1 Secure Software Principles

Three pillars

- Risk Management
- Touchpoints
- Knowledge

1.1 80% / 20% Problem Reduction

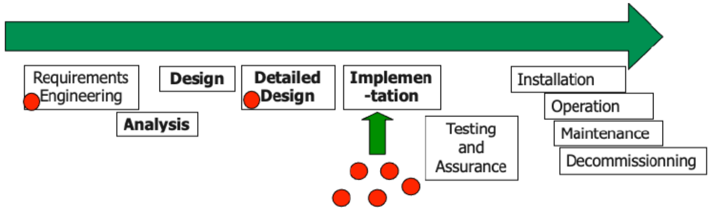
1. Secure the weakest link
2. Practice defence in depth
3. Fail securely
4. Follow the principle of least privilege
5. Compartmentalize
6. Keep it simple
7. Promote privacy
8. Remember that hiding secrets is hard
9. Be reluctant to trust (Off-the-shelf software)
10. Use your community resources (crypto algorithms)

2 Secure Software Lifecycle

SW Engineering processes: Waterfall, Spiral, Prototyping, XP, Adaptive Programming, ...
None of these really support security.

2.1 Software Engineering & Security

- Often ad hoc or not covered at all
- Issues:
  - Non systematic approach
  - Bad early decisions
  - Hard to manage and modify



2.2 Requirements / Analysis

- Identify security requirements
- Quantify the security risks
- Know the entire problem domain of the system

2.2.1 Identify Security Requirements

1. Identify stakeholders
2. Identify assets from different stakeholders (sensitive info / resources)
3. Identify requirements on these assets

Result: high-level security policy
STRIDE: Systematic approach for threat identification.

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

2.2.2 Quantify Security Risks

- Estimate and quantify the risk of the previously identified threats
  1. Importance or severity (critical vs. non-critical)
  2. Cost
  3. DREAD (Damage potential, Reproduceability, Exploitability, Affected users, Discoverability)
- Order the requirements and identify the relevant subset: Risk mitigation strategy

Result: high-level security policy revisited

2.2.3 Example

General formulated:
The software must validate all user input to ensure it does not exceed the size specified fot that type of input.
The system must encrypt sensitive data transmitted over the Internet between the server and the browser.
Non-functional requirements:

1. **Security Property Requirement**: specifies the characteristics that software must exhibit.
2. **Constraint or Negative Requirements** limit what software functionality can be allowed to behave.
3. **Security Assurance Requirements** are rules or best practices by which the software security functions will be built, deployed and operated.

2.3 Detailed Design

Main goals:

1. Identify security technologies that meet relevant security requirements
2. Bind these technologies to the application

2.3.1 Identify security technologies

- Select requirements that should be addressed at this level
- Identify security technologies that address requirements

STRIDE Countermeasures: Tampering

- Use data hashing and signing
- Digital signatures
- Strong authorization
- Tamper-resistant protocols across communication links
- Secure communication links with protocols that provide message integrity

Information Disclosure

- Strong authorization
- Strong encryption
- Secure communication links with protocols that provide message confidentiality
- Do not store secrets in plaintext

...

2.3.2 Bind Technologies to Application

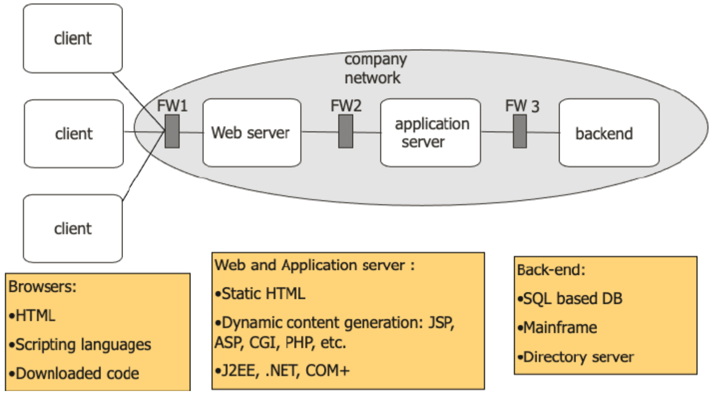
- For coarse-grained or simple requirements
  - Countermeasure can be realized in the operating system/middleware (SSL)
- For fine-grained and complex requirements
  - Implement as part of the application (hard to get right)

