COMP482
Cybersecurity
Week 8 - Wednesday

Dr. Nicholas Polanco
(he/him)

KALAMAZOO COLLEGE

Important Notes

- We will touch base during the activity to see if we need Friday as an additional work day, that would mean **this is my last lecture**
  - We have Monday off next week and Wednesday is (as of right now) a Project work day. We will discuss later.
- Your project presentations are due a week from today (Week 9 Wednesday) at midnight.
  - I won't be granting any extensions for this, you need to have them done.
- All assignments should be added to kit for the rest of the term, please let me know if I am missing anything
- Would we rather select the order on Friday of this week (Week 8) or Wednesday of next week (Week 9)?

Important Dates (Week 8)

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Project Deliverable: Midway Report		Extra Credit: AES or RSA				

Outline

1. What is Safe Software?
2. Principles of Secure Software Development
3. Risk Assessment
4. Threat Models
5. Activity: Safe Software

What is Safe Software?

What is Safe Software?

Safe software is software that operates reliably under expected and unexpected conditions without causing harm to users, systems, data, or the environment.

It is designed, developed, and maintained to prevent accidental or intentional actions that could lead to unsafe behavior, security breaches, or system failure.

Principles of Secure Software Development

Principles of Secure Software Development

Avoid 3 Common Failures:

1. Organization has no formal policy. Thus, personnel cannot consistently make necessary decisions.
2. Organization has no **reasonable response** plans for violations, incidents, and disasters.
3. Plans don’t work when needed because they haven’t been regularly tested, updated, and rehearsed. (e.g., failure of operational security)

Principles of Secure Software Development (continued)

What Can We Do?

Build in Security from the Start

- Have a plan!
  - We should think about the system and how security can be added before implementation.
- Clear understanding of security policy, and **have a security policy**
- We should be prepared to respond if something occurs, and make sure we are ready to handle it

Image Credit
https://www.etsy.com/listing/1239850637/fire-safety-fire-drill-routine-classroom

KALAMAZOO COLLEGE

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Secure Software Development Lifecycle (SSDLC)

- Ensuring that the final product meets the needs and expectations of the stakeholders and users
  - Need a good grasp of our requirements
- Incorporating security at every phase of the software development lifecycle—from requirements to deployment and maintenance.

Design with Care

- We want to use good tools and practices

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Understand Users

- We should identify our assets in a system, and ensure we protect relevant data.
  - Narrow focus is better than broad, expanding capabilities too much can cause some issues. Why?
- Understand that there is no “average user”
  - What do I mean by that?

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Understand Balance Between Features and Security

- Find the middle ground between safety and customer satisfaction.
- Manage complexity and change
  - We want easy to test so we can verify its strength, fewer flaws
  - Design of security should be as small and simple as needed

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Employ Testing

- We have lots of tools to help with testing now
- What types of tools can we use to help with this?

The Principle of Open Design

- Our security mechanism should not be a secret!
- Who agrees? Why or what not?

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Set Security Goals

- Consistency - The property to ensure that a consistent view of each data item is shown to every user
- Control - The processes in place to protect from dangerous network vulnerabilities and data hacks
- Audit - A comprehensive analysis and review of your IT infrastructure

