Protecting Information Assets

- Unit #14 -

Computer Application Security
Agenda

• Introduction
• Software development life cycle (SDLC)
• SDLC and security
• Test taking tip
• Quiz
Application Security

As applications become more accessible though the web, cloud and mobile devices, organizations are being forced to abandon their reactive approach to security and, instead, to take a proactive approach by minimizing risk directly in the software they buy, create and use to serve themselves and their customers.
Usual trend

1. Buggy software is released to the market to beat the competition.

2. Hackers find new vulnerabilities and weaknesses in new software.

3. Websites post these vulnerabilities and how to exploit them.

4. Vendor develops and releases patch to fix vulnerabilities.

5. The new patch goes on the stack of software patches that all network administrators need to test and install.
Perimeter security solutions are often relied on as a solution to insecure application development practices.
Perimeter security solutions are often relied on as a solution to insecure application development practices.
Past and current situation….

• Application developers are not security professionals
  – Software vendors skip proper security architecture, design and testing steps as they race to beat competitors to market with new features
• Secure application development practices have not historically been taught in computer science and other academic departments, and are only recently being considered and adopted by developers
• Development projects’ scope and budgets focus on functionality, not security
• Security professionals typically not software developers
  – Often lack insight for understanding of software vulnerabilities
• IT customers...
  – “Trained” to expect to receive flawed software needing upgrades and patches
  – Unable to control flaws in software they purchase, so they rely on perimeter protection
Security strategy needs to be a consideration at each level of the architecture.
Best Practice: **Build Security In**

**Security Architecture**

Creation, use and enforcement of System Architecture standards provides the basic building blocks for developing, implementing and maintaining secure applications.

**Software Development Life Cycle**

Attention to security throughout the Software Development Life Cycle (SDLC) is the key to creating secure, manageable applications regardless of platform or technologies.

**Procurement Standards**

Describing the process and detailed criteria that will be used to assess the security level of third party software enables companies to make strategic, security-sensitive decisions about purchased software purchases.
Software Development Life Cycle

Requirements
- Why the software was created (i.e. goals)
- Who the software was created for
- What the software is intended to do

Design
- Specifications identifying how software and data will be formed to accomplish goals and used to meet requirements

Development
- Programming software code implemented and integrated to meet specifications

Testing-Validation
- Assuring software and data works as planned to meet the goals

Release-Maintenance
- Deploying software and data, and assuring they are properly configured, patched and monitored
Software Development Life Cycle (SDLC)

1. Requirements analysis

2. Design

3. Develop ("make") / Implement ("buy")

4. Testing/Validation

5. Release/Maintenance
Software Development Life Cycle (SDLC)

1. Requirements analysis
   - Informational, functional, behavioral, and performance specifications...

2. Design
   - Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models...

3. Develop (“make”) / Implement (“buy”)
   - Source code control system, code reviews, daily builds, automated CASE tools...

4. Testing/Validation
   - Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing

5. Release/Maintenance
   - Release testing
SDLC and Security

1. Requirements analysis
   - *Informational, functional, behavioral, and performance specifications...*
   + CIA risk assessment, + Risk-level acceptance,...

2. Design
   - *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models...*
   + Threat modeling, + Attack surface analysis,...

3. Develop ("make") / Implement ("buy")
   - *Source code control system, code reviews, daily builds, automated CASE tools...*
   + Developer security training, + Static analysis, + Secure code repositories,...

4. Testing/Validation
   - *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*
   + Dynamic analysis, + Fuzzing,...

5. Release/Maintenance
   - *Release testing*
   + Separation of duties, +Change management,...
SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications...*
- + CIA risk assessment, + Risk-level acceptance,...