2.4 Other phases

- Avoiding implementation vulnerabilities
- Security testing
- Automated patching

2.5 Case study: Web Applications

2.5.1 High level Architecture

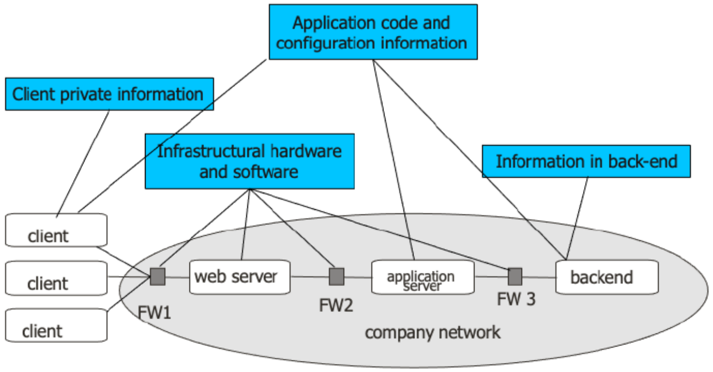


Browsers:
- HTML
- Scripting languages
- Downloaded code

Web and Application server :
- Static HTML
- Dynamic content generation: JSP, ASP, CGI, PHP, etc.
- J2EE, .NET, COM+

Back-end:
- SQL based DB
- Mainframe
- Directory server

2.5.2 Owners and Assets



Application code and configuration information
Client private information
Infrastructural hardware and software
Information in back-end
client
client
client
web server
application server
backend
FW1
FW2
FW 3
company network

2.5.3 Threat Agents and Threats

Any hacker on the internet:

- Spoof client or server
- Eavesdrop on connections / modify data in transit
- Bring down infrastructure components
- Gain unauthorized access to application or data

Authorized user of the application: (additionally)

- Repudiate transactions
- Elevate privilege

Malicious server:

- Steal private client information
- Spread spyware/viruses

2.5.4 Infrastructural countermeasures

Authentication:

- Network level: IPSEC
- Transport level: HTTP authentication mechanisms
- OS level: Windows authentication
- Single sing-on systems based on federation or windows active directory (Kerberos)
- Web authentication products: IAM products

Data protection:

- IPSEC (Network)
- Kerberos (OS)
- TLS (Transport)

Access control:

- Firewall
- In the webserver (URL)
- RBAC in the web container or application server
- File system based access control
- Access control products

Sandboxing:

- At the OS level: low-privileged accounts
- Language level: Java security architectures

Others:

- Filtering
- Throttling
- Shielding
- ...

2.5.5 Typical vulnerabilities

Bugs in functional parts

- Input validation
- Race conditions
- Bad error handling

Broken countermeasures

- Access control
- Authentication
- Crypto

2.5.6 Available Countermeasures at coding level

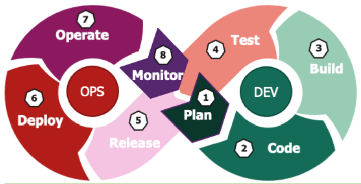
- Security technologies
- Quality improvements
  - Choice of programming language
  - Coding guidelines
  - Source code scanners
  - Security testing and audit
  - Code review

