Software Security Testing

Dr. Mark C. Paulk

SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Jonsson School of Engineering and Computer Science

Security

A measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized.

An action taken against a computer system with the intention of doing harm is called an attack.

- **an unauthorized attempt to access data or services**
- **an unauthorized attempt to modify data**
- **intended to deny services to legitimate users**

L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, Third Edition, 2012.

Security Characteristics

CIA approach to security

- **Confidentiality**
- **Integrity**
- **Availability**

Authentication

Nonrepudiation

Authorization

Confidentiality & Integrity

Confidentiality

- **data or services are protected from unauthorized access**
  - **a hacker cannot access your income tax returns on a government computer**

Integrity

- **data or services are not subject to unauthorized manipulation**
  - **your grade has not been changed since your instructor assigned it**

Availability & Authentication

Availability

- **the system will be available for legitimate use**
  - **a denial-of-service attack won't prevent you from ordering book from an online bookstore**

Authentication

- **verifies the identities of the parties to a transaction and checks if they are truly who they claim to be**
  - **when you get an email purporting to come from a bank, authentication guarantees that it actually comes from the bank**

Nonrepudiation & Authentication

Nonrepudiation

- **guarantees that the sender of a message cannot later deny having sent the message, and that the recipient cannot deny having received the message**
  - **you cannot deny ordering something from the Internet, or the merchant cannot disclaim getting your order**

Authorization

- **grants a user the privileges to perform a task**
  - **an online banking system authorizes a legitimate user to access his account.**

Information Security Controls (UTD ISO)

Confidentiality

- data classification
- encryption
- malware prevention
- physical security
- users provided only necessary access

Integrity

- access control
- file version control
- file change monitoring

Availability

- disaster recovery
- file backups
- malware prevention
- network drives and cloud storage

Accountability

- event logging
- individual accounts for each user
- two-factor authentication

Security General Scenario

Source

- **Human or another system which may have been previously certified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.**

Stimulus

- **Unauthorized attempt is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.**

Scenario: Artifact & Environment

Artifact

- **system services**
- **data within the system**
- **a component or resources of the system**
- **data produced or consumed by the system**

Environment: System is

- **either online or offline**
- **either connected to or disconnected from a network**
- **either behind a firewall or open to a network**
- **fully operational, partially operational, or not operational**

Scenario: Response

Transactions are carried out in fashion such that

- **Data or services are protected from unauthorized access.**
- **Data or services are not being manipulated without authorization.**
- **Parties to a transaction are identified with assurance.**
- **The parties to the transaction cannot repudiate their involvements.**
- **The data resources and system services will be available for legitimate use**

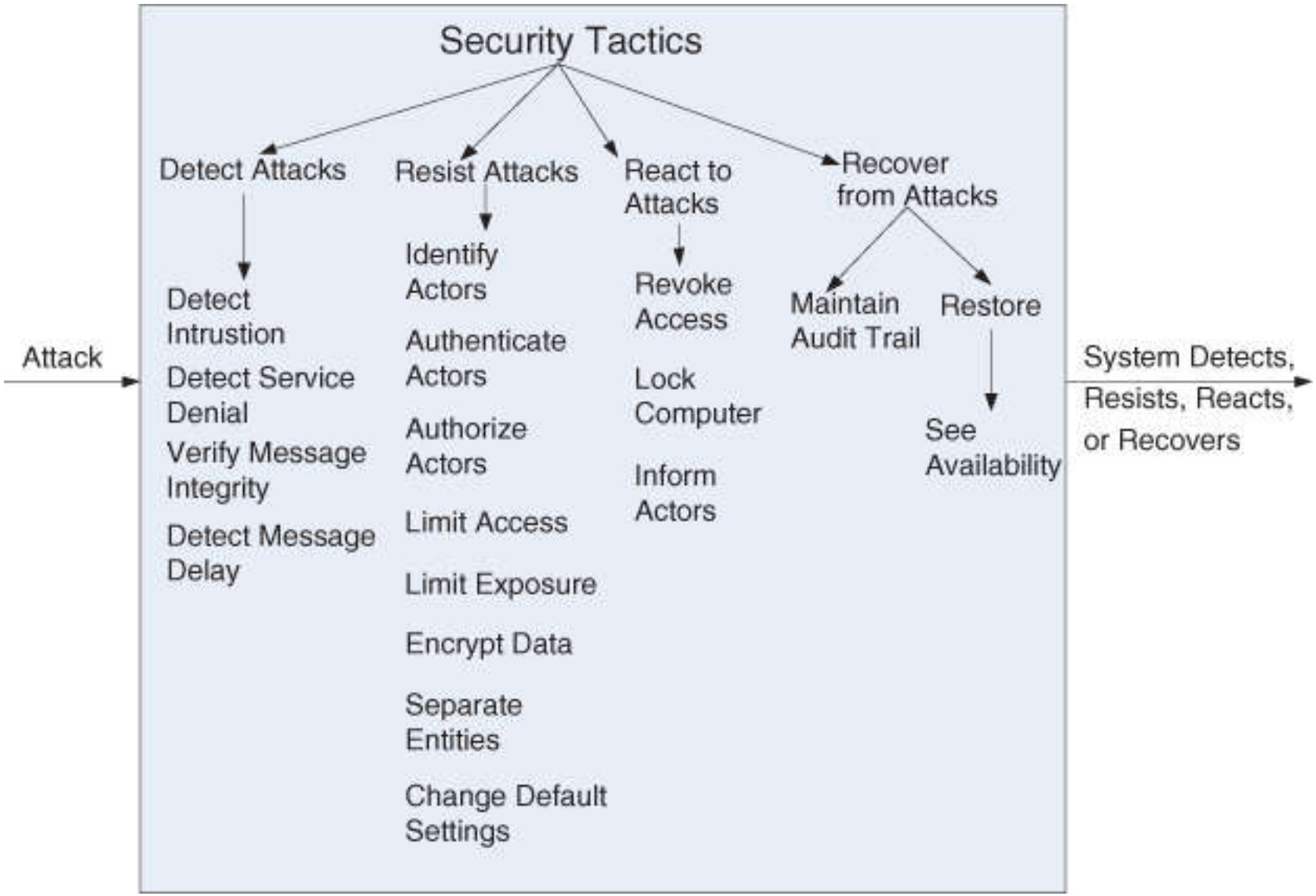
The system tracks activities within it by

