6.857 Computer and Network Security: Lecture Notes (Draft)

Part 1 : Introduction

Version: September 14, 1999

This course is about computer and network security. It is intended primarily for seniors and first-year graduate students. Students should feel comfortable with computer systems, networks, algorithmic concepts and modular arithmetic. The material covered in this course has several different styles:

Case studies: applications, 'hacks and attacks' including electronic voting, Hotmail issues, Y2K vulnerabilities, e-commerce, electronic tollbooths, intellectual property and software distribution.

Principles and Perspectives: as opposed to the bottom-up approach of case studies, looking at the principles and perspectives of security allows us to take a top-down approach.

Techniques, mechanisms, tools for achieving more secure computer systems, including encryption, digital signatures, and operating system approaches such as memory access protection.

Specific topics covered in this course may include:

Physical security: Computer site security. Magnetic media. Tempest.

Cryptography:

Unconditional security: One-time pad. One-time authentication.

Secret-key cryptography: DES, AES.

Public-key cryptography: RSA, El-Gamal; digital signatures.

Crypto protocols: Secret sharing, oblivious transfer, commitment, certificate authorities, zero–knowledge proofs, key exchange, key agreement. Pay TV security.

Crypto policy: Key recovery, export regulations.

Computer Security: Authentication, authorization, auditing, computer viruses, covert channels, secure operating systems, smart cards, and biometric technologies like thumbprints and eyescans. We may also discuss formal models for computer security, modes of risk analysis, methods of attacks, and security testing.

Network Security and advanced applications: We have a special focus on the Internet. We will talk about the Internet Protocol and its security issues and study firewalls and network protocols like Kerberos. We also will look at the World Wide Web, electronic commerce, and electronic payments. In addition, we will study electronic voting and electronic intellectual property protection.

1 Security Objectives

We will learn how, given a computer system, we can identify and possibly solve its security problems. This means that we have to define what we mean by security. The computer system can be anything you may think of: a shared system, a Web server, a bank ATM, the cable-box on your TV, and so on. For each case, defining security means specifying a *security policy*—a set of desired goals. For example, in an electronic voting system, we decide to set up the system so that only registered voters can vote; for a Web server, we may decide that clients accessing the server should pay a fee; in a shared computer system, only authorized users should be able to log in; and so on.

A security policy may be motivated by a desire to protect resources from attack or misuse (e.g. a password is required to login to a computer), to ensure correct operation (e.g. in an electronic voting system), or to comply with relevant law (e.g. requirements for protecting the confidentiality of patient information). European laws on individual privacy tend to be much more stringent than U.S. laws.

Once a security policy has been specified we need to implement *security mechanisms* to make sure that the desired goals are met. For example, a verification process for registered voters should ensure that only registered voters can vote. A payment protocol should ensure fee payments for clients accessing a Web server. A password scheme should give access to a shared system only to authorized users.

Once the security mechanisms are put in place, they need to be reviewed in order to assess possible *vulnerabilities*, i.e., weakness that leave the system open to attacks. Most systems are not bulletproof. Hostile parties who exploit weaknesses in order to create damage are known as *threats*.

If a vulnerability is identified, then a *safeguard* or *countermeasure* must be designed to eliminate or reduce the weakness.

1.1 Security Policy: Goals

There may be many different security goals depending on the purpose of the computer system. Many goals fit, however, into the following three broad categories:

Confidentiality: Contents of a computer file or of a message sent across a network should not be accessible by unauthorized parties. For example, if all the employees of a company are equipped with Global Positioning System (GPS) locator devices, the goal of confidentiality would dictate that information should only be disclosed in accordance with the security policy: only certain people should be allowed to access the location of an employee.

Integrity: Contents of a computer file or of a message sent across a network should not be modifiable by unauthorized users. For example, only professors should be able to modify grades, and no one should be able to alter them if they are transmitted across a network. Businesses are often much more worried about integrity than confidentiality.

Availability: The system should be available to authorized users. A *denial of service attack* is an attack that infringes upon availability of the system. For example, students should be able to view their own grades.

Other goals may also come up. For example, in an electronic payment protocol, anonymity of the payee might be required. Such an example does not seem to fit neatly into any of the categories above (although confidentiality is probably the closest).

Often a system may involve several parties with conflicting goals. In an electronic payment system, the interests of the bank, the customer, and the merchant may be different. Each party may have his own security requirements.

1.2 The Design of a System

The evolution of a security system usually involves a loop of three main steps:

Design: Starting from a given system purpose and associated security policy, a system design is created, and designers present an argument that the system design is secure.

Implementation: The system is implemented according to the specified design, and its security is tested and verified to the extent possible.

Breakdown: The security system fails. The process returns to design, possibly with revised system purpose and/or security policy.

Security mechanisms evolve over long periods of time. For example, the current design of bank vaults took years to evolve; they were improved as thieves found new ways to break into them. Similarly for the design of pay phones.