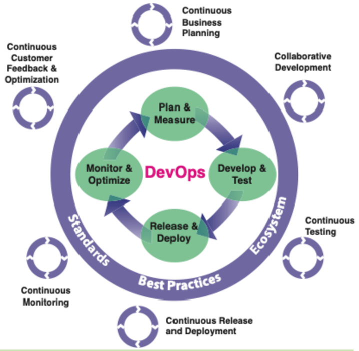
2.6 Daily Sins

1. Buffer Overruns
2. Format String problems
3. Integer Overflows
4. SQL Injection
5. Command Injection
6. Failing to Handle Errors
7. XSS
8. Failing to protect Network traffic
9. Use of magic URLs and Hidden Forms
10. Improper use of SSL and TLS
11. Use of Weak Password-Based systems
12. Failing to store and protect data securely
13. Information leakage
14. Improper file access
15. Trusting Network Name Resolution
16. Race Conditions
17. Unauthenticated Key Exchange
18. Cryptographically strong random numbers
19. Poor usability

3 Threat Modeling

3.1 Securing DevOps





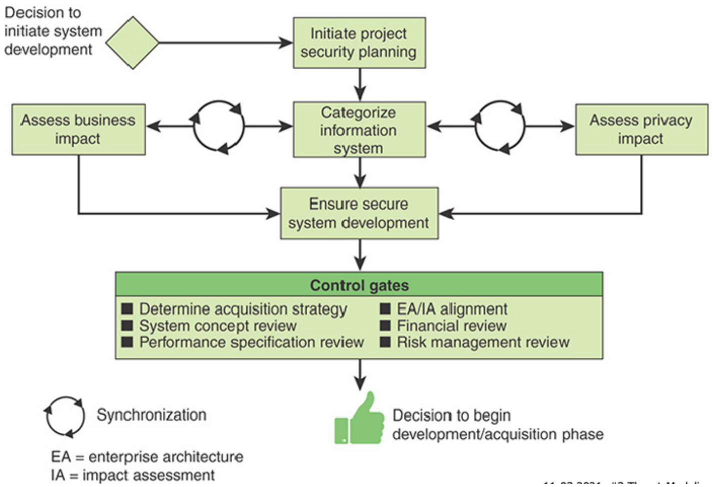
3.2 Four Major Activities

- Plan and measure
- Develop and test
- Release and deploy
- Monitor and optimize

3.3 Continous Secutity

- Prepare the organisation
- Protect the software
- Produce well secured software
- Respond to vulnerabilities

3.4 Security in the Phases



Decision to initiate system development
Initiate project security planning
Assess business impact
Categorize information system
Assess privacy impact
Ensure secure system development
Control gates
Determine acquisition strategy
System concept review
Performance specification review
EA/IA alignment
Financial review
Risk management review
Synchronization
Decision to begin development/acquisition phase
EA = enterprise architecture
IA = impact assessment

Major Security Activities: A number of security-related activities are needed to assure that security is incorporated effectively in that design phase
Expected Outputs: A key to success is to define specific deliverables for each activity
Synchronization: A feedback loop between tasks provides opportunities to ensure that the SDLC is implemented as a flexible approach that allows for appropriate and consistent communication and the adaptation of tasks and deliverables as the system is developed
Control gates: Decision points at the end of each phase when the system is evaluated and management determines whether the project should continue as is, change direction, or be discontinued

3.4.1 Initiate project security planning

- Identify key security roles
- Identify the standards and regulations for the system
- Develop an overall plan for security milestones
- Get Stakeholders to have a common understanding (security implications, considerations, requirements)
- Enable developers to design security features
- **Output:** Supporting documents of all decisions

3.4.2 Categorize information system

- Identify information that will be transmitted, processed or stored
- Define applicable levels of information categorization (based on impact analysis)
- **Result:** Catalog of information types
- **Outputs:** Definitions of categories, level of effort estimate

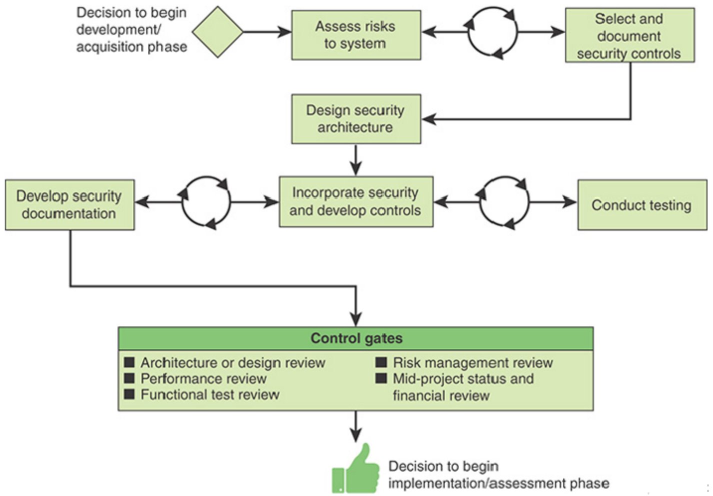
3.4.3 Ensuring secure system development

