application Firewall (WAF):** Monitors and filters HTTP traffic to prevent file inclusion attacks.

**Network Diagram**

Internet

|

[ Firewall ]

|

[ Router ]

|

------------------------+--------------------------

| | |

[ Testing Machine ] [ Local Server ] [ Web Application ]

(OWASP ZAP, Burp Suite) (Vulnerable Web App) (Web Application)

|

[ Web Application Firewall (WAF) ]

|

[ Database Server ]

1. **IMPLEMENTATION:**

The implementation phase of the project involves setting up the necessary infrastructure, conducting vulnerability assessments, exploiting identified vulnerabilities, and applying defensive measures to mitigate file inclusion attacks. Here is a detailed breakdown of the implementation process:

#### 4.1 Infrastructure Setup

The infrastructure setup involves creating a controlled environment for testing and demonstrating file inclusion vulnerabilities.

**4.1.1 Virtual Environment**

* **Tools Used:** Oracle VirtualBox
* **Setup:** Two virtual machines were created:
  + **Kali Linux VM:** Used as the attack machine equipped with OWASP ZAP, Burp Suite, Curl, Netcat, and Metasploit Framework.
  + **OWASP WebGoat VM:** Hosted a vulnerable web application for testing purposes.

**Steps to Install VirtualBox:**

1. **Download and Install VirtualBox:** Go to the VirtualBox download page, download the appropriate version for your OS, and run the installer.
2. **Launch VirtualBox:** Open VirtualBox from your applications or start menu.

**Steps to Install Kali Linux:**

1. **Download Kali Linux ISO:** Go to the Kali Linux download page and download the ISO file.
2. **Create a New VM:** In VirtualBox, click New, name the VM "Kali Linux," select Linux and Debian (64-bit), allocate at least 2 GB RAM, create a virtual hard disk (at least 20 GB).
3. **Attach the ISO:** Go to Settings > Storage, select the empty optical drive, choose the downloaded ISO file.
4. **Install Kali Linux:** Start the VM, follow the on-screen instructions to complete the installation.

**Steps to Install OWASP WebGoat Using OWASP Virtual Box Setup:**

1. **Download the OWASP Broken Web Applications VM:**
   * Go to the OWASP Broken Web Applications Project page.
   * Download the VM image.
2. **Import the VM into VirtualBox:**
   * Open VirtualBox, go to File > Import Appliance.
   * Select the downloaded OWASP VM image file and follow the prompts to import the appliance.
3. **Start the OWASP VM:**
   * Select the imported OWASP VM and click Start.
   * The VM will boot up, providing access to various vulnerable web applications including WebGoat.

#### 4.2 Identification of Vulnerabilities

The identification phase involves scanning the web application to discover potential Local File Inclusion (LFI) and Remote File Inclusion (RFI) vulnerabilities.

**4.2.1 Automated Scanning with OWASP ZAP**

* **Setup:** OWASP ZAP was configured on the Kali Linux VM to scan the WebGoat application.
* **Execution:** An active scan was performed to identify potential file inclusion vulnerabilities.
* **Results:** The scan results were analyzed to pinpoint specific endpoints vulnerable to LFI and RFI.

**4.2.2 Manual Testing with Burp Suite**

* **Proxy Setup:** Burp Suite was configured on the Kali Linux VM to intercept HTTP requests.
* **Testing:** Manual testing was conducted by modifying intercepted requests to include payloads aimed at exploiting LFI and RFI vulnerabilities.
* **Verification:** The responses were analyzed to confirm the presence of vulnerabilities.

#### 4.3 Exploitation of Vulnerabilities

This phase involves exploiting the identified vulnerabilities to demonstrate their potential impact.

**4.3.1 Local File Inclusion (LFI)**

* **Payloads:** Custom payloads were crafted to include local files such as /etc/passwd on Unix-based systems.
* **Execution:** The payloads were sent using Curl and Burp Suite from the Kali Linux VM.
* **Results:** Successful exploitation was documented with screenshots and detailed descriptions of the steps taken.

