CNT 4419: Secure Coding [Fall 2023]
Final Exam

NAME: ____

Instructions:

1) This test is 11 pages in length.

2) You have 120 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. Responses will be graded as described on the syllabus. Respond at the level of detail discussed in class. Avoid using bullet points and enumerated lists.

4) This test is closed books, notes, papers, phones, smartwatches, 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] [Short essay]
Briefly describe 5 general principles of secure software development discussed in class.

2. [3 points] Compare and contrast the two main classes of XSS attacks. [2-3 sentences]

3. [2 points]
Intuitively, why do static mechanisms generally exhibit false positives? [1-2 sentences]

4. [2 points] [1 sentence]
What does one typically aim for, regarding soundness and completeness, in practice?

5. [2 points]
What are two criticisms we discussed in class of the CIA classification? [1-2 sentences]

6. [2 points] Why might SIDs be sent in HTTP cookies? [1-2 sentences]

7. [4 points] Explain use-after-free vulnerabilities. What can you as an application-level programmer do to mitigate them? [Short essay]

8. [5 points] [Short *essay* (as always, essays need to be written in prose)]
Name and briefly describe the OSI layers. Make it clear, for example, what Layer 2 is.

9. [8 points] Show and briefly explain 2 example SQLIAs based on (a) “piggybacking” and (b) timing inference.

10. [8 points] [Pseudocode plus 1-3 sentences of explanation]
You want to implement Java web-application functionality to take some relational-database-table-column c as user input and display, for the user, the data in column c of a table t (where t is fixed), at whatever row has a value of uid equal to the value returned by `String session.getUID()` (which looks up the current session's username in the database). What do you do? I.e., how do you code the desired functionality securely?

11. [6 points]

a) Define an example property policy, first using set-builder notation and then using a one-sentence English description. Do not use an example discussed in class or an example that appeared on a previous test. Prove that your example is indeed a property.

b) Define an example nonproperty policy, first using set-builder notation and then using a one-sentence English description. Do not use an example discussed in class or an example that appeared on a previous test. Prove that your example is not a property.

12. [10 points] For each of the following classes of attacks, write a short paragraph summarizing what application-level coders can do to mitigate those attacks.

a) SQL-injection attacks

b) CSRF attacks

c) XSS attacks

13. [4 points] What are the tradeoffs of communicating through a channel where all data is encrypted using AES versus RSA? [Short essay]

14. [8 points] Prove that a policy is both safety and liveness iff it is trivial.

15. [11 points]
Consider a database with tables created using the following SQL DDL statements.
CREATE TABLE players(pID int, name varchar(255), team varchar(255));
CREATE TABLE gamePerformance(gameID int, pID int, playerScore int);

Write a SQL query to do each of the following. Do not include repetitions in the results.
a) Return the names of all current players who have never played in a known game. ("Current" players are those appearing in the players table; "known" games are those appearing in the gamePerformance table.)

b) Return the names of all current players who have scored in a known game.

c) Return the gameIDs for all known games in which a noncurrent player played.

16. [20 points] Consider the following code, assuming a 64-bit architecture, that all needed #include's are present, that each int is stored in 4 bytes and each char is stored in 1 byte, and that getUI securely inputs an unsigned int from the user. Also assume an unoptimized memory layout, where system calls are given their own stack frames. According to its documentation, the 2nd argument to fgets, here an unsigned int, "is the maximum number of characters to be read (including the final null-character)".

```
0       #define MAX 512
1       void readMsg(char *ca, unsigned int size) {
2          char msg[MAX];
3          printf(ca);
4          if(size+1 <= MAX) { //add 1 to be sure there's enough space for the null-char
5               fgets(msg, size, stdin);
6          }
7       }
8       int main(int argc, char *argv[]) {
9          char ca[MAX];
10         unsigned int size;
11         fgets(ca, MAX, stdin);
12         size = getUI();
13         readMsg(ca, size);
14      }
```

a) Describe how an attacker could overflow a buffer in this code. E.g., which buffer could be overflowed, on which line of code, which inputs would the attacker provide, and what would those inputs enable?

b) Using your basic technique from Part a, explain a stack-smashing attack on this code. Assume that the system lacks ASLR, NX, and StackGuard.

c) Describe a format-string attack on this code in detail, assuming the system uses ASLR, NX, and StackGuard with 4-byte canaries. Describe the information an attacker gains from this attack.

d) Referencing Parts a-c as helpful, describe an attack on this code assuming ASLR, NX, and StackGuard are in use.