CSE127 — Midterm

Yee **Winter '03**

Name and Student ID: Answer Key

Wait until everybody has a copy before you start. This exam is closed book, closed notes.

There are a total of 14 questions on 8 pages. There are 105 points possible. You might not have time to finish the entire exam—don't be discouraged. Advice: skim through the entire test to determine which of the problems you can solve quickly and work on those first, rather than getting stuck on a hard problem early and wasting too much of your time on it.

When you start, you should first make sure that you have all the pages, and write your name and your student ID on the first page, and your student ID on the top of *all subsequent pages*. Pages of this exam will be separated and graded separately—if you fail to write your ID at the top of a page, you will not receive credit for answers on that page. **Write clearly**: if we cannot read your handwriting or your pencil smudges, you will not properly get credit for your answers.

No electronic computation aids are allowed.

Prob	1	2	3	4	5	6	7	8	9	10	11	12	13	14	**Total**
Score															
Poss	3	12	3	5	5	6	8	8	10	10	10	10	10	5	105

1 (Class basics) What does the compute security code of ethics which you signed say? (You do not need to have it memorized; a paraphrasing is okay.)

(3pts)

I will not use the knowledge learned from CSE 127 to break into computer systems uninvited. I will not share that knowledge with others unless they also agree to the Computer Security Code of Ethics.

2 (Basic Concepts) Give definitions of the following

A. Confidentiality

B. Integrity

C. Accountability / Traceability / Non-Repudiation

D. Availability

(12pts)

A. Confidentiality: Only authorized users are allowed to *read* confidential data files; others are denied access.

B. Integrity: Only authorized users are allowed to modify (*write*) protected files; others are denied access.

C. Accountability: System activities are logged so that users can be held responsible for their actions.

D. Availability: Otherwise unused system resources remain available for use, rather than be tied up inappropriately.

3 (Basic Concepts) What is the KISS rule?

(3pts)

____ A. Use excessive makeup when trying to impress clients.

____ B. No kissing in class/the work place.

____ C. Krispy-kreme Indulgence Syndrome Sufferers: munchies attacks caused by computer insecurity worries.

_√_ D. Keep It Simple, Stupid: keep designs simple when possible to avoid bugs.

____ E. Kernel Interface Stabilization System: using it eliminates kernel-side race conditions.

4 (Basic Concepts) What is a Threat Model? What is Risk Management? Explain what these concepts are and how they relate to computer security.

(5pts)

Threat Models and Risk Management form part of the security model. To build a threat model, we identify the potential threats to the system—who the attackers are, what the means and methods of attack might be, e.g., sources of risks such as known as well as undiscovered bugs, operational security risks, physical security risks, etc.

After identifying the threats in threat modeling, we estimate the likelihood that the threats will materialize, determine the security assets that we wish to protect, and identify the risk mitigation strategies that we may wish to deploy and gauge their effectiveness and cost. The expected loss is the risk that must be managed—e.g., by chosing the appropriate risk mitigation technique (cost-effective security mechanisms or tighter usage policies)—so that the expected cost becomes acceptable.

5 (Expectation Values) Suppose you are invited to play a new (gambling) game g: Roll a (hypothetical) 10 sided fair die, and the dealer pays you according to the following payoff table

die value	payoff	die value	payoff
1	\$1	6	\$5
2	\$1	7	\$5
3	\$1	8	\$10
4	\$2	9	\$10
5	\$2	10	\$50

(1) What is the expected winnings from playing one game, i.e., what is $E(g)$? (2) If you have to pay the dealer \$10 to play g, should you play it?

(5pts)

(1)

$$\begin{aligned} E(g) &= \sum_{e \in \text{Events}} P(e) \cdot \text{value}(e) \\ &= \frac{1}{10} \cdot \$1 + \frac{1}{10} \cdot \$1 + \frac{1}{10} \cdot \$1 + \frac{1}{10} \cdot \$2 + \frac{1}{10} \cdot \$2 + \\ &\quad \frac{1}{10} \cdot \$5 + \frac{1}{10} \cdot \$5 + \frac{1}{10} \cdot \$10 + \frac{1}{10} \cdot \$10 + \frac{1}{10} \$50 \\ &= \frac{1}{10} \cdot (\$1 + \$1 + \$1 + \$2 + \$2 + \$5 + \$5 + \$10 + \$10 + \$50) \\ &= \frac{1}{10} \cdot \$87 \\ &= \$8.70 \end{aligned}$$

So the expected payoff for playing a game is \$8.70.

(2) Since this game costs \$10.00 to play, the net result is a loss of \$1.30 per game. I should not play this game.

6 (Defense) Suppose a security consultant to your company suggested that you should buy a new security device to protect your company's computers. The device costs $170,000 for the first year (initial licensing, training, etc), and $10,000 per year to operate thereafter (license renewal, update, staff costs). The consultant assures you that the device will completely eliminate security incidents from the average of 5 per year, which costs your company about $10,000 per incident (in labor, lost customer confidence, etc). There are, of course, uncertainties for all of these estimates/claims.

(1) Suppose this device operates as claimed for the forseeable future (at least, oh, 5 years). Should you adopt this device?

(2) Suppose you believe that the attackers will learn about how the device works and develop new, more sophisticated attacks that will be bypass it. You estimate that after 4 years of use the device will provide only partial protection, lowering the incidence rate to one per year, should you adopt this device?

(6pts)

(1) Yes. If the device continues to operate as claimed for at least 4 years, it have cost the company $\$170,000 + 3 \cdot \$10,000 = \$200,000$ and saved the company $4 \cdot 5 \cdot \$10,000 = \$200,000$; this is when the cross-over occurs: at each subsequent years the company saves $\$40,000$ per year.

year	cumulative cost	cumulative savings	net savings
1	$170,000	$50,000	-$120,000
2	$180,000	$100,000	-$80,000
3	$190,000	$150,000	-$40,000
4	$200,000	$200,000	$0
5	$210,000	$250,000	$40,000

(2) Yes, after 4 years we break even, and thereafter we save $30,000 per year.

year	cumulative cost	cumulative savings	net savings
1	$170,000	$50,000	-$120,000
2	$180,000	$100,000	-$80,000
3	$190,000	$150,000	-$40,000
4	$200,000	$200,000	$0
5	$210,000	$240,000	$30,000
6	$220,000	$280,000	$60,000

7 (Attack) What are "buffer overflow" bugs? How can they let attackers penetrate a system?

(8pts)

Buffer overflow bugs occur in languages where there is no array-bounds checking. If input can be provided to a program which causes array indexing to go out of bounds on a write operation, unintended memory changes can occur; in the common case of an input buffer array allocated in stack memory, by overflowing such an array an attacker can change control state such as return addresses to cause control to be transferred to inappropriate code (or to data that is interpreted as code).

8 (Concepts) What is a Trusted Computing Base? Name some of the typical components.

(8pts)

A Trusted Computing Base (TCB) is that portion of the system that must be trusted—it implements the security mechanisms which enforces the system security policy. Typically this includes the hardware, the operating system kernel, the user authentication subsystem (e.g., login program), and system administration tools. The compiler may be part of the TCB—when compiling and installing new system software from source is part of the system administration activity—or might not, e.g., when the system is used for running programs and the programs are written and compiled elsewhere.

9 (Algorithms) Give the modular exponentiation algorithm described in class.

(10pts)

```
Integer mod_exp(Integer x, Integer n, Bit e[])
{
        int i;
        Integer y = 1, z = x;

        for (i = 0; i < e.length; i++) {
                if (e[i] == 1) {
                        y = y * z mod n;
                }
                z = z * z mod n;
        }
        return y;
}
```

10 (Differential * Analysis) What are statistical characteristics used in differential timing/power analysis of modular exponentiaton? Why are they important/how are they used in a DTA/DFA attack?