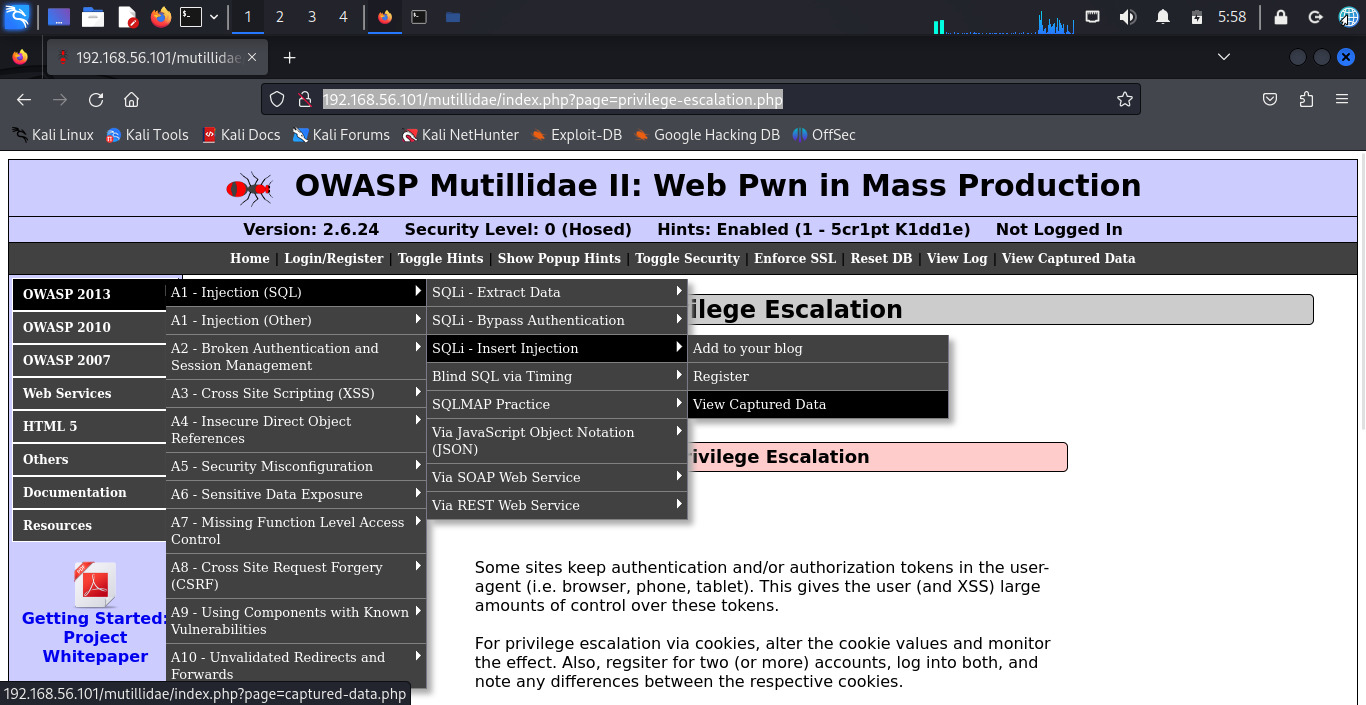
**To find a LFI vulnerable web page:**

Look for URLs with parameters that might include file paths. These are often found in query strings or path segments, such as:

