Name: ___________________________

CIS/TCOM 551 Midterm 2
March 31, 2005

1	/10
2	/15
3	/20
4	/25
5	/20
6	/10
Total	/100

- Do not begin the exam until you are told to do so.
- You have 80 minutes to complete the exam.
- There are 9 pages in this exam.
- Make sure your name is on the top of this page.

1. True or False (10 points)

Circle the appropriate answer.

(a) T F C and C++ should be be used only when the programming task requires control over low-level details such as memory layout. (And not even then if you can avoid it. . .)

(b) T F A system with a larger trusted computing base is generally more trustworthy.

(c) T F The epidemic model suggests that blacklisting is much more effective than filtering based on signatures for limiting worm propagation.

(d) T F The source IP address is often a good indicator in worm signatures.

(e) T F A metamorphic worm is unlikely to be caught by current fingerprinting techniques.

(f) T F Recent studies have shown that the reaction time needed to effectively contain advanced worms is less than a few minutes.

(g) T F Perl taint-checking provides needed protection against malicious code.

(h) T F The exception handling mechanisms provided by modern languages like Java and C# are motivated by the principle of least privileges.

(i) T F The Slammer worm did not contain any malicious payload, but caused denial of service due to the overwhelming network traffic it generated.

(j) T F Improved cryptographic techniques have the potential to significantly reduce the number of security problems on the Internet.

2. Code Safety (15 points)

   Consider the following C program:

```
 1   char *copy(char *s) {
 2     char dup[256];
 3     strncpy(dup, s, 256);
 4     return dup;
 5   }
 6
 7   int main(int argc, char **argv) {
 8     char input[256];                /* allocate input string */
 9     char *buf;                      /* buf initialized to null */
10     if (gets(input) != NULL) {
11       buf = copy(input);
12     }
13     printf(argv[1]);
14     printf(buf);
15     return 0;
16   }
```

   There are several bugs in this code. Indicate each bug by giving the line number (or range of line numbers) and a one-sentence explanation.

3. Java Stack Inspection (20 points)

Consider the Java program in Figure 1. It consists of a `Trusted` class and an `UntrustedApplet` class. The `writeFile` method is similar to the example from lecture, but it prints either "A" (for "Allowed") or "D" (for "Denied") to the terminal rather than writing to disk. This program also has additional methods `m1`, `m2`, and `m3` implemented as shown in the figure. Recall from lecture that `enablePrivilege(p)` is a no-op if the code does not have static privilege `p`, `disablePrivilege(p)` adds a mark to the stack frame that causes stack inspection for `p` to fail if reached, and that `checkPermission(p)` throws a `ForbiddenException` if the stack inspection algorithm determines that access should be denied. Assume that stack inspection defaults to denying access if the bottom of the stack is reached.

(a) (6 points) Suppose that the `Trusted` class is in a protection domain with permission set containing `AllPermission` and that the `UntrustedApplet` class is in a protection domain that is associated with only the `FilePermission("/tmp/*", "write")` permission. What is the printed by the program when execution starts at `Trusted.main()`? Your answer should be the sequence consisting of characters "A" and "D" printed to the output.

(b) (6 points) Now suppose that the protection domain for the `UntrustedApplet` class is extended to include both the old permission `FilePermission("/tmp/*", "write")` and a new one, `FilePermission("/home/stevez/*", "write")`. What is the new output of the program when execution starts in the `Trusted.main()` method? Your answer should again be the sequence of "A" and "D" characters printed to the output.

```
class Trusted {
  static SecurityManager sm = System.getSecurityManager();

  public static void main(...) {
    FilePermission fp = new FilePermission("/tmp/*", "write");
    sm.enablePrivilege(fp);
    UntrustedApplet.m1();
    UntrustedApplet.m2();
  }

  static void writeFile(String filename, String s) {
    FilePermission fp = new FilePermission(filename, "write");
    try {
      sm.checkPermission(fp);
      System.out.println("A");            // Access allowed
    } catch (ForbiddenException e) {
      System.out.println("D");            // Access denied
    }
  }

  static m3() {
    FilePermission fp1 = new FilePermission("/home/stevez/*", "write");
    sm.enablePrivilege(fp1);

    FilePermission fp1 = new FilePermission("/tmp/*", "write");
    sm.disablePrivilege(fp2);

    writeFile("/home/stevez/grades.xls", "A+");
    UntrustedApplet.m1();
  }
}

class UntrustedApplet {
  public static m1() {
    Trusted.writeFile("/home/stevez/grades.xls", "A+");
    Trusted.writeFile("/tmp/foo.txt", "cis551 is great");
  }

  public static m2() {
    Trusted.m3();
    Trusted.writeFile("/home/stevez/grades.xls", "A+");
    Trusted.writeFile("/tmp/foo.txt", "cis551 is great");
  }
}
```

