Attention: This exam has 5 sections and you must answer all of them. You are expected to finish the exam in no more than 150 minutes and only using your notes, the papers, and my lecture presentations to aid you. I’m trusting you to abide by this, and I think it should be easily feasible.

As with any exam, you should read through the questions first and start with those that you are most comfortable with. Be sure to answer all parts of each question you answer. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer. For partial credit, be sure to show how you arrived at your answer as well as the answer itself.
1. Side Channels

Dean Tullsen, a professor in CSE, is one of the inventors of a concept called “Simultaneous Multi-threading”. Microprocessors using this feature implement two independent sets of architectural processor state (i.e., registers) but share all other processor resources like the execution pipeline, L1 and L2 caches and so on. The processor then executes instructions from both “threads” – interleaving them as needed. Suppose you have such a processor and thread #1 is executing a program that is either accessing memory location A or memory location B (and the choice is a secret).

a) Describe what you – as an adversary – might do using thread #2 to determine what memory location thread #1 is accessing? If you feel you need to make any assumptions about the architecture or operating system, please state them accordingly.

b) Describe how you might defend against the attack you developed either by changing the architecture or the operating system?


2. Cross-site scripting

Suppose a Web site specializes its content to each user based on a database keyed on their email address. Thus, my Web page might be:

http://www.securesite.com/search.asp?email=savage@cs.ucsd.edu

The Web page itself constructs a SQL query constructed as follows:

```sql
sql = "SELECT lastname, firstname, nickname, email FROM usertable
WHERE email=’" & Request.QueryString("email") & "’;"
```

a) How might you make a request such that my nickname is changed to “loser”? Explain why this works. (NOTE: SQL syntax is for writing a record field is “UPDATE tablename SET fieldname=”value” WHERE fieldname=”key”, although don’t sweat the details)

b) What could you do to prevent such an attack?
3. Usability

At the core of most usability research is user testing – where users are observed in trying to complete tests under a control condition and using the system under test. However, one criticism of this approach is that since these are inevitably “toy” situations in which the user is playing a “role” that they have no risk and hence do not behave as they would in “real life”. For example, consider testing the effectiveness of anti-phishing indicators on Web browsers (that indicate if a Web site is “suspicious” or not and let user know not to enter personal information).

a) Do you think this criticism is a reasonable one? Why or why not?

b) How might you design a user study to address this problem?

c) (unrelated to parts a and b) Recall Whitten & Tygar’s list of reasons why useable security is difficult. Now consider how this list in the context of anonymity systems such as Tor. Explain what you think this means for building a “useable” anonymity service and where the most significant hurdles are likely to be.
4. Authentication

In 2010, a biometric scanner hits the market that can completely sequence a human genome in 1 second. It quickly becomes popular for use in ATM machines. User’s swab the inside of their cheek (single use swabs are provided nearby, much like check deposit envelopes today) and then insert the swab into the reader, where it is processed. The sequenced genome is then communicated back to the bank where it can be matched against the appropriate customer record.

a) Would you expect this system to solve the problem of authenticating yourself to the bank? Why or why not?

b) What advantages or disadvantages does this approach have as an authentication mechanism when compared with the bank card + PIN approach we use today?
5. Software vulnerabilities

a) In most operating systems today the stack grows “downward” meaning that a new value pushed upon the stack is stored at a lower memory address than the previous value pushed. Thus, if a function’s local buffer variable is overflowed, this will directly write over the function’s own return address. Now, consider what would happen if the stack grew “up” instead of down. That is, new values were pushed on the stack towards higher addresses. How would this impact stack-based buffer overflows? Are they still possible? Explain either way.

b) Suppose you replaced all integer operations with symbolic interpretation. Rather than a native binary representation, you used strings of decimal digits and implemented operations such as addition and subtraction explicitly to create a new string of digits. Assume the use of a safe string class that will return an error if there is insufficient memory to complete an operation. Would such an approach still be vulnerable to integer overflow attacks? Explain.
5. Software vulnerabilities (continued)

c) Recall the race condition vulnerability example from class:

```c
/* access returns 0 on success */
if(!access(file, W_OK)) {
    f = fopen(file, "wb+");  
    write_to_file(f);
} else {
    fprintf(stderr, "Permission denied trying to open %s.\n", file);
}
```

Is the modified version (below) still susceptible to a race condition attack? Explain?

```c
f = fopen(file, "wb+");  
for (i=1; i<10; i++) {
    if (!access(file, W_OK)) {
        tf = fopen(file, "wb+");  
        if (tf points to a different file than f) {
            printf("Race attack!");  
            return -1;
        }
    } else {
        fprintf(stderr, "Permission denied trying to open %s.\n", file);  
        return -1;
    }
}
write_to_file(f);
```