<table>
<thead>
<tr>
<th>Operating Unit / Function</th>
<th>Process name</th>
<th>Name of Asset</th>
<th>Personal Identifying Information (PII) (Y/N)</th>
<th>Personal Health Information (PHI) (Y/N)</th>
<th>Critical Infrastructure Information (CII) (Y/N)</th>
<th>Customer Data (Y/N)</th>
<th>Organization Data (Y/N)</th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Availability</th>
<th>Categorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal Distribution System</td>
<td>ChilledWater</td>
<td>TDS</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Thermal Distribution System</td>
<td>HeatedWater</td>
<td>TDS</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Thermal Distribution System</td>
<td>TDS</td>
<td>TDS</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Communication</td>
<td>Data</td>
<td>COM</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Communication</td>
<td>Voice</td>
<td>COM</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Communication</td>
<td>Security</td>
<td>COM</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Communication</td>
<td>Free</td>
<td>COM</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Public Works</td>
<td>Sewer</td>
<td>Utilities</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Public Works</td>
<td>Stormwater</td>
<td>Utilities</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Public Works</td>
<td>Water</td>
<td>Utilities</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Public Works</td>
<td>Utilities</td>
<td>Utilities</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>External</td>
<td>Parcels</td>
<td>Parcels</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
Software requirements often specified with...

1. **Information model** – Type and content of information that will be processed and how it will be processed

2. **Functional model** – Tasks and functions the application needs to carry out

3. **Behavioral model** – States the application will be in and transition among
Software requirements specifications documents help support:

- **Validation**
  - “Did they build the right application?”
  - In large complex applications it is easy to lose sight of the main goal?
  - Does the application/system provide the solution for the intended problem?

- **Verification**
  - “Did they build the application right?”
  - Applications can be built that do not match the original specifications
    - *Often not designed/developed with security requirements in mind...*
  - Determines if the application accurately represent and meets the specifications
  - Ensures that the specifications were met properly
1. Informational Model
Informational Model

Entity-Relational or UML Data Model

Entity

Data dictionary

MIS 5206 Protecting Information Assets
Verification

**Did they build the application right?**

**Does it match the data model?**
Functional model

**Figure 2. Use Case Hierarchy Diagram**

**Validation**

*Did they build the right application?*
2. Functional model

Validation

Did they build the right application?

MIS 5206 Protecting Information Assets
3. Behavioral models – swim lane model

**Validation**

“Did they build the right application?”

**Verification**

“Did they build the application right?”
3. Behavioral model

Validation
“Did they build the right application?”

Verification
“Did they build the application right?”
Behavioral model

Illustration of status transition model throughout the workflow among departments

For the example:
- PIF = Project Information Form
- SIF = Study Information Form
- WSSPU = Water and Sewer Planning Unit
3. Behavioral model – workflow/status transition model

For the example:
• PIF = Project Information Form
• SIF = Study Information Form
SDLC and Security

Requirements analysis

– *Informational, functional, behavioral, and performance specifications*...
+ CIA risk assessment, + Risk-level acceptance,...

Design

– *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models*...

+ Threat modeling, + Attack surface analysis,...
Threat modeling is a systematic approach for understanding how different threats could be realized and a successful attack could take place.
Attack surface is what is available to be used by an attacker against the application itself.

Goal of attack surface analysis is to identify and reduce the amount of code and functionality accessible to untrusted users.

Development team should reduce the attack surface as much as possible to remove “resources” that can be used as avenues for the attacker to use.
MITRE’s Common Application Vulnerabilities

- Development Concepts
  - Configuration - (16)
  - Data Processing Errors - (19)
  - Pathname Traversal and Equivalence Errors - (21)
  - Numeric Errors - (189)
  - 7PK - Security Features - (254)
  - 7PK - Time and State - (361)
  - Error Conditions, Return Values, Status Codes - (389)
  - Resource Management Errors - (399)
  - Channel and Path Errors - (417)
  - Handler Errors - (429)
  - Behavioral Problems - (438)
  - Business Logic Errors - (840)
  - Web Problems - (442)
  - User Interface Security Issues - (355)
  - Initialization and Cleanup Errors - (452)
  - Pointer Issues - (465)
  - Mobile Code Issues - (490)
  - Often Misused: Arguments and Parameters - (559)
  - Expression Issues - (569)
  - Violation of Secure Design Principles - (657)
  - Bad Coding Practices - (1006)
Insecure Interaction Between Components
These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

<table>
<thead>
<tr>
<th>CWE ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>CWE-651</td>
<td>URL Redirection to Untrusted Site ('Open Redirect')</td>
</tr>
</tbody>
</table>

Porous Defenses
These weaknesses are related to defensive techniques that are often misused, abused, or just plain ignored.