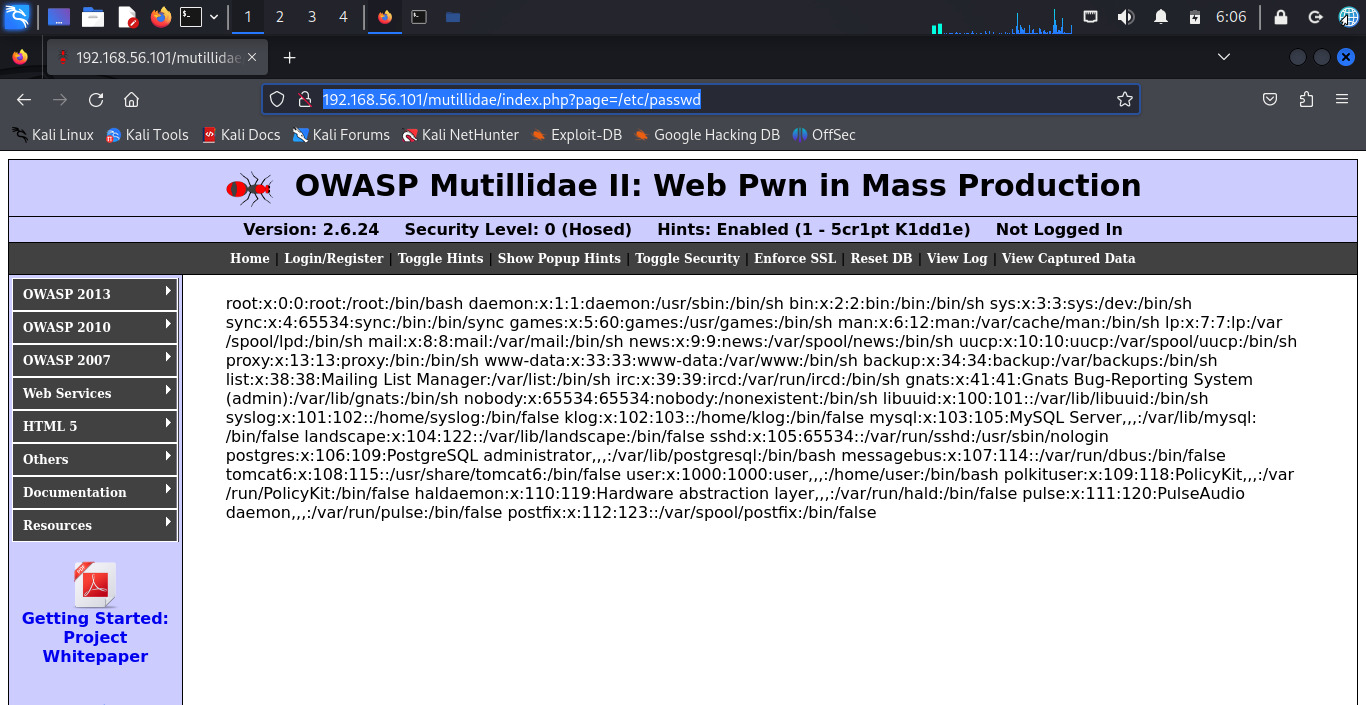
* example.com/page.php?file=home
* example.com/index.php?page=about
* example.com/view.php?doc=report

**Screenshots: LFI Vulnerability in Action** :These screenshots demonstrate LFI attack where the attacker includes a sensitive files like (/etc/passwd),(/etc/hosts). The content of the file is displayed, revealing user information on the web page.