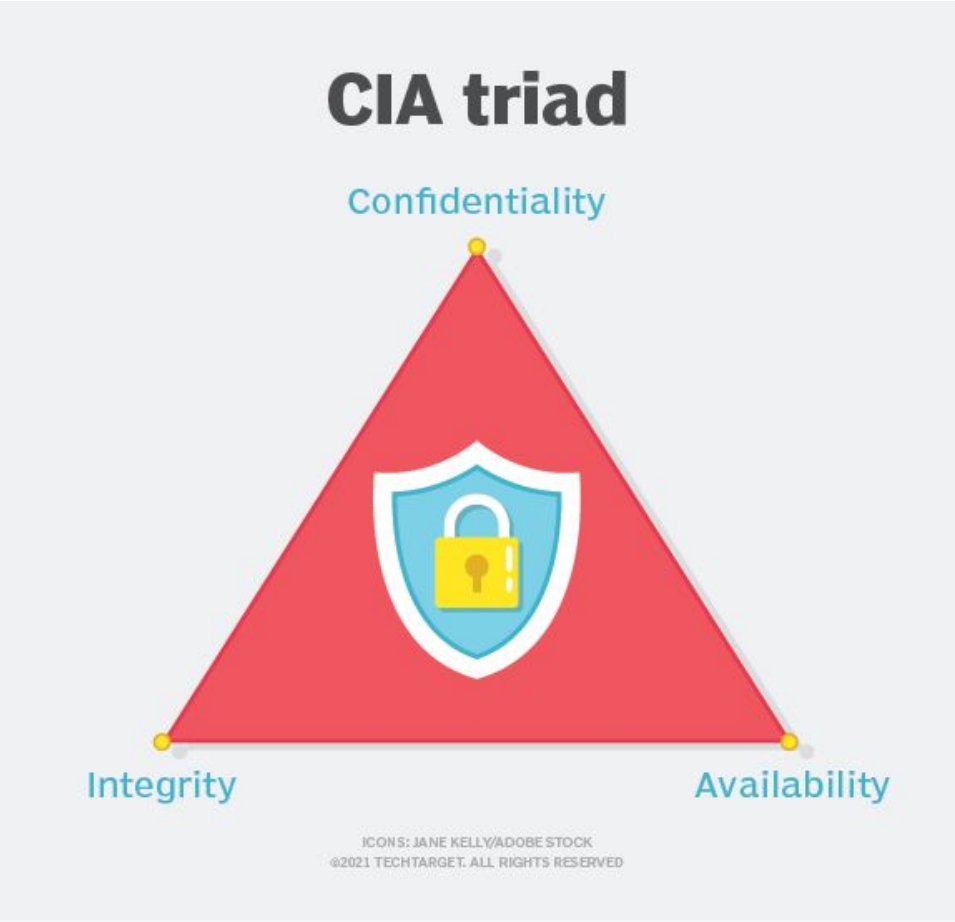

Image Credit
https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA

KALAMAZOO COLLEGE

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Seek Good Security Means

- Limiting what happens, who can make it happen, how it happens, who can change the system

Fail Securely

- Software should default to a secure state in the event of failure.
  - Example: If a login system fails, it should deny access—not grant it.

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Set a Security Plan

- Risk assessment
- Cost-benefit analysis
- Creating policies to reflect your needs
- Implementation
- Audit and incident response

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Know When Software is Secure

- Does not disclose information
- Does not allow unauthorized access
- Does not allow unauthorized change
- Maintains quality of service despite input and load
- Preserves authenticity and control
- Avoid surprises….

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Least Privilege

- Every module, process, or user should have the minimum level of access—or permissions—needed to perform its function.
  - Example - A database user only allowed to read data shouldn't have permission to delete it.

Secure Authentication and Authorization

- Authentication is verifying users are who they claim to be (e.g., via strong passwords, MFA).
- Authorization is ensuring users can only access resources they're permitted to.

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Defense in Depth

- We add multiple layers of security controls that are placed throughout the system. Cheese?
  - Example - Input validation on both client and server, plus a Web Application Firewall (WAF).

Input Validation and Sanitization

- All external inputs must be validated and sanitized to prevent injection attacks.

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Keep Security Simple (KISS Principle)

- Avoid overly complex security mechanisms that are hard to understand or audit.
- Why would we want a simple system?

Secure Defaults

- The default configuration of a system should be secure.
  - Example: Disable unused services by default; require strong passwords out of the box

Principles of Secure Software Development (continued)

What Can We Do? (continued)

Logging and Monitoring

- We want to maintain logs of security-relevant events and monitor them for anomalies.
  - Should we also ensure logs are also protected against tampering?

Risk Assessment

Risk Assessment

A cybersecurity risk assessment is a process that analyzes an organization's environment to identify and prioritize potential threats and vulnerabilities.

Risk Assessment (continued)

3 Questions to Answer:

1. What am I trying to protect (asset)?
2. What do I need to protect against (threats)?
3. How much time, effort and money am I willing to expend to obtain adequate protection?

3 Key Steps:

1. Identify Assets
2. Identify Threats
3. Calculate Risks

Risk Assessment (continued)

Tangibles

What are some tangibles you can think about?

Intangibles

What are some intangibles you can think about?

Risk Assessment (continued)

Tangibles

Computers, disk drives, proprietary data, backups and archives, manuals, printouts, commercial software distribution media, communications equipment & wiring, personnel records, audit records

Intangibles

Safety & health of personnel, privacy of users, personnel passwords, public image & reputation, customer/client goodwill, processing availability, configuration information

Risk Assessment (continued)

What are some risks?

- Illness/loss of key people
- Loss of phone/network services
  - Key to a productive workday
- Loss of utilities (phone water, electricity) for a short or prolonged time
- Natural disaster
  - Lightening or flood
- Theft of disks, tapes, key person's laptop or home computer

Risk Assessment (continued)

What are some risks? (continued)

- Introduction of a virus
- Computer vendor bankruptcy
- Bugs in software
- Labor unrest
- Motivated hackers

Risk Assessment (continued)

Estimate likelihood of each threat occurring

- If an event happens on a regular basis, you can estimate based on your records
  - Power company: official estimate of likelihood for power outage during coming year
  - Insurance company: actuarial data on probabilities of death of key personnel based on age & health

Risk Assessment (continued)

Create a Policy

- Defines what you consider to be valuable and what steps should be taken to safeguard those assets
  - States the responsibility for that protection.
  - Provides grounds upon which to interpret and resolve any later conflicts that might arise
- Policy should be general and change little over time
  - Should not list specific threats, machines or individuals by name

Threat Models

Threat Models

A threat model is a structured approach used to identify, assets, and prioritize potential security threats to a system, application, or organization.