- **Recording access or modification**
- **Recording attempts to access data, resources, or services**
- **Notifying appropriate entities (people or systems) when an apparent attack is occurring**

Scenario: Response Measure

One or more of the following:

- **How much of the system is compromised when a particular component or data value is compromised?**
- **How much time passed before an attack was detected?**
- **How many attacks were resisted?**
- **How long does it take to recover from a successful attack?**
- **How much data is vulnerable to a particular attack?**


Security Tactics
Detect Attacks
Resist Attacks
React to Attacks
Recover from Attacks
Detect Intrustion
Detect Service Denial
Verify Message Integrity
Detect Message Delay
Identify Actors
Authenticate Actors
Authorize Actors
Limit Access
Limit Exposure
Encrypt Data
Separate Entities
Change Default Settings
Revoke Access
Lock Computer
Inform Actors
Maintain Audit Trail
Restore
See Availability
Attack
System Detects, Resists, Reacts, or Recovers

CERT Top 10 Security Practices

1) **Validate input.**
2) **Heed compiler warnings.**
3) **Architect and design for security policies.**
4) **Keep it simple.**
5) **Default deny.**
6) **Adhere to the principle of least privilege.**
7) **Sanitize data sent to other software.**
8) **Practice defense in depth.**
9) **Use effective quality assurance techniques.**
10) **Adopt a software construction security standard.**

R.C. Seacord, The CERT C Secure Coding Standard, 2008.

SWEBOK on Security

Software design reviews can evaluate security.

Data flows (and therefore data flow diagrams) can be used for security analysis.

- **they offer identification of possible paths for attack and disclosure of confidential information**

Reused and off-the-shelf software components should meet the same security requirements as new software.

The choices of allowable programming language subsets and usage standards are important aids in achieving higher security.

Construction languages and their implementations (for example, compilers) can be serious contributors to security vulnerabilities.

- **uncritical usage of C and C++ are questionable choices from a security viewpoint**

Coding

- **prevention of code-level security breaches**
  - **buffer overflows or array index bounds, for example**
- **faults introduced during construction can result in serious security vulnerabilities**
  - **not only faults in security functionality but also faults elsewhere that allow bypassing of this functionality and other security weaknesses or violations**

Fuzz testing (or fuzzing) is a special form of random testing aimed at breaking the software often used for security testing.

Security, in terms of access control and the backup facilities, is a key aspect of library management.

Secure Software Development (SWEBOK)

Builds security in software by following a set of established and/or recommended rules and practices in software development.

- **Secure software maintenance complements secure software development by ensuring the no security problems are introduced during software maintenance.**

A generally accepted view concerning software security is that it is much better to design security into software than to patch it in after software is developed.

- **Security must also be taken into consideration when performing software maintenance as security faults and loopholes can be and often are introduced during maintenance.**

OWASP Top 10 Most Critical Web Application Security Risks 2017 (draft)

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Broken Access Control
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Insufficient Attack Protection
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities
A10 – Underprotected APIs

Software Requirements Security
(SWEBOK)

Deals with the clarification and specification of security policy and objectives into software requirements.

