Secure Software Systems

CYBR 200 | Fall 2018 | University of the Pacific | Jeff Shafer

Goals and Requirements

Content adapted from CS 5430 (System Security), Cornell University, Dr. Michael Clarkson

Schedule

This Week

- Tue September 4
  - Beyond the Attacks
  - Goals and Requirements
- Thur September 6
  - Goals and Requirements
  - Assurance

Next Week

- Tue September 11
  Thur September 13
  - *Architectural Approaches to Security*

Project 1

- *For each group, discuss….*
  - *Team Members?*
  - *Selected application?*
  - *What does application do?*
  - *Why is security important to it?*
- Proposals due Thursday! (11:59pm)
  - **Will provide go/no-go feedback this week**
- Chapter 1 due Tuesday Sept 18th (11:59pm)

Trivia

CVE – Common Vulnerabilities and Exposures

"**Common Vulnerabilities and Exposures (CVE®)** is a list of common identifiers for publicly known cyber security vulnerabilities. Use of CVE IDs ensures confidence among parties when used to discuss or share information about a unique software vulnerability, provides a baseline for tool evaluation, and enables data exchange for cyber security automation."

- https://cve.mitre.org/

MITRE

- Origins – Group of scientists/engineers:
  - MIT Lincoln Laboratory
  - USAF SAGE Project - 1950's project to combine multiple radars into single "national airspace" view
    - Computers, networking, algorithms, command-and-control systems, etc…
- Many decades of federal R&D dollars
- Today
  - Non-profit engineering/security research corporation

CVE

- Q: Who can assign CVE IDs?
- Ans: Not *just* MITRE
  - CVE Numbering Authorities (CNA)
    - Bug bounty programs
    - National and Industry CERTs (Computer Emergency Response Team)
    - Vendors/Projects
  - 73 in September 2017
  - https://cve.mitre.org/cve/cna.html

CVE

- Q: Is CVE a “vulnerability database”?
- Ans: No – it’s a list of identifiers (with a *brief* description)
  - Allows vulnerabilities databases to be linked together to produce security tools & services
  - CVE is missing information on:
    - Risk
    - Impact
    - How to fix
    - Detailed technical details

NVD

- Q: Where *can* I find a vulnerability database?
- Ans: National Vulnerability Database
  - https://nvd.nist.gov/
  - https://nvd.nist.gov/general/nvd-dashboard

CWE – Common Weakness Enumeration

"**CWE™** is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts."

- https://cwe.mitre.org/
- 705 listed as-of September 2017

CWE Examples

- General Coding
  - CWE-457: Use of Uninitialized Variable
  - …
- Dynamic Memory
  - CWE-415: Double Free
  - CWE-416: Use After Free
  - …

CWE Examples

- Math
  - CWE-682: Incorrect Calculation (parent)
  - CWE-190: Integer Overflow or Wraparound
  - …
- Race Conditions
  - CWE-362: Race Condition (parent)
  - CWE-366: Race Condition Within a Thread
  - CWE-367: Time-of-Check Time-of-Use (TOCTOU) Race Condition
  - …

CWE Examples

- Buffer Overflow
  - CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer (parent)
  - CWE-121: Stack-based Buffer Overflow
  - CWE-122: Heap-based Buffer Overflow
  - CWE-125: Out-of-bounds Read
  - CWE-129: Unchecked Array Indexing
  - CWE-131: Incorrect Calculation of Buffer Size
  - CWE-193: Off-by-one Error
  - …

2011 Top CWE - Porous Defenses

- Execution with Unnecessary Privileges - (250)
- Improper Restriction of Excessive Authentication Attempts - (307)
- Incorrect Authorization - (863)
- Incorrect Permission Assignment for Critical Resource - (732)
- Missing Authentication for Critical Function - (306)
- Missing Authorization - (862)
- Missing Encryption of Sensitive Data - (311)
- Reliance on Untrusted Inputs in a Security Decision - (807)
- Use of Hard-coded Credentials - (798)
- Use of a Broken or Risky Cryptographic Algorithm - (327)
- Use of a One-Way Hash without a Salt - (759)

https://cwe.mitre.org/data/index.html

Goals and Requirements

Recap

- Aspects of Security
  - **C**onfidentiality, **I**ntegrity, **A**vailability
- Key Concepts
  - Harm, threat, vulnerability, attack, countermeasure
- Principles
  - Accountability, least privilege, defense in depth, …