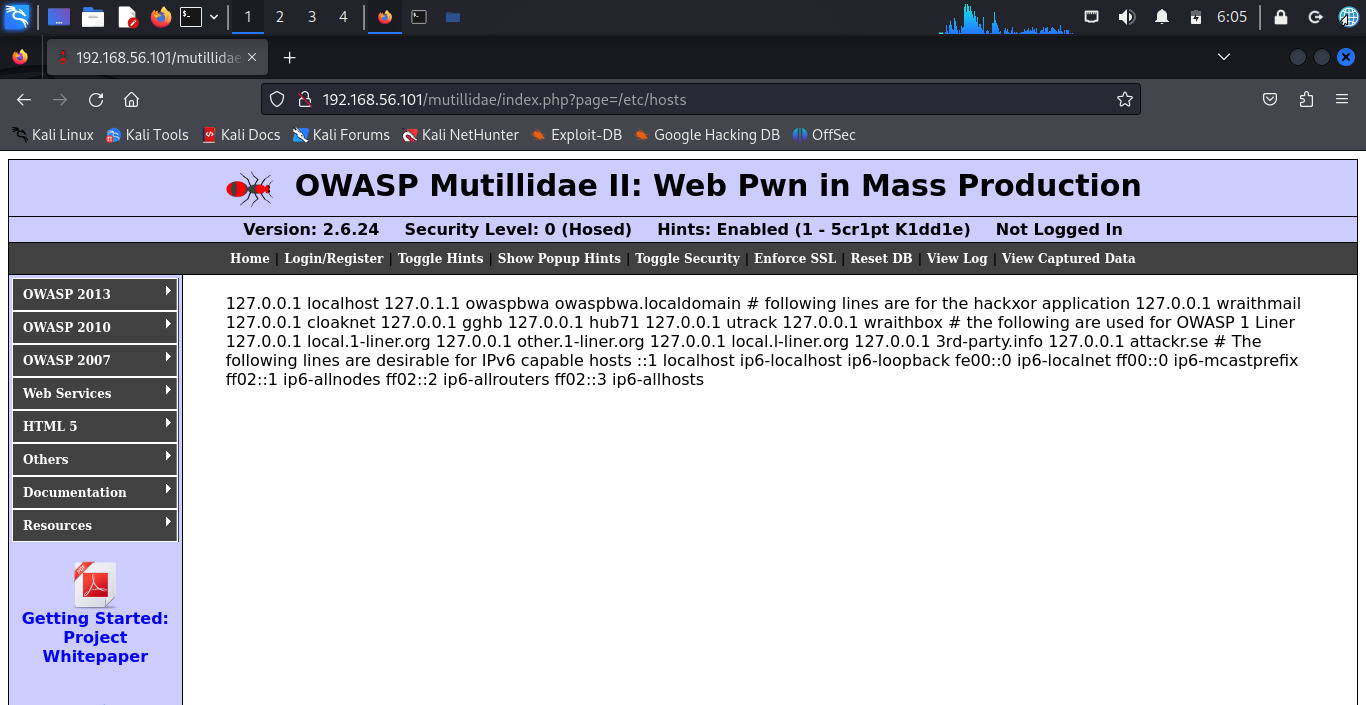
1.OWASP website for testing vulnerability



2.Including page (/etc/passwd):it run the page means the website is LFI vulnerable