*It helps you understand what you’re defending, what you’re defending against, and how to defend against it.

Threat Models (continued)

Key Elements of a Threat Model

System

1. **Assets** – What you're trying to protect (e.g., data, systems, user credentials).
2. **Threats** – Potential events or actors that can cause harm (e.g., hackers, insiders, malware).
3. **Vulnerabilities** – Weaknesses that could be exploited (e.g., unpatched software, misconfigurations).
4. **Attack vectors** – The paths or methods used to exploit vulnerabilities (e.g., phishing, SQL injection).
5. **Mitigations** – Security controls or strategies to reduce risk (e.g., encryption, access controls).

Threat Models (continued)

Examples Threat Models

1. Web Application Threat Model

- Assets: User data, backend database
- Threats: SQL injection, XSS, CSRF
- Vulnerabilities: Lack of input validation, poor session management
- Attack Vectors: Malicious user inputs, stolen cookies
- Mitigations: Input sanitization, HTTPS, secure authentication mechanisms

Threat Models (continued)

Examples Threat Models

2. Cloud Infrastructure Threat Model

- Assets: Customer data stored in cloud services
- Threats: Misconfigured storage, privilege escalation
- Vulnerabilities: Default credentials, exposed ports
- Attack Vectors: phishing
- Mitigations: regular audits, encryption

Threat Models (continued)

Examples Threat Models

3. IoT Device Threat Model

- Assets: Device firmware, sensor data
- Threats: Physical tampering, remote code execution
- Vulnerabilities: Hardcoded passwords, lack of firmware updates
- Attack Vectors: Local access, open wireless interfaces
- Mitigations: Secure boot, firmware signing, network segmentation

Threat Models (continued)

<u>Common Threat Modeling Methodologies</u>

- STRIDE (by Microsoft)
  - We will walk through STRIDE together
- DREAD (not as popular today)
- PASTA
- OCTAVE

Threat Models (continued)

STRIDE

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

STRIDE THREAT MODEL

Enter your sub headline here

	Threat	Property Violated	Threat Definition
S	Spoofing	Authentication	Pretending to be something or someone other than yourself
T	Tampering	Integrity	Modifying something on disk, network, memory, or elsewhere.
R	Repudiation	Non-Repudiation	Claiming that you didn't do something or we're not responsible. Can be honest or false
I	Information Disclosure	Confidentiality	Providing information to someone not authorized to access it.
D	Denial of service	Availability	Exhausting resources needed to provide service.
E	Elevation of Privilege	Authorization	Allowing someone to do something they are not authorized to do.

Image Credit
https://dzone.com/articles/stride-threat-modeling-guide-secure-implementation

KALAMAZOO COLLEGE

Threat Models (continued)

Example for STRIDE (A Mobile Banking Application)

- **Spoofing:** An attacker impersonates a legitimate user to steal funds from their account.
  - Mitigation: Implement multi-factor authentication (MFA) and use biometrics (e.g., fingerprint or face recognition) for login.
- **Tampering:** An attacker modifies a transaction request to transfer money to their own account.
  - Mitigation: Encrypt transactions (e.g., use HTTPS), validate inputs, and implement transaction integrity checks (e.g., digital signatures).
- **Repudiation:** A user denies making a transaction, and there's no way to prove it was them.
  - Mitigation: Keep secure logs of all actions, with non-repudiation features like digital signatures and audit trails.

Threat Models (continued)

Example for STRIDE (A Mobile Banking Application) (continued)

- **Information Disclosure:** A hacker gains access to a user's personal financial information via a vulnerability.
  - Mitigation: Encrypt sensitive data, ensure proper access controls, and regularly audit access logs.
- **Denial of Service (DoS):** A DDoS attack overloads the app's servers, preventing legitimate users from accessing their accounts.
  - Mitigation: Use rate-limiting and DDoS protection services like Cloudflare, and ensure the infrastructure is scalable and resilient.
- **Elevation of Privilege:** A low-level user exploits a vulnerability to gain administrative access and transfer funds from other accounts.
  - Mitigation: Apply the principle of least privilege, regularly patch security vulnerabilities, and enforce role-based access control.

Threat Models (continued)

Why Threat Modeling Matters:

- Helps prioritize security efforts
- Guides secure design decisions
- Enhances incident response readiness
- Reduces risk in software and infrastructure

Activity: Safe Software

Activity: Safe Software

In this assignment, you are designing a new software-based system from a security-first perspective.

You won’t be writing full production code—instead, your focus will be on the secure design of the system.

You will identify potential threats, outline system components, and propose security controls to reduce risk. Your final deliverable will include a design document and a threat model.

Attendance

https://forms.office.com/r/u62XL60DH5

*This does not record names, I will take attendance up here. We can chat afterwards.

Week 8 Wednesday - Attendance

KALAMAZOO COLLEGE