Name: ____________________________

CIS 551 / TCOM 401 Midterm 1
February 14, 2006

1	/10
2	/16
3	/22
4	/15
5	/17
6	/20
Total	/100

- Do not begin the exam until you are told to do so.
- You have 80 minutes to complete the exam.
- There are ?? 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 Carrying out a successful stack-smashing buffer overflow attack (as in Project 1) requires access to the source code of the vulnerable program.

b. T F For a message of length m, it is possible to achieve *perfect security* with a polyalphabetic substitution cipher by using m substitutions.

c. T F Testing a software system can show only the absence of security flaws, not their presence.

d. T F Neither the DES nor the RSA encryption algorithm has been shown to be uncrackable except by brute force.

e. T F In the paper "The Protection of Information in Computer Systems," Saltzer and Schroeder introduce a number of design principals for secure systems, including *psychological acceptability*—the idea that security mechanisms should be easy for humans to use and match their intuitions about how they should be applied.

f. T F TEMPEST is a collection of guidelines for reducing the chances that electromagnetic radiation generated by computing devices leaks secret information.

g. T F The larger its trusted computing base (TCB) the more secure a system is.

h. T F If there are n subjects and m objects in an access control matrix, it can take $O(n)$ time to revoke all rights of a given subject when the matrix is represented as capabilities

i. T F In mandatory access control (MAC) settings, subjects do not necessarily have the authority to grant access to objects they own.

j. T F The principal of least privileges suggests that all access to underlying OS resources by a Java applet must be regulated by the Java Virtual Machine.

2. Unix Access Control (16 points)

Recall that RUID stands for "Real User ID", EUID stands for "Effective User ID", and SUID stands for "Saved User ID"; recall also that programs can change their EUID (in restricted ways) by making the `seteuid` system call.

Suppose that a Unix directory contains the following files with permissions set as shown. Assume that all SetGID and Sticky bits are turned off.

File Description			Permissions			
Filename	Owner	Group	SetUID	Owner	Group	Other
`foo.txt`	15	15	-	rw-	r--	---
`bar.txt`	15	99	-	---	rw-	rw-
`baz.txt`	75	75	-	rw-	-w-	---
`quk.txt`	25	25	-	rw-	r--	r--
`wordpro`	25	99	y	--x	--x	---
`Userver`	25	75	-	--x	--x	---

Assume that user `15` is in groups `15`, `99`, and `75`.

Assume that user `25` is only in group `25`.

a. Consider a process running with RUID=25, EUID=25, and SUID=25. Assuming no other process changes any of the file permissions, which of these files could it read?

b. Consider a process running with RUID=15, EUID=15, and SUID=15. Assuming no other process changes any of the file permissions, which of these files could it write?

c. Suppose a process running with RUID=0, EUID=0, and SUID=0 calls the `exec` system call to run the program `wordpro`. Assuming no other process changes any of the file permissions, which of these files could that instance of `wordpro` read?

d. Suppose a process running with RUID=15, EUID=15, and SUID=15 calls the `exec` system call to run the program `Userver`. Assuming no other process changes any of the file permissions, which of these files could that instance of `Userver` execute?

3. Java Stack Inspection (22 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 `t1`, `u1`, and `u2` implemented as shown in the figure. Recall from lecture that `enablePrivilege(p)` marks the stack frame with permission `p` if the code has static privilege `p` and is a no-op otherwise, `disablePrivilege(p)` adds a mark $\neg$`p` 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.

In this problem, we also consider a new stack inspection operation called `revertPrivilege(p)`. The behavior of `revertPrivilege(p)` is to *erase* a mark of `p` or $\neg$`p` that appears on the corresponding stack frame. In the code, there is a line of method `t1` marked with `(*)`. The symbol `<???>` indicates that either "`disablePrivilege`" or "`revertPrivilege`" will be called at that location, according to the scenarios described below.

