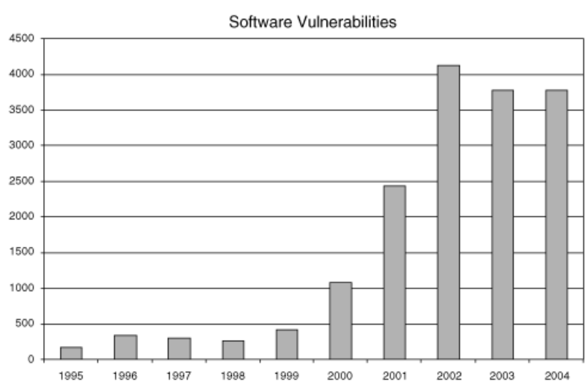
System Testing cont…

Dr. Durga Prasad Mohapatra
Professor, Dept. of CSE,
NIT Rourkela

SECURITY TESTING

Software Problem



vulnerabilities Reported by CERT/CC

- More than half of the vulnerabilities are due to buffer overruns.
- Others such as race conditions, design flaws are equally prevalent.

Software security

- It is about
  - Understanding software-induced security risks and how to manage them
  - Leveraging software engineering practice,
  - Thinking security early in the software lifecycle
  - Knowing and understanding common problems
  - Designing for security
  - Subjecting all software artifacts to thorough objective risk analyses and testing
- It is a knowledge intensive field

Software Security

- Renewed interest
  - "idea of engineering software so that it continues to function correctly under malicious attack"
  - Existing software is riddled with design flaws and implementation bugs
  - "any program, no matter how innocuous it seems, can harbor security holes"

Security Testing

- Security is a protection system that is needed to assure the customers that their data will be protected.
  - For example, internet users feel that their personal data/ information is not secure, the system loses its accountability.
- Security may include controlling access to data, encrypting data in communication, ensuring secrecy of stored data, auditing security events, etc.
  - Security breaches can result in loss of information, privacy violations, denial of service, etc.

Types of Security Requirements

- While performing security testing, the following security requirements must be considered:
  - Security requirements should be associated with each functional requirement.
    - Each functional requirement, most likely, has a specific set of related security issues to be addressed in the software implementation.
    - For example, the log on requirement in a client server system must specify the number of retries allowed, the action to be taken if the log-on fails, and so on.

Example of Security Concerns

- In addition to the security concerns that are directly related to particular requirements, a software project has security issues that are global in nature, and hence related to the application's architecture and overall implementation.
  - For example, a web application may have a global requirement that all private customer data of any kind is stored in encrypted form in the database.
  - In another example, a system wide security requirement is to use SSL to encrypt the data sent between the client browser and the web server - the testing team should verify the correctness of SSL.

Security Concerns cont…

- The problem with security testing is that security-related bugs are not as obvious as other types of bugs.
- It may be possible that the security system has failed and caused the loss of information without the knowledge of loss.
- Thus, the tester should perform security testing with the goals to identify the bugs that are very difficult to identify.

Software Vulnerability

- Vulnerability is an error that an attacker can exploit.
- Security vulnerabilities are of the following types:
  - Bugs at the implementation level, such as local implementation errors or inter-procedural interface errors.
  - Design level mistakes.
- Design level vulnerabilities are the hardest defect category to handle, but they are also the most prevalent and critical.

Software Vulnerability cont…

- Unfortunately, ascertaining whether a program has design level vulnerabilities requires great expertise
  - which makes finding not only difficult but particularly hard to automate.
- Examples: problem in error handling, unprotected data channels, incorrect or missing access control mechanisms, and timing errors especially in multithreaded systems.

How to Perform Security Testing

- Testers must use a risk based approach, grounded in both the system's architectural reality and the attacker's mindset,
  - to gauge software security adequately.
- By identifying risks and potential loss associated with those risks in the system and creating tests driven by those risks
  - The tester can focus on areas of code in which an attack is likely to succeed.
- Therefore, risk analysis can help in identifying potential security problems.
  - Once identified and ranked, can help in security testing.

