CIS/TCOM 551: Computer and Network Security **Final**
Spring 2005 **Apr. 28, 2005**

1	/10
2	/12
3	/12
4	/28
5	/38
Total	/100

- Do not begin the exam until you are told to do so.
- You have 120 minutes to complete the exam.
- There are 11 pages in this exam.
- Make sure your name is on the top of every page.

1. True or False (10 points)

Circle the appropriate answer.

a. T F Under mandatory access controls, the owner of a file may not be able to grant read access for the file to another user.

b. T F Macro viruses are so successful because most software designers fail to apply the principle of minimizing the trusted computing base.

c. T F A system composed of many different kinds of computing platforms and operating systems is more resilient to viruses and worms than one composed of identical machines running identical operating systems.

d. T F The Internet Protocol (IP) enables a common denial-of-service attack that simply floods a server with too many SYN packets.

e. T F Most recent worms and macro viruses have caused the most damage by corrupting the integrity of sensitive system files.

f. T F In Java and C#, stack inspection provides complete protection against untrusted applet code.

g. T F To achieve the highest Common Criteria rating (EAL 7) the software must be formally verified to meet both its functional and security specifications.

h. T F Recent research on the effectiveness of Internet quarantining estimates that if a worm probes at a rate of more than 60 hosts/second, even if the top 100 ISPs all perform content filtering with instantaneous deployment, it is not possible to contain a worm to a 10% infection rate.

i. T F The principle of least privileges suggests that the only way to allocate or deallocate a system resource should be through a call to the OS.

j. T F It is possible to use the C standard library function `gets` securely.

2. Unix Access Control (12 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	-	--x	--x	---
`Userver`	0	75	y	--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=15, EUID=15, and SUID=15. Assuming no process changes any of the file permissions, which of these files could it read?

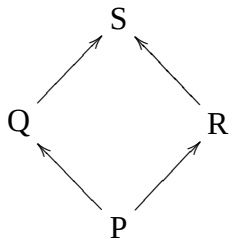
b. Consider a process running with RUID=25, EUID=15, and SUID=25. Assuming no process changes any of the file permissions, which of these files could it write?

c. Suppose a process running with RUID=25, EUID=25, and SUID=25 calls the `exec` system call to run the program `wordpro`. Assuming no 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 process changes any of the file permissions, which of these files could that instance of `Userver` write?

3. Multilevel Security (12 points)

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



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

Assume that the data labels are applied at the file level of granularity so that the entire contents of a file is given one label.

For each of the following questions, your answers should be a subset of {P, Q, R, S}.

a. Users with which clearance level(s) can read a file containing data derived from two sources, one labeled P and the other labeled Q?

b. Users with which clearance level(s) can read a file containing data derived from two sources, one labeled Q and the other labeled R?

c. Users with which clearance level(s) can write data to a file labeled Q?

d. Users with which clearance level(s) can write data to a file labeled S?

4. Short Answer (28 points, 7 points each)

a. Briefly explain what a worm or virus signature is and describe an example of such a signature that might be used to detect attacks against the Blame server used in the projects.

b. Recall that network-based intrusion detection systems (such as Bro) observe network traffic to look for possible attacks or malicious behavior. Describe two features of network traffic that make it difficult to build such a monitoring systems.

c. What is a certificate authority and what is it used for?

d. Describe and advantage that access control lists have over capabilities and an advantage that capabilities have over access control lists.

5. Security Analysis (38 points)

In this problem, we consider a new hardware-based security mechanism that has recently been proposed by researchers at Stanford.[1] The idea is to provide cryptographic hardware support for tamper-resistant software through a technology called eXecute Only Memory (or XOM for short). Intuitively, XOM processors provide a kind of memory that can only be executed, not read or written. Such memory is useful for storing software.

