## **Report on Remote File Inclusion (RFI) Vulnerability in DVWA**

### **1. Introduction**

The purpose of this exercise is to identify and exploit a Remote File Inclusion (RFI) vulnerability in the DVWA (Damn Vulnerable Web Application). RFI is a web application vulnerability that allows an attacker to include a remote file, typically a malicious PHP file, to execute commands on a vulnerable server.

In this report, I will cover the steps taken to identify the RFI vulnerability, exploit it, and the implications of this vulnerability. Furthermore, I will provide recommendations for securing web applications against such attacks.

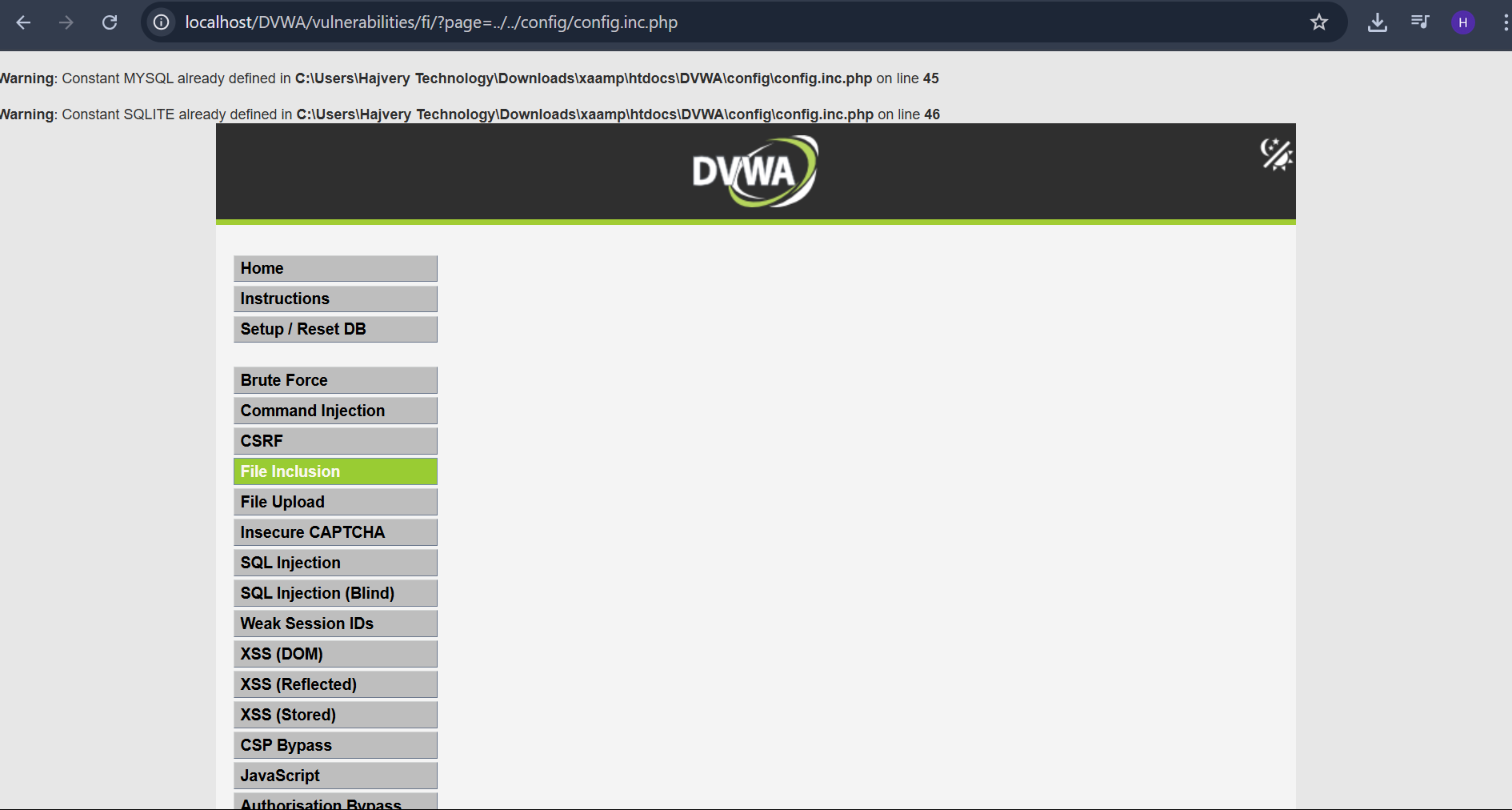
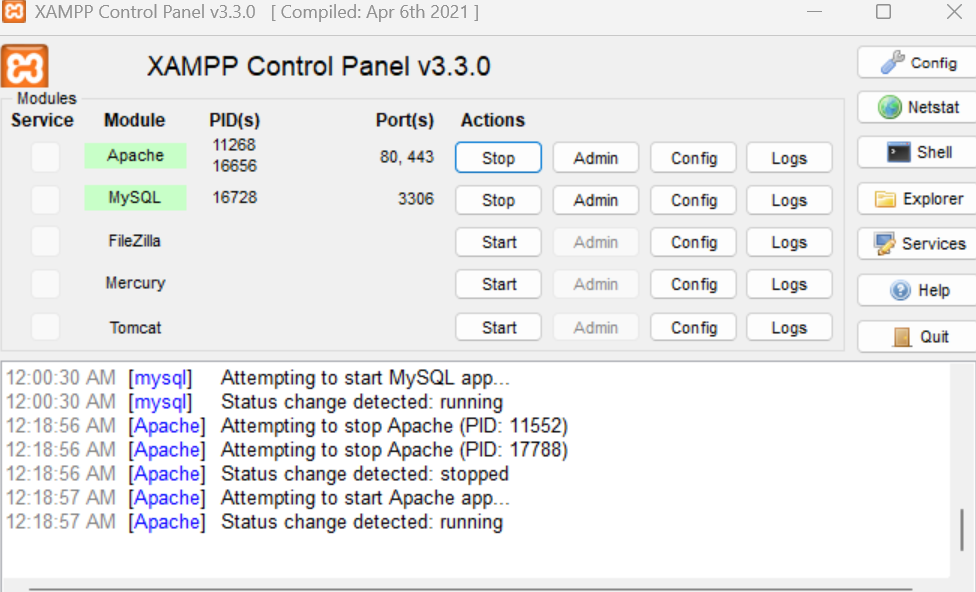
### **2. Task Overview**