Engineering Methodology

1. Functional Requirements
2. Threat Analysis
3. Harm Analysis
4. Security Goals
5. Feasibility Analysis
6. Security Requirements

Functional Requirements (1)

- Should be **testable** – 3rd party can determine if requirement is met
- User stories – brief description of a single kind of interaction user can have with system
  - As a *user* I can *action* so that *purpose*
- Examples from Course Management System (e.g. *Canvas*)
  - As a *professor*, I can *create a new assignment* by specifying its name, number of possible points, and due date
  - As a *student*, I can *submit a file* as a solution to an assignment
- These stories reveal system *assets*

Threat Analysis (2)

- Identify threats of concern to system
  - Especially malicious, human threats
  - What kinds of attackers will system resist?
  - What are their motivations, resources, and capabilities?
- Identify non-threats
  - Trusted hardware?
  - Trusted environment?
    - Physically secure machine room, only trusted system operators have access

Harm Analysis (3)

- Harm: Action adversely affects value of asset
- Harm to: **C**onfidentiality, **I**ntegrity, **A**vailability
- "Performing *action* on/to/with *asset* could cause *harm*"
  - *"Stealing money could cause loss of revenue"*
  - *"Erasing account balances could cause loss of customers"*

Harm Triples

- <action, asset, harm>
  - <theft, money, loss of revenue>
  - <erasure, account balance, loss of customer>
- Methodology
  - Start with asset
  - Brainstorm: What actions could harm this asset?
  - Let CIA triad inspire you

Example: GMS

- Imagine Grade Management System (GMS)
  - Manages just the final grade for a course
- **Functional Requirements? (and assets?)**
- **Threat Analysis?**
- **Harm Analysis?**

Example: GMS

Functional Requirements

- As a student, I can view my final grade
- As a professor, I can view and change final grades for all students in my courses
- As an administrator, I can add or remove students and professors to/from the course
- Asset: Letter grade for each student

Example: GMS

Threat Analysis

- Students:
  - Motivations: Increase their own grade, lower others' grades, learn others' grades
  - Capabilities: Network access to system, physical access to other students' computers, social engineering. Limited computational or financial resources
- Out of scope: Assume that threats cannot physically access any servers; professors and sysadmins are trusted

Example: GMS

Harm Analysis

- Performing *action* with *asset* could cause *harm*
- **Brainstorm some harm triples**
  <action, asset, harm>

Security Goals (4)

- “The system shall prevent/detect *action* on/to/with *asset*.”
- **Specify what not how**
- Examples
  - “The system shall prevent theft of money”
  - “The system shall prevent erasure of account balances”
- Poor Goals
  - “The system shall use encryption to prevent reading of messages”
  - “The system shall use authentication to verify user identities”
  - “The system shall resist attacks”

Feasibility Analysis (5)

- Not all goals are **feasible** to achieve
- Relax goals
  - “Prevent theft of items from a vault”
    - Too hard!
  - “Resist penetration for 30 minutes”
    - Realistic and testable
  - “Detect theft of items from a vault”
    - Realistic and testable

Goals -> Requirements

- **Goals**: What should never happen in any situation
  - Not testable
- **Requirements:** What should happen in specific situations
  - Testable

Security Requirements (6)

- Constraint on functional requirements, in service of security goals
- Example
  - Functional requirement: allow customers to cash checks
  - Security goal: Prevent loss of revenue through bad checks
  - Security requirement:
    - Check must be drawn on bank where it's being cashed (so funds can be verified), or
    - Customer must be account holder at bank and depositing funds in account (so funds could be reversed)

Security Requirements (6)

- Constraint on functional requirements, in service of security goals
- Example
  - Functional requirement: Allow two users to chat using IM
  - Security goal: Prevent disclosure of message content to other users
  - Security requirement:
    - (Poor) Contents of message cannot be read by anyone other than the two users
    - (Better) Message is encrypted by key shared with the two users
      - *Don't be too specific with technical details here*