Figure 1: A Java program that uses stack inspection.

(c) (8 points) Even though the `fileWrite` method checks the appropriate permissions before writing to a file, it is still possible for badly designed code in the `Trusted` class to permit a security violation. Briefly describe how the `Trusted` code and `UntrustedApplet` code might interact so that a call to p `fileWrite` succeeds (prints an "A") even though its `filename` and `s` arguments can be arbitrarily chosen by the `UntrustedApplet`. Assume that the protection domain permissions are set as in part (a).

4. Worms and Viruses — Modeling (25 points)

Recall that the epidemic model of worm propagation developed in class yielded the following equation for determining the proportion of infected hosts at time t:

$$i(t) = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}}$$

This model was based on the following definitions:

2^{32}	size of IP address space
N	size of the total vulnerable population
$S(t)$	susceptible hosts at time t minutes
$I(t)$	infective/infected hosts at time t minutes
β	Contact likelihood
$s(t) = S(t)/N$	proportion of susceptible population
$i(t) = I(t)/N$	proportion of infected population
T	time (in minutes) at which 50% of vulnerable hosts are infected

(a) (8 points) Suppose that a worm generates target IP addresses uniformly at random and that it attempts to infect $2^7 = 128$ such randomly chosen hosts per minute. If there are 2^{20} (roughly a million) vulnerable hosts on the Internet, what is a reasonable value for β according to the model?

(b) (7 points) As part of your job at the National Internet Security Agency (NISA), you monitor the progress of a new worm called Blister whose behavior is described by the epidemic model above—it generates target addresses uniformly at random and propagates according to the infection function $i(t)$. At time $t = 180$ minutes Blister manages to infect 10,000 hosts (out of a possible 1,000,000 vulnerable hosts). From observing this worm in the wild, you are able to deduce parameters $\beta_{Blister}$ and $T_{Blister}$ that describe it accurately (for this question, their precise values don't matter).

A few weeks later, a new, more aggressive variant of Blister (Blister-v2) appears on the Internet. This version uses an initial hit list of 10,000 vulnerable hosts to start the spread of the worm much faster. After the initial hit list is infected, Blister-v2 spreads at random just like the original Blister worm. Assuming that infecting the hit list takes negligible time, what is the equation for $i(t)$ that models Blister-v2?

(c) (10 points) If $T_{Blister} = 300$ minutes, sketch the graph of $I(t)$ for Blister-v2 that corresponds to your solution to part (b). Label the axes appropriately. Be as precise as you can given the information above, but you *do not* have to evaluate the function $I(t)$ a large number of times—instead, indicate the general shape of the curve and highlight any particular points of interest (e.g. the inflection point of the graph).

5. Worms and Viruses — Automatic Fingerprinting (20 points)

   Describe two assumptions about worm-generated network traffic that are used in the paper "Automatic Worm Fingerprinting" by Singh, *et al.* For each assumption, briefly describe a way that a hacker could program a worm to violate the assumption, thereby making it harder to detect his worm.

6. Software Security (10 points)

   Describe a tradeoff between security and some other desirable property that arises in the context of software development. Briefly explain the tradeoff using a concrete example and suggest a reasonable way to resolve the problem.