CIS 418/518 – Secure Software Engineering
Software Security Practices and Processes

Jagadeesh Nandigam

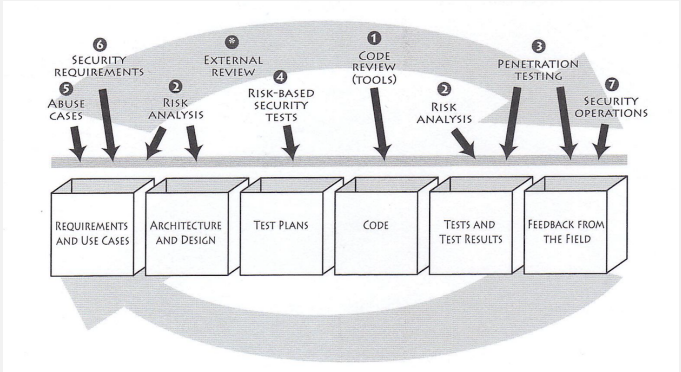
School of Computing
Grand Valley State University
nandigaj@gvsu.edu

Outline

1. Software Security Touchpoints
2. A Risk Management Framework
3. Microsoft Security Development Lifecycle (SDL)
4. Other Secure Software Development Life Cycle Processes
5. Information Security Organizations

Software Security Touchpoints

- A set of software security best practices applied to various artifacts produced during software development.
- Practices are designed to be process agnostic – can be applied to any software process used to build software.



Software Security Touchpoints

- List of best practices, in order of effectiveness:
  1. Code review
  2. Architectural risk analysis
  3. Penetration testing
  4. Risk-based security tests
  5. Abuse cases
  6. Security requirements
  7. Security operations
- The above ordering takes the following into account:
  - What software artifacts are likely to be available
  - What kinds of tools exist and how good they are
  - Challenge presented by cultural change and organization commitment
  - Early is better approach preferred to reactive strategy of "penetrate-and-patch" approach

Touchpoint: Code Review

- Code review aims to find implementation bugs
- Uses static analysis tools and manual code review
- Static analysis tools
  - Apply checks thoroughly and consistently without any bias
  - Can point to the root cause of a security problem, not just one of its symptoms
  - Produce too much noise in the form of *false positives* or *false alarms*
  - From a security perspective, *false negatives* are much worse than false positives
  - Static analysis tools that target security usually produce more false positives to minimize false negatives
- Use manual code review for "critical" components of an application
- The best a code review can uncover is around 50% of the security problems

Touchpoint: Architectural Risk Ananlysis

- Architectural risk analysis aims to find design flaws (the other 50% of the security problems)
- Designers, architects, and analysts should document assumptions and identify possible attacks
  - Misuse/abuse cases
  - Attack patterns
  - Threat models

Touchpoint: Penetration Testing

- Penetration testing (pen-testing) is a white box security analysis of a software system performed by skilled security professionals simulating the actions of a hacker.
- The pen-tester will have *permission* from the owner of the software system being tested and will be responsible to provide a report.
- Pen-testing is extremely useful if architectural risk analysis informs the tests (i.e., white box in nature).
- Be aware that network penetration tests are not the same as application or software-faced penetration tests.
- It is important for the pen-tester to keep detailed notes about how the tests were done so that the results can be verified and any issues that were uncovered can be resolved.

Touchpoint: Risk-Based Security Tests

- A good security test plan encompasses two strategies:
  - Testing of security functionality with standard functional testing techniques.
  - Risk-based security testing based on abuse/misuse cases, attack patterns, risk analysis results.
- Standard QA is about making sure good things happen.
- Security testing is about making sure *bad* things don't happen.
- Thinking like an attacker is essential in security test planning.

Touchpoint: Abuse Cases

- Abuse cases (sometimes also known as misuse cases) describe the system's behavior under attack.
- Use cases from the point of view of an actor *hostile* to the system under design.
- Abuse cases specify behavior not wanted in the proposed system for the purpose of eliciting security requirements.
- Focus is on what should be protected, from whom, and for how long.

Touchpoint: Security Requirements

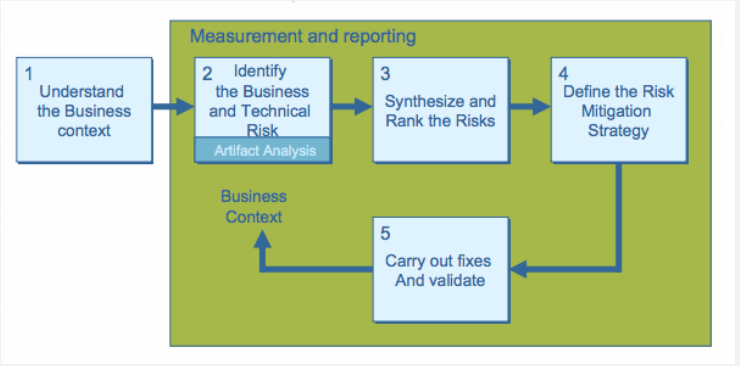
- Security must be explicitly addressed in the requirements phase.
- Good security requirements cover both overt functional security and covert emergent characteristics
- Captured via security features, abuse cases, and attack patterns.

