Software Development Security domain refers to the controls that are included within systems and application software and the steps used in their development (e.g., SDLC).

Software refers to system software (operating systems) and application programs such as agents, applets, software, databases, data warehouses, and knowledge-based systems. These applications may be used in distributed or centralized environments.

The candidate should fully understand the security and controls of the system development process, system life cycle, application controls, change controls, data warehousing, data mining, knowledge-based systems, program interfaces, and concepts used to ensure data and application integrity, security, and availability.

Reference: CISSP CIB, January 2012 (4.17.14 Rev. 13)
Introduction

Current State of Insecurity in Federal Agencies


<table>
<thead>
<tr>
<th>% of System with a:</th>
<th>FY 2005</th>
<th>FY 2006</th>
<th>FY 2007</th>
<th>FY 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certification and Accreditation (C&amp;A)</td>
<td>85%</td>
<td>88%</td>
<td>92%</td>
<td>96%</td>
</tr>
<tr>
<td>Tested Contingency Plan</td>
<td>61%</td>
<td>77%</td>
<td>86%</td>
<td>92%</td>
</tr>
<tr>
<td>Tested Security Controls</td>
<td>72%</td>
<td>88%</td>
<td>95%</td>
<td>93%</td>
</tr>
<tr>
<td>Total Systems Reported</td>
<td>10,289</td>
<td>10,595</td>
<td>10,304</td>
<td>10,679</td>
</tr>
</tbody>
</table>

• Yet, “20 of 24 major agencies indicated that inadequate information security controls were either a significant deficiency or a material weakness.”*

* Source: GAO-08-496, Information Security—Although Progress Reported, Federal Agencies Need to Resolve Significant Deficiencies, February 14, 2008
Introduction

Current State of Insecurity in Federal Agencies

- # of security incidents keeps growing*…

Security Incidents - FY'05 to FY'11

<table>
<thead>
<tr>
<th>Security Incident Type</th>
<th>FY'05</th>
<th>FY'06</th>
<th>FY'07</th>
<th>FY'08</th>
<th>FY'09</th>
<th>FY'10</th>
<th>FY'11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Unauthorized Access</td>
<td>304</td>
<td>706</td>
<td>2,321</td>
<td>3,214</td>
<td>4,848</td>
<td>5,782</td>
<td>6,959</td>
</tr>
<tr>
<td>2. Denial of Service</td>
<td>31</td>
<td>37</td>
<td>36</td>
<td>26</td>
<td>48</td>
<td>28</td>
<td>33</td>
</tr>
<tr>
<td>3. Malicious Code</td>
<td>1,806</td>
<td>1,465</td>
<td>1,607</td>
<td>2,274</td>
<td>6,977</td>
<td>12,926</td>
<td>11,556</td>
</tr>
<tr>
<td>4. Improper Usage</td>
<td>370</td>
<td>638</td>
<td>3,305</td>
<td>3,762</td>
<td>6,148</td>
<td>7,334</td>
<td>8,372</td>
</tr>
<tr>
<td>5. Scans/Probes/Attempted Access</td>
<td>976</td>
<td>1,388</td>
<td>1,661</td>
<td>1,272</td>
<td>1,152</td>
<td>69,832</td>
<td>66,057</td>
</tr>
<tr>
<td>6. Under Investigation</td>
<td>82</td>
<td>912</td>
<td>4,056</td>
<td>7,502</td>
<td>10,826</td>
<td>11,534</td>
<td>13,601</td>
</tr>
</tbody>
</table>

* Source: US-CERT

What happened here?
• The software flaw statistics are also trending upward…

• According to an analysis by Software Engineering Institute (SEI): “Most software security vulnerabilities arise from common causes; more than 90 percent are caused by known software defect types.” Where the top 10 causes account for about 75 percent of all vulnerabilities.

*Source: National Vulnerability Database (http://nvd.nist.gov)*
## 2011 CWE/SANS Top 25 Most Dangerous Programming Errors

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>93.8</td>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>2</td>
<td>83.3</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>3</td>
<td>79.0</td>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>4</td>
<td>77.7</td>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>5</td>
<td>76.9</td>
<td>CWE-306</td>
<td>Missing Authentication for Critical Function</td>
</tr>
<tr>
<td>6</td>
<td>76.8</td>
<td>CWE-862</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>7</td>
<td>75.0</td>
<td>CWE-798</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>8</td>
<td>75.0</td>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>9</td>
<td>74.0</td>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>10</td>
<td>73.8</td>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>11</td>
<td>73.1</td>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>12</td>
<td>70.1</td>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>13</td>
<td>69.3</td>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td>
</tr>
<tr>
<td>14</td>
<td>68.5</td>
<td>CWE-494</td>
<td>Download of Code Without Integrity Check</td>
</tr>
<tr>
<td>15</td>
<td>67.8</td>
<td>CWE-863</td>
<td>Incorrect Authorization</td>
</tr>
<tr>
<td>16</td>
<td>66.0</td>
<td>CWE-829</td>
<td>Inclusion of Functionality from Untrusted Control Sphere</td>
</tr>
<tr>
<td>17</td>
<td>65.5</td>
<td>CWE-732</td>
<td>Incorrect Permission Assignment for Critical Resource</td>
</tr>
<tr>
<td>18</td>
<td>64.6</td>
<td>CWE-676</td>
<td>Use of Potentially Dangerous Function</td>
</tr>
<tr>
<td>19</td>
<td>64.1</td>
<td>CWE-327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
</tr>
<tr>
<td>20</td>
<td>62.4</td>
<td>CWE-131</td>
<td>Incorrect Calculation of Buffer Size</td>
</tr>
<tr>
<td>21</td>
<td>61.5</td>
<td>CWE-307</td>
<td>Improper Restriction of Excessive Authentication Attempts</td>
</tr>
<tr>
<td>22</td>
<td>61.1</td>
<td>CWE-601</td>
<td>URL Redirection to Untrusted Site ('Open Redirect')</td>
</tr>
<tr>
<td>23</td>
<td>61.0</td>
<td>CWE-134</td>
<td>Uncontrolled Format String</td>
</tr>
<tr>
<td>24</td>
<td>60.3</td>
<td>CWE-190</td>
<td>Integer Overflow or Wraparound</td>
</tr>
<tr>
<td>25</td>
<td>59.9</td>
<td>CWE-759</td>
<td>Use of a One-Way Hash without a Salt</td>
</tr>
</tbody>
</table>

