Assignment - 3

Security Engineering

Introduction to Buffer Overflow Vulnerabilities

Instructor -

Sambuddho Chakravarty

Submitted By -

Deepak Kumar Sood

MT15013

MTech 1st year

Date - 17-03-2016

Prerequisites:

Turn off ASLR (Address Space Layout Randomization) by following command:
Turn Off - `echo 0 | sudo tee /proc/sys/kernel/randomize_va_space`
Turn On - `echo 2 | sudo tee /proc/sys/kernel/randomize_va_space`

Question 1:

Assembly Code for Reading and Writing from a file.

In question 1, a shellcode has to be written for opening a file and reading the contents of a file and then finally writing the content of the file to terminal (standard output). For this we had made a text file test.txt with content to read.
To find the system call number we can see unistd_64.h
Location: /usr/include/x86_64-linux-gnu/asm/unistd_64.h
This file contains all the System Call with their respective system call numbers.

Commands -

1. `nasm -f elf64 que1.asm` //for making an object file from the shellcode/assembly code
2. `ld que1.o -o que1` //for linking the object file to executable file
3. `./que1` //for running the file

Running from makefile -

1. make que1
2. make clean1

For Bonus -

1. There is an error reporting if no file is present for reading.
2. Error is displayed if file is present but the program is unable to read the file if file is locked in any case.
3. Error is displayed if file is read but the program is unable to write the output to standard output i.e. terminal.

Question 2:

Using que1 as shellcode.

Commands -

1. For generating shellcode -

```
for i in `objdump -d que1 | tr '\t' ' ' | tr ' ' '\n' | egrep '^[0-9a-f]{2}$' ` ;
do echo -n "\\x$i" ; done
```

2. Now copy the shellcode generated and paste it to the que2.c shellcode char array.
3. compile the program

```
gcc -g -Wall -fno-stack-protector -z execstack que2.c -o que2
```

4. Run the program
   `./que2`

Running from makefile -

1. make que2
2. make clean2

Question 3:

Function call flow diversion.

In this program we have to use buffer overflow to overwrite the return address so that function target is called when returning from function1 to main. This is done through injecting shellcode using memset and passing constant characters along with return address for target function got via objdump on object file.

Commands -

1. gcc -fno-stack-protector -zexecstack -o que3 que3.c
2. objdump -d que3
3. perl -e '(print "0"x24).(print "\xf4\x06\x40\x00")' | ./que3

Running from makefile -

1. make que3
2. make clean3

Explanation -

- -fno-stack-protector is used for for disabling the stack protection bit.
- -zexecstack is used for making the stack executable.
- Objdump is used to find the return address of the target function to put it in buffer overflow attack.
- perl is used for making the hex code for stack overflow and pipe it to ./que3 as input.
- Little endian format is used for hex code therefore the 64 bit address is written in reverse order.

For Bonus -

I have used perl script to generate the shellcode that can be directly fed to the program main function.