It is difficult to know precisely when security has been achieved because it is much harder to prove a negative (that no security hole exists) than it is to prove a positive (that a concrete security flaw exists). Also, a security mechanism can be proved correct, but doing so requires assumptions about what potential adversaries can do (passive attacks, active attacks, etc.). Such assumptions are sometimes difficult to pin down and often new ones arise as situations change.

1.3 Countermeasures and Safeguards

We will explore many possible security mechanisms that help eliminate vulnerabilities in computer systems. Some typical ones follow.

Identification: Identifying the person who performed an action can allow accountability, which can often deter a threat from exploiting a vulnerability.

Authentication: Along with identification comes authentication – requiring users to prove their identity before they can perform an action. This will reduce the risk of unauthorized people making use of the system.

Access control: Only people authorized to have privileges to certain machines, files, etc., should have access to them. In UNIX, for example, access control lists specify which users can read, write, execute a given file.

Redundancy: Duplicating data (backups) or equipment can reduce the risk of loss of unavailability.

Auditing: Recording all the security-relevant activities that take place in a system makes it possible to detect security breaches that could not be prevented.

Physical protection: Locks and keys prevent unauthorized access to buildings where computer systems reside. Shredders make snooping in the trash harder. Degaussers can actually destroy the contents of a disk file, compared to a DELETE or ERASE command that typically just removes a pointer to the file.

Cryptography: As we will see later in the class, encryption and other cryptographic techniques can be used in different ways to enforce both confidentiality and integrity.

There are a number of approaches to building countermeasures and safeguards.

Perhaps the simplest countermeasure is *avoidance*: for instance, a number of organizations avoid the entire problem of crackers on the Internet by completely disconnecting their network from the rest of the world.

Some security mechanisms are aimed at *prevention* of security breaches while others are aimed at their *detection*. The latter is particularly important since we never can be sure that we have ruled out all possible attacks. It is therefore important to detect something that has gone wrong and to take necessary countermeasures.

A third approach is to focus on *recovery*. If we assume that we will be compromised, then we can plan ahead and know how to limit the potential damage. For instance, new vulnerabilities are often introduced in the implementation of the error handling for a given protocol; anticipating and preventing these issues would be all but impossible.

When there are multiple parties involved with different interests, it may be important to determine in whose interest a given mechanism is operating. For example, is a smart ATM card acting in the bank's interest, the user's interests, or both? Why should each party trust the mechanism?

1.4 Points of View and Perspectives

The design of effective security mechanisms involves some general *security principles*.

Don't aim for perfection: Implementing a secure system is *hard*. Perfection is basically impossible. Instead, be prepared to practice risk management, to detect problems, to design backups and countermeasures, and to recover from attacks. ("There are no secure systems, only degrees of insecurity."–Adi Shamir.)

Secure systems can be expensive: "To halve the insecurity, you double the cost."–Adi Shamir.

Principle of least privilege: Give a user or process only privileges needed to perform task at hand. No more, no less. This extends to the principle of "separation of privilege" or "separation of duties": when possible, require two users to cooperate to perform a task (e.g. dual locks required to fire a missile or open a safe).

Minimize number of trusted components: Identify which components of the system need to be trusted and aim to keep those small and simple. (Also called "economy of mechanism.")

Keep it simple: Because of the ease of introducing vulnerabilities in a large system, and the consequent difficulty of verifying the security properties of a large system, it is important to keep security mechanisms as simple as possible.

Be skeptical: Don't accept people's claims that something is secure. Investigate for yourself.

Be paranoid: People put a lot of effort into attacking a system. At an invited lecture at a CRYPTO conference, Robert Morris said, "Never underestimate the amount of time and effort that an adversary will put into attacking your system." Your enemy may be putting in orders of magnitude more work to break your security system than you put into designing and testing it.

User awareness: Educating users about the security policies, mechanisms, and ethics, along with the risks, is one of the most effective and important techniques for improving security.

Personnel policies: Many (most?) security problems are caused by "insiders" – authorized individuals who abuse the authority given to them. Careful hiring, education, and review can help prevent such incidents.

Defense in depth: Introducing several layers or rings of protection, which must be compromised separately, will force the attacker to spend time and energy evaluating each layer of defense.

No "security through obscurity": It's usually the case that the mechanism you're using becomes public. Systems should be secure even if all mechanisms are known.

Crypto is tremendously useful: You can do amazing things with cryptographic techniques. (But not everything. Be aware of the limitations.)

"Crypto is bypassed, not broken." (quote from Adi Shamir)

Easy of use is *important*: In general, there is a tradeoff between how well you meet your security objectives, and how much inconvenience is involved for the user. If security features aren't easy to use, people won't use them.

Privacy is important: You should take into account the amount of privacy your users need, and determine how to ensure that privacy. For instance, an electronic toll system might keep track of who goes by, but it might be possible to design it so it doesn't need to record customer identities.