<table>
<thead>
<tr>
<th>CWE ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-306</td>
<td>Missing Authentication for Critical Function</td>
</tr>
<tr>
<td>CWE-302</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>CWE-796</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>CWE-863</td>
<td>Incorrect Authorization</td>
</tr>
<tr>
<td>CWE-732</td>
<td>Incorrect Permission Assignment for Critical Resource</td>
</tr>
<tr>
<td>CWE-327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
</tr>
<tr>
<td>CWE-307</td>
<td>Improper Restriction of Excessive Authentication Attempts</td>
</tr>
<tr>
<td>CWE-799</td>
<td>Use of a One-Way Hash without a Salt</td>
</tr>
</tbody>
</table>

Risky Resource Management
These weaknesses are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

<table>
<thead>
<tr>
<th>CWE ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td>
</tr>
<tr>
<td>CWE-494</td>
<td>Download of Code Without Integrity Check</td>
</tr>
<tr>
<td>CWE-829</td>
<td>Inclusion of Functionality from Untrusted Control Sphere</td>
</tr>
<tr>
<td>CWE-676</td>
<td>Use of Potentially Dangerous Function</td>
</tr>
<tr>
<td>CWE-131</td>
<td>Incorrect Calculation of Buffer Size</td>
</tr>
<tr>
<td>CWE-134</td>
<td>Uncontrolled Format String</td>
</tr>
<tr>
<td>CWE-190</td>
<td>Integer Overflow or Wraparound</td>
</tr>
</tbody>
</table>
T10 - OWASP Top 10 Application Security Risks – 2017

A1: 2017 - Injection
A2: 2017 - Broken Authentication
A3: 2017 - Sensitive Data Exposure
A4: 2017 - XML External Entities (XXE)
A5: 2017 - Broken Access Control
A6: 2017 - Security Misconfiguration
A7: 2017 - Cross-Site Scripting (XSS)
A8: 2017 - Insecure Deserialization
A9: 2017 - Using Components with Known Vulnerabilities
A10: 2017 - Insufficient Logging & Monitoring

https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet
SDLC and Security

Requirements analysis

– *Informational, functional, behavioral, and performance specifications*...
+ CIA risk assessment, + Risk-level acceptance,...

Design

– *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models*...
+ Threat modeling, + Attack surface analysis,...

Develop (“*make*”) / Implement (“*buy*”)

– *Source code control system, code reviews, daily builds, automated CASE tools*...
+ Developer security training, + Static analysis, + Secure code repositories,...
## Secure Software Development Curriculum

<table>
<thead>
<tr>
<th>Level</th>
<th>Course</th>
<th>Certification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 1</strong></td>
<td>DEV522: Defending Web Applications Security Essentials</td>
<td>GWEB</td>
</tr>
<tr>
<td></td>
<td>DEV531: Defending Mobile Applications Security Essentials</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEV534: Secure DevOps: A Practical Introduction</td>
<td></td>
</tr>
<tr>
<td><strong>Level 2</strong></td>
<td>DEV541: Secure Coding in Java/JEE: Developing Defensible Applications</td>
<td>GSSP-JAVA</td>
</tr>
<tr>
<td></td>
<td>DEV543: Secure Coding in C &amp; C++</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEV544: Secure Coding in .NET: Developing Defensible Applications</td>
<td>GSSP-.NET</td>
</tr>
<tr>
<td><strong>Specialty Courses</strong></td>
<td>SEC542: Web App Penetration Testing and Ethical Hacking</td>
<td>GWAPT</td>
</tr>
<tr>
<td></td>
<td>SEC642: Advanced Web App Penetration Testing, Ethical Hacking, and Exploitation Techniques</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HST: Certified Secure Software Lifecycle Professional (CSSLP®) CBK® Training Seminar</td>
<td></td>
</tr>
</tbody>
</table>
List of tools for static code analysis

From Wikipedia, the free encyclopedia

This article needs additional cita[Learn how and when to remove this template]

This is a list of tools for static code analysis.

