**Question 1 - math expression**

Complete this quiz when you have completed project 1. The questions for the quiz were presented in the description of the project, so you should just have to enter your answers here. To make sure your answers match, avoid spurious whitespace.

There is a stack-based overflow in the program. What is the name of the stack-allocated variable that contains the overflowed buffer?

\*A: wis

Feedback: The wis variable is allocated on the stack and can get overflowed by the call to gets.

Default Feedback:

**Question 2 - numeric**

Consider the buffer you just identified: Running what line of code will overflow the buffer? (We want the line number, not the code itself.)

\*A: 62.0

Feedback: Due to the gets overrunning the target buffer

Default Feedback:

**Question 3 - math expression**

There is another vulnerability, *not dependent at all on the first*, involving a *non*-stack-allocated buffer that can be indexed outside its bounds (which, broadly construed, is a kind of buffer overflow). What variable contains this buffer?

\*A: ptrs

Feedback: Note that l->data can be overflowed, but only if wis is overflowed, so it would be dependent on the first.

Default Feedback:

**Question 4 - text match**

Consider the buffer you just identified: Running what line of code overflows the buffer? (We want the number here, not the code itself.)

\*A: 100

Feedback: This overflow happens by allowing the index variable to be too large. Properly, this is at line 102, but you can make arguments that earlier lines set this up to happen.

\*B: 101

Feedback: This overflow happens by allowing the index variable to be too large. Properly, this is at line 102, but you can make arguments that earlier lines set this up to happen.

\*C: 102

Feedback: This overflow happens by allowing the index variable to be too large. Properly, this is at line 102, but you can make arguments that earlier lines set this up to happen.

Default Feedback:

**Question 5 - text match**

What is the address of $$\color{red}{\verb|buf|}$$ (the local variable in the $$\color{red}{\verb|main|}$$ function)? Enter the answer in either hexadecimal format (a 0x followed by 8 “digits” 0–9 or a-f, like $$\color{red}{\verb|0xbfff0014|}$$) or decimal format. Note here that we want the address of $$\color{red}{\verb|buf|}$$, not its contents.

\*A: 3221221680

Feedback: break at wisdom-alt.c:100 and print &buf

\*B: 0xbffff130

Feedback: break at wisdom-alt.c:100 and print &buf

Default Feedback:

**Question 6 - text match**

What is the address of $$\color{red}{\verb|ptrs|}$$ (the global variable) ? As with the previous question, use hex or decimal format.

\*A: 0x804a0d4

Feedback: Again, at the first breakpoint you can print &ptrs to get the answer

\*B: 0x0804a0d4

Feedback: Again, at the first breakpoint you can print &ptrs to get the answer

\*C: 134521044

Feedback: Again, at the first breakpoint you can print &ptrs to get the answer

Default Feedback:

**Question 7 - text match**

What is the address of $$\color{red}{\verb|write\_secret|}$$ (the function) ? Use hex or decimal.

\*A: 0x8048534

Feedback: Easy: Print &write\_secret from gdb

\*B: 0x08048534

Feedback: Easy: Print &write\_secret from gdb

\*C: 134513972

Feedback: Easy: Print &write\_secret from gdb

Default Feedback:

**Question 8 - text match**

What is the address of $$\color{red}{\verb|p|}$$ (the local variable in the $$\color{red}{\verb|main|}$$ function) ? Use hex, or decimal format.

\*A: 0xbffff534

Feedback: Same drill as the earlier questions

\*B: 3221222708

Feedback: Same drill as the earlier questions

Default Feedback:

**Question 9 - numeric**

What input do you provide to the program so that $$\color{red}{\verb|ptrs[s]|}$$ reads (and then tries to execute) the contents of stack variable $$\color{red}{\verb|p|}$$ instead of a function pointer stored in the buffer pointed to by $$\color{red}{\verb|ptrs|}$$? As a hint, you can determine the answer by performing a little arithmetic on the addresses you have already gathered. If successful, you will end up executing the $$\color{red}{\verb|pat\_on\_back|}$$ function. Provide the smallest positive integer.

\*A: 7.71675416E8

Feedback: This is the result of doing (unsigned int)((int \*)&p - (int\*)&ptrs) in gdb. Note that doing (unsigned int)&p - (unsigned int)&ptrs) won't work because the difference will be in bytes, not pointer-sized words. We need the difference to be in words so using s in ptrs[s] does the right thing.

Default Feedback:

**Question 10 - numeric**

What do you enter so that $$\color{red}{\verb|ptrs[s]|}$$ reads (and then tries to execute) starting from the 65th byte in $$\color{red}{\verb|buf|}$$, i.e., the location at $$\color{red}{\verb|buf[64]|}$$? Enter your answer as an (unsigned) integer.

\*A: 7.71675175E8

Feedback: (unsigned int)((int \*)&buf[64] - (int \*)&ptrs) in gdb. Same process as the previous question, but now you are using a different starting point.

Default Feedback:

**Question 11 - text match**

What do you replace \color{red}{\verb|\xEE\xEE\xEE\xEE|} with in the following input to the program (which due to the overflow will be filling in the 65th–68th bytes of \color{red}{\verb|buf|}) so that the \color{red}{\verb|ptrs[s]|} operation executes the \color{red}{\verb|write\_secret|} function, thus dumping the secret? (Hint: Be sure to take endianness into account.)

\color{red}{\verb|771675175\x00AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\xEE\xEE\xEE\xEE|}

\*A: \x34\x85\x04\x08

Feedback: This is the address of write\_secret, which is 0x08048534, but entered in hex bytes and accounting for little endianness

Default Feedback:

**Question 12 - numeric**

Suppose you wanted to overflow the $$\color{red}{\verb|wis|}$$ variable to perform a stack smashing attack. You could do this by entering 2 to call $$\color{red}{\verb|put\_wisdom|}$$, and then enter enough bytes to overwrite the return address of that function, replacing it with the address of $$\color{red}{\verb|write\_secret|}$$. How many bytes do you need to enter prior to the address of $$\color{red}{\verb|write\_secret|}$$?

\*A: 148.0

Feedback: This number comes from the following calculation:

128 bytes for the buffer;

12 bytes for the three local variables (r, l, and v);

4 bytes for saved EBP;

4 bytes for saved EDI;

And finally the return address, to be overflowed

Default Feedback: