COMS 4995-2: Introduction to Security — October 2005

Rules

- Remember to write your name and UNI on the blue exam book.
- No book, notes or calculators are allowed during this examination.
- This is a time-limited test. All papers must be turned in 75 minutes after the beginning of the test.
- The total points add up to 100.
- Good luck, and may the Force be with you.

Questions

1. (5 points) Explain the role and limitations of cryptography in computer security.

Cryptography provides confidentiality and integrity protection. It is also used in many authentication techniques. (It's sometimes possible to use cryptogrpahy to protect availability, but we haven't discussed that in this clas.) However, most security problems are due to buggy software. Cryptography does not help with that.

2. (15 points) The following fragment of C code, which creates a unique temporary file and returns a file descriptor for it, contains a security flaw. Identify it, and explain how to fix it. (The code is syntactically correct and actually runs.)

```
struct timeval tv;
char buf[200];
int try;

gettimeofday(&tv, NULL);
for (try = 0; try < 26; try++) {
        sprintf(buf, "/tmp/%ld-%ld-%c", tv.tv_sec,
            tv.tv_usec, 'a'+try);
        if (access(buf, F_OK) < 0)
                return open(buf, O_WRONLY|O_CREAT, 0666);
}
return -1;
```

Notes: the `F_OK` operand to `access()` checks if the file exists and is reachable by the real UID. `O_WRONLY` means "open the file for output only"; `O_CREAT` means "create it if it doesn't exist". `0666` is the mode for this file, before modification by `umask`.

There is a race condition between the `access()` and `open()` calls. The file might not exist when `access()` is called, but someone might create a link to some other file before `open()` is called. The simplest solution is to use the O_EXCL flag, which will cause the `open()` to fail if the file exists. The `sprintf()` call is *not* a problem; the enemy can't tamper with the time of day.

3. (5 points) In what sort of environments is mandatory access control useful? Why?

Mandatory access control is useful when individual users should not have permission to change file permissions. This is most commonly used in the government, to prevent declassification of files by those not authorized to make such decisions. It's also useful in a situation where so-called "Trojan Horse" programs are a risk, since MAC will prevent such programs from changing file permissions and then overwriting the files or stealing their contents.

4. (10 points) Describe the limitations of biometric authentication. But it does work well in some environments. Explain what characteristics would make biometrics a good choice.

Biometrics suffer from usage problems (staring into scanners, the association of fingerprints with criminals), false positive and false negative rates, fake body parts, "bit replay", non-reproducibility, and lack of sufficient backup body parts. They can work well if accepted locally or under observation, especially if biometrics are used to unlock a local authentication token.

5. (15 points) Alice encrypts the message $P_1, P_2, \ldots, P_8$ to Bob using a shared key and Cipher Block Chaining (CBC) mode, yielding the message

$$IV, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8$$

An enemy tampers with the message, so Bob receives

$$IV, C_1, (C_2 \oplus 1), C_3, C_6, C_7, C_8, C_4, C_5, C_6$$

where $\oplus$ is exclusive-OR. What plaintext will Bob see after decryption? If you invent any new symbols, define them.

Note: it may help if you remember the basic equations for CBC mode:

$$\{P_i \oplus C_{i-1}\}_k \rightarrow C_i$$
$$\{C_i\}_{k^{-1}} \oplus C_{i-1} \rightarrow P_i$$

Remember the basic error propagation properties of CBC: if a block of ciphertext is changed, the corresponding block of plaintext is garbage, while the following block is changed predictably. Bob thus sees the following for each block of the message:

Message	Plaintext	Decryption Operation
IV	—	
C_1	P_1	
$(C_2 \oplus 1)$	garbage	$\{\{P_2 \oplus C_1\}_K \oplus 1\}_{K^{-1}} \oplus C_1$
C_3	$P_3 \oplus 1$	$\{\{P_3 \oplus C_2\}_K\}_{K^{-1}} \oplus (C_2 \oplus 1)$
C_6	garbage	$\{\{P_6 \oplus C_5\}_K\}_{K^{-1}} \oplus C_3$
C_7	P_7	
C_8	P_8	
C_4	garbage	$\{\{P_4 \oplus C_3\}_K\}_{K^{-1}} \oplus C_8$
C_5	P_5	
C_6	P_6	

Strictly speaking, C_6 and C_4 do not decrypt to pure garbage; C_4, for example, decrypts to $P_4 \oplus C_3 \oplus C_8$. However, since C_3 and C_8 are not controllable by the attacker, the change is knowable but not controllable. Contrast that with the change when decrypting C_3, where the attacker precisely controls the difference in the resulting plaintext. C_2 decrypts to pure garbage, since the ciphertext being fed to the block cipher for decryption is not the result of any known encryption.