Software requirements
- **specific functions that are required for the sake of security**

Threats/risks
- **possible ways that the security of software is threatened**

Security Requirements (Sommerville 2016)

Security requirements are "shall not" requirements.

- **they specify what should not happen rather than functionality or required behavior**

It is not usually possible to define this unwanted behavior as simple constraints to be checked by the system.

You can demonstrate, in principle, that a system meets its functional requirements; it is impossible to prove that a system does not do something.

- **you can generate requirements to protect against some known types of attack**
- **you cannot derive requirements for unknown types of attack**

Software Design Security (SWEBOK)

Deals with the design of software modules that fit together to meet the security objectives specified in the security requirements.

Factors considered may include frameworks and access modes that set up the overall security monitoring/enforcement strategies, as well as the individual policy enforcement mechanisms.

Software Construction Security (SWEBOK)

Concerns the question of how to write actual programming code for specific situations such that security considerations are taken care of.

Can mean the way a specific function is coded, such that the coding itself is secure

Can mean the coding of security into software

Example Security Coding Rules
(SWEBOK)

Structure the process so that all sections requiring extra privileges are modules.

- **The modules should be as small as possible and should perform only those tasks that require those privileges.**

Ensure that any assumptions in the program are validated.

- **If this is not possible, document them for the installers and maintainers so they know the assumptions that attackers will try to invalidate.**

Ensure that the program does not share objects in memory with any other program.

The error status of every function must be checked.

- **Do not try to recover unless neither the cause of the error nor its effects affect any security considerations.**
- **The program should restore the state of the software to the state it had before the process began, and then terminate.**

Error Messages and Security (SWEBOK)

Error messages should

- **clearly explain what is happening so that users know what is going on in the software**
- **pinpoint the cause of the error, if at all possible**
- **be displayed right when the error condition occurs.**
- **not overload the users with too much information and cause them to ignore the messages all together**

However, messages relating to security access errors should not provide extra information that would help unauthorized persons break in.

Security Checklist Examples
(Sommerville 2016)

Do all files created in the application have appropriate access permissions?

- **The wrong access permissions may lead to these files being accessed by unauthorized users.**

Does the system automatically terminate user sessions after a period of inactivity?

- **Sessions that are left active may allow unauthorized access through an unattended computer.**

If the system is written in a programming language without array bounds checking, are there situations where buffer overflow may be exploited?

- **Buffer overflow may allow attackers to send code strings to the system and then execute them.**

If passwords are set, does the system check that passwords are "strong"?

- **Strong passwords consist of mixed letters, numbers, and punctuation, and are not normal dictionary entries. They are more difficult to break than simple passwords.**

Are inputs from the system's environment always checked against an input specification?

- **Incorrect processing of badly formed inputs is a common cause of security vulnerabilities.**

Some Proven Application Security Principles (OWASP)

- **Apply defense in depth (complete mediation)**
- **Use a positive security model (fail-safe defaults, minimize attack surface)**
- **Fail securely**
- **Run with least privilege**
- **Avoid security by obscurity (open design)**
- **Keep security simple (verifiable, economy of mechanism)**
- **Detect intrusions (compromise recording)**
- **Don’t trust infrastructure**
- **Don’t trust services**
- **Establish secure defaults (psychological acceptability)**

Build Security Into the Software Process (SWEBOK)

Software is only as secure as its development process goes.

Secure Development Lifecycle (SDL)

- **a classical spiral model that takes a holistic view of security from the perspective of software lifecycle and ensures that security is inherent in software design and development, not an afterthought later in production.**
- **SDL process is claimed to reduce software maintenance costs and increase reliability of software concerning software security related faults**

Adding Security Processes (SWEBOK)

Security concerns during software development may necessitate one or more software processes to protect the security of the development environment and reduce the risk of malicious acts.

Security Testing (IEEE 29119.4 A2.15)

The purpose of security testing is to evaluate the degree to which a test item and its associated data are protected so that unauthorized persons or systems cannot use, read, or modify them and authorized persons or systems are granted required access to them.

Security testing uses a model of the test item that specifies its security requirements

- **including any required security design standards to which the test item must conform**

Security requirements are concerned with the ability to protect the data and functionality of a test item from unauthorized users and malicious use.

Security Testing Techniques

Penetration testing

- **involves attempted access to a test item (including its functionality and/or private data) by a tester that is mimicking the actions of an unauthorized user**

Privacy testing

- **involves attempted access to private data and verification of the audit trail (i.e., trace) that is left behind when users access private data**

Security auditing

- a type of static testing in which a tester inspects, reviews, or walks through the requirements and code of a test item to determine whether any security vulnerabilities are present

Vulnerability scanning

- involves the use of automated testing tools to scan a test item for signs of specific known vulnerabilities

Summary – Things to Remember

Confidentiality, integrity, and availability

Authentication, non-repudiation, and authorization

Questions and Answers