DEPI Graduation Project

Penetration testing track

Project Name

Web Application Penetration Testing

Project team names

1- abdulrhman fouad mohamed alfiky
2- Ahemd Mostafa
3- Ahmed Waleed gamal eldin
4- Amir Wagih

1. Target Website Overview

- **Website URL**: http://192.168.1.4/mutillidae/index.php?page=home.php
- **Organization**: Metasploitable
- **Purpose**: a learning vulnerabilities framework to explore and learn more about types of security attacks in a web application .

2- Web Server: Apache

- **Access URL: http://192.168.1.4**

3- Directory Enumeration:

- Common Directories: /admin, /uploads, /cgi-bin
- Tools used: Dirb, Gobuster

Vulnerability identification and exploitation on Metasploitable framework

In this report i'm using SQL injection and XSS and will discuss common vulnerabilities

For SQL injection

1. Error-Based SQL Injection

Error-based SQLi leverages detailed database error messages to extract information. When an application exposes errors, attackers can deduce the structure of the database by injecting queries that intentionally produce errors.

2. Union-Based SQL Injection

Union-based SQL injection exploits the `UNION` operator to combine the results of multiple SELECT queries into one result. This technique allows attackers to extract data from other tables by injecting `UNION` queries into vulnerable parameters.

3- **Time-Based Blind SQL Injection**

Time-based SQLi exploits the fact that databases can be made to delay their responses. By injecting a query that intentionally causes a delay (e.g., `SLEEP` function), an attacker can infer whether the query is executed and thus identify vulnerabilities.

For XSS

1. Stored XSS (Persistent XSS)

Stored XSS occurs when a website stores malicious scripts that attackers input and displays them to other users at a later time. This type of XSS is particularly dangerous because the malicious script can affect multiple users when they visit a vulnerable page.

2. Reflected XSS (Non-Persistent XSS)

Reflected XSS occurs when a malicious script is reflected off a web application onto a user's browser, often through URL parameters. Unlike stored XSS, reflected XSS is not stored on the server and is typically triggered when a user clicks on a malicious link.

3. DOM-Based XSS (Client-Side XSS)

DOM-Based XSS occurs when the vulnerability exists in the client-side JavaScript code rather than the server-side. In this case, the attacker manipulates the DOM (Document Object Model) environment on the client side, causing the browser to execute malicious code.

4. Self-XSS (Self-inflicted XSS)

Self-XSS tricks users into running malicious scripts in their own browser. Attackers often convince victims to paste malicious JavaScript code into their browser's developer console, exploiting the victim’s misunderstanding of how web security works. Although it’s less common as a widespread vulnerability, it can still lead to serious consequences.