Here are the important features of the XOM hardware:

- Each XOM processor P has a unique RSA public/private key pair K_P, k_p. The private key k_P is embedded in the hardware and is not available in any other way. The public key K_P is available to anyone, but in particular, the owner of the hardware has access to it.
- When XOM software $prog_i$ is purchased, the software is encrypted by the vendor with a freshly generated symmetric key K_{Si} (e.g. for AES). The symmetric key is encrypted by the processor's public key and both the encrypted software $K_{Si}\{prog_i\}$ and the encrypted key $K_P\{K_{Si}\}$ are installed on the disk (note that the disk is *not* trusted).
- When the user wants to run $prog_i$, the XOM processor can decrypt the shared key K_{Si} and use that software key to access the code for $prog_i$. The unencrypted code never leaves the processor chip.
- The XOM processor may execute several different programs $prog_1 \ldots prog_n$ (each with different symmetric keys $K_{S1} \ldots K_{Sn}$) in an interleaved way to provide multithreading. To prevent code and data from different programs from mixing, XOM associates with each program $prog_i$ a *compartment tag* t_i, which is associated with key K_{Si} in a table in the processor's core. When the XOM processor is executing $prog_i$, all data it manipulates is tagged with t_i. Any attempt to read or write to data with a tag other than the one currently running results in an exception.
- Whenever data with tag t_i leaves the processor (to be written to main memory, for example) it is encrypted and hashed with key K_{Si}. Whenever data is loaded into the processor (by reading it from main memory, for example) the hash is verified to check for data integrity.
- For backwards compatibility and for certain kinds of I/O, the XOM processor supports a "null" tag t_0 that means that there is no encryption or hashing of that code or data. Any process can explicitly choose to relabel its data to the null tag using a special `moveToNull` instruction, which means that any program might be able to read such data. Similarly, and process can explicitly import possibly untrustworthy data by using the `moveFromNull` instruction (this is necessary to obtain user input, for instance).
- XOM makes use of some fairly sophisticated hardware tricks to mitigate the performance overheads introduced by encryption and decryption, resulting in a 50% overhead in execution speed.

[1] See `http://www-vlsi.stanford.edu/~lie/xom.htm`.

The XOM architecture is intended to provide strong software tamper resistance even in the case where the memory and operating system are not trusted. XOM can potentially be applied to the problem of digital rights management: an application vendor can use XOM to prevent hackers from reverse engineering their software—this would make it difficult for users to crack software and enable vendors to use proprietary standards for things like music, video, and game formats.

a. (5 points) Suppose that a program is running with a non-null compartment tag (meaning that its code and data are encrypted). Does XOM help prevent buffer overflow attacks against it? Explain.

b. (5 points) Suppose that an e-mail client is running with a non-null compartment tag (meaning that its code and data are encrypted). Does XOM help prevent macro viruses that target this e-mail client? Explain.

c. (5 points) Even if *all* of the data output by a program is encrypted, there are still covert channels available to learn some information about a program running in the XOM processor—for example, one might watch the processor's power consumption. Describe another example of a covert channel (other than power consumption) that might reveal information about a XOM program.

d. (8 points) In order to allow the OS to manage system resources, the XOM architecture must allow a program with tag t_i to move to memory, but not tamper with or read, data owned by a program with tag t_j. To facilitate this operation, XOM provides a pair of operations `saveSecure` and `restoreSecure`.

Suppose that some machine register r_3 contains the data value v with tag t_j. What should the XOM architecture write to memory address $addr$ when it executes the instruction "`saveSecure` r_3 $addr$" so that when the corresponding "`restoreSecure` $addr$ r_3" instruction is invoked, the processor can validate and restore the contents of r_3 exactly to its former contents? Note that your implementation (like XOM's) should prevent replay attacks due to an untrusted main memory, so some additional state is needed. Briefly explain your implementation strategy.

e. (7 points) If two XOM programs wish to share data (for instance to allow for cut-and-paste between a text editor and an e-mail client), but they don't want other programs to (potentially) see the data, they must establish a secure channel using the "null" tagged memory. Briefly sketch a way of achieving this. Assume that the two programs have tags t_1 and t_2 (neither of which is null). Don't worry about the low-level implementation details, but give the high-level protocol the programs should follow.

f. (8 points) Is XOM a good idea? Who would want to use this technology and why? What tradeoffs are involved when using this technology? What are its drawbacks, limitations, and advantages? (This question is intentionally open ended.)

(Extra space.)