3.Including page (/etc/hosts):It page means the website is LFI vulnerable

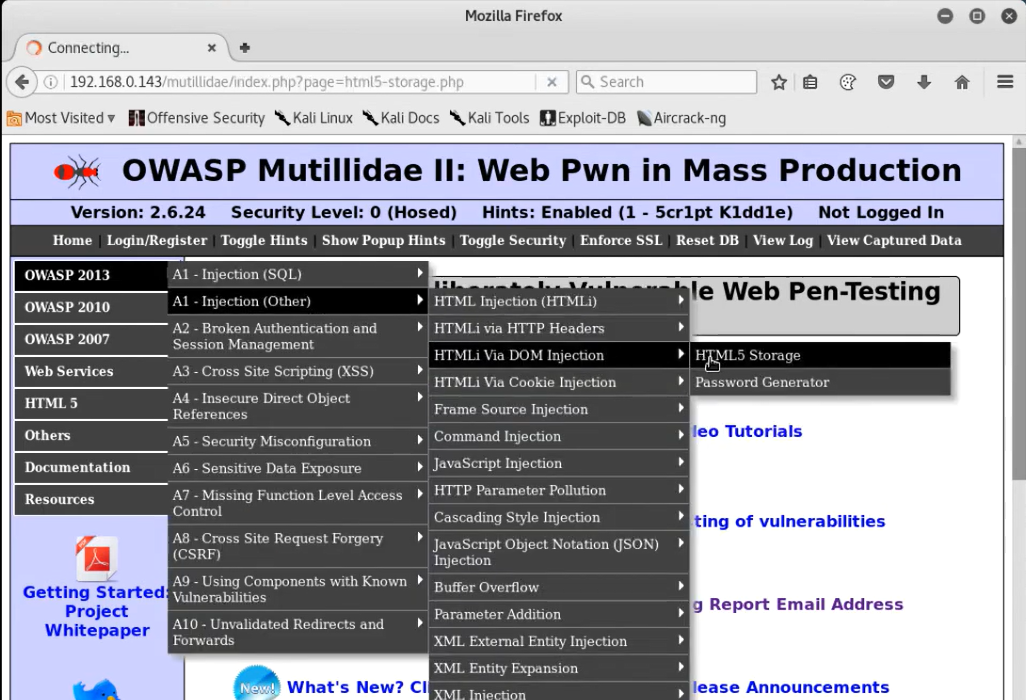


**4.3.2 Remote File Inclusion (RFI)**

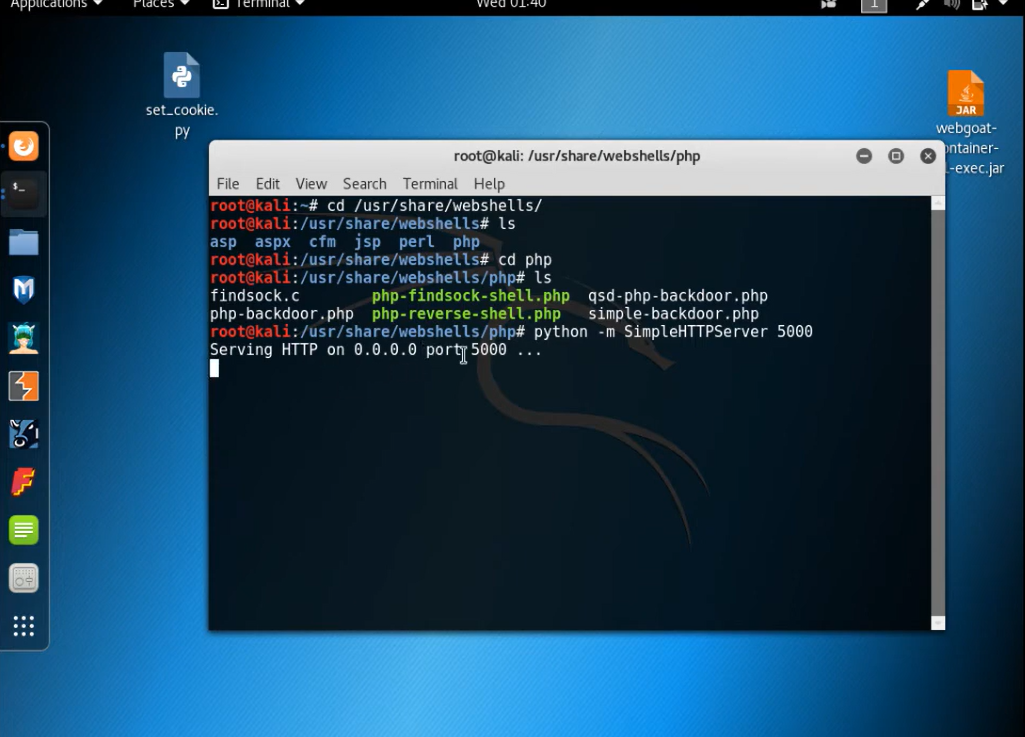
* **Payloads:** URLs pointing to remote scripts hosted on an attacker-controlled server were crafted.
* **Execution:** The payloads were sent using Curl and Burp Suite from the Kali Linux VM.
* **Netcat Listener:** A Netcat listener was set up on the Kali Linux VM to capture data sent from the exploited server.