Risk Management

- Risk management and security testing
  - Software security practitioners perform many different tasks to manage software risks, including:
    - Creating security abuse/misuse cases
    - Listing normative security requirements
    - Performing architectural risk analysis
    - Building risk based security test plans
    - Wielding static analysis tools
    - Performing security tests.

Risk Management cont…

- Three tasks, i.e. architectural risk analysis, risk-based security test planning, and security testing, are closely linked because a critical aspect of security testing relies on probing security risks.
- Based on design-level risk analysis and ranking of security related risks, security test plans are prepared which guide the security testing.

Risk Management cont…

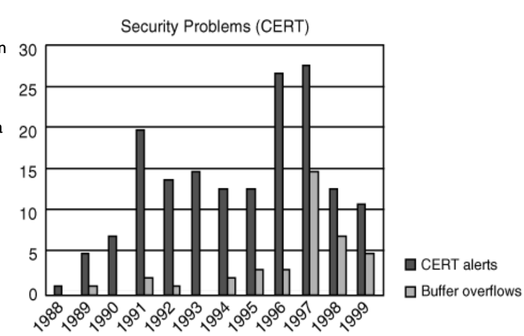
- Thus, security testing must necessarily involve two diverse approaches:
  - Testing security mechanisms to ensure that their functionality is properly implemented.
  - Performing-risk based security testing motivated by understanding and simulating the attacker's approach.

Elements of Security Testing

- Authentication
  - To establish the validity of a transmission, message, or originator
- Authorization
  - It is the process of determining that a requester is allowed to receive a service or perform an operation.
- Availability
  - It assures that the information and communication services will be ready for use when expected.
- Non-repudiation
  - A measure intended to prevent the later denial that an action happened, or a communication took place, etc.

Security problems in software

- Bug
  - An implementation level software problem
- Flaw
  - A problem at a deeper level
- Bugs + Flaws
  - leads to Risk



Solution …
Three pillars of security

Pillar I:
Applied Risk Management

- Architectural risk analysis
  - Sometimes called threat modeling or security design analysis
  - Is a best practice and is a touchpoint
- Risk management framework
  - Considers risk analysis and mitigation as a full life cycle activity

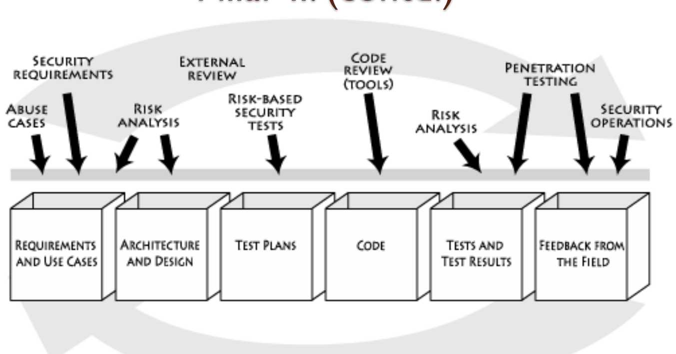
Pillar II:
Software Security Touchpoints

- "Software security is not security software"
  - Software security
    - is system-wide issues (security mechanisms and design security)
    - Emergent property
- Touchpoints in order of effectiveness (based on experience)
  - Code review (bugs)
  - Architectural risk analysis (flaws)
    - These two can be swapped
  - Penetration testing
  - Risk-based security tests
  - Abuse cases
  - Security requirements
  - Security operations

Pillar II: (contd.)

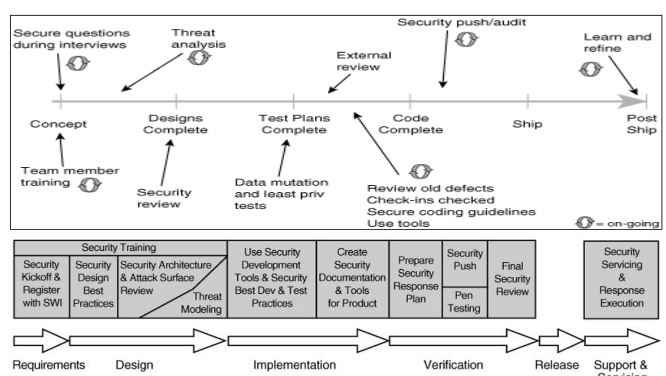
- Many organization
  - Penetration first
    - Is a reactive approach
- Code Review and Architecture Risk Analysis can be switched however skipping one solves only half of the problem
- Big organizations may adopt these touchpoints simultaneously

Pillar II: (contd.)



