Lecture 4: Security by Design

CS 181S

September 17, 2018

Engineering methodology

1. Functional requirements
2. Threat analysis
3. Harm analysis
4. Security goals
5. Feasibility analysis
6. Security requirements

1. Functional requirements

- **Security** = **does what it should** + nothing more
- Should be **testable**: a 3rd party could determine whether requirement is met
- User stories:
  - brief description of single kind of interaction user can have with system
  - "As a *user* I can *action* so that *purpose*"
  - Examples from Sakai:
    - 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 upload a file as a solution to an assignment.
- These stories reveal system assets

Assets

- Types of Assets:
  - physical objects (e.g., money)
  - intangible objects (e.g., bank account balance)
- In computer systems:
  - information is typically the main asset
  - hardware and software could be assets
  - people are not typically considered to be assets

Stakeholders

- Anything of value to a **stakeholder** in system could be an asset
  - direct value: damage affects asset itself
  - indirect value: damage affects something else, e.g. reputation
- An object is not an asset if it doesn't have value to some stakeholder
- A principal isn't a stakeholder if it doesn't value some system object
  - We won't consider a generic "attacker" to be a stakeholder

Example: GMS

Grade Management System: manages just the final grade for a single course

Functional requirements:

- As a student, I can view my final grade.
- As a professor, I can view and change final grades for all students.
- As an administrator, I can add/remove students and professors to/from the course

2. Threat analysis

- Identify threats of concern to system
  - Especially **malicious, human threats**
  - What kinds of attackers will system resist?
  - What are their **motivations, resources, and capabilities**?
- Best if analysis is specific to system and its functionality
- Non threats?
  - Trusted hardware
  - Trusted environment
  - e.g., physically secured machine room reachable only by trustworthy system operators

Example: GMS

Threat analysis:

- Students:
  - Motivations: increase their own grade, lower others' grades, learn others' grades
  - Capabilities: network access to servers, some physical access to others' computers, social engineering; probably not extensive computational or financial resources
- Out of scope: assume that threats cannot physically access any servers, profs are trusted, system admins

3. Harm analysis

- Harm: a negative consequence to a system asset
- Harm to...
  - **confidentiality**: disclosure, interception
  - **integrity**: modification, fabrication
  - **availability**: deprivation, interruption
- "Performing *action* on/to/with *asset* could cause *harm*"
  - e.g., "stealing money could cause loss of revenue"
  - e.g., "erasing account balances could cause loss of customers"

Harm triples

- <action, asset, harm>
  - e.g., <theft, money, loss of revenue>
  - e.g., <erasure, account balance, loss of customer>
- Useful methodology:
  - start with asset
  - brainstorm actions that could harm asset
  - let brainstorming be guided by CIA triad

Ex: GMS

- **Asset:** a numeric score for each student
- **Functional requirements:** students view grades, profs view and change grades, admins manage enrollment
- **Threat analysis:** students might be malicious or curious; profs are trusted; threats can't access servers physically
- **Harm analysis:** performing *action* on/to/with *asset* could cause *harm*
- **Exercise:** invent some harm triples <*action, asset, harm*>

4. Security goals

- "The system shall prevent/detect *action* on/to/with *asset*."
  - e.g., "The system shall prevent theft of money"
  - e.g., "The system shall prevent erasure of account balances"
- Specify **what** not **how**
- 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"

Ex: GMS

Exercise: transform harm triples

<action, asset, harm>

into security goals

the system shall prevent/detect
action *on/to/with* asset

5. Feasibility analysis

- Not all goals are **feasible** to achieve
- Relax goals:
  - "prevent theft of items from a vault"
  - to "resist penetration for 30 minutes"
  - or to "detect theft of items from a vault"

From goals to requirements

- **Goals**: what should never happen in any situation
  - not testable
- **Requirements**: what should happen in specific situations
  - testable

6. Security requirements

- Constraints on functional requirements, in service of security goals
- Example:
  - **Functional requirement:** allow people 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

- Constraints on functional requirements, in service of security goals
- Another example:
  - **Functional requirement:** allow two users to chat using IM
  - **Security goal:** prevent disclosure of message contents to other users
  - **(Poor) security requirement:** contents of message cannot be read by anyone other than the two users
  - **(Improved) security requirement:** message is encrypted by key shared with the two users
    - doesn't over-commit to encryption algorithm, key size, etc.

