CS 493
Secure Software Systems

Introduction

Dr. Williams
Central Connecticut State University

Objectives

- Common terminology in the software security environment
- The software environment and some current threats to software
- Incorporate security into software design
- Primary attributes of security on which software design should be focused
- The tools available to provide security
- Common techniques to improve security of a system

Goals

- Identify threat agents in the software environment.
- Identify mitigation techniques to prevent compromise of specific system attributes.
- Identify how security affects the software design process.
- Understand common techniques for improving system security.
- Understand the context of applying security to software products.

The World Upside Down

- *Does the software do what it is supposed to do?*
- *Does the software refrain from doing what it is not supposed to do?*
- Given incorrect input or used inappropriately, will your software still behave within its boundaries?
- Will it safely handle the invalid input without producing any harmful side effects?

The Lingo

- What does secure mean anyway? It means “free from or not exposed to danger or harm; safe.”
- A **vulnerability** is simply a design flaw or an implementation bug that allows a potential attack on the software in some way.
- A **threat** is a possible exploit of a vulnerability while an **attack** is the actual use of such an exploit.

Common Vulnerabilities

- **Lack of input validation**
- **Insecure configuration management**
- **Lack of bounds checking on arrays and buffers**
- **Unintentional disclosure**
- A **countermeasure** is a means to eliminate the possibility of an attack or at least to mitigate the amount of damage caused if it occurs, such as failing safely or successfully tolerating a fault.
- Anything or anyone who could potentially harm your software is classified as a **threat agent.**

The Usual Suspects

- **Time/logic bomb:** This is code that does not activate a malicious payload until specific conditions are met within a program.
- **Bots:** This is a device that is controlled through a lightweight script by a central host with the purpose of directing traffic to a specified destination.
- A **virus** is a segment of code that attaches to a host file.

The Usual Suspects

- A **worm** is a stand-alone executable that operates like a virus, with the exception that it does not need a host file on which to reside.
- A **Trojan** offers a service that a user would find desirable, such as photo editing or even, ironically, virus protection. It may or may not deliver that service, but as it does so, it is also performing malicious activity on the host machine.

The Usual Suspects

- **Extortionware** (or ransomware) is a type of software that, when activated, either locks certain functionality on a user's computer or blocks all access until a payment is made to deactivate the software.
- **Spyware** is a type of malicious software designed to steal personal information by running undetected on your machine, and it has found a pervasive home on the Internet.

The Usual Suspects

- A **key logger** is a specific type of spyware that records the keystrokes of a user.
- A **sniffer** is a passive eavesdropper that listens to communications within a host or on a network. For example, a packet sniffer will record all packets that are transmitted on a broadcast network for later use.

The Usual Suspects

- A **vulnerability scanner** applies to any software that looks for weaknesses on a system.
- **Backdoor** is a method of circumventing normal authentication procedures and allowing unwanted access into a computer system.
- **Rootkits** are pieces of software that actually subvert the legitimate control of a software system by its operators, operating at the highest level of permission on the system.

The Usual Suspects

- The Human Element- Social engineering is a dangerous tactic to even the most hardened security. Because the human element is often the weakest link in security, it only makes sense that attackers would choose that as their angle of attack.
- Staying Hidden- Secrecy is the biggest ally of an attacker. Giving away the presence of an attack allows a response to the attack. Earlier incarnations of malicious software did not attempt to hide themselves from the user because they assumed the user could do nothing about the attack even with knowledge of it.

The Many Hats of Hackers

- The term **hacker** started out referring simply to someone who was proficient with computers, particularly in the field of networks, and they were the foundation of what has led to the technology revolution in which we all find ourselves.
- A **Cracker** is a person who wishes to use technical proficiency to exploit weaknesses and break systems.

The Many Hats of Hackers

- **Hacktivism** is the use of coding and technology for political purposes. It has been responsible for the open invitation of law enforcement officials to hacker conventions such as HoHoCon (the Christmas convention) and Hackers On Planet Earth (HOPE).

