Peyrin Kao<br>
Fall 2023

# CS 161 Computer Security

Discussion 3

**Question 1** ***C Memory Defenses***

Mark the following statements as True or False and justify your solution. Please feel free to discuss with students around you.

Q1.1 Stack canaries completely prevent a buffer overflow from overwriting the return instruction pointer.

Q1.2 A format-string vulnerability can allow an attacker to overwrite values below the stack pointer.

Q1.3 ASLR, stack canaries, and NX bits all combined are insufficient to prevent exploitation of all buffer overflow attacks.

**Short answer!**

Q1.4 What vulnerability would arise if the stack canary was between the return address and the saved frame pointer?

Q1.5 Assume ASLR is enabled. What vulnerability would arise if the instruction **jmp *esp** exists in memory?

*This content is protected and may not be shared, uploaded, or distributed.*

**Question 2** ***Robin***

Consider the following code snippet:

```
 1 void robin(void) {
 2     char buf[16];
 3     int i;
 4
 5     if (fread(&i, sizeof(int), 1, stdin) != 1)
 6         return;
 7
 8     if (fgets(buf, sizeof(buf), stdin) == NULL)
 9         return;
10
11     _________
12 }
```

Assume that:

- There is no compiler padding or additional saved registers.
- The provided line of code in each subpart compiles and runs.
- `buf` is located at memory address `0xffffd8d8`
- Stack canaries are enabled, and all other memory safety defenses are disabled.
- The stack canary is four completely random bytes (**no null byte**).

For each subpart, mark whether it is possible to leak the value of the stack canary. If you put possible, provide an input to Line 5 and an input to Line 8 that would leak the canary. If the line is not needed for the exploit, you must write "Not needed" in the box.

Write your answer in Python syntax.

Q2.1 (3 min) Line 11 contains `gets(buf);`.

○ Possible

○ Not possible

Line 5:

Line 8:

*This content is protected and may not be shared, uploaded, or distributed.*

Q2.2 (5 min) **For this subpart only, enter an input that allows you to leak a single character from memory address `0xffffd8d7`. Mark "Not possible" if this is not possible.** Line 11 contains `printf("%c", buf[i]);`.

○ Possible

○ Not possible

Line 5:

Line 8:

Q2.3 (6 min) Line 11 contains `printf(buf);`.

○ Possible

○ Not possible

Line 5:

Line 8:

Q2.4 (6 min) Line 11 contains `printf(i);`.

○ Possible

○ Not possible

Line 5:

Line 8:

*This content is protected and may not be shared, uploaded, or distributed.*