Touchpoint: Security Operations

- Software security can benefit greatly from network security and security operations people.
- Operations people set up and monitor fielded systems during use to enhance the security posture.
- Knowledge gained by understanding attacks and exploits should be cycled back into software development.

A Risk Management Framework

- An iterative risk management approach that is deeply integrated throughout the software development lifecycle to identify, rank, track, and mitigate software security risk over time.



A Risk Management Framework

The risk management framework (RMF) consists of five fundamental activity stages:

- Understand the business context
  - Idenify and document *business goals*, priorities, and circumstances to determine what kinds of software risks to care about and which business goals are paramount.
- Identify the business and technical risks
  - Analyze data (gathered from research, risk questionnaires, and interviews) to identify and list business risks associated with each business goal
  - Determine technical risks that lead to business risks impacting business goals using either top-down or bottom-up approach:
    - Business goals ⇒ Business risks ⇒ Technical risks
    - Technical risks ⇒ Business risks ⇒ Business goals

A Risk Management Framework

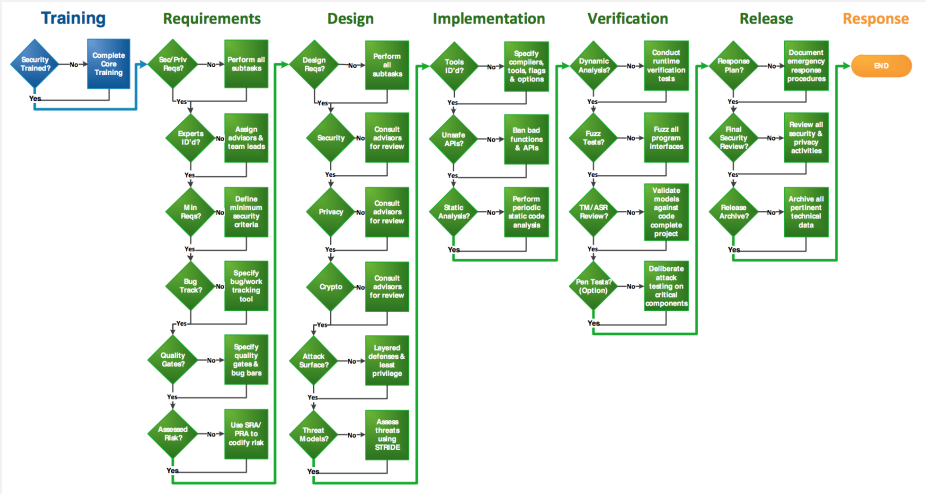
- Synthesize and rank the risks
  - Establish relationships between business goals, business risks, and technical risks
  - Create goal-to-risk relationship table to visualize relationships between business goals, business risks, and technical risks
  - Identify technical risk severity by business goals
- Define the risk mitigation strategy
  - Risk analysis is only as good as the mitigation strategy it contains
  - Create a coherent strategy for mitigating the risks in a cost-effective manner
  - Author a risk analysis report for peer review and editing
- Carry out the fixes and validate
  - Execute the risk mitigation strategy defined
  - Validate fixes to gain confidence that risks have been properly mitigated using the validation plan
  - Measure progress made against the risk mitigation strategy

Microsoft Security Development Lifecycle (SDL)

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

- Focuses on applying proven security practices at distinct points in the software development life cycle
- Sixteen *mandatory* security activities to comply with the SDL process
- Optional security activities (manual code review, penetration testing, vulnerability analysis of similar applications, etc.) are generally performed in critical applications
- Technology-agnostic (not specific to Microsoft or the Windows platform) and development methodology independent

Microsoft SDL Process Illustration



Other Secure Software Development Life Cycle Processes

- Systems Security Engineering Capability Maturity Model (SSE-CMM)
- Team Software Process for Secure Software Development (TSP-Secure)
- OWASP Software Assurance Maturity Model (OWASP-SAMM)
- Software Security Framework (as Software Integrity Toolbelt) by Synopsys
- Correctness by Construction Methodology – incorporates formal specification languages and methods into development activities

Information Security Organizations

- Several organizations provide useful information for security professionals.
  - Detecting and responding to established and emerging security threats
  - Early warning system for Internet security issues
  - Best practices for security
  - Security training
  - Security certifications
  - Unbiased and objective review of security products

Information Security Organizations

- CERT/CC (`https://www.sei.cmu.edu/about/divisions/index.cfm`)
- US-CERT (`https://www.us-cert.gov/`)
- SANS Institute (`https://www.sans.org/`)
- ISC^2 (`https://www.isc2.org/`)
- Common Criteria (`https://www.commoncriteriaportal.org/`)
- The Federal Information Processing Standard (FIPS - `https://www.corsec.com/fips-140-2/`)
- ICSA Labs (`https://www.icsalabs.com/`)
- The OWASP Foundation (`https://www.owasp.org/index.php/Main_Page`)

References

- Gary McGraw, "Software Security - Building Security In", Chapters 2 & 3, Addison Wesley.
- Microsoft SDL, https://www.microsoft.com/en-us/SDL/process/training.aspx
- A Simplified Implementation of Microsoft SDL, https://www.microsoft.com/en-us/download/details.aspx?id=12379
- Secure Software Development Life Cycle Processes `https://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_297287.pdf`