(10pts)

The characteristic must exhibit *data dependence* as well as *key dependence*—in differential timing analysis of modular exponentiation, the execution time of the modular multiply depends on the input values which are under external control; in differential power analysis, it is the amount of power consumed that depends on the input. In both cases, whether the characteristic is present in a measured value or not depends on the value of a bit in the hidden key. The data dependence permits measurements to be classifed and the presence or absence of the characteristic determined statistically. The presence or absence of the characteristic reveals information about the value of the key which we can use to refine the search. For modular exponentiation, the characteristic is the execution time/power associated with

```
if (e[i] == 1) {
        y = y * z mod n;
}
```

The cost of performing this modular multiplication, if all lower order bits of the exponent is known and thus the inputs y and z are known, can be determined from a timing (or power) model. By classifying overall timing (or power) values according to our predictions, we should detect this signal if the exponent bit is one and detect nothing if it is zero. This allows us to determine the exponent one bit at a time, starting at the low-order bits.

11 (Attack) Describe the Tenex password attack and explain clearly how it works.

(10pts)

In the Tenex operating system, switching to another user ID is done by an application supplying a username and password via a system call to the kernel. The kernel has access to a small file with the correct password for all the users, and after lookuping up the user the kernel performs a string comparison between that and the password provided by the application. The string compare operation terminates as soon as the result is known, i.e., it returns a failure at the first unequal character.

By accessing memory held in different pages of memory, a program can control which virtual pages will be RAM resident and which will be paged out to disk. Using this control and by allocating the string buffer containing the password string so that it spans a page boundary, we can measure the switch-user system call response time to detect whether the password string is correct on a character-by-character basis: start with the first character in a RAM resident page and the rest in a page that is paged out. Incorrect guesses for the first character will return quickly, without paging in the page containing the rest of the string buffer. A correct guess, however, will cause the other page to fault in before seeing the next (incorrect) character of the password string, and the extra time required is an easily detectable characteristic. Repeat by shifting the string one byte lower in memory so that one unknown character is in the RAM-resident page and testing as before, until all the password characters are determined. This is a linear time operation.

12 (Networking) Describe how `inetd` can reduce system resource usage.

(10pts)

Instead of having daemons start at boot time and consume process table slots and paging space, we run them as "inferior daemons" under `inetd`, the master daemon. At system boot `inetd` reads a configuration file `/etc/inetd.conf` describing all the inferior daemon service ports and programs which implement them, and `inetd` creates sockets for each of those ports and accepts connections on all of them. Only when a client connect to one of these ports do the inferior daemon get run: `inetd` forks a subprocess which sets up I/O descriptors so that the socket corresponding to the client becomes the standard input and output of the inferior daemon program.

13 (RPCs) Explain what *marshalling* means and explain when it occurs in the context of RPCs.

(10pts)

The term *marshalling* refers to the process of copying data into a message buffer. When the client-side stub is invoked, the input arguments are marshalled into a request message, which also includes an RPC number to identify the request. This request is then sent (over the network) to the server. The server unmarshalls the arguments and invokes the service routine, which returns with output values. The output values are marshalled into a response message, which is sent as a reply to the client. The client stub unmarshalls the output values and returns them to its caller.

14 (Bonus, work on this only when done with everything else) Consider the following game $g(N)$: You pay \$$N$ to the dealer to play. The dealer flips a fair coin. If it comes up heads, you win \$$2N$ from the dealer (net win of \$$N$). Otherwise you get nothing. (1) Should you play?

Consider the strategy $\mathcal{S}$. You pay \$1 to the dealer to play $g(1)$. If you win, you stop, having won \$1. If you lose, you pay \$2 to the dealer to play $g(2)$. If you win, you will have won a net of \$1 and you will stop. If you lose, have lost a net of \$3, so you pay \$4 to the dealer and play again at a higher stake. Eventually at some game $g(2^k)$ worth \$$2^k$, you will win, and you will then have won a net of \$1. It appears that by utilizing $\mathcal{S}$, you will always win a dollar by playing many games of $g(N)$. (2) Should you play using strategy $\mathcal{S}$?

(5pts)

(1) g is a fair game.
(2) It is impossible to convert a sequence of fair games into a non-fair game. The proper analysis involves taking the limit of a sequence of finite-length games played using this doubling strategy:
Let G_k be the game where we play g according to the described strategy, with at most k steps, so $G_1 = \{g(1)\}$, $G_2 = \{g(1), g(2)\}$, ..., $G_4 = \{g(1), g(2), g(4), g(8)\}$, ...
For all k, the expectation value of G_k is 0, so $\lim_{k\to\infty} E(G_k) = 0$.