Exploitation

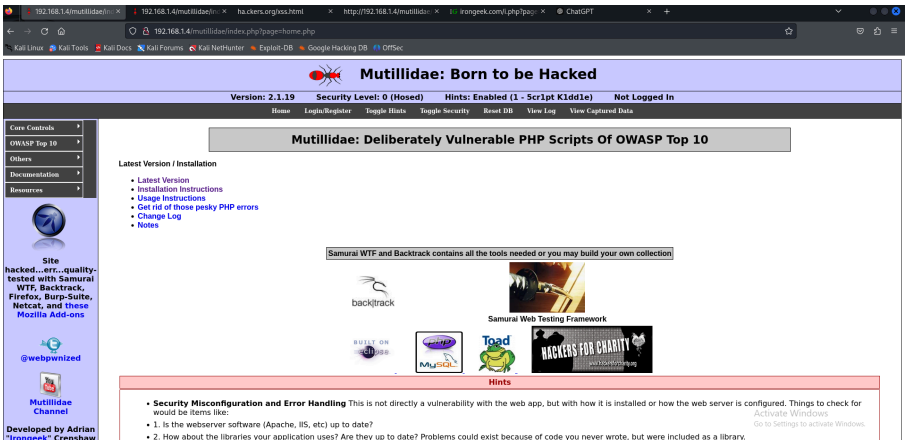
- ## SQL injection

- Error-Based SQL Injection

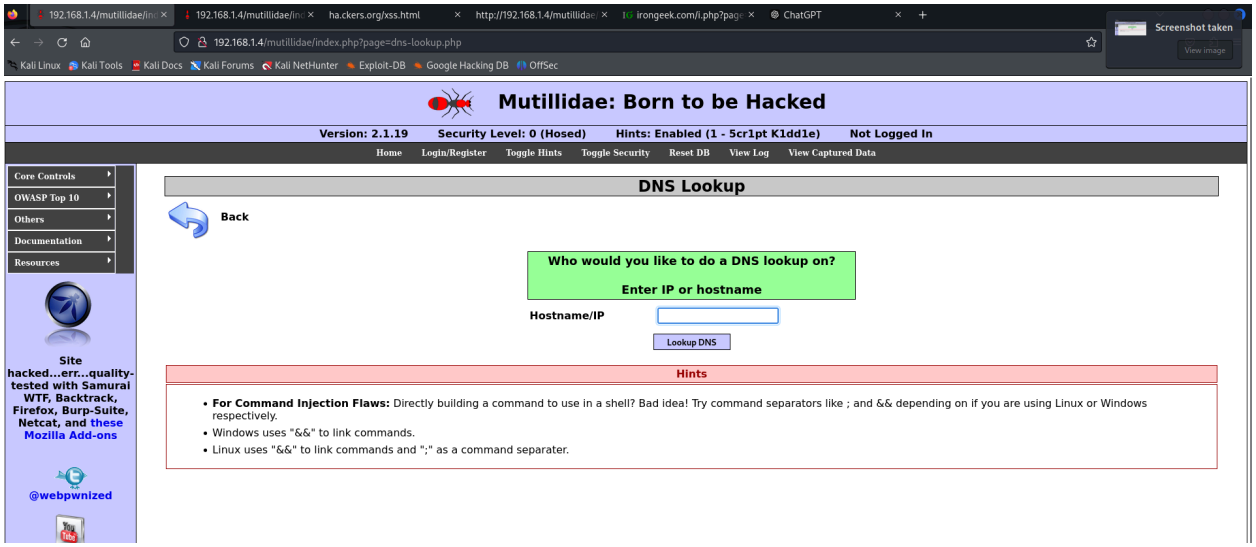
In this attack i try to attack the target (mutillidae) in DNS look up module,

Steps to do this attack

1- when i enter the website



2- enter sql injection DNS lookup



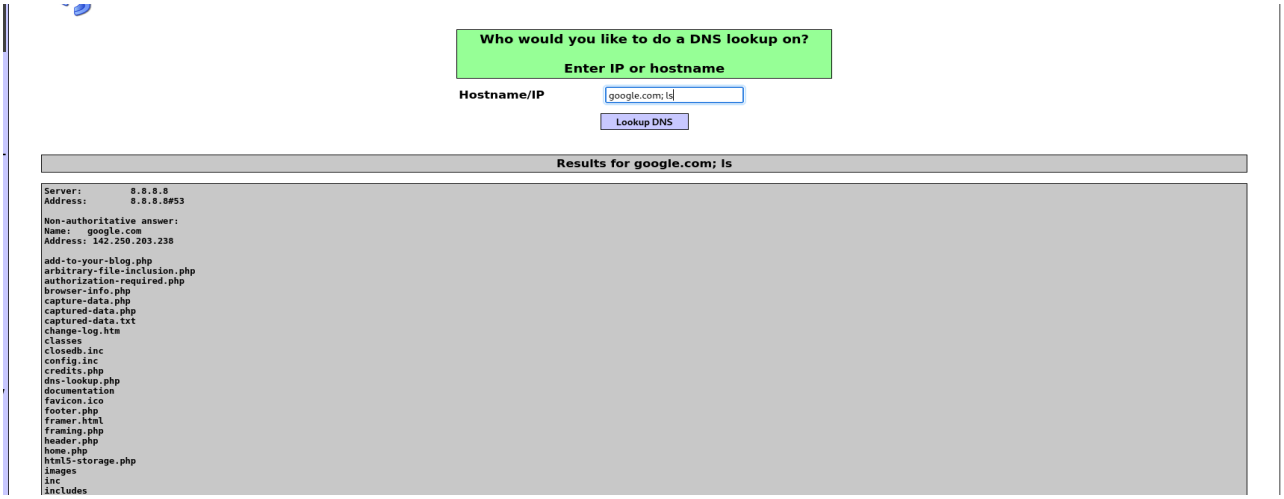
2- try to enter malicious input like entering a live url then enter a command that can retrieve sources of website like this (www.google.com; ls)

3- then you will see that the website have retrieved all sources and php files of the website

SEVERITY:- HIGH

IMPACT :- Attackers can use error-based SQL injection to:

- Enumerate database tables.
- Extract sensitive information.
- Cause denial of service by dropping tables or corrupting data.
- Escalate attacks to further exploit the system.



Suggested Remediation

1. Input Validation

- Sanitize Input: Ensure all user inputs are validated against a strict set of rules. For example, if a field should only accept numeric values, ensure that only numbers are processed.
- Whitelist Validation: Use a whitelist approach to specify valid inputs rather than trying to identify invalid ones. For example, if you expect a username, validate it against a pattern that only allows alphanumeric characters.