Reference: http://cwe.mitre.org/top25/
## Introduction

Today’s problems are about same as yesterday’s

### Open Web Application Security Project (OWASP) Top 10

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Injection</td>
<td>A1 – Injection</td>
</tr>
<tr>
<td>A3</td>
<td>Broken Authentication and Session Management</td>
<td>A2 – Broken Authentication and Session Management</td>
</tr>
<tr>
<td>A2</td>
<td>Cross-Site Scripting (XSS)</td>
<td>A3 – Cross-Site Scripting (XSS)</td>
</tr>
<tr>
<td>A4</td>
<td>Insecure Direct Object References</td>
<td>A4 – Insecure Direct Object References</td>
</tr>
<tr>
<td>A6</td>
<td>Security Misconfiguration</td>
<td>A5 – Security Misconfiguration</td>
</tr>
<tr>
<td>A7</td>
<td>Insecure Cryptographic Storage – Merged with A9</td>
<td>A6 – Sensitive Data Exposure</td>
</tr>
<tr>
<td>A8</td>
<td>Failure to Restrict URL Access – Broadened into</td>
<td>A7 – Missing Function Level Access Control</td>
</tr>
<tr>
<td>A5</td>
<td>Cross-Site Request Forgery (CSRF)</td>
<td>A8 – Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>&lt;br&gt;buried in A6: Security Misconfiguration&gt;</td>
<td>A9 – Using Known Vulnerability Components</td>
<td></td>
</tr>
<tr>
<td>A10</td>
<td>Un-validated Redirects and Forwards</td>
<td>A10 – Un-validated Redirects and Forwards</td>
</tr>
<tr>
<td>A9</td>
<td>Insufficient Transport Layer Protection</td>
<td>Merged with 2010-A7 into new 2013-A6</td>
</tr>
</tbody>
</table>

**Source:** OWASP Top Ten Project (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)
Topics
Software Development Security Domain

Governance & Management
  • System Life Cycle and Security
  • Software Environment and Security Controls
  • Programming Languages
  • Database and DB Warehousing Vulnerabilities, Threats, and Protections
  • Software Vulnerabilities and Threats
Number of connections (or interfaces) = $n \times (n - 1) / 2$

“As project size increases, errors usually come more from requirements and design…” (Boehm 1981, Grady 1987, Jones 1998)
Governance & Management

Information Security Governance

- **Policy.** Management directives that establish expectations (goals & objectives), and assign roles & responsibilities.
- **Standards.** Functional specific mandatory activities, actions, and rules.
- **Procedure.** Step-by-step implementation instructions.
- **Baseline (or Process).** Mandatory description of how to implement security packages to ensure consist security posture.
- **Guidelines.** General statement, framework, or recommendations to augment baselines or procedures.
Governance & Management

Clinger-Cohen Act of 1996 (CCA)

• The Clinger-Cohen Act of 1996 (a.k.a. ITMRA) defined the Federal agencies and DoD’s acquisition, management, and usage of IT.

• Key Elements
  – Defines the roles & responsibilities of Federal agencies and their executives (i.e. directors and CIOs.)
  – Requires Federal agencies to implement performance and result-based management for capital planning and investment control (CPIC).
  – Defines the IT acquisition process.
  – Requires IT architecture be defined for all Federal agencies. (i.e. Federal Enterprise Architecture (FEA)).
In 1992, GAO reported: “Defense’s mission-critical systems continue to have significant software development problems. Numerous GAO reports and Defense studies have identified many problems, including a lack of management attention, ill-defined system requirements, and inadequate testing. The highly complex nature of mission-critical systems and millions of lines of software required to support them contribute to the continuation of serious software development problems.”

E.g., *
- Cheyenne Mountain Upgrade (CMU), etc.
- Strategic Defense Initiative (SDI)
- Patriot surface-to-air missile system (Patriot)
- Army Tactical Command and Control System (ATCCS)
- AN/BSY-2 combat system for SSN-21 Seawolf submarine (BSY-2)
- AN/FQ-93 computer for the North American Aerospace Defense Command
- C-17 transport aircraft
- F-14D Tomcat fighter aircraft, etc.

