15213 Lecture 9: Advanced aka Security POGIL Activity Solutions

1 Getting Started

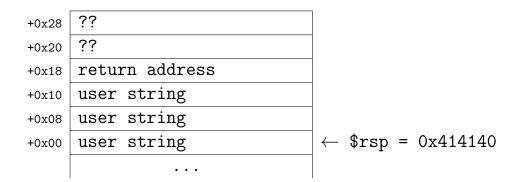
- 1. The struct is allocated on the stack, as a local variable.
- 2. Each callq instruction pushes the (correct) return address onto the stack.
- 3. This function call overwrote the return address for fun() with an address pointing to some non-program memory, causing attempted execution of either invalid operations or non-executable operations.

2 Gets

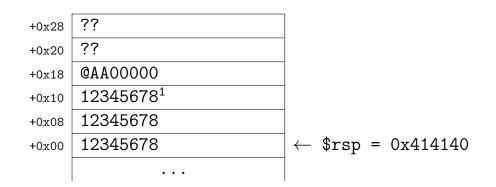
- 1. We don't!
- 2. When it encounters a newline or the end of the user input stream.
- 3. No, they purely depend on how many characters the user inputs.

3 Overwriting Stack

- 1. 0x18 (24) bytes are allocated on the stack in echo().
- 2. The input buffer is at most 0x18 (24) bytes long. (Note that the compiler may have inserted padding to align the stack.) The user may or may not enter a string shorter than this safe length.



4. The solution is given using ASCII bytes with addresses increasing left to right.



¹Assuming the upper 4 bytes of the original return address was all 0's

4 Exploit

1. Starting from echo()'s call to gets(), at 0x4006d6:

```
0x4006d6 -> 0x4006db (mov) -> 0x4006de (puts) -> 0x4006e3 (add) -> 0x4006e7 (retq) -> [USER INPUT] 0x414140 (xor) -> 0x414143 ... etc.
```

- 2. When control was going to be returned to echo()'s caller, control was instead transferred to user input on the stack.
- 1. movl \$decafbad, %eax
- 2. The instruction bytes would replace the first 1–5 characters of the input string.
- 1. See footnote 1, question 3.3

5 Defense

- 1. fgets() is safer than gets() because it is called with a maximum number of bytes to read from the given stream.
- 2. Execution would jump to an unknown section of memory, almost certainly executing non-executable or invalid code before being terminated by the OS.
- 3. By randomizing the starting address of the stack at runtime.

```
1. <...echo()'s code...>
%rax = 0x28;
*(%rsp + 8) = %rax;
%rax = 0;
<...echo()'s code...>
%rax = *(%rsp + 8);
if (%rax != 0x28) stack_chk_fail()
<...echo()'s return...>
```

- 2. We would overwrite *(%rsp + 8), causing stack_chk_fail() to be executed and our program to terminate.
- 3. This method of defense makes our program slower to compile, as the compiler needs to determine where and how to insert these 'canaries,' slower to execute, as it involves adding instructions, and makes our program larger (in the given example echo() grew by 37 bytes).

6 ROP

- 1. We overwrote the return address for echo()'s caller before executing retq.
- 2. The byte value c3 corresponds to a ret instruction.
- 3. The sequence of bytecode instructions appears at address 0x4004d3.
- movq %rax, %rdi retq
 - <...instructions starting at the second next stack address prior to running this code block...>
- 2. If there is no return instruction at the end of the gadget, execution will never jump to the next gadget in the chain.