CS 161 Spring 2026 — Introduction to Computer Security — Exam Prep 2

Q1 *Memory Safety: Jonah Dreams of Sheep* (12 points)

Consider the following vulnerable C code:

```
1  void sleep() {
2    char dream[24];
3    fread(dream, 24, 1, stdin);
4    count_sheep(dream);
5  }
6
7  void count_sheep(char* input) {
8    size_t num_sheep;
9    fread(&num_sheep, sizeof(size_t), 1, stdin);
10
11   if (num_sheep > 64) {
12     exit(1);
13   }
14   printf(input);
15 }
```

Stack at Line 9

RIP of `sleep`
(1)
(2)
(3)
RIP of `count_sheep`
SFP of `count_sheep`
`num_sheep`

Assumptions:
- All memory safety defenses are disabled.
- There is a copy of `SHELLCODE` at address `0xdeadbeef`.
- There is no compiler padding.
- RIP of `sleep` is located at address `0xffffd634`.

Q1.1 (0.5 points) What goes in blank (1) in the stack diagram above?

◯ `dream` ◯ `input` ◯ `num_sheep` ◯ RIP of `sleep` ◯ SFP of `sleep`

Q1.2 (0.5 points) What goes in blank (2) in the stack diagram above?

◯ `dream` ◯ `input` ◯ `num_sheep` ◯ RIP of `sleep` ◯ SFP of `sleep`

Q1.3 (0.5 points) What goes in blank (3) in the stack diagram above?

◯ `dream` ◯ `input` ◯ `num_sheep` ◯ RIP of `sleep` ◯ SFP of `sleep`

Q1.4 (1.5 points) Which of the following memory safety vulnerabilities are present in the above code?

☐ Stack Buffer Overflow ☐ Off-by-one ◯ None of the above

☐ Heap Buffer Overflow ☐ Format String Vulnerability

Warning: Q1.5 and Q1.6 are very hard. Consider attempting all other questions before spending too much time on these two parts.

Q1.5 (5 points) Provide an input to the `fread` call on Line 3 that will help us execute `SHELLCODE`.

If a part of the input can be any non-zero value, use `'A' * n` to represent `n` bytes of garbage.

For the purposes of this question, the `%*u` specifier reads one argument from the stack (call it `m`), treats it as an integer, and prints `m` bytes. It then proceeds to read a second argument off the stack, but does nothing with it.

Hint: When processed by printf, `%hhn` writes one byte to the address given by the corresponding argument; the value written is the number of characters printed so far.

`'[ ]' + '%*u' + ('%c' * [ ]) +`

`[ ] + [ ] + ('A' * [ ])`

Q1.6 (2 points) Provide an input to the `fread` call on Line 9 that, together with your answer to the previous part, will execute `SHELLCODE`.

If a part of the input can be any non-zero value, use `'A' * n` to represent `n` bytes of garbage.

[ ]

Q1.7 (1 point) Which memory safety defenses would cause the correct exploit (without modifications) to fail? Consider each choice independently.

☐ ASLR ☐ Non-Executable Pages ○ None of the above

Q1.8 (1 point) Would the same exploit work if the call to `fread` on line 3 was replaced with `fgets` (with the same parameters)?

○ Yes, because we can still write in the correct exploit input into `dream`.

○ Yes, because `fgets` does the same thing as `fread` when given the same number of bytes to read.

○ No, because `fgets` will overwrite the last byte of the SFP of `sleep` with a null terminator.

○ No, because the amount of space in `dream` required to perform the exploit will no longer be sufficient.