File inclusion

- file inclusion is a vulnerability that is made by poor input sanitation which allows attackers to access sensitive files or hidden one's
- file inclusion is strongly related and chained with path traversal in which attackers consider both as one.
- file inclusion vulnerability has tow types :
  1. LFI : lfi (local file inclusion) is when attackers try to access directory's out of the provided scope (e.x: ../../../../../etc/passwd) this file no user should have access to it but hackers by accessing the poor filtered input try to access it along with some important files as the logs files .
  2. RFI : rfi (remote file inclusion) is a vulnerability that appears with poor input sanitation that mainly leads to RCE (remote code execution) the main goal of it is including an remote files This can lead to the execution of malicious scripts or code on the server.
- PHP wrappers are one of the most common way of obtaining (lfi) that's because php wrappers give the user access to streams of data that if it's not sanitised , e.g: **attackers can use the filter method in which they can edit the data before its written** using the function filter filter is used as ```

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

  this will give us the etc/passwd file but encoded in base 64 , another way that is used in php wrappers is Data **data allows inline data embedding. It is used to embed small amounts of data directly into the application code.** using the following we can access the php wrapper info

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

- PHP sessions is also vulnerable to lfi where attackers can use the session used by the php session in the cookies where we can use the next

```
sessions.php?page=/var/lib/php/sessions/sess_sessionID
```

- LOG Poisoning is a method of exploiting lfi that is done by injecting a malicious file into the log files where it gets executed as a normal php file on the server side In a log poisoning attack, the attacker must first inject malicious PHP code into a log file. This can be done in various ways, such as crafting an evil user agent, sending a payload via URL using Netcat, or a referrer header that the server logs. Once the PHP code is in the log file, the attacker can exploit an LFI vulnerability to include it as a standard PHP file. This causes the server to execute the malicious code contained in the log file, leading to RCE this can be done as the following :

```
$ nc 10.10.54.34 80
<?php echo phpinfo(); ?>
HTTP/1.1 400 Bad Request
Date: Thu, 23 Nov 2023 05:39:55 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Length: 335
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at 10.10.54.34.eu-west-1.compute.inter
</body></html>
```

Obfuscation :

1. first way that is used to bypass the validation is to url encode the string(../file path) so when the server decodes it it will bypass the validation and become the normal (../ file path)
2. second way of mitigation is double encoding for the string in which if the server double decodes the input twice so it will be returned to the original path (../file path)
3. third method is using the double string such as (....//) it's done by adding double (../) inside each other so when the server removes the pattern (../) it will end up with (../)

- j2ee scan