

Building Secure Software

Building Secure Software
How to Avoid Security Problems the Right Way
Chapter 1-3

Yngve Espelid (yngvee@ii.uib.no)
5. September 2005

INF 329 – Utvalgte emner i programutviklingsteori:
Utvikling av sikre applikasjoner

Motivation

- Computer security impacts our everyday lives
- Today's media focus:
 - What firewalls are.
 - What cryptography is.
 - Which antivirus is best.
 - Malicious attacks and viruses
- Missed focus – the root of the problem:
 - Bad software
- Software is everywhere

Approach to Software Security

- Risk management
 - Know your threats
 - Design for security
 - Subject design to
 - Risk analysis
 - Testing
- Acknowledge *security* as concern among many
 - Time-to-market
 - Cost
 - Flexibility
 - Reusability
 - Ease of use
- Set priorities and identify cost of pursuing each concern

Adversary

- Malicious hackers
 - Exploit security holes which are the result of bad software design and implementation.
 - Build exploits and distribute them as scripts.

Popular sources for vulnerability information

- Bugtraq
 - E-mail discussion list devoted to security issues
 - Following the principle of full disclosure
 - Making complete information about security issues public
- CERT Coordination Center
 - Research and development center studying Internet security vulnerabilities
 - Reported 4129 vulnerabilities in 2003 – 70 % increase over 2002 and four times as much as 2001
- RISKS Digest forum
 - Mailing list

Technical trends

- Complexity
 - Complex systems
 - Are difficult to understand, hard to analyze and hard to secure.
 - May hide malicious or flawed code.
- Extensibility – host accept extensions (mobile code)
 - Increases the risk of intentional introduction of malicious behavior
- Connectivity
 - Number of Internet-connected computers are increasing
 - The number of attack vectors increases
 - Increases the ease with which an attack can be made
 - Automated attacks

What is security?

- System-wide emergent property
- Behavioral property of a complete system in a particular environment
- Book definition: Enforcing a policy that describes rules for accessing resources
 - Explicit and implicit policies.
- Security is relative. No such thing as 100 % security.
- Secure against what and from whom?

Penetrate and patch approach

- Little attention is paid to security when developing the software.
- Patches are distributed after deployment to fix surfaced vulnerabilities.
- Problems
 - Only problems known to developers get patched
 - Patches are rushed out
 - Patches go unapplied

Security Goals

- Prevention
 - Discovered attacks spread like wildfire (automation)
- Traceability and Auditing
 - Attacks will happen and recovering require knowledge of who did what.
- Monitoring
 - Real-time auditing.
 - Example: Intrusion detection systems.
- Privacy and Confidentiality
 - Keeping secrets.

Security Goals

- Multilevel Security
 - Different information require different security measures
- Anonymity
- Authentication
 - Know who to trust
 - Enforcing security policy
- Integrity
 - Data staying unchanged

Software Project Goal

- **Functionality**
 - Number-one driver in most software projects
- **Usability**
 - Security concerns impact convenience
- **Efficiency**
 - Security often comes with significant overhead
- **Time-to-market**
 - Risk management – severe time constraint
- **Simplicity**

Software risk management

- Require expert knowledge of security
 - Recognizing situations in which common attacks can be launched
- Recognizing the risks
 - Architectural problems
 - Implementation issues
- For instance, network attacks
 - Eavesdropping
 - Tampering
 - Spoofing
 - Hijacking
 - Capture/replay

Software risk management

- High-quality software engineering methodology
 - Enable good risk management practice
- Spiral model
 - Refining the product based on new experiences
- Best practices
 - Deriving Requirements
 - Risk Assessment
 - Design for Security
 - Implementation
 - Security Testing

Security Personnel

- A gap between developers and IT department is common.
- Solution: Make software security somebody's job
 - Qualifications
 - Deep understanding of software development
 - An understanding of security
 - From experience

Best practices

- Deriving requirements
 - Common questions
 - What needs to be protected?
 - From whom things need to be protected?
 - For how long things need to be protected?
 - How much it is worth to keep things protected?
 - Resulting in a solid specification
 - What the system does
 - Why the should behave in such a way