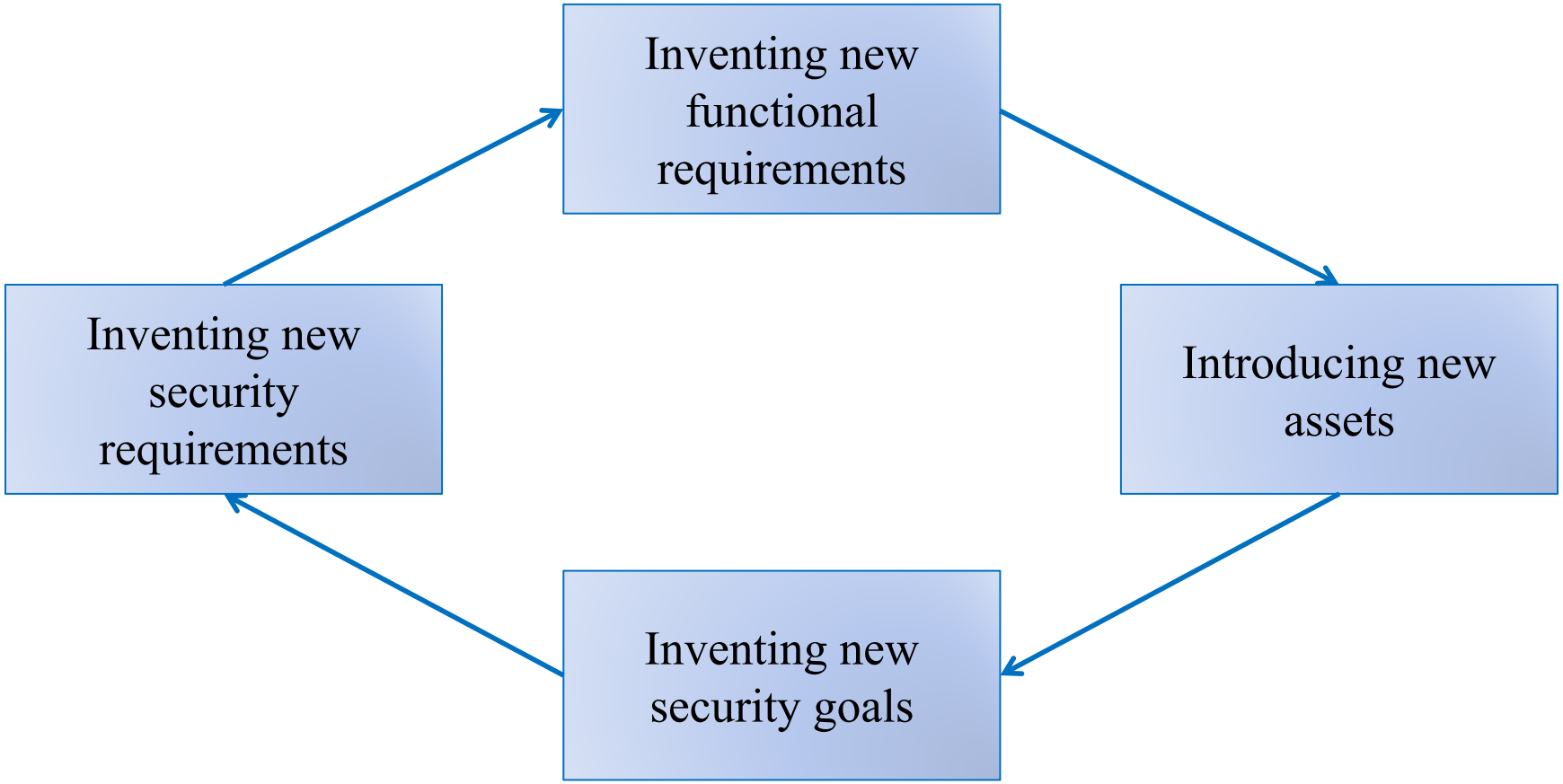
Goals vs. requirements

Goals	Requirements
Broad scope	Narrow scope
Apply to system	Apply to individual functional requirements
State desires	State constraints
Not testable	Testable
Not about design/implementation details	Provide some details

Example: GMS

- **Functional requirements:** students view grades, profs view and change grades, admins manage enrollment
- **Security goals:** ...
- **Security requirements:** *combine functional requirements with goals to invent constraints on system*

Iteration


Inventing new functional requirements
Introducing new assets
Inventing new security goals
Inventing new security requirements

yelp

Find tacos, cheap dinner, Max's | Near Ithaca, NY

Sign Up

Restaurants | Nightlife | Home Services | Write a Review | Events | Talk | Log In

Eleanor's place Ithaca, NY

Showing 1-1 of 1

$ | $$ | $$$ | $$$$ | Open Now | All Filters

1. **Eleanor's Place**

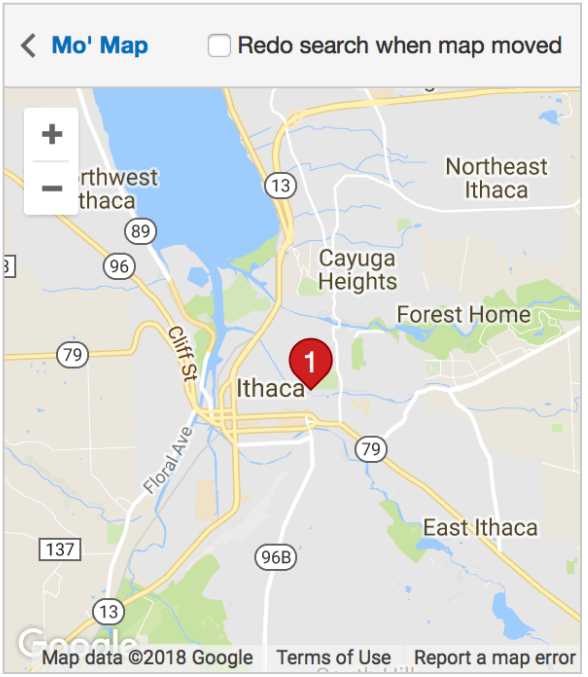
2 reviews

Bars

120 Linn St
Ithaca, NY 14850

Excellent atmosphere, totally cool spot with quite an exotic wine selection and solid scotch menu as well. read more

Page 1 of 1



Ads by Google

Example: ELEANOR'S PLACE

- New restaurant in Claremont
- You are contracted to build a system for online reservations
- What are the functionality requirements for this reservation system?
- What is the threat model?
- What confidentiality, integrity, and availability harms does Eleanor's Place face?
- What security goals should it have?
- Are they feasible? How could these be refined to security requirements?