**Task 1: Identifying File Inclusion Point**

The first task was to identify a file inclusion point in the application. By manipulating the URL parameter to include sensitive system files, I confirmed that the application is vulnerable to **Local File Inclusion (LFI)**.

Steps:

* Opened the URL: <http://localhost/vulnerabilities/fi/>.
* The page showed a form that accepted a page parameter (?page=include.php).
* I modified the URL to: <http://localhost/vulnerabilities/fi/?page=../../../../etc/passwd>.
* This exposed the contents of the /etc/passwd file, indicating that the application is vulnerable to **LFI**.



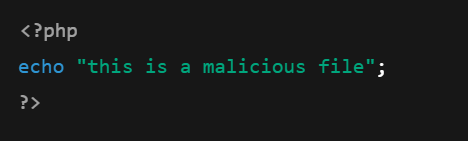
**Task 2: Exploiting Remote File Inclusion (RFI)**

#### **Objective:**

To demonstrate a Remote File Inclusion (RFI) vulnerability by injecting and executing a malicious PHP file hosted on a local server.

#### **Steps Performed:**

1. **Created a Malicious File**
   1. File name: evil.php
   2. Location: C:\xampp\htdocs\evil.php
   3. Content:



**Started a Local PHP Server:**

* Command used:
* The file was hosted locally and accessible via:  
   <http://localhost:8000/evil.php>