For each of the following scenarios, give the sequence of "A" and "D" characters printed by the program when execution starts at `Trusted.main()`. (4 points each)

Let:

```
 P1  =  AllPermission
FP1  =  FilePermission("/tmp/*", "write")
FP2  =  FilePermission("/home/stevez/*", "write")
```

a. `Trusted` class is in a protection domain with permissions {`P1`}.
`UntrustedApplet` class is in a protection domain with permissions {`FP1`}.
`<???>` is `disablePrivilege`.

b. `Trusted` class is in a protection domain with permissions {`P1`}.
`UntrustedApplet` class is in a protection domain with permissions {`FP1,FP2`}.
`<???>` is `disablePrivilege`.

c. `Trusted` class is in a protection domain with permissions {`P1`}.
`UntrustedApplet` class is in a protection domain with permissions {`FP1,FP2`}.
`<???>` is `revertPrivilege`.

```
class Trusted {
  static SecurityManager sm = System.getSecurityManager();
  public static void main(...) {
    FilePermission fp = new FilePermission("/tmp/*", "write");
    sm.enablePrivilege(fp);
    UntrustedApplet.u1();
  }

  public 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
    }
  }

  public static void t1() {
    FilePermission fp = new FilePermission("/tmp/*", "write");
    sm.disablePrivilege(fp);
    UntrustedApplet.u2();
    sm.<???>(fp);                         // See problem description      (*)
    Trusted.writeFile("/tmp/bar.txt", "Ceci n'est pas une buffer overflow.");
    Trusted.writeFile("/home/stevez/grades.xls", "A++");
  }
}

class UntrustedApplet {
  public static void u1() {
    FilePermission fp = new FilePermission("/home/stevez/*", "write");
    Trusted.sm.enablePrivilege(fp);
    UntrustedApplet.u2();
    Trusted.t1();
  }

  public static void u2() {
    Trusted.writeFile("/tmp/foo.txt", "Even better than the real thing.");
    Trusted.writeFile("/home/stevez/grades.xls", "C++ = F-");
  }
}
```

Figure 1: A Java program that uses stack inspection.

d. (10 points) As described above, the `enablePrivilege(p)` command is a no-op if the code executing it does not have the statice privilege `p`. How would the security of a Java system change if `enablePrivilege(p)` threw an exception instead of doing nothing in this case? What tradeoff is being made by choosing between these two possible options?

4. Multilevel Security (15 points)

Consider a Bell & LaPadula style multilevel security setting in which all data is labeled with one of four security levels drawn from the following hierarchy (higher is more confidential):

$$\begin{tikzcd}
 & S & \\
Q \arrow[ur] & & R \arrow[ul] \\
 & P \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

P	=	Public
Q	=	Quarantined
R	=	Restricted
S	=	Secret

Consider the following program that calculuates output variables v, w, x, y, and z from input variables p, q, r, and s. Assume that the input variables have security labels that correspond to their names (i.e. variable p has label P, etc.). Assume that all output variables are initially set to 0.

```
v = p + q;
if (r > 0) then {
  w = 1;
  if (v > 0) then {
    x = 1;
  }
} else {
  y = 1;
}
z = s - s;
```

For each output variable indicate the *minimal* security label it can be given so that system is secure according to the information-flow policy. Each answer should be one of {P, Q, R, S}.

- label of v = ______
- label of w = ______
- label of x = ______
- label of y = ______
- label of z = ______

5. **RSA Encryption** (17 points)

Consider the RSA algorithm where $p = 7$ and $q = 11$.

a. (12 points) Find a public–private keypair. For full credit show steps you take.

b. (5 points) Use the public key to encrypt the message whose numerical encoding is 2.

6. Short Answer (20 points)

a. (10 points) Describe two distinct mechanisms of reducing vulnerability due to buffer overflows. In both cases, discuss a tradeoff versus other desirable system properties that must be made to apply the mechanism.

b. (10 points) Recall that three important qualities of a system's security are *confidentiality*, *integrity*, and *availability*. These three properties are very often interdependent.

i. Give an example where the failure to protect confidentiality leads to a compromise of availability. Be as specific as you can.

ii. Give an example where the failure to protect integrity leads to a compromise of confidentiality. Be as specific as you can.