CSCI 1951E Computer Systems
Security: Principles and Practices

Security Through Software Engineering
John E. Savage
Brown University
Outline

• Security modeling including access control
• Federal security regulations and standards
• Software vulnerability assessments
• Microsoft’s Security Development Lifecycle
Policy, Models and Trust

• To have secure systems, engineers need clear
  – Security goals
  – Effective implementation strategy

• A security policy puts constraints on actions that subjects can take with respect to objects in the system in order to achieve security goals.
Security Policy Components

• Subjects
  – Individual or group agents interacting with a system.

• Objects
  – Informational/computational resources affected by policy.

• Actions
  – Actions that may be done to objects, e.g. read, update, remove

• Permissions
  – Rules for actions that subjects may take on objects.

• Protections
  – Features of policy, e.g confidentiality, availability, anonymity.
Security Model

• An abstraction providing conceptual language to specify security policies.
  – E.g. Unclassified (U), Confidential (C), Secret (S), Top Secret (TS)
  – Special compartments for special information such as human intelligence, satellite observations
Two Models of Access Control

• Discretionary access control
  – Owner may specify permissions on files
  – More relaxed

• Mandatory access control
  – Administrator fixes permissions in advance.
  – More strict

• Rules have subjects, parties requesting access, and objects, those things being accessed.
Bell-LaPadula (BLP) Access Control Model

• Applies to confidentiality – dates from ‘70s
• Object x & user u have security levels L(x) and L(u)
  – Examples of security levels: U, C, S, TS
• For users u and v, if L(u) ≤ L(v), u can pass info to v but not vice versa.
• No “read up” (user can’t see more secure data)
  – User u can read x only if L(x) ≤ L(u)
• No “write down” (user can’t use more secure data)
  – User u can write to object x only if L(u) ≤ L(x).
Ken Biba (‘77) Access Control Model

• BLP model weakness: handles only confidentiality
• Biba’s model deals with integrity or trustworthiness
  – User u and object x have integrity I(u) and I(x)
  – The higher I(u) or I(x), the more trustworthy or accurate the user or object is.
• Don’t corrupt data by reading from lower integrity level, don’t write to higher integrity level.
  • User u can read object x only if I(u) ≤ I(x).
  • User u can write to object x only if I(x) ≤ I(u).
Role-Based Access Control

• Components: users, roles, permissions, sessions
  – A role is a collection of users.
  – A session is an interaction for a period of time.

• Role hierarchy is defined, as in a corporation.
  – President IsA manager IsA employee
  – Higher role user inherits permissions of lower one
  – Is this always a good idea?

• Role constraints may be imposed
  – Example goal: avoid conflicts of interest.
Software Vulnerability Assessment

• The problem: systems can be enormous
  – Mac OS X 10.4 has > 86 million lines of code!
  – Code can have both performance & security bugs

• Approaches to vulnerability assessment:
  – Black-box analysis
    • Penetration testing without knowledge of innards.
  – White-box analysis
    • Same but with full knowledge of hardware/software, network environment, etc.
Code Analysis for Privacy/Security

• Problem: Cybercrime is here to stay.
• Goal: Find and remove privacy/security hazards.
  – Static code analysis examines static programs.
  – Dynamic analysis examines running programs.
• Good analysis requires training and investment
  – Software engineers generally not educated on this.
  – Microsoft’s Security Development Lifecycle (SDL) represents a big step forward.
• Benefits: Improved security, privacy and reliability.
2013 OWASP Top 10 Threats

• A1 – Injection
• A2 – Broken Authentication and Session Management
• A3 – Cross-Site Scripting (XSS)
• A4 – Insecure Direct Object References
• A5 – Security Misconfiguration
• A6 – Sensitive Data Exposure
• A7 – Missing Function Level Access Control
• A8 – Cross-Site Request Forgery (CSRF)
• A9 – Using Known Vulnerable Components
• A10 – Unvalidated Redirects and Forwards
Security Principles I