- Develop a set of principles of security expectations
  - Secure concept of operations
  - Standards and processes
  - Security training for development team
  - Quality management
  - Secure environment
  - Secure code practices and repositories
- **Output:** Plans for development security training / quality assurance

3.4.4 Control Gates

- Determine acquisition strategy
- System concept review
- Performance specification review
- EA/IA alignment
- Financial review
- Risk management review

3.5 Develop & Test

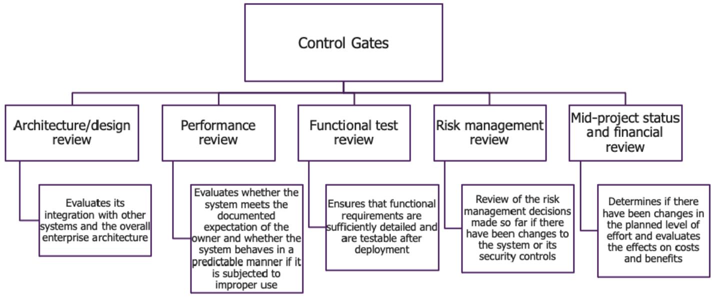


Decision to begin development/ acquisition phase
Assess risks to system
Select and document security controls
Design security architecture
Develop security documentation
Incorporate security and develop controls
Conduct testing
Control gates
Architecture or design review
Performance review
Functional test review
Risk management review
Mid-project status and financial review
Decision to begin implementation/assessment phase

3.5.1 Designing the security architecture

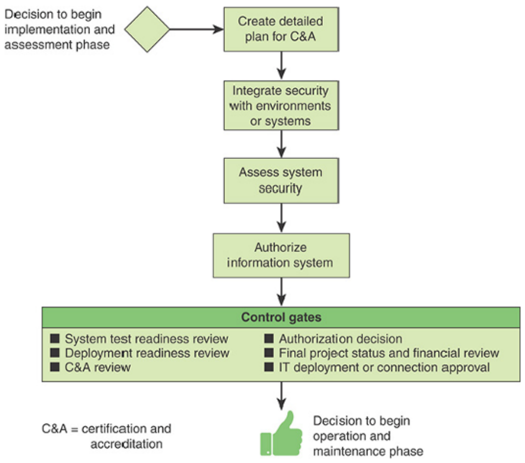
- Produce a detailed architecture
- Incorporate security features and controls into the system design
- **Outputs:**
  - Schematic of security integration
  - List of shared services
  - Identification of common controls used by the system

3.5.2 Control Gates



Control Gates
Architecture/design review
Performance review
Functional test review
Risk management review
Mid-project status and financial review
Evaluates its integration with other systems and the overall enterprise architecture
Evaluates whether the system meets the documented expectation of the owner and whether the system behaves in a predictable manner if it is subjected to improper use
Ensures that functional requirements are sufficiently detailed and are testable after deployment
Review of the risk management decisions made so far if there have been changes to the system or its security controls
Determines if there have been changes in the planned level of effort and evaluates the effects on costs and benefits