**Screenshots: RFI Vulnerability in Action**

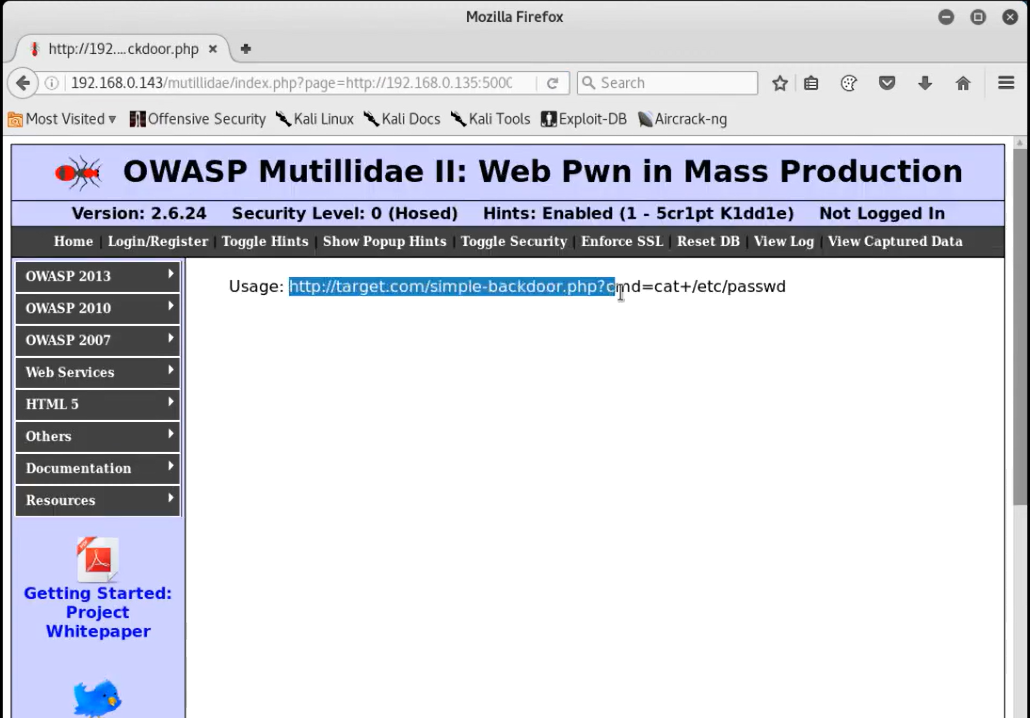
1.OWASP website to test RFI

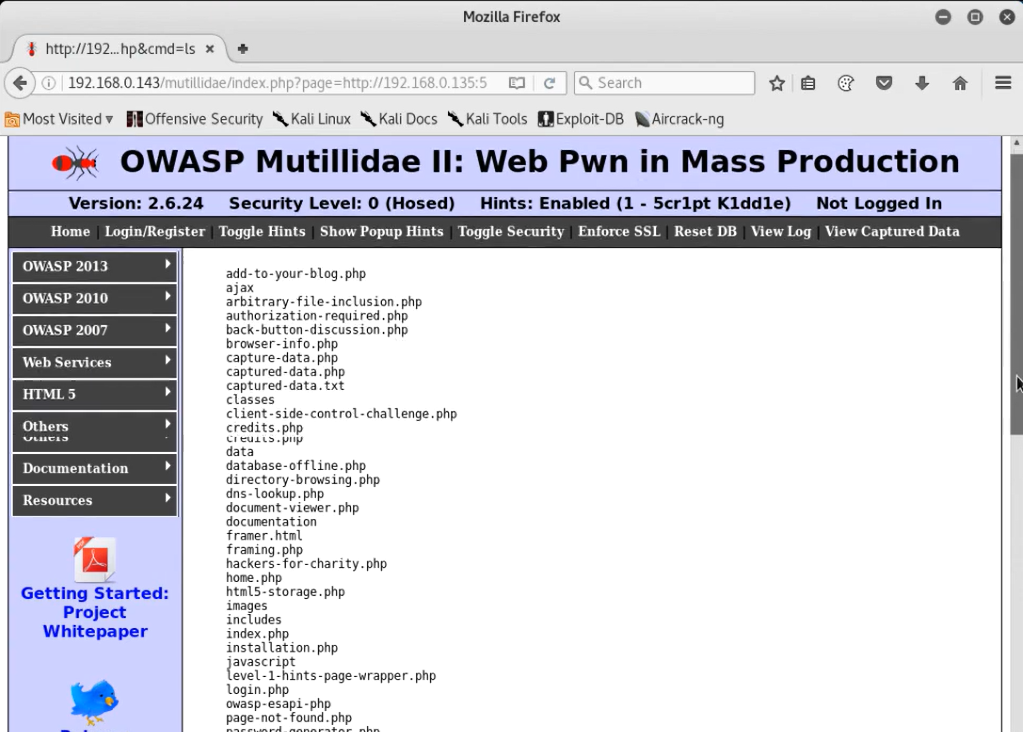


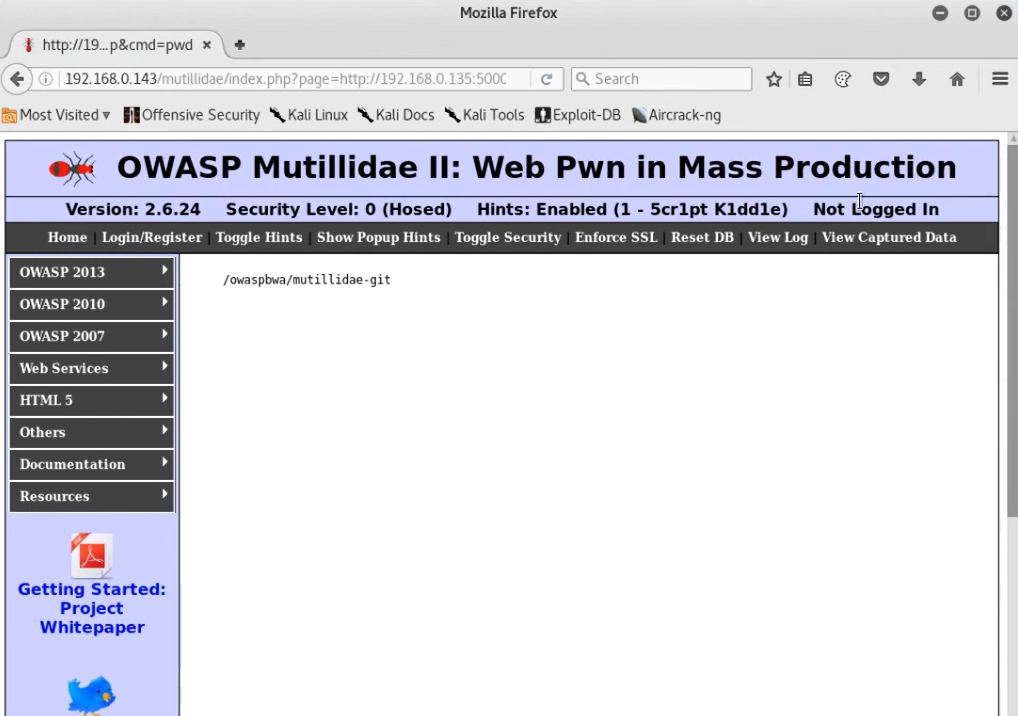
2.Hosting php server to do RFI



3.Injecting hosted php web to testing website







**Results:** Successful exploitation was documented with screenshots and detailed descriptions of the steps taken.

#### 4.4 Implementation of Defensive Measures

Defensive measures were applied to mitigate the identified vulnerabilities.

**4.4.1Input Validation**

* **Techniques:** Input validation was implemented to ensure that only allowed inputs were processed by the application.
* **Implementation:** Code snippets and configuration settings were provided to demonstrate how input validation was applied in the WebGoat application.

**4.4.2 Secure Coding Practices**

* **Guidelines:** OWASP Secure Coding Guidelines were followed to improve the security of the web application.
* **Implementation:** Examples of secure coding practices were provided, focusing on proper handling of file inclusion functions in the WebGoat application.