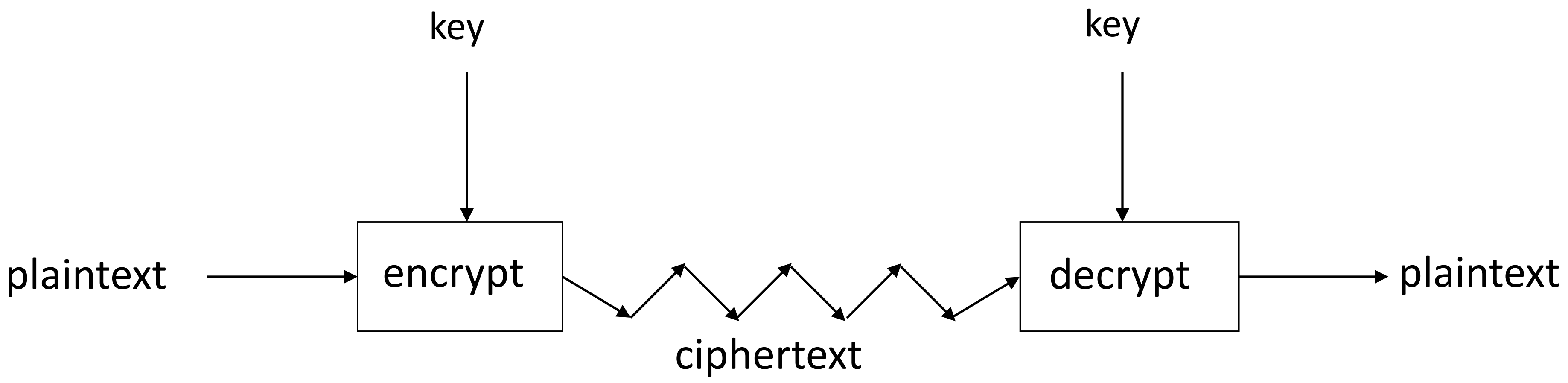
The CIA Triad

- **Confidentiality** in an application means that the private and sensitive data handled by the application cannot be read by anyone who is not explicitly authorized to view it.
- **Integrity** means that the data processed by an application is not modified by any unauthorized channels or any unauthorized persons.
- **Availability** is defined as system's ability to remain operational even in the face of failure or attack.

Cryptography

- **Cryptography** is a mathematical approach to transforming data such that, without the necessary piece of information, a key to unlock it, the information cannot be read.
- **Cleartext** or **plaintext** is information that is not encrypted is said to be in the clear.
- **Ciphertext** is information that has been encrypted.

Crypto as Black Box



A generic view of crypto

Encryption

- Is transforming from plaintext to ciphertext based on a transformation dictated by a cipher and key
- Using different transformations and keys can create many different ciphertexts for the exact same message
- Note that while with **symmetric cryptography** the **key** must be kept secret, with **asymmetric/public key** cryptography the encryption key may be publically known

Decryption

- To decrypt a ciphertext requires the recipient to know how to undo the transform or *decipher* to obtain the original message
- The key used to decrypt a message may be the same as the one used to encrypt it (symmetric) or a different key may be required (asymmetric)
- Goal of a cipher is that it cannot be broken without trying all available transformations/keys

Symmetric key cryptography

- **Symmetric cryptography** states that the same key is needed for both the encryption and decryption process.
- Implication is both the sender and receiver must have the same key which had to be exchanged at some point
- Thus the safety of a message is only as protected as the method the key is distributed
- Public key cryptography makes this easier, but still requires a “trusted” source

Cryptography

- The two principles that define good encryption are called **confusion** and **diffusion**.
- **Perfect secrecy/Provably secure** involves complex mathematics, but boils down to the fact that perfect secrecy is achieved only when the plaintext and ciphertext are statistically independent.
  - *Note – term used by textbook "**perfect secrecy**" is **<u>not</u>** the same as "**perfect forward secrecy**"*
- The only cipher known to achieve this is the one-time pad, which uses a single character (similarly a **single bit** in computer terms) to encrypt each character (or bit) of plaintext.