Reference:
* GAO/IMTEC-93-13, Defense Attempting to Address Major Software Challenges, December 24, 1992
Federal Enterprise Architecture (FEA) Framework

- Federal Enterprise Architecture Framework (FEAF) focuses on BUSINESS

• Control Objectives for Information and related Technology (COBIT) is an IT Governance Framework created by Information Systems Audit and Control Association (ISACA)

• COBIT controls can encompass:
  – Information security controls (e.g., NIST SP 800-53, CNSS 1253, ISO/IEC 27001:2005)
  – IT processes management frameworks (e.g., ITIL, CMMI, ISO/IEC 27000 IT Service Management, PMBOK)

• COBIT governance is composed of 5 focus areas:
  – Strategic alignment
  – Value delivery
  – Resource management
  – Risk management
  – Performance measurement

Reference: COBIT 4.1 (http://www.isaca.org/)
Augment IT Governance with Information Security

- Information security is an ubiquitous practice…

InfoSec Controls:
- Management
- Operational
- Technical

Interrelationship of COBIT Components…

Reference: COBIT 4.1 (http://www.isaca.org/)
Governance & Management

Life Cycle Stages in Defense Acquisition System


---

Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System

- Material Solution Analysis Phase
- Technology Development Phase
- Engineering & Manufacturing Development Phase
- Production & Deployment Phase
- Operations & Support Phase

Joint Capabilities Integration & Development System

Oversight & Review

Contracting

Major Products

Logistical Support

Defense Acquisition System (acquisitions)

Technical System Engineering and Integration Support

Acquisition Management

Planning, Programming, Budgeting & Execution Process (personnel-driven)

---

Governance & Management

Each Life Cycle Stage has Milestone & Review

<table>
<thead>
<tr>
<th>Defense Acquisition Life Cycle (DoD 5000)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User needs &amp; Technology Opportunities</td>
<td>Materiel Solution Analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISO/IEC 15288/IEEE 1220 Systems and Software Engineering Life Cycle</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Design</td>
<td>Preliminary Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Systems Engineering Life Cycle using Structured Analysis and Design Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept Development Stage</td>
<td>Engineering Development Stage</td>
</tr>
<tr>
<td>Needs Analysis</td>
<td>Concept Exploration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information Systems Security Engineering (ISSE) Life Cycle</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover Information Protection Needs</td>
<td>Define Requirements</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Development: Rational Unified Process</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Elaboration</td>
</tr>
<tr>
<td>Business Modeling Requirements</td>
<td>Analysis &amp; Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>McGraw’s Software Security Touch Points</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements and Use Cases</td>
<td>Architecture &amp; Design</td>
</tr>
</tbody>
</table>

Focus on software structural defects

Focus on software weaknesses
• By Development Stage, 85% of LCC has already been committed.*

• Ratio of structural/design defects (flaws) vs. implementation weaknesses (bugs) is **50:50**.

• If **structural/design flaws** have not been discovered, mitigating them will add **20 to 100 times** to the plan cost. (And up to 1000 x in Production/Test Stage.)*

• Running source code analysis tools doesn’t help, because they are mostly for finding implementation weaknesses.**

Reference:
Governance & Management

Capability Maturity Model (CMM) – History

In 1986, Software Engineering Institute (SEI) and MITRE began developing an assessment framework for measuring the maturity of an organization’s [system/] software engineering process.

- **Process capability** describes expected results.
- **Process performance** represents the actual results achieved.
- **Process maturity** is the degree which a process is explicitly defined, managed, measured, controlled, and effective.

Governance & Management

Software Capability Maturity Model (SW-CMM)

• **Level 1: Initial**
  – The software development process is characterized as ad-hoc. Success depends on individual effort and heroics.

• **Level 2: Repeatable**
  – Basic project management (PM) processes are established to track performance, cost, and schedule.

• **Level 3: Defined**
  – Tailored software engineering and development processes are documented and used across the organization.

• **Level 4: Managed**
  – Detailed measures of product and process improvement are quantitatively controlled.

• **Level 5: Optimizing**
  – Continuous process improvement is institutionalized.
• System Security Engineering – Capability Maturity Model (SSE-CMM)
SSE-CMM is composed of two domains:

- **Security Base Practice** (11 x Process Areas)
  - Administer Security Controls
  - Assess Impact
  - Assess Security Risk
  - Assess Threat
  - Assess Vulnerability
  - Build Assurance Argument
  - Coordinate Security
  - Monitor Security Posture
  - Provide Security Input
  - Specify Security Needs
  - Verify & Validate Security

- **Project & Organizational Base Practice** (11 x Process Areas)
  - Ensure Quality
  - Manage Configuration
  - Manage Project Risks
  - Monitor & Control Technical Effort
  - Plan Technical Effort
  - Define Organization’s SE Process
  - Improve Organization’s SE Process
  - Manage Product Line Evolution
  - Manage SE Support Environment
  - Provide Ongoing Skills & Knowledge
  - Coordinate with Suppliers
Meeting the **assurance requirements** is a part of “due diligence” processes.

- Example:
  SC-3: Security Function Isolation. The information system isolates security functions from non-security functions.

Meeting the **functional requirements** is a part of “due care” processes.