Software security best practices applied to various software artifacts

Pillar II: (contd.)
Microsoft's move …



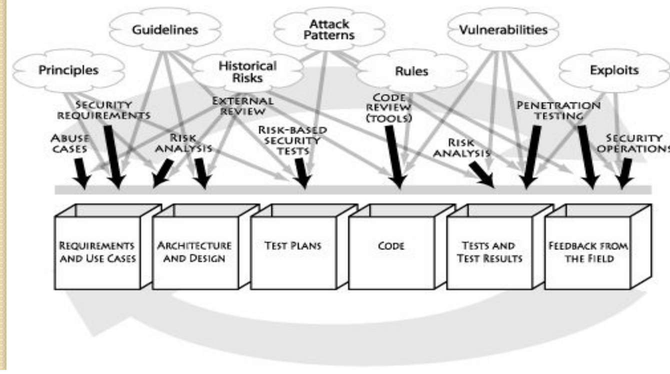
Pillar III: Knowledge

- Involves
  - Gathering, encapsulating, and sharing security knowledge
- Software security knowledge catalogs
  - Principles
  - Guidelines
  - Rules
  - Vulnerabilities
  - Exploits
  - Attack patterns
  - Historical risks

Can be put into three categories

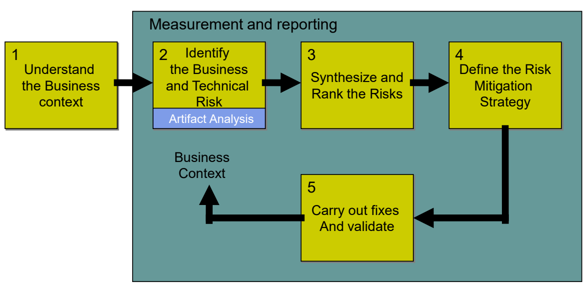
Prescriptive knowledge
Diagnostic knowledge
Historical knowledge

Pillar III: Knowledge catalogs to s/w artifacts



Risk management framework: Five Stages

- RMF occurs in parallel with SDLC activities



Stage 1: Understand Business Context

- Risk management
  - Occurs in a business context
  - Affected by business motivation
- Key activity of an analyst
  - Extract and describe business goals – clearly
    - Increasing revenue; reducing dev cost; meeting SLAs; generating high return on investment (ROI)
  - Set priorities
  - Understand circumstances

Stage 2: Identify the business & technical risks

- Business risks have several impacts
  - Direct financial loss; loss of reputation; violation of customer or regulatory requirements; increase in development cost
- Severity of risks
  - Should be captured in financial or project management terms
- Key is –
  - tie technical risks to business context

Stage 3: Synthesize and rank the risks

- Prioritize the risks alongside the business goals
- Assign risks appropriate weights for resolution
- Risk metrics
  - Risk likelihood
  - Risk impact
  - Number of risks mitigated over time

Stage 4: Risk Mitigation Strategy

- Develop a coherent strategy
  - For mitigating risks
  - In cost effective manner; account for
    - Cost Implementation time
    - Completeness Impact
    - Likelihood of success
- A mitigation strategy should
  - Be developed within the business context
  - Be based on what the organization can afford, integrate and understand
  - Must directly identify validation techniques