3.5.3 Begin Implementation / Assessment phase

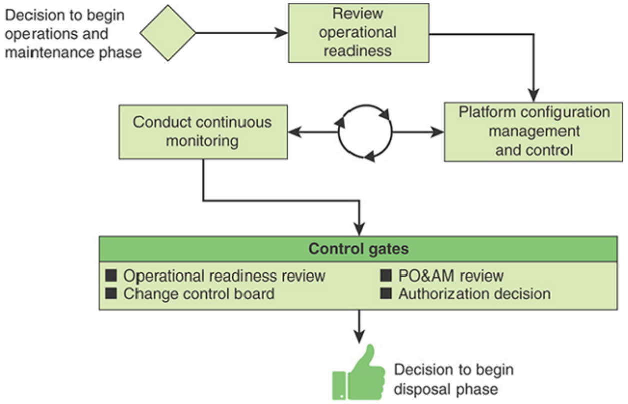


Decision to begin implementation and assessment phase
Create detailed plan for C&A
Integrate security with environments or systems
Assess system security
Authorize information system
Control gates
System test readiness review
Deployment readiness review
C&A review
Authorization decision
Final project status and financial review
IT deployment or connection approval
C&A = certification and accreditation
Decision to begin operation and maintenance phase

3.5.4 Control Gates

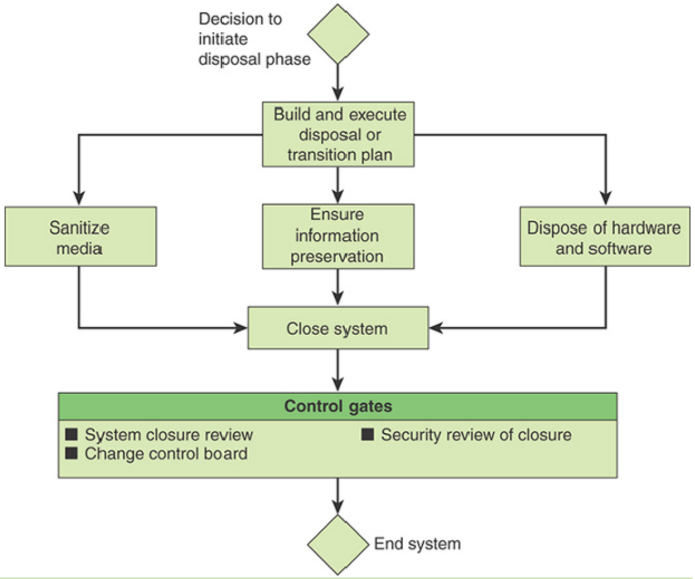
- System test readiness review
- Deployment readiness review
- Certification and accreditation review
- Authorization decision
- Final project status and financial review
- IT deployment or connection approval

3.5.5 Begin Operations and Maintenance phase



Decision to begin operations and maintenance phase
Review operational readiness
Conduct continuous monitoring
Platform configuration management and control
Control gates
Operational readiness review
Change control board
PO&AM review
Authorization decision
Decision to begin disposal phase

3.5.6 Initiate disposal phase


Decision to initiate disposal phase
Build and execute disposal or transition plan
Sanitize media
Ensure information preservation
Dispose of hardware and software
Close system
Control gates
System closure review
Change control board
Security review of closure
End system

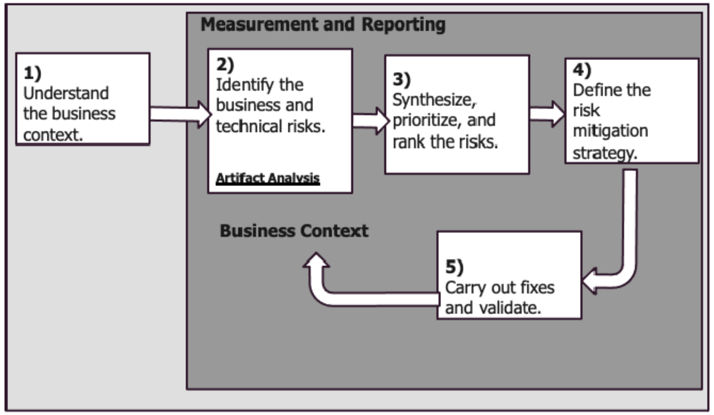
3.6 Continous Security

1. Test driven Security
2. Monitor and respond to attacks
3. Assess risks and mature security

3.7 Risk Management

3.7.1 Activities

Summary:


Measurement and Reporting
1) Understand the business context.
2) Identify the business and technical risks.
Artifact Analysis
3) Synthesize, prioritize, and rank the risks.
4) Define the risk mitigation strategy.
Business Context
5) Carry out fixes and validate.

