Security Properties

- Confidentiality, integrity, availability
- Privacy
- Vulnerability, attack, threat
- The role of humans

Host Security

- Hardware features — timers, memory protection, mode bit
- Multics and ring structures
- Principle of Least Privilege
- Role of OS

File Permissions

- Unix file permissions
- ACLs
- DAC and MAC
- Complex access control, and how to fake it

Labeled Security

- Lattice model
- Protecting integrity
- *-property and other formalisms

Privileges

- What are privileges?
- What are some sample privileges?
- Acquiring privilege
- SetUID
- The problem with setUID
- Message-passing

Cryptography

- Keys
- Symmetric ciphers
- Stream and block ciphers
- Modes of operation (what they are)
- Public key cryptography
- Hash functions; HMAC

Public Key Cryptography

- Basic properties
- Digital signatures
- Message integrity
- Certificates; CAs

Authentication

- Types of authentication
- Passwords and their problems
- How to store passwords
- Salt
- Tokens

Biometrics

- Types
- Advantages and disadvantages
- Systems properties

Authentication Scenarios

- Parties
- Training
- Scale
- Environment
- Resource being protected

Case Studies

- What are the roles?
- Who has what permissions?
- Threats
- Role of scale
- Role of tools
- Role of privilege

Software Security

- The role of bugs
- Specification errors
- Format string errors

Buffer Overflows

- What are they?
- Why do they occur?
- How to defend against them?

Input Validation

- What’s the problem?
- Specifications
- Defenses?

Other Holes

- Macro injection
- Environment variables
- File descriptors

File Name Parsing

- When it’s needed
- The “..” problem
- Application syntax

TOCTTOU

- Attacking
- Defending

Opening Files

- Shed privilege
- File descriptor passing
- Lock directories

Protecting Clients

- Network clients
- Mobile code
- Smart cards
- Physical attacks

Designing Systems

- Location of the value
- Role of tokens
- Balancing needs

DRM

- General approaches
- How it works
- How it fails
- Watermarking

Cryptographic Engineering

- Handling keys
- File and disk encryption
- Cryptographic hardware

Random Numbers

- Importance
- Generating them
- Hardware vs. software

Server Security

- File permissions
- Application-enforced permissions
- The role of users
- The role of scripts

Keys and Passwords

- Storing them
- The difficulty of password-changing

Confinement

- How to confine applications?
- What are we protecting?
- Various primitives
- VMs, sandboxes
- Covert channels

Chroot

- What is it?
- What does it do?
- How can it be attacked?
- What are its limits?

Test Conditions

- Open book
- Open notes
- No computers
- 75 minutes

Excerpts from a Previous Midterm

Full questions and answers available on my Fall ’05 web site

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.

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.)

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.

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.

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.

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*.

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.