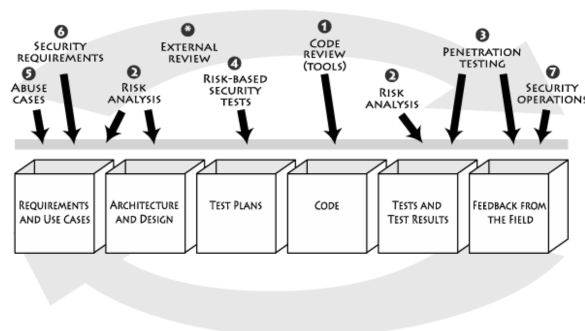
Stage 5: Carry out Fixes and Validate

- Execute the chosen mitigation strategy
  - Rectify the artifacts
  - Measure completeness
  - Estimate
    - Progress, residual risks
- Validate that risks have been mitigated
  - Testing can be used to demonstrate
  - Develop confidence that unacceptable risk does not remain

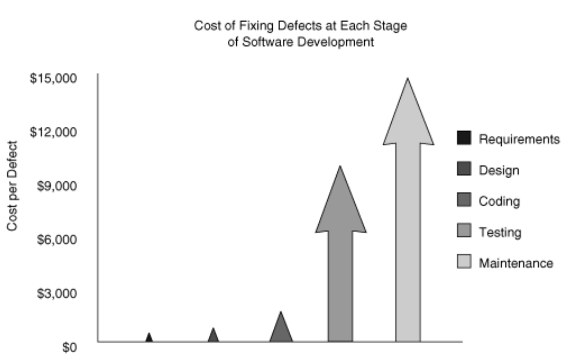
Risk Mitigation Framework - A Multi-loop

- Risk management is a continuous process
  - Five stages may need to be applied many times
  - Ordering may be interleaved in different ways
    - Risk can emerge at any time in SDLC
      - One way – apply in each phase of SDLC
    - Risk can be found between stages
- Level of application
  - Primary – project level
    - Each stage must capture complete project
  - SDLC phase level
  - Artifact level
- It is important to know that Risk Mitigation is
  - Cumulative
  - At times arbitrary and difficult to predict

Seven Touchpoints



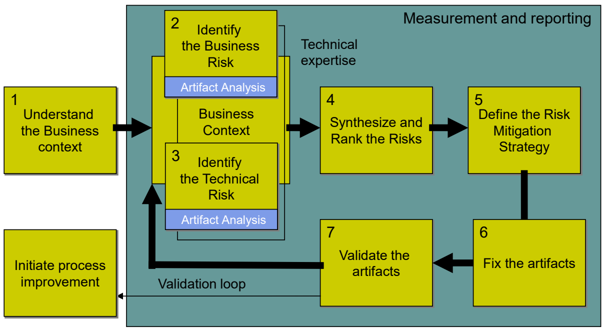
Cost of fixing defect at each stage



Architectural risk analysis

- Design flaws
  - about 50% of security problem
  - Can't be found by looking at code
    - A higher level of understanding required
- Risk analysis
  - Track risk over time
  - Quantify impact
  - Link system-level concerns to probability and impact measures
  - Fits with the RMF

ARA within RMF



ARA process

- Attack resistance analysis
  - Steps
    - Identify general flaws using secure design literature and checklists
      - Knowledge base of historical risks useful
    - Map attack patterns using either the results of abuse case or a list of attack patterns
    - Identify risk based on checklist
    - Understand and demonstrate the viability of these known attacks
      - Use exploit graph or attack graph

- Note: particularly good for finding known problems

ARA process

- Ambiguity analysis
  - Discover new risks – creativity requried
  - A group of analyst and experience helps – use multiple points of view
    - Unify understanding after independent analysis
  - Uncover ambiguity and inconsistencies
- Weakness analysis
  - Assess the impact of external software dependencies
  - Modern software
    - is built on top of middleware such as .NET and J2EE
    - Use DLLs or common libraries
  - Need to consider
    - COTS
    - Framework
    - Network topology
    - Platform
    - Physical environment
    - Build environment

Software penetration testing

- Most commonly used today
- Currently
  - Outside->in approach
  - Better to do after code review and ARA
  - As part of final preparation acceptance regimen
  - One major limitation
    - Almost always a too-little-too-late attempt at the end of a development cycle
      - Fixing things at this stage
        - May be very expensive
        - Reactive and defensive