- Example:
  - VLAN technology shall be created to partition the network into multiple mission-specific security domains.
  - The integrity of the internetworking architecture shall be preserved by the access control list (ACL).
### Assurance Requirements – Federal Agencies

<table>
<thead>
<tr>
<th>CLASS</th>
<th>FAMILY</th>
<th>IDENTIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>Risk Assessment</td>
<td>RA</td>
</tr>
<tr>
<td></td>
<td>Planning</td>
<td>PL</td>
</tr>
<tr>
<td></td>
<td>System and Services Acquisition</td>
<td>SA</td>
</tr>
<tr>
<td></td>
<td>Certification, Accreditation, and Security Assessment</td>
<td>CA</td>
</tr>
<tr>
<td></td>
<td>Program Management</td>
<td>PM</td>
</tr>
<tr>
<td>Operational</td>
<td>Personnel Security</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>Physical and Environmental Protection</td>
<td>PE</td>
</tr>
<tr>
<td></td>
<td>Contingency Planning</td>
<td>CP</td>
</tr>
<tr>
<td></td>
<td>Configuration Management</td>
<td>CM</td>
</tr>
<tr>
<td></td>
<td>Maintenance</td>
<td>MA</td>
</tr>
<tr>
<td></td>
<td>System and Information Integrity</td>
<td>SI</td>
</tr>
<tr>
<td></td>
<td>Media Protection</td>
<td>MP</td>
</tr>
<tr>
<td></td>
<td>Incident Response</td>
<td>IR</td>
</tr>
<tr>
<td></td>
<td>Awareness and Training</td>
<td>AT</td>
</tr>
<tr>
<td>Technical</td>
<td>Identification and Authentication</td>
<td>IA</td>
</tr>
<tr>
<td></td>
<td>Access Control</td>
<td>AC</td>
</tr>
<tr>
<td></td>
<td>Audit and Accountability</td>
<td>AU</td>
</tr>
<tr>
<td></td>
<td>System and Communications Protection</td>
<td>SC</td>
</tr>
</tbody>
</table>
Governance & Management

Assurance Requirements – DoD

DoDI 8500.2, *Information Assurance (IA) Implementation*

- **Confidentiality Controls** + **Controls for Integrity & Availability** (i.e. Mission Assurance Category (MAC))

<table>
<thead>
<tr>
<th>CONFIDENTIALITY CONTROLS</th>
<th>INFORMATION CLASSIFICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4.A4 (High)</td>
<td>Classified Information</td>
</tr>
<tr>
<td>E4.A5 (Medium)</td>
<td>Sensitive Information</td>
</tr>
<tr>
<td>E4.A6 (Basic)</td>
<td>Public Information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBJECT AREA NAME</th>
<th>ABBREVIATION</th>
<th>NUMBER OF CONTROLS IN SUBJECT AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4.A1 (MAC I)</td>
<td>DC</td>
<td>31</td>
</tr>
<tr>
<td>E4.A2 (MAC II)</td>
<td>IA</td>
<td>9</td>
</tr>
<tr>
<td>E4.A3 (MAC III)</td>
<td>EC</td>
<td>48</td>
</tr>
<tr>
<td>Security Design &amp; Configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification &amp; Authentication</td>
<td>IA</td>
<td>9</td>
</tr>
<tr>
<td>Enclave &amp; Computing Environment</td>
<td>EC</td>
<td>48</td>
</tr>
<tr>
<td>Enclave Boundary Defense</td>
<td>EB</td>
<td>8</td>
</tr>
<tr>
<td>Physical &amp; Environmental</td>
<td>PE</td>
<td>27</td>
</tr>
<tr>
<td>Personnel</td>
<td>PR</td>
<td>7</td>
</tr>
<tr>
<td>Continuity</td>
<td>CO</td>
<td>24</td>
</tr>
<tr>
<td>Vulnerability &amp; Incident Management</td>
<td>VI</td>
<td>3</td>
</tr>
</tbody>
</table>
## Assurance Requirements – Industry


<table>
<thead>
<tr>
<th>CONTROL CATEGORY</th>
<th>SUB-CATEGORY OF CONTROLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Policy</td>
<td>Information security policy</td>
</tr>
<tr>
<td>Organization of Information Security</td>
<td>Internal organization; External parties</td>
</tr>
<tr>
<td>Asset Management</td>
<td>Responsibility for assets; Information classification</td>
</tr>
<tr>
<td>Human Resource Security</td>
<td>Prior to employment; During employment; Termination or change of employment</td>
</tr>
<tr>
<td>Physical and Environmental Security</td>
<td>Secure areas; Equipment security</td>
</tr>
<tr>
<td>Communications and Operations Management</td>
<td>Operational procedures and responsibilities; Third party service delivery management; System planning and acceptance; Protection against malicious and mobile code; Back-up; Network security management; Media handling; Exchange of information; Electronic commerce services; Monitoring</td>
</tr>
<tr>
<td>Access Control</td>
<td>Business requirement for access control; User access management; User responsibilities; Network access control; Operating system access control; Application and information access control; Mobile computing and teleworking</td>
</tr>
<tr>
<td>Information Systems Acquisition, Development, and Maintenance</td>
<td>Security requirements of information systems; Correct processing in applications; Cryptographic controls; Security of system files; Security in development and support processes; Technical vulnerability management</td>
</tr>
<tr>
<td>Information Security Incident Management</td>
<td>Reporting information security events and weaknesses; Management of information security incidents and improvements</td>
</tr>
<tr>
<td>Business Continuity Management</td>
<td>Information security aspects of business continuity management</td>
</tr>
<tr>
<td>Compliance</td>
<td>Compliance with legal requirements; Compliance with security policies and standards, and technical compliance; Information system audit considerations</td>
</tr>
</tbody>
</table>
### Governance & Management
### Assurance Requirements – Credit Card Payment Industry

**Payment Card Industry – Data Security Standard (PCI-DSS), Requirements and Security Assessment Procedures, Version 2.0, October 2010**