<table>
<thead>
<tr>
<th>Name</th>
<th>Manager</th>
<th>Established</th>
<th>Server side:</th>
<th>Client side:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allof</td>
<td>Debian Project</td>
<td>2003</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Assembla</td>
<td>Assembla, Inc.</td>
<td>2005</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>Betavine</td>
<td>Vodafone</td>
<td>2007</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>Bitbucket</td>
<td>Atlassian</td>
<td>2008</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CloudForge</td>
<td>CollabNet</td>
<td>2000</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>CodePlex</td>
<td>Microsoft</td>
<td>2006-05</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>GitHub</td>
<td>GitHub, Inc.</td>
<td>2008-04</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Gna!</td>
<td>Unknown</td>
<td>2004-01</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GNU Savannah</td>
<td>Savannah Administration</td>
<td>2001-01</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Kallithea</td>
<td>SFC</td>
<td>2014[9]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Launchpad</td>
<td>Canonical</td>
<td>2004</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OSDN</td>
<td>OSDN K.K.</td>
<td>2002-04</td>
<td>Unknown</td>
<td>Yes</td>
</tr>
<tr>
<td>Ourproject.org</td>
<td>Comuntes Collective</td>
<td>2002</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OW2 Consortium</td>
<td>OW2 Consortium</td>
<td>Unknown</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>Rosetta Code</td>
<td>Unknown</td>
<td>2007</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>SEUL</td>
<td>Unknown</td>
<td>1997-05</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>SourceForge</td>
<td>BizX LLC</td>
<td>1999-11</td>
<td>Yes[12][13]</td>
<td>Yes</td>
</tr>
<tr>
<td>Tigris.org</td>
<td>(community)</td>
<td>2000</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>Team Foundation Server</td>
<td>Microsoft</td>
<td>2012-2005[16]</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>Visual Studio Team Services</td>
<td>Microsoft</td>
<td>2012[17]</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Language

Multi-language

- BlueOptima – Coding Effort Analytics objectively measure the productivity and maintainability of a code base.
- CAST Application Intelligence Platform – Detailed, audience-specific data for major databases.

### Table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Manager</th>
<th>Established</th>
<th>Server side:</th>
<th>Client side:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allof</td>
<td>Debian Project</td>
<td>2003</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Assembla</td>
<td>Assembla, Inc.</td>
<td>2005</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>Betavine</td>
<td>Vodafone</td>
<td>2007</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>Bitbucket</td>
<td>Atlassian</td>
<td>2008</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CloudForge</td>
<td>CollabNet</td>
<td>2000</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>CodePlex</td>
<td>Microsoft</td>
<td>2006-05</td>
<td>No</td>
<td>Unknown</td>
</tr>
<tr>
<td>GitHub</td>
<td>GitHub, Inc.</td>
<td>2008-04</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Gna!</td>
<td>Unknown</td>
<td>2004-01</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GNU Savannah</td>
<td>Savannah Administration</td>
<td>2001-01</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Kallithea</td>
<td>SFC</td>
<td>2014[9]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Launchpad</td>
<td>Canonical</td>
<td>2004</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OSDN</td>
<td>OSDN K.K.</td>
<td>2002-04</td>
<td>Unknown</td>
<td>Yes</td>
</tr>
<tr>
<td>Ourproject.org</td>
<td>Comuntes Collective</td>
<td>2002</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OW2 Consortium</td>
<td>OW2 Consortium</td>
<td>Unknown</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>Rosetta Code</td>
<td>Unknown</td>
<td>2007</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>SEUL</td>
<td>Unknown</td>
<td>1997-05</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>SourceForge</td>
<td>BizX LLC</td>
<td>1999-11</td>
<td>Yes[12][13]</td>
<td>Yes</td>
</tr>
<tr>
<td>Tigris.org</td>
<td>(community)</td>
<td>2000</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>Team Foundation Server</td>
<td>Microsoft</td>
<td>2012-2005[16]</td>
<td>Unknown</td>
<td>No</td>
</tr>
<tr>
<td>Visual Studio Team Services</td>
<td>Microsoft</td>
<td>2012[17]</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications...*
  + CIA risk assessment, + Risk-level acceptance,...