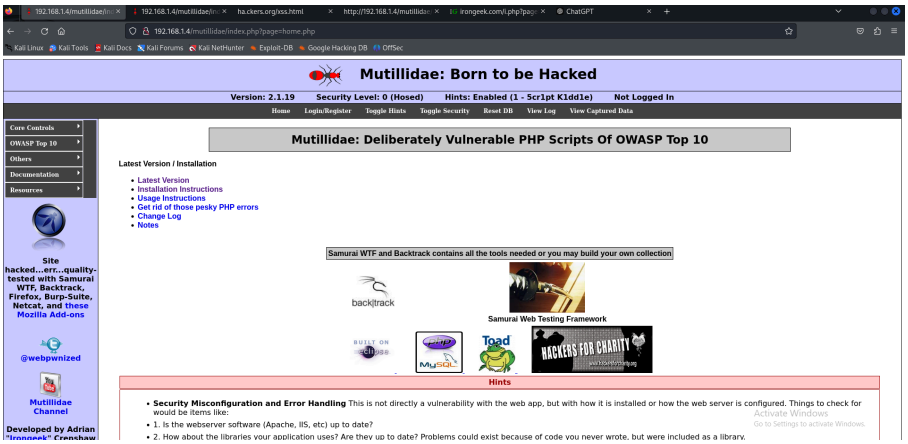
2. Database Permissions

- Least Privilege Principle: Use the principle of least privilege for database user accounts. Ensure that the application's database user has only the necessary permissions to perform its functions (e.g., no DROP TABLE privileges).
- Role-based Access Control: Implement role-based access control to restrict access to sensitive data.

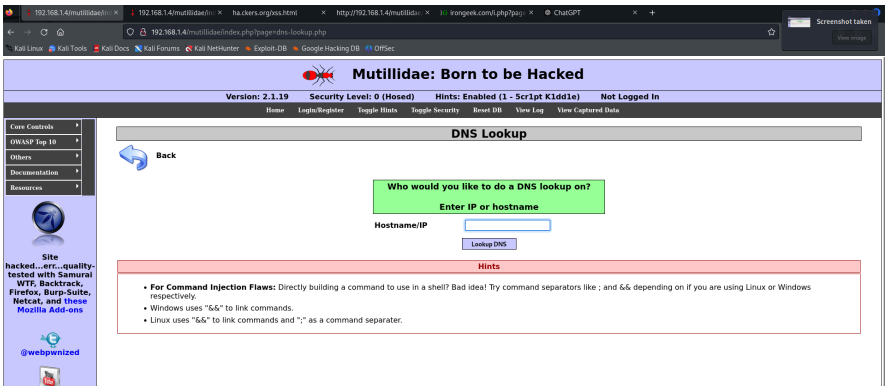
-XSS ATTACK (Stored XSS ATTACK)

Steps to do the attack

1- when i enter the website

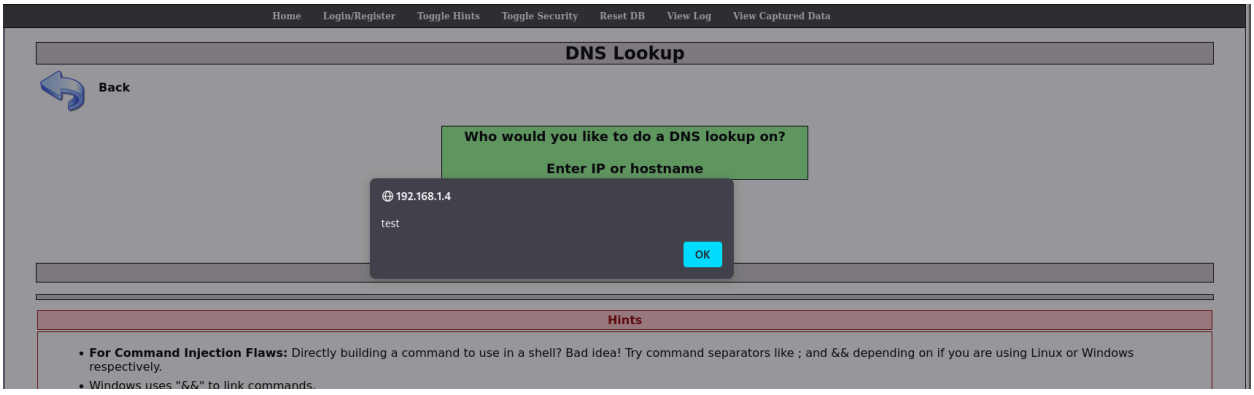


2- enter DNS lookup



3- then will try to insert an alert script to the input field to check the behavior (**<script>alert('test')</script>**)

4- it will give me the expected alert



SEVERITY:- HIGH

IMPACT :- Attackers can use XSS stored Attack in the following:

- Attackers can use this vulnerability to execute arbitrary JavaScript code in the victim's browser.
- It can be used for **session hijacking**, **credential theft**, **phishing attacks**, or **redirecting users** to malicious websites.

Suggested Remediation

Input Validation: Ensure that any input from users (such as domain names) is validated on the server and does not include any executable code.

Output Encoding: Reflecting user input on the page should be encoded properly to prevent execution of scripts. Use functions like `htmlspecialchars()` in PHP to encode `<`, `>`, and other characters.

Content Security Policy (CSP): Implement a strict CSP to prevent the execution of inline JavaScript or unauthorized scripts.