<table>
<thead>
<tr>
<th>Assessment Procedures</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build and Maintain a Secure Network</td>
<td>Req. 1: Install and maintain a firewall configuration to protect cardholder data.</td>
</tr>
<tr>
<td></td>
<td>Req. 2: Do not use vendor-supplied defaults for system passwords and other security parameters.</td>
</tr>
<tr>
<td>Protect Cardholder Data</td>
<td>Req. 3: Protect stored cardholder data.</td>
</tr>
<tr>
<td></td>
<td>Req. 4: Encrypt transmission of cardholder data across open, public networks.</td>
</tr>
<tr>
<td>Maintain a Vulnerability Management Program</td>
<td>Req. 5: Use and regularly update anti-virus software or programs.</td>
</tr>
<tr>
<td></td>
<td>Req. 6: Develop and maintain secure systems and applications.</td>
</tr>
<tr>
<td>Implement Strong Access Control Measures</td>
<td>Req. 7: Restrict access to cardholder data by business need to know.</td>
</tr>
<tr>
<td></td>
<td>Req. 8: Assign a unique ID to each person with computer access.</td>
</tr>
<tr>
<td></td>
<td>Req. 9: Restrict physical access to cardholder data.</td>
</tr>
<tr>
<td>Regular Monitor and Test Network</td>
<td>Req. 10: Track and monitor all access to network resources and cardholder data.</td>
</tr>
<tr>
<td></td>
<td>Req. 11: Regular test security systems and processes.</td>
</tr>
<tr>
<td>Maintain an Information Security Policy</td>
<td>Req. 12: Maintain a policy that addresses information security for all personnel.</td>
</tr>
</tbody>
</table>
Governance & Management

Assurance Requirements – Other PCI Security Standards

• Payment Application Data Security Standard (PA-DSS) Requirement and Security Assessment Procedure, Version 2.0, October 2010

• Payment Card Industry PIN Transaction Security (PCI PTS)

• Payment Card Industry Point-to-Point Encryption (PCI P2PE)
  – P2PE Hardware Solution Requirements and Testing Procedures, April 2012.

Software Development Security Domain

- Governance & Management
- System/Software Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats
System Development Life Cycle (SDLC) Models and Processes

- **Waterfall Development Models**

- **Iterative Development Models**
  - Boehm’s *Spiral Model*.
  - *Rapid Application Development (RAD) & Joint Application Development (JAD)*

- **SDLC Processes**
System/Software Development Life Cycle (SDLC)

Waterfall Development Models

- Classic Waterfall: DoD-STD-2167A
- Modified Waterfall: MIL-STD-498
Other SDLC Models – Modified Waterfall w/ Subprojects

System/Software Development Life Cycle (SDLC)

Boehm’s Spiral Model

Reference: http://csse.usc.edu/people/barry.html
System/Software Development Life Cycle (SDLC)

Rapid Application Development (RAD) Model

- Iterative, but spiral cycles are much smaller.
- Risk-based approach, but focus on “good enough” outcome.
- SDLC fundamentals still apply...
  - Requirements, configuration, and quality management, design process, coding, test & integration, technical and project reviews etc.

Reference:
- S. McConnel, Rapid Development: Taming Wild Software Schedules
- http://www.cs.bgsu.edu/maner/domains/RAD.htm
Evolutionary Prototyping Model

- The system concept is refined continuously...
  - The focus is on “good enough” concept, requirements, and prototype.
  - However, it is difficult to determine level of effort (LOE), cost, and schedule.

## System/Software Development Life Cycle (SDLC)

### The need for speed... Agile Development Approach

<table>
<thead>
<tr>
<th>Project Terms</th>
<th>Agile Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNS</td>
<td>Vision</td>
</tr>
<tr>
<td>CONOPS</td>
<td>User Stories</td>
</tr>
<tr>
<td>SDP</td>
<td>Release &amp; Iteration</td>
</tr>
<tr>
<td>PMR/MS Reviews</td>
<td>Retrospectives, Product Demo</td>
</tr>
</tbody>
</table>

### Diagram

```
PKG [SE Deliverables] Traceability
```

- **Mission Needs (Product Vision)**
  - **CONOPS (User Stories)**
    - **Operational Req's (Product Backlog)**
      - **Functional Req's (Sprint Backlogs)**
        - **Design Specs. (Design Patterns)**
          - **Functional Components**
            - **Subsystems**
              - **System**
                - **System in I&T/Operating Environment**
                  - **Field Test**
                    - **Validation**
                      - **Qualification Test (Demonstration)**
                        - **System Test (Demonstration)**
                          - **System Integration Test (Demonstration)**
                            - **Unit Test (Build/Test)**
                              - **Agile practices applied in all software development & test activities**
```