Software penetration testing

- A better approach
  - Penetration testing from the beginning and throughout the life cycle
  - Penetration test should be driven by perceived risk
  - Best suited for finding configuration problems and other environmental factors
  - Make use of tools
    - Takes care of majority of grunt work
    - Tool output lends itself to metrics
    - Tools for penetration testing
      - fault injection tools;
      - attacker's toolkit: disassemblers and decompilers; coverage tools monitors

Risk based security testing

- Testing must be
  - Risk-based
  - grounded in both the system's architectural reality and the attacker's mindset
    - Better than classical black box testing
  - Different from penetration testing
    - Level of approach
    - Timing of testing
      - Penetration testing is primarily on completed software in operating environment; outside->in

Risk based security testing

- Security testing
  - Should start at feature or component/unit level testing
  - Must involve two diverse approaches
    - Functional security testing
      - Testing security mechanisms to ensure that their functionality is properly implemented
    - Adversarial security testing
      - Performing risk-based security testing motivated by understanding and simulating the attacker's approach

Abuse cases

- Creating anti-requirements
  - Important to think about
    - Things that you don't want your software to do
    - Requires: security analysis + requirement analysis
  - Anti-requirements
    - Provide insight into how a malicious user, attacker, thrill seeker, competitor can abuse your system
    - Considered throughout the lifecyle
      - indicate what happens when a required security function is not included

Abuse cases

- Creating an attack model
  - Based on known attacks and attack types
  - Do the following
    - Select attack patterns relevant to your system – build abuse case around the attack patterns
    - Include anyone who can gain access to the system because threats must encompass all potential sources
  - Also need to model attacker

Security requirements and operations

- Security requirements
  - Difficult tasks
  - Should cover both over functional security and emergent characteristics
    - Use requirements engineering approach
- Security operations
  - Integrate the security operations
    - e.g. software security should be integrated with network security

Research Motivations

- Absence of Data-set: So we are in need of an efficient data-set for analyzing Intrusion Detection Systems.
- Machine-Learning Techniques will improve the security by around 200%.
- Marking threats according to their effect on vulnerability is needed to draw architecture for software security.
- All Viruses, Trojans and Malwares should be studied and handled carefully to make our system strong enough against these, in real- time.

Latent Errors: How Many Errors are Still Remaining?

- Make a few arbitrary changes to the program:
  - Artificial errors are seeded into the program.
  - Check how many of the seeded errors are detected during testing.

Error Seeding

- Let:
  - N be the total number of errors in the system
  - n of these errors be found by testing.
  - S be the total number of seeded errors,
  - s of the seeded errors be found during testing.

Error Seeding

- $n/N = s/S$
- $N = S \quad n/s$
- Remaining defects:

  $N - n = n \quad ((S - s)/ s)$

Quiz 2

- 100 errors were introduced.
- 90 of these errors were found during testing
- 50 other errors were also found.
- Find errors remaining in code.

Quiz 2: Solution

- 100 errors were introduced.
- 90 of these errors were found during testing
- 50 other errors were also found.
- Remaining errors=

  $50 \ (100-90)/90 = 6$

Challenges in Error Seeding

- The kinds of seeded errors should match closely with existing errors:
  - However, it is difficult to predict the types of errors that exist.
- Categories of remaining errors:
  - can be estimated by analyzing historical data from similar projects.

Quiz 3

- Before system testing 100 errors were seeded.
- During system testing 90 of these were detected.
- 150 other errors were also detected
- How many unknown errors remain after system testing?

Summary

- Discussed security testing in detail.
- Discussed Types of Security Requirements
- Explained Software Vulnerability.
- Presented the Elements of Security Testing.
- Discussed the Security problems in software and the solution.
- Discussed how to know the latent errors.

References

1. Rajib Mall, Fundamentals of Software Engineering, (Chapter – 10), Fifth Edition, PHI Learning Pvt. Ltd., 2018.
2. Naresh Chauhan, Software Testing: Principles and Practices, (Chapter – 7), Second Edition, Oxford University Press, 2016.

THANK YOU