CNT 4419: Secure Coding [Fall 2019]
Test 4

NAME: _____

Instructions:

1) This test is 5 pages in length.

2) You have 40 minutes to complete and turn in this test.

3) Short-answer and essay questions include guidelines for how much to write. Respond in complete English sentences. Avoid bullet points. Responses will be graded as described on the syllabus.

4) This test is closed books, notes, papers, smartphones, laptops, friends, neighbors, etc.

5) Use the backs of pages in this test packet for scratch work. If you write more than a final answer in the area next to a question, circle your final answer.

1. [5 points]
Explain NOP sleds and how they are used. [2-3 sentences]

2. [4 points]
Name four standard functions that can be avoided to mitigate buffer overflows.

3. [15 points]
a) Compare and contrast mandatory and discretionary access control. [2-3 sentences]

b) Name and describe two models discussed in class for mandatory access control. Be sure to describe each model's advantages and disadvantages. [1-2 paragraphs]

4. [24 points]
a) What is ASLR and how does it mitigate attacks? [2-3 sentences]

b) Describe the limitations of ASLR. [1-2 sentences]

c) What is StackGuard and how does it mitigate attacks? [2-3 sentences]

d) Describe the limitations of StackGuard. [1-2 sentences]

e) What is CFI and how does it mitigate attacks? [2-3 sentences]

f) Describe the limitations of CFI. [1-2 sentences]

5. [20 points]

Consider the following code. Assume a 16-bit architecture, that all needed #include directives are present, that each character is stored in 1 byte and each integer in 4 bytes, and that the get_input function returns a user-entered string allocated on the heap.

```
1       int g(char *input) {
2           printf(input);
3           return 0;
4       }
5       int f(char *input) {
6           char a[16];
7           a[0] = 'b';
8           g(input);
9           return 0;
10      }
11      int main(int argc, char *argv[])
12          f(get_input());
13          return 0;
14      }
```

a) Draw a representation of the program memory segments (including their contents when known) right before the printf is executed, at the level of detail shown in class. Assume the system does not use stack canaries but does use an optimized layout of memory, as discussed in class, where printf is not given its own frame.

b) Assuming input is “abcd%p%n”, describe what happens when running the printf, at the level of detail discussed in class. [1-2 sentences]

6. [32 points]
Consider the following function f called by function g. Assume a 32-bit architecture, that all needed #include directives are present, that each character is stored in 1 byte and each integer in 4 bytes, that memory is laid out as in class (optimized version), and that s is user supplied, allocated on the heap, has a max size of 512, and cannot be overflowed.

```
1       int f(char *s) {
2           char a[128];
3           printf(s);
4           gets(a);
5           return 0;
6       }
```

a) Assuming that the system lacks NX bits but is using ASLR and StackGuard, describe how a user could attack this program in such a way that using NX bits would prevent the attack. Describe the attack at the level of detail we described attacks in class, including drawing memory when appropriate. [1 paragraph]

b) Assuming that the system is using NX bits, ASLR, and StackGuard, describe how a user could attack this program. Again, describe the attack at the level of detail we described attacks in class. [1 paragraph]

c) How could the attack you described in Part (b) be prevented? [1-2 sentences]