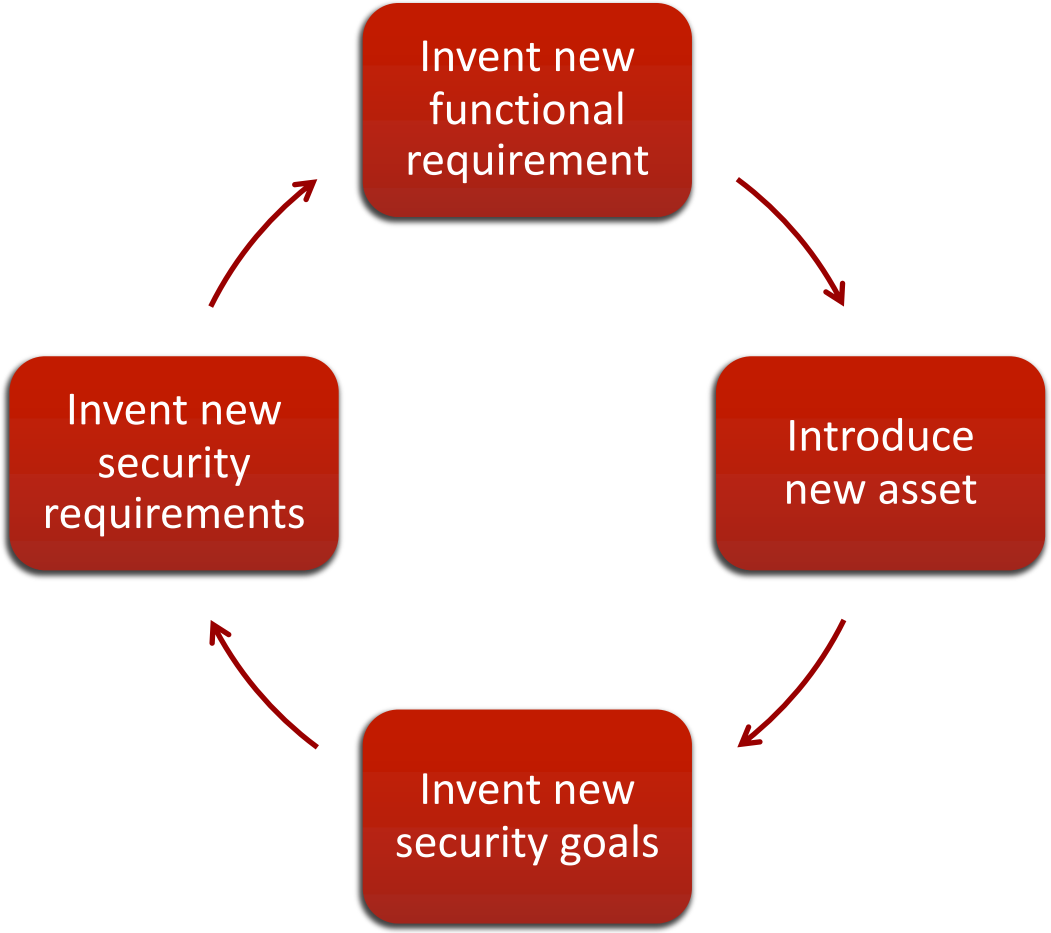
Example: GMS

- Functional Requirements
  - Students view grades
  - Professors view and change grades
  - Admins manage enrollment
- **Security goals?**
  - *“The system shall prevent/detect action on/to/with asset."*
- **Security Requirements?**
  - *Combine functional requirements with goals to invent constraints on system*

Engineering Methodology

1. Functional Requirements
2. Threat Analysis
3. Harm Analysis
4. Security Goals
5. Feasibility Analysis
6. Security Requirements

Iteration



Invent new functional requirement

Introduce new asset

Invent new security goals

Invent new security requirements

Goals vs Requirements

Goals	Requirements
Broad scope	Narrow scope
Apply to system	Apply to individual functional requirements
State desires	State constraints
Not testable	Testable
No design/implementation details	Limited design/implementation details