1. Understand the business context
   - Extract and describe business goals
   - Set prios
   - Understanding what risks to consider
   - Gathering the artifacts
   - Conducting project research to the scope
2. Identifying the business and technical risks (priorize / rank)
   - Business risks impact business goals
   - Map technical risks to business goals
   - Develop a set of risk questionnaires
   - Interview the target project team
   - Analyse the research interview data
   - Evaluate software artifacts
3. Synthesize, prioritize and rank the risks
   - Prio the risks based on business goals
   - Apply risk metrics
   - Number of risks emerging over time
   - What shall be done first?
   - What is the best allocation of resources?
4. Define the risk mitigation strategy
   - Take into account: Cost, Implementation time, Likelihood of success, Competence and impact
   - Identify the validation techniques
   - Metrics are financial in nature
5. Carry out fixes and validate their correctness
   - Implement the mitigation strategy
   - Rectify artifacts

3.7.2 Summary

- Relies on continuous and consistent identification of risks.
- The five fundamentals should be applied repeatedly
- Use project management tools to track risk information

3.8 Attack Trees

- Tree-structured graph
- Showing how a system can be attacked

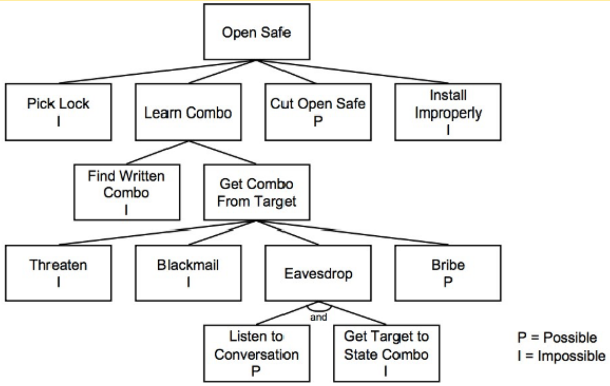
Construction:

1. Identify goals (1 Tree per goal)
2. Identify attacks against goals
3. Existing sub-trees can be plugged in

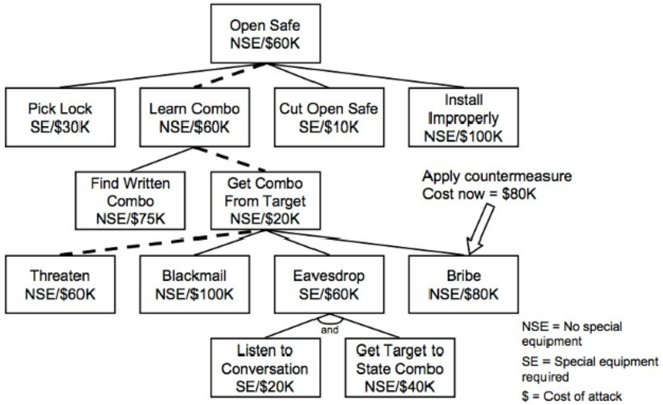
Usage:

- Propagate up the tree
- You can specify values that represent other different meanings (Equipment / Cost)
- Combine Node Values

3.8.1 Example



Countermeasures:


Open Safe NSE/$60K
Pick Lock SE/$30K
Learn Combo NSE/$60K
Cut Open Safe SE/$10K
Install Improperly NSE/$100K
Find Written Combo NSE/$75K
Get Combo From Target NSE/$20K
Apply countermeasure Cost now = $80K
Threaten NSE/$60K
Blackmail NSE/$100K
Eavesdrop SE/$60K
Bribe NSE/$80K
and
Listen to Conversation SE/$20K
Get Target to State Combo NSE/$40K
NSE = No special equipment
SE = Special equipment required
$ = Cost of attack