• Secure the weakest link
  – If security like a chain, weakest link should be strong

• Defend in depth
  – Layer defenses, be redundant, be diverse

• Fail securely
  – Don’t fail “open,” e.g. don’t offer to change password after three failures

• Grant least privilege
  – Be miserly in doling out privilege

Security Principles II

• Separate privileges
  – Require multiple privileges to access critical resources

• Economize mechanism
  – Apply KISS principle – complexity is enemy of security

• Do not share mechanisms
  – Don’t put key inward facing apps on a remote cloud.
  – Don’t share objects and access methods

• Be reluctant to trust
  – Assume the environment is hostile! Know that trust is transitive! Limit access to your APIs. Check for spoofing
Security Principles III

• Assume your secrets are not safe
  – Security ≠ obscurity. Every tool available to attacker

• Mediate completely
  – Check every access and object every time. If permissions change, recheck. Don’t cache grants of authority

• Make security usable
  – Don’t make it more odious than necessary for the task

• Promote privacy
  – Limit what PII that you collect. Store securely

• Use your resources
  – Doubt security; don’t go it alone. Security is very hard!
The Only Secure Computer System

• One that has its disk drive demagnetized
• Turned off
• Inside a Faraday cage
• Buried in a hole and
• Sealed in concrete
Risk Management Framework (RMF)

• **Risk** = probability × impact
  – What are probability and impact?
• **Security is risk management**
  – Perfection impossible, continuously reassess
• **An RMF is a philosophy of software engineering**
  – Integrate it into the software development lifecycle
• **Approach:**
  – “Identify, rank, track & understand software security risk as it changes over time”

The NIST Risk-Based Approach

• **Categorize** info system and info itself by impact
• **Select** baseline security controls, adjust as needed
• **Implement** security controls & document deployment
• **Assess** implementation & operation of controls
• **Authorize** info system operation based on above
• **Monitor** continuously effectiveness of approach

McGraw’s Five Stages of RMF

1. Understand business context
2. Identify business and technical risks
3. Synthesize and rank risks
4. Define risk mitigation strategy

• Remember: The RMF is a multilevel loop
  – Risks can appear at any time, even between stages
Microsoft’s Security Development Lifecycle (SDL)

• Bill Gates inaugurated Microsoft’s Trustworthy Computing Initiative in 2002.
  – Success with major new corporate initiatives often requires support from top management.

• Every product that impacts privacy or may be used by children needs security analysis.
  – This means almost all hardware/software products

• Average cost of fixing a security bug was $300K!
Current Development Methods Fail

• There must be an incentive to review code
• Knowledge of security bugs is limited
• No critical mass of knowledgeable coders
• “Given enough eyeballs, all bugs are shallow” 😞
• Duration of undiagnosed security bugs:
  – 15 years Sendmail e-mail server
  – 10 years MIT’s Kerberos authentication protocol
  – 2 years OpenSSL – aka Heartbleed
Microsoft’s SDL

- Personnel must be trained.
- Security requirements, risk assessment needed
- Must do threat modeling (STRIDE) and reduce attack surface.
  - Spoofing, Tampering, Repudiation, Information Disclosure, Denial of service, Elevation of privilege.
- Implementation requires good tools to protect against attacks
- Must plan for post-release handling of errors
- Verification needed via dynamic analysis including fuzzing.
Is Open Source Software a Panacea?

- Software is available for modification under liberal copyright policy.
- Do many eyeballs on the code make it secure?
  - “… in reality that doesn’t happen” Cowan 2002.
- Russia believes it – avoids US software.
  - Putin orders Russian government to move to Open Source Software by 2015. (12/28/2010)
- Problems: No incentive to find bugs. Coders not trained to find them. It is hard!
Review

- Security modeling including access control
- Federal security regulations and standards
- Software vulnerability assessments
- Microsoft’s Security Development Lifecycle