6. (20 points) Consider the following set of Unix permissions:

Path	*User,Group,Other*	*File User*	*File Group*
`/`	`rwx,r-x,r-x`	`root`	`root`
`/home`	`rwx,r-x,r-x`	`root`	`root`
`/home/smb`	`rwx,r-x,--x`	`smb`	`faculty`
`/home/smb/upload`	`rwx,-wx,-wx`	`smb`	`faculty`
`/home/smb/upload/td`	`rw-,---,---`	`smb`	`faculty`
`/home/smb/upload/xy`	`rw-,r--,r--`	`robin`	`student`
`/home/chris`	`rwx,--x,--x`	`chris`	`other`
`/home/chris/proj`	`r-x,rwx,---`	`chris`	`proj`

Below is a list of commands, preceded by the user.group of the process. Which commands will succeed? For command (e), assume that the process's current working directory is `/home/smb/upload`. For commands that fail, explain which permission entry caused the failure.

```
(a)  chris.root     date >/home/chris/proj/now
(b)    pat.student  echo data >/home/smb/upload/p1
(c)  chris.other    mv /home/smb/upload/td /home/chris
(d)    joe.student  ls -l /home/smb/upload/xy
(e)    sue.faculty  cd ../../chris
```

a Fails. Chris does not have write permission on `proj`, in the last rule.

b Succeeds — the `upload` directory is world-writable, and the path to it is searchable

c The intended answer: Succeeds — you need write permission to delete a file from a directory or to add a file to one; chris has write permission on both directories.

However, there was a flaw in the question. I had assumed that the two directories were on the same file system. If they're on different file systems, the `mv` operation is implemented as a copy, which requires chris to have read permission on the file. But that's not true, so the operation would fail.

d Succeeds — joe has x permission on all of the directories, and hence can trace a path to the file. The permissions on the file itself are irrelevant. Trying to list `/home/smb/upload` would have failed.

e Succeeds — `..` is a path, and requires `x` permission, not `r`.

7. (20 points) You're designing a secure communication system for a company with five employees. All five will use this system to connect over the Internet to the company's server. Only standard PCs are available. Sketch a design. Include discussions of authentication, encryption types, key generation, and operating system mechanisms needed.

There is no one right answer to this question. A number of possible answers are correct *if* they're properly justified. That said, there are certain principles that do constrain the answer.

The company is quite small. This implies that it has limited resources for computing. Anything requiring custom work is likely to be impossible. This suggests, for example, that there will be no special random number generation hardware available. Similarly, there is probably no money for secure, dedicated authentication servers. This suggests that software random number generators — see the next question! — are needed for key generation; similarly, it suggests that either passwords or certificates purchased from a commercial CA would be used for authentication. (If you noted that some brands of computers, such as Thinkpad laptops, come with fingerprint readers, that's acceptable.)

Nothing special is needed in the way of encryption. If certificates are used, a hybrid public key scheme can be used. LIfe is harder if only passwords are used, though there are systems — SSL, for example — that can use passwords for authentication but public key technology for traffic key distribution. Finally, if you assume that virtually all communication is from the employees' machine to a central file/email/web server, it's possible to use passwords as a key-encrypting key, and use that to transfer a traffic key. Any of these answers are acceptable *if justified*.

8. (10 points) You wish to write a system library function that provides random numbers for cryptographic purposes. Describe how you would construct such a function. Assume an ordinary PC with no special hardware.

You need sources of very unpredictable numbers. Good choices are the *low-order* bits of inter-character time, mouse position and timing, disk read timing, packet arrival, and high-precision time of day, and perhaps process ID. Most PCs have sound cards, so it's reasonable to turn the gain up to the maximum and look at the low-order bits of the line or microphone input. In addition, a file with a few hundred bytes of randomness can be consulted; the contents of the file should be stirred together with the new inputs and rewritten.