Format Strings Vuln's

Takes adv of formatted str's in

printf ('%s', ...)

If as an attacker we can control arg's to printf

printf(buffer)

We can make printf so look on the stack for values

↙ First Param

print (buffer) "%p , %p

↑ second param

Registers are used to hold params in 64 bit, half on stack

32 bit is all stack

Rsi	1	
Rdx	2	
rcx	3	
r8	4	
r9	5	
[rsp]	6	stack
[rsp-8]	7	
[rsp-16]	8	

format in Rdi

Stack starts at #6

Format 0 Challenge

Steps to solve it

%<Number> p = %1p = First Parameter

Format Strings are used when you don't know the size of the buffer

Since it allows us to print values off the stack in 8 bytes at a time

We can find the saved return address of vuln() and overwrite it with the address for win()

The 1st address we see is the saved ret

How do we go from parameter to offset?

####d90 = libc_start_main

Offset = ret_addr - buffer_addr

Offset = 16

Found this by looking at the hex input of format string and counting the bytes until you see what looks like a return address

padding + win_addr = exploit

Look out for system() calls in functions because it expects 16 byte aligned stack

Take whatever addr we leak off the stack and add 6 to it to reach the ret in main

Write a script to leak the p's from the stack.

The %n Formatter

will take the # of char's printed to the screen and hold it in a variable

We can control where to write (the argument)
We can control what to write (the string)

"Write-what-where"

%hhn single byte to overwrite

If we use multiple %hhn format strings we can write one byte at and address at a time.

Could rewrite ret of main with a few bytes at a time.

Format 1 Challenge

- Need to overwrite target
- The start address of buffer has to be aligned
- Small Buffer sizes do not gaurantee alignment