Public Key Cryptography

- **Public key cryptography** states that two keys are needed for the process; one key is used to encrypt and one is used to decrypt. These two distinct keys allow the process to happen because the one used to encrypt does not reveal any information about the one used to decrypt.
- Public key cryptography also makes signatures possible to allow non-repudiation

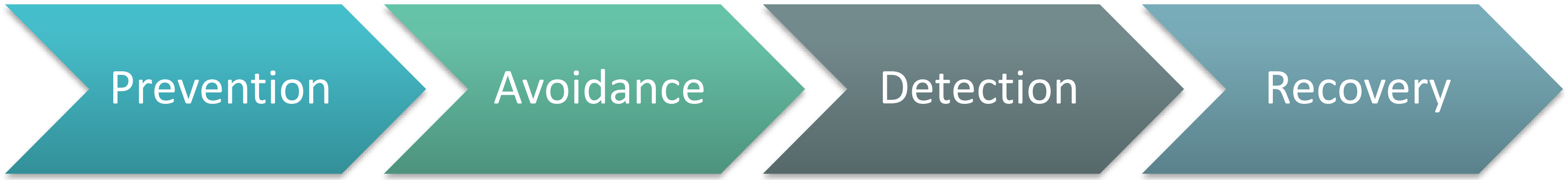
Integrity

- Integrity is used to prevent data modification, insertion or deletion by unauthorized parties. One means to accomplish this is through the use of a message digest or **cryptographic hash.**
- A cryptographic hash ensures that the message hasn't changed since the hash was created, but does not guarantee the hash was produced by that user
  - To ensure hash created by a trusted source must either by encrypted by a shared symmetric key or signed using public key cryptography
- The government has its own standard for digital signatures (called the Digital Signature Standard), which requires the use of keys as well.

Availability

- **Availability** is the more difficult attribute of security to ensure.
- **Redundancy** can be used to provide availability.
- **Off-site backup** may work in the midst of an attack, but it is costly to keep it running when there is no emergency situation requiring it.

Fundamental strategies for secure software



Prevention

- **Prevention** is a tremendously difficult task. This is an assertion that an attack absolutely cannot happen to or through your system.
- One way to do this is to sandbox the application, meaning you test to see what it will do, given user input, before you allow it to actually process.

Avoidance

- **Avoidance** is a best attempt at making sure that attacks do not affect your system. This means that you make every effort in the system to avoid compromise.
- A way to provide avoidance is to code defensively.
- Closing off holes, restricting access, and limiting points of entry into the system are all ways to code defensively.

Detection

- Handling exceptions in code is a good way to approach this. When an attack occurs, it is generally given that the application behaves in an unexpected way.
- **Checkpoints** in code are another way to do this; these are just milestones of execution that indicate everything is still working correctly.

Recovery

- The final option for response is **recovery**. In this case, the attack is allowed to occur when it happens.
- An example of this would be rolling back a database to the state it was in before the attack (something that can only be accomplished if transactions are being recorded) or restarting an application at the last safe state that was recorded in memory.

Changing the Design

- The term **scope creep** comes up a lot in software engineering; this is the term for a software. system taking on more and more functionality as it is developed. Specifically, this is functionality that was not intended when the software system was originally conceived.
- If you keep going and keep adding, you are adding holes in your software and increasing its complexity beyond its scope.

The Shape of Things

- Apply defense in depth
- Minimize the attack surface
- Fail safely
- Run with least privilege
- Avoid security by obscurity
- Keep security simple
- Detect intrusions and record compromise

The Shape of Things

- Do not trust infrastructure
- Do not trust services
- Establish secure defaults

Some Key Ideas

- Vulnerability, Threat, Attack
- Counter measure
- Threat agent
- CIA Triad
- Prevention, Avoidance, Detection, Recovery

Chapter Summary

- Software security is an increasing concern in the modern era of connectivity with its consistency of threats from attackers.
- An attacker can be an automated tool, an intentional insider, an unintentional insider, a malicious outsider, or even an external system destruction.