```
Appropriate level of System Architecture & Detailed Design
```
Agile SDLC Model – Scrum

- Scrum is an agile software development methodology and model that is both iterative and incremental.
- The concept derived from the development of commercial products, where:
  - **Product owner** provides the vision and roadmap;
  - **Scrum master** specifies activities and ensures deliverables meet the sprint and iteration goals;
  - **Team** executes the specified scrum activities.
- The process is executed in a series of “time-boxed” sprints and iterations, where:
  - A “**sprint**” is usually 2 to 4 weeks; and
  - The end-product is a “**iteration**”.

Reference:
Agile SDLC Model – Scrum

- The **product vision** is translated into a list of project requirements;
- This “list” is called the **product backlog**. It encompasses all the project requirements and work;
- The scrum master works with the product owner to plan and divide the product backlog into a series of **sprint backlog**.
- The **self-organized team** composed of domain and SMEs. The team is empowered to select, plan, and make decisions on its work task
- The daily stand-up team meeting is called the **daily-scrum**. It keeps the team members focused on their tasks. Both product owner and scrum master are required to participate.

Reference:
Are there other SDLC models?

DevOps*

- Idea observed from cloud computing...
- 2009, Flickr reported doing 10 deployments per day
- Amazon EC2 reported in May 2011:**
  - Mean time between deployments: 11.6 seconds
  - Maximum # of deployments in an hour: 1,079
  - Mean # of hosts can simultaneously receive a deployment: 10k
  - Maximum # of hosts can simultaneously receive a deployment: 30k

- [http://youtu.be/o7-luYS0iSE](http://youtu.be/o7-luYS0iSE) ***

Reference:

** Jon Jenkins, *Velocity Culture*, O'Reilly Velocity 2011, (http://www.youtube.com/watch?v=dxk8b9rSKOo)
Philosophy behind the Rugged DevOps

- Seamless integration of software development and IT operations
- Focus on the “big picture” rather than security controls
  - Standard configuration
  - Process discipline
  - Controlled access to production systems
- Results
  - 75% reduction in outages triggered by software deployment since 2006
  - 90% reduction in outage minutes triggered by software deployments
  - Instantaneous automated rollback
  - Reduction in complexity
- Back to our study...

Reference:
- Jon Jenkins, *Velocity Culture*, O’Reilly Velocity 2011, (http://www.youtube.com/watch?v=dxk8b9rSKOo)
System/Software Development Life Cycle (SDLC)

History of Systems/Software Engineering Process Standards

**Systems Engineering**
- MIL-STD 499 (1969)
- EIA/IS 632 (Interim) (1994)

**Software Engineering**
- DOD-STD 2167A (1988)
- DOD-STD 1703 (1987)
- MIL-STD 498 (1994)
- DOD-STD 7935A (1988)

<<Based on>>
<<Referenced in>>
System/Software Development Life Cycle (SDLC)

Software & System Engineering Management Processes

• There are more and more “software-intensive” systems…
  – Systems are getting more complex. Hardware problems are often addressed through software;
  – Operating environments are stochastic. Software are more flexible than hardware.

• As SDLC models evolves, management processes are evolving too…
  – DoD-STD-2167A: Waterfall SDLC + SE Process
  – MIL-STD-498: Modified Waterfall SDLC + SE Process
  – IEEE 1220: System Engineering Process
Everything must be traceable

- **Verification**: “The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.”
- **Validation**: “Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.”

**ISO/IEC 15288* encompasses:**
- Systems/software engineering processes (Technical Processes)
- Project management processes
- Project support infrastructure (Organizational Project-Enabling Processes)
- Contract/business management processes (Agreement Processes)

*Note: ISO/IEC 15288 is identical to IEEE Std 15288*
System/Software Development Life Cycle (SDLC)

<table>
<thead>
<tr>
<th>System Context Processes</th>
<th>Software Specific Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agreement Processes</strong></td>
<td><strong>SW Implementation Processes</strong></td>
</tr>
<tr>
<td>Acquisition Process</td>
<td>Software Implementation Process</td>
</tr>
<tr>
<td>Supply Process</td>
<td>Software Requirements Analysis Process</td>
</tr>
<tr>
<td><strong>Project Processes</strong></td>
<td>Software Architectural Design Process</td>
</tr>
<tr>
<td>Project Planning Process</td>
<td>Software Detailed Design Process</td>
</tr>
<tr>
<td>Project Assessment and Control Process</td>
<td>Software Construction Process</td>
</tr>
<tr>
<td>Decision Management Process</td>
<td>Software Integration Process</td>
</tr>
<tr>
<td>Risk Management Process</td>
<td>Software Qualification Testing Process</td>
</tr>
<tr>
<td>Configuration Management Process</td>
<td>Validation Process</td>
</tr>
<tr>
<td>Information Management Process</td>
<td>Validation Process</td>
</tr>
<tr>
<td>Management Process</td>
<td><strong>SW Support Processes</strong></td>
</tr>
<tr>
<td></td>
<td>Software Documentation Process</td>
</tr>
<tr>
<td><strong>Organizational Project-Enabling Processes</strong></td>
<td>Software Configuration Management Process</td>
</tr>
<tr>
<td>Life Cycle Model Management Process</td>
<td>Software Quality Assurance Process</td>
</tr>
<tr>
<td>Infrastructure Management Process</td>
<td>Software Verification Process</td>
</tr>
<tr>
<td>Project Portfolio Management Process</td>
<td>Software Validation Process</td>
</tr>
<tr>
<td>Quality Management Process</td>
<td>Software Audit Process</td>
</tr>
<tr>
<td></td>
<td>Software Problem Resolution Process</td>
</tr>
<tr>
<td><strong>Technical Processes</strong></td>
<td>Software Reuse Processes</td>
</tr>
<tr>
<td>Stakeholder Requirements Definition Process</td>
<td>Domain Engineering Process</td>
</tr>
<tr>
<td>Requirements Analysis Process</td>
<td>Reuse Program Management Process</td>
</tr>
<tr>
<td>Architecture Design Process</td>
<td>Software Development Life Cycle Processes</td>
</tr>
<tr>
<td>Implementation Process</td>
<td>Validation Process</td>
</tr>
<tr>
<td>Integration Process</td>