**4.4.3 Web Application Firewall (WAF) Configuration**

* **Rules:** Custom WAF rules were implemented to detect and block file inclusion attacks.
* **Testing:** The effectiveness of the WAF rules was validated by attempting to exploit the vulnerabilities post-implementation.
* **Results:** The results of the testing phase were documented, showing the successful prevention of attacks by the WAF.

**5.CONCLUSION & RECOMMENDATIONS:**

### 5.1 Conclusion

This project aimed to explore and understand the intricacies of file inclusion attacks, particularly Local File Inclusion (LFI) and Remote File Inclusion (RFI), and to develop effective strategies for mitigating these vulnerabilities in web applications. The project successfully demonstrated the identification, exploitation, and defense mechanisms against file inclusion vulnerabilities using a combination of automated tools, manual testing, and secure coding practices.

**Key Achievements:**

* **Comprehensive Vulnerability Assessment:** Successfully identified and exploited LFI and RFI vulnerabilities in a controlled environment using tools like OWASP ZAP and Burp Suite.
* **Practical Exploitation:** Demonstrated real-world attack scenarios through practical exploitation of LFI and RFI vulnerabilities, highlighting the potential impact on web applications.
* **Effective Mitigation:** Implemented various defensive measures, including input validation, secure coding practices, and web application firewall (WAF) rules, to mitigate the identified vulnerabilities.
* **Documentation and Recommendations:** Provided detailed documentation of the identification, exploitation, and mitigation processes, along with best practices and guidelines for securing web applications against file inclusion attacks.

By gaining a deep understanding of file inclusion attacks and implementing robust defense strategies, this project contributes to enhancing the security posture of web applications, making them more resilient to such threats.

### 5.2 Recommendations

Based on the findings and experiences from this project, the following recommendations are made to further strengthen web application security against file inclusion attacks:

**Regular Security Assessments:**

* + Conduct regular security assessments using both automated tools (e.g., OWASP ZAP, Burp Suite) and manual testing methodologies to identify potential vulnerabilities.
  + Schedule periodic security audits to ensure that new vulnerabilities are promptly detected and mitigated.

**Input Validation and Sanitization:**

* + Implement strict input validation and sanitization techniques to ensure that user inputs are thoroughly checked and sanitized before processing.
  + Utilize whitelisting approaches for input validation, allowing only known good inputs and rejecting all others.

**Secure Coding Practices:**

* + Follow secure coding guidelines provided by reputable sources such as OWASP to develop robust and secure web applications.
  + Regularly train and educate developers on secure coding practices to prevent the introduction of vulnerabilities during the development phase.

**Web Application Firewall (WAF):**

* + Deploy a WAF to provide an additional layer of defense against file inclusion attacks and other web-based threats.
  + Regularly update and configure WAF rules to adapt to evolving attack techniques and patterns.

**Least Privilege Principle:**

* + Apply the principle of least privilege to restrict access rights for users, processes, and systems to the minimum necessary to perform their functions.
  + Limit the permissions of the web server and application to access only required directories and files, reducing the risk of unauthorized access.

**Logging and Monitoring:**

* + Implement comprehensive logging and monitoring mechanisms to detect and respond to suspicious activities and potential attack attempts.
  + Regularly review and analyze logs to identify patterns indicative of file inclusion attacks and other security incidents.

**Patch Management:**

* + Keep all software components, including web servers, applications, and libraries, up-to-date with the latest security patches and updates.
  + Regularly review and apply security patches to address known vulnerabilities in a timely manner.

By adhering to these recommendations and maintaining a proactive security stance, organizations can significantly reduce the risk of file inclusion attacks and enhance the overall security of their web applications.

**6.LIST OF REFERENCES:**

Learning resources/links for understanding the problem and solution:

1. "Understanding File Inclusion Vulnerabilities" - Article by OWASP
2. "Exploiting Local File Inclusion (LFI) Vulnerabilities" - Tutorial by Pentester Academy
3. "Preventing File Inclusion Attacks with Secure Coding Practices" - Whitepaper by SANS Institute
4. "Web Application Firewall (WAF) Configuration for File Inclusion Protection" - Guide by Imperva