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models...*
  + Threat modeling, + Attack surface analysis,...

Develop (“make”) / Implement (“buy”)

- *Source code control system, code reviews, daily builds, automated CASE tools...*
  + Developer security training, + Static analysis, + Secure code repositories,...

Testing/Validation

- *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*
  + Dynamic analysis, + Fuzzing,...
Testing/validation

Issue Resolution Workflow

- Opening
- Resolving
- Closing

New issue is identified and documented
- Task Leader assigns
- Opened & Assigned
- Developer is finished with issue
- Developer takes possession
  - Resolution not satisfactory
  - Resolution verified
- Issue is reopened
- Issue is closed

issue finder
developer

Jira Issue Management System
Testing/validation

Issue Resolution Workflow

- Opening
- Resolving
- Closing

Graph:
- Open Issues
- Resolved Issues
- Closed Issues

Time ->
Figure 1. Magic Quadrant for Application Security Testing

Application security testing tool providers

Source: Gartner (March 2018)
Dynamic code testing result reports

- Applications should not be accepted until *all high and medium issues resolved!*
Dynamic code testing result reports

- Applications should not be accepted until **all high and medium issues resolved!**
SDLC and Security

Requirements analysis
– *Informational, functional, behavioral, and performance specifications*...
+ CIA risk assessment, + Risk-level acceptance,...

Design
– *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models*...
+ Threat modeling, + Attack surface analysis,...

Develop ("make") / Implement ("buy")
– *Source code control system, code reviews, daily builds, automated CASE tools*...
+ Developer security training, + Static analysis, + Secure code repositories,...

Testing/Validation
– *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*
+ Dynamic analysis, + Fuzzing,...

Release/Maintenance
– *Release testing*
+ Separation of duties, +Change management, +Operational practices...
Separation of Duties

Different environments (development, testing, and production) should be separated, without overlapping access to code, applications and systems.

*The access and ability of developers to modify application code make them the “most powerful” insider threats and vulnerabilities to information systems*

- Developers should not have access to modify code used in production
- Code should be tested, submitted to a library, and then sent to the production environment
Releases/Maintenance

• Commercial Off The Shelf (COTS) products and Open Source products should have their security patches
• Installation programs should be removed from production
• File and program settings and privileges should be reviewed
Operational concerns

- Commercial Off The Shelf (COTS) software sources of risk
- Open source libraries sources of risk
- Operational Practices
  - System Security Plan (SSP) updates
  - Contingency Plan (BCP/DRP) updates
  - Awareness and training updates
  - Documentation updates
Operational Practices

• Support training classes
• User administration and access privileges
• Backup and restoration
  – Data, applications, configurations, restart instructions and procedures
  – Performing backups: How often? In which ways?
  – Performing backups
  – Offsite storage
  – Testing restoration
• Ensure implementation of only approved and accredited systems
• Cryptography keys
  – Generation and Use
  – Protection and storage
• Audit logs
  – How collected?
  – Where stored?
  – How protected?
  – How analyzed?
Operational Assurance Activities

• Review
  – Interdependencies among applications and systems
  – Runtime operation
  – Technical controls

• Verify documentation
  – Of access permissions
  – Is current and accurate

• Verify proper deregistration
  – i.e. removal of users and privileges

• Is availability and distribution of output products secure?
• Are software & hardware licenses fulfilled and warranties in place?
Other topics: Disposal

- Storage and protection of cryptographic keys
- Legal requirements of records retention
- Archiving federal information
- Sanitize media
Test Taking Tip

Focus on addressing each question individually

• As you take the test, if you don't know an answer, don't obsess over it

• Answer the best way you can or skip over the question and come back to it after you've answered other questions
Quiz
Agenda

✓ Introduction
✓ Software development life cycle (SDLC)
✓ SDLC and security
✓ Test taking tip
✓ Quiz