<td><strong>Technical Processes</strong></td>
</tr>
<tr>
<td>Verification Process</td>
<td>Software Implementation Process</td>
</tr>
<tr>
<td>Transition Process</td>
<td>Software Requirements Analysis Process</td>
</tr>
<tr>
<td>Validation Process</td>
<td>Software Architectural Design Process</td>
</tr>
<tr>
<td>Operation Process</td>
<td>Software Detailed Design Process</td>
</tr>
<tr>
<td>Maintenance Process</td>
<td>Software Construction Process</td>
</tr>
<tr>
<td>Disposal Process</td>
<td>Software Integration Process</td>
</tr>
<tr>
<td></td>
<td>Software Qualification Testing Process</td>
</tr>
</tbody>
</table>

* Note: ISO/IEC 12207 is identical to IEEE Std 12207

IEEE 1220: System Life Cycle (SLC)

- Concept Stage
- Development Stage
- Production Stage
- Support Stage
- Disposal Stage

- System Definition
- Preliminary Design
- Detailed Design
- Fabrication Assembly, Integration & Test (FAIT)
IEEE 1220 defined System Engineering Process (SEP) within System Life Cycle (SLC)

IEEE 1220: System Life Cycle (SLC)

Reference: IEEE STD 1220: Standard for Application and Management of the Systems Engineering Process
### System/Software Development Life Cycle (SDLC)

#### Introducing Security into SDLC

<table>
<thead>
<tr>
<th>Defense Acquisition Life Cycle (DoD 5000)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User needs &amp; Technology Opportunities</strong></td>
</tr>
<tr>
<td>Conceptual Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISO/IEC 15288 Systems and Software Engineering Life Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conceptual Design</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Systems Engineering Life Cycle using Structured Analysis and Design Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Needs Analysis</strong></td>
</tr>
</tbody>
</table>

### Information Systems Security Engineering (ISSE) Life Cycle

<table>
<thead>
<tr>
<th><strong>Discover Information Protection Needs</strong></th>
<th><strong>Define Requirements</strong></th>
<th><strong>Design System Architecture</strong></th>
<th><strong>Develop Detailed System Design &amp; Security Controls</strong></th>
<th><strong>Implement System &amp; Security Controls</strong></th>
<th><strong>Continuous Monitoring</strong></th>
</tr>
</thead>
</table>

### Software Development: Rational Unified Process

<table>
<thead>
<tr>
<th><strong>Inception</strong></th>
<th><strong>Elaboration</strong></th>
<th><strong>Construction</strong></th>
<th><strong>Transition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Modeling</strong></td>
<td><strong>Requirements</strong></td>
<td><strong>Analysis &amp; Design</strong></td>
<td><strong>Implementation</strong></td>
</tr>
</tbody>
</table>

### McGraw’s Software Security Touch Points

<table>
<thead>
<tr>
<th><strong>Requirements and Use Cases</strong></th>
<th><strong>Architecture &amp; Design</strong></th>
<th><strong>Test Plans</strong></th>
<th><strong>Code</strong></th>
<th><strong>Test &amp; Test Results</strong></th>
<th><strong>Feedback From The Fields</strong></th>
</tr>
</thead>
</table>

Focus on software structural defects
Focus on software weaknesses
System/Software Development Life Cycle (SDLC)

Security Considerations in SDLC

1. Initiation Phase (IEEE 1220: Concept Stage)
   - Survey & understand the policies, standards, and guidelines.
   - Identify information assets (tangible & intangible).
   - Define information classification & protection level (security categorization).
   - Define rules of behavior & security CONOPs.
   - Conduct preliminary risk assessment.

2. Acquisition / Development Phase (IEEE 1220: Development Stage)
   - Conduct risk assessment.
   - Define security requirements and select security controls (categories & types).
   - Perform cost/benefit analysis (CBA).
   - Security planning (based on risks & CBA).
   - Practice Information Systems Security Engineering (ISSE) Process to develop security controls.
   - Develop security test & evaluation (ST&E) plan for verification & validation of security controls.

3. Implementation Phase (IEEE 1220: Production Stage)
   - Implement security controls in accordance with system security plan (SSP).
   - Perform Security Certification & Accreditation of target system.

4. Operations / Maintenance Phase (IEEE 1220: Support Stage)
   - Configuration management & perform change control.
   - Continuous monitoring – Perform periodic security assessment.

5. Disposition Phase (IEEE 1220: Disposal Stage)
   - Preserve information. archive and store electronic information
   - Sanitize media. Ensure the electronic data stored in the disposed media are deleted, erased, and over-written
   - Dispose hardware. Ensure all electronic data resident in hardware are deleted, erased, and over-written (i.e. EPROM, BIOS, etc.)

System/Software Development Life Cycle (SDLC)

Information Systems Security Engineering (ISSE) Process

• Phase 1: Discover Information Protection Needs
  – Ascertain the system purpose.
  – Identify information asset needs protection.
• Phase 2: Define System Security Requirements
  – Define requirements based on the protection needs.
• Phase 3: Design System Security Architecture
  – Design system architecture to meet on security requirements.
• Phase 4: Develop Detailed Security Design
  – Based on security architecture, design security functions and features for the system.
• Phase 5: Implement System Security
  – Implement designed security functions and features into the system.
• Phase 6: Assess Security Effectiveness
  – Assess effectiveness of ISSE activities.

Reference: Information Assurance Technical Framework (IATF) Rel. 3.1
# System/Software Development Life Cycle (SDLC)

## Security starts at the beginning...

<table>
<thead>
<tr>
<th>IEEE 1220</th>
<th>DoD Acquisition SDLC</th>
<th>Key System Engineering Tasks</th>
<th>Key Security Engineering Tasks*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Concept Stage</strong></td>
<td><strong>User Needs &amp; Technology Opportunities</strong></td>
<td>Task 1: Discover Mission/Business Needs</td>
