Nate Le
CS 479
2/4/2024

Introduction

In our task, we aimed to craft 64-bit x86 Linux shellcode. It's designed to use the execve system function. Its job? Transform the existing process into a terminal shell like /bin/sh. We'll dissect the assembly code, go through each line. We'll also talk about the shellcode length.

Shellcode Explanation:

```
.text
        .global _start
_start:
        xorl %eax, %eax
        movb $11, %al          # syscall number for execve
        leal shell_path, %ebx  # address of "/bin/sh"
        xorl %ecx, %ecx        # null for command-line arguments
        xorl %edx, %edx        # null for environment variables
        int $0x80

        xorl %eax, %eax
        inc %eax               # syscall number for exit
        int $0x80

.section .data
        shell_path: .asciz "/bin/sh"
```

movl $11, %eax
The assembly code loads the syscall number for execve (11 for 32-bit Linux) into the eax register.

leal shell_path, %ebx
The address of the string "/bin/sh" is loaded into the ebx register.

xorl %ecx, %ecx
xorl %edx, %edx
Registers ecx and edx are set to null, indicating no command-line arguments and no environment variables.

```
int $0x80
```
The int 0x80 instruction is used to trigger the system call.

```
movl $1, %eax  # syscall number for exit
xorl %ebx, %ebx  # exit code 0
int $0x80
```
The program then exits using the exit system call.

```
.section .data
shell_path: .asciz "/bin/sh"
```
The .data section contains the definition of the /bin/sh string.

Shellcode length

I use python script to find my shellcode bytes, this is the result

My shellcode is 435 bytes long.
Here they are: -- 2E 74 65 78 74 0A 20 20 20 20 2E 67 6C 6F 62 61 6C 20 5F 73 74 61 72 74
0A 5F 73 74 61 72 74 3A 0A 20 20 20 20 78 6F 72 6C 20 25 65 61 78 2C 20 25 65 61 78 0A 20
20 20 20 6D 6F 76 62 20 24 31 31 2C 20 25 61 6C 20 20 20 20 20 20 20 20 20 20 20 23 20 73
79 73 63 61 6C 6C 20 6E 75 6D 62 65 72 20 66 6F 72 20 65 78 65 63 76 65 0A 20 20 20 20 6C
65 61 6C 20 73 68 65 6C 6C 5F 70 61 74 68 2C 20 25 65 62 78 20 20 20 23 20 61 64 64 72 65
73 73 20 6F 66 20 22 2F 62 69 6E 2F 73 68 22 0A 20 20 20 20 78 6F 72 6C 20 25 65 63 78 2C
20 25 65 63 78 20 20 20 20 20 20 20 20 20 23 20 6E 75 6C 6C 20 66 6F 72 20 63 6F 6D 6D 61
6E 64 2D 6C 69 6E 65 20 61 72 67 75 6D 65 6E 74 73 0A 20 20 20 20 78 6F 72 6C 20 25 65
64 78 2C 20 25 65 64 78 20 20 20 20 20 20 20 20 20 23 20 6E 75 6C 6C 20 66 6F 72 20 65 6E
76 69 72 6F 6E 6D 65 6E 74 20 76 61 72 69 61 62 6C 65 73 0A 20 20 20 20 69 6E 74 20 24 30
78 38 30 0A 0A 20 20 20 20 78 6F 72 6C 20 25 65 61 78 2C 20 25 65 61 78 0A 20 20 20 20 69
6E 63 20 25 65 61 78 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 23 20 73 79 73 63 61
6C 6C 20 6E 75 6D 62 65 72 20 66 6F 72 20 65 78 69 74 0A 20 20 20 20 69 6E 74 20 24 30 78
38 30 0A 0A 2E 73 65 63 74 69 6F 6E 20 2E 64 61 74 61 0A 20 20 20 20 73 68 65 6C 6C 5F 70
61 74 68 3A 20 2E 61 73 63 69 7A 20 22 2F 62 69 6E 2F 73 68 22 0A

Conclusion

In conclusion, the assembly code successfully achieves the goal of invoking the execve system call to execute the shell at /bin/sh. The shellcode length was determined, and its ASCII representation was examined using the provided Python script. But my shellcode takes a lot of bytes 435 bytes. I will look into more efficient way in the future