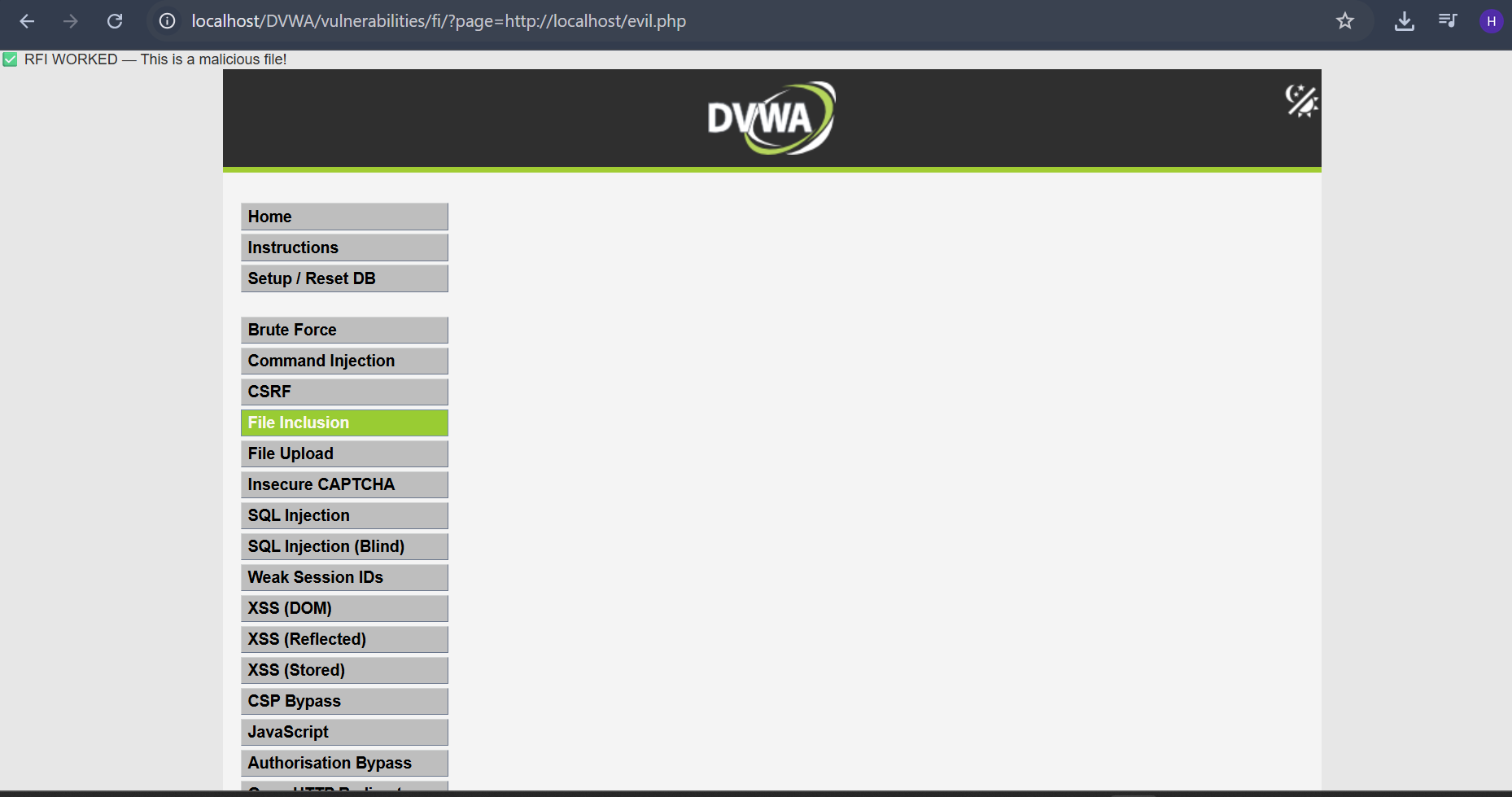
**Executed RFI via DVWA:**

* Accessed DVWA File Inclusion page:  
   <http://localhost/DVWA/vulnerabilities/fi/?page=http://localhost:8000/evil.php>

**Result:**

The browser displayed:  
 "this is a malicious file"

* This confirms the remote PHP file was included and executed successfully on the server.



**Task 3: Observation and Findings**

By exploiting the RFI vulnerability, I was able to execute system commands on the server. Specifically, the malicious file included on the server returned the output of the id command, which displayed the server's user ID.

The vulnerability works because the application allowed external files to be included and executed, opening up the possibility for attackers to run arbitrary commands on the server.

### **3. Implications of RFI Vulnerability**

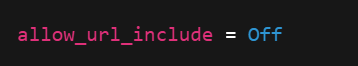
**Remote File Inclusion (RFI)** is a severe security risk because it allows attackers to:

* **Execute arbitrary code** on the server: By including a malicious file, attackers can execute system commands and gain control of the server.
* **Compromise sensitive data**: If an attacker can access files such as database credentials or system files, they can steal sensitive information.
* **Complete server compromise**: An attacker can escalate the attack, gaining full control of the server, leading to data loss, server downtime, and unauthorized access.

### **4. Recommendations for Mitigation**

To prevent RFI and other file inclusion vulnerabilities, I recommend the following best practices:

1. **Disable allow\_url\_include**:  
    Ensure that allow\_url\_include is disabled in the php.ini configuration file. This will prevent PHP from including files from external URLs.



1. **Sanitize User Input**:  
    Never trust user input when including files. Always validate and sanitize the page or file parameters to ensure only legitimate files are included.
2. **Use Whitelists for File Inclusion**:  
    Instead of allowing arbitrary file inclusion, maintain a list of allowed files that can be included. This can be done by checking user input against the list of predefined safe files.
3. **Keep Software Updated**:  
    Ensure that all software, including PHP and web servers, is kept up to date with the latest security patches to prevent known vulnerabilities from being exploited.
4. **Implement Proper Access Controls**:  
    Limit access to sensitive files on the server, such as configuration files and database credentials, by setting strict permissions on them.
5. **Use Web Application Firewalls (WAFs)**:  
    Deploy a WAF to monitor and block malicious traffic attempting to exploit vulnerabilities like RFI.

### **5. Conclusion**

The **Remote File Inclusion (RFI)** vulnerability in the DVWA application was successfully identified and exploited. The attack demonstrated how an attacker can inject malicious files and execute arbitrary code on the server, leading to a potential full server compromise. By following the best practices outlined above, such vulnerabilities can be mitigated and the security of web applications can be significantly improved.