Best practices

- Risk assessment
 - Identify risks (based on the specification)
 - Rank risks in order of severity
 - Context-sensitive
 - Depends on system needs and goals
 - Essential to later allocating testing and analysis resources
 - If possible: analysis should be done by an impartial person

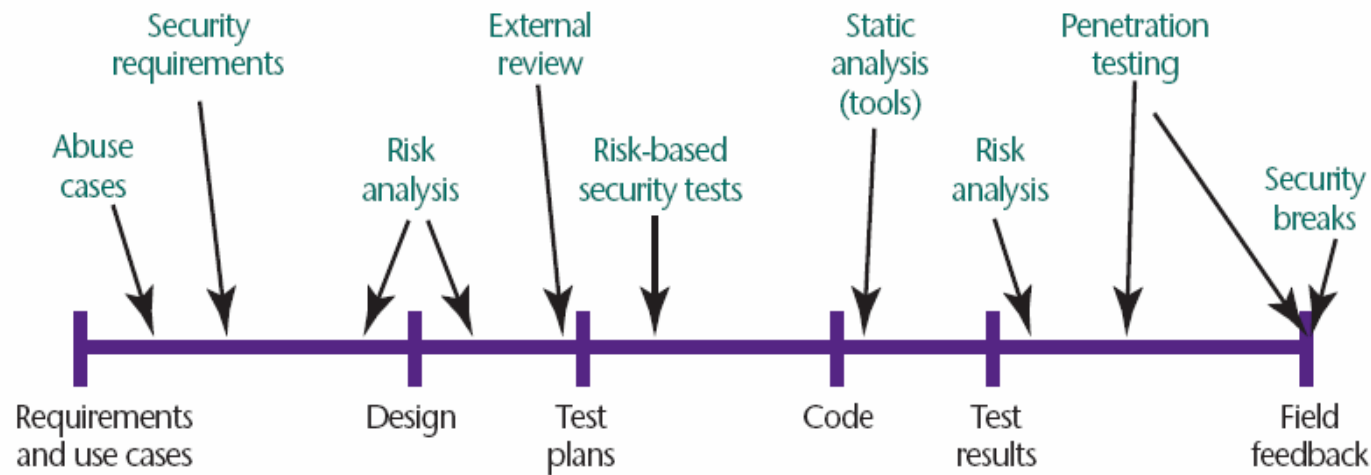
Best practices

- Design for security
 - Focus on identifying
 - Data flow between components
 - Users, roles and rights
 - Trust relationships between components
 - Solutions for recognized problems
- Implementation
 - Code review

Best practices

- Security testing
 - Probing a system in a way an attacker might probe it
 - Bounded by identified risks and the security expertise of the tester
 - Code coverage
 - Code not exercised during testing – suspect in terms of security
- Inefficient approaches
 - Black Box Testing
 - Red Teaming

Best practices applied to various software artifacts



- From G. McGraw's article series "Building Security In" – *Security & Privacy*

Common Criteria

- A security assurance system that can be systematically applied to diverse security-critical systems
- Standardize an evaluation approach across nations and technologies
- ISO standard
- Defines a set of security classes, families, and components designed to be appropriately combined to define a protection profile for any type of IT product, including hardware, firmware, or software.

Selecting technologies

- Comparing technologies and selecting those who best meet the derived requirements.
 - Consider all possible tradeoffs
- Activity early during the life cycle
 - Specification or design phase

Selecting technologies

- Common choices security practitioners must make
 - Choosing which programming language to use for implementation
 - Efficiency: leading to C or C++
 - Often sole justification for language choice
 - Security risks present in C
 - Exception handling
 - More static error checking = more reliable
 - Security features
 - Sandboxing

Selecting technologies

- Choosing a Distributed Object Platform
 - CORBA
 - DCOM
 - RMI
- Choosing an Operating System
- Authentication technologies
 - Host-Based Authentication
 - IP, MAC, cookies
 - Physical Tokens
 - Credit cards, smart cards
 - Biometric Authentication
 - Physical or behavioral characteristics of a human
 - Cryptographic Authentication
 - Defense in depth