<td>Task 1: Discover Information Protection Needs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Understand customer’s mission/business goals (i.e., initial capability, project risk assessment)</td>
<td>• Understand customer’s information protection needs (i.e., infosec. risk assessment)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Understand system concept of operations (CONOPS)</td>
<td>• Understand operating environment (i.e., sensitivity of information assets, mode of operations)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Create high-level entity-data relations model (i.e., system context diagram)</td>
<td>• Create information management model (IMM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Define engineering project strategy and integrate into the overall project strategy</td>
<td>• Define information protection policy (IPP) and integrate into the project strategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Create system engineering management plan (SEMP)</td>
<td>• Create system security plan (SSP) and integrate into SEMP</td>
</tr>
<tr>
<td><strong>Milestone A</strong></td>
<td>Task 6: Assess project performance in meeting mission/business needs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Reference: Information Assurance Technical Framework (IATF), Release 3.1

## Key Deliverables

- Mission Needs Statement / Project Goal(s) and Objectives
- System Capabilities
- Preliminary CONOPS
- Preliminary System Context Descriptions
- Project Risk Assessment
- Draft System Engineering Management Plan (SEMP)
<table>
<thead>
<tr>
<th>Development Stage</th>
<th>Technology Development</th>
<th>Key System Engineering Tasks</th>
<th>Key Security Engineering Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestone B</td>
<td>Task 2: Define System Requirements</td>
<td>• Refine system context (e.g., functional components)</td>
<td>Task 2: Define Security Requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Define system requirements (e.g., functional, performance, operational, support, etc.)</td>
<td>• Select assurance requirements and define security functional requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Refine CONOPS</td>
<td>• Refine IMM and SSP</td>
</tr>
<tr>
<td></td>
<td>Task 3: Design System Architecture</td>
<td>• Determine &amp; select architecture framework</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task 4: Develop Detailed System Design (Logical &amp; Physical)</td>
<td>• Design system architecture and allocate system requirements to subsystems and components (i.e., RTM)</td>
<td>• Allocate system security requirements to subsystems and service components (i.e., RTM)</td>
</tr>
<tr>
<td></td>
<td>Task 6: Assess project performance in meeting mission/business needs</td>
<td>• Analyze gaps (i.e., risk assessment)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Refine entity-data relations model (i.e., UML diagrams, data-flow, network, etc.)</td>
<td>• Refine IMM, embed security controls into system design products (i.e., UML, data-flow, network, etc.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Perform system synthesis analysis to assure system integration (i.e., system design, system architecture, system requirements, and project mission/business needs)</td>
<td></td>
</tr>
</tbody>
</table>

- **Key Deliverables**
  - System Requirements
  - Functional Definitions (+ allocation of system requirements)
  - System Architecture (Contextual + Logical)
  - Detailed System Design (Logical + Physical)
  - Requirements Traceability Matrix (RTM)
### Task 5: Implement System Design
- Procure system components / construct system
- Code/ customize/ configure system functional components
- Conduct code inspection/ walk-through/ unit test
- Perform system integration
- Conduct system test

### Task 5: Implement Security Controls
- Conduct security test & evaluation (ST&E)

### Task 6: Assess project performance in meeting mission/business needs
- Generate system operations procedure (SOP) and users guide/ manual
- Conduct system readiness review
- Obtain system certification
- Deploy system
- Assess security effectiveness
- Obtain approval to operate (ATO)

### Key Deliverables
- Implement detailed system design
- Perform test & evaluations (unit, system, security tests)
- Test reports
- Standard Operating Procedure (SOP) + User Manuals
- Deploy system
- Conduct acceptance tests
System/Software Development Life Cycle (SDLC)

Rational Unified Process (RUP)

Disciplines

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Mgmt
- Project Management
- Environment

Phases

- Inception
- Elaboration
- Construction
- Transition

Iterations

- Initial
- E1
- E2
- C1
- C2
- CN
- T1
- T2

**System/Software Development Life Cycle (SDLC)**

**Rational Unified Process (RUP)**

<table>
<thead>
<tr>
<th>Software Development: Rational Unified Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
</tr>
<tr>
<td>Business Modeling</td>
</tr>
</tbody>
</table>

**McGraw’s Software Security Touch Points**

| Requirements and Use Cases | Architecture & Design | Test Plans | Code | Test & Test Results | Feedback From The Fields |

- **Use cases drives requirements** *(Business Needs/Concept Exploration)*
  - System, software, and security engineers create operational use cases (e.g., operational, functions, threat, risks models)
  - Use cases drives operational requirements

- **System design drives design specifications** *(Concept Definition/Detailed Design)*
  - Operational requirements are decomposed into system functions and functional requirements
  - Architecture organizes system functions allocation of functional requirements
  - Architecture is further decomposed into detailed system design
  - Detailed system design is explained in design specifications

- **Design specifications drives programming of software codes** *(Implementation/Coding/Integration/Testing)*
  - Software components integrated into functional components/subsystems (Unit Testing)
  - Functional subsystems integrated into system (/systems) (System Testing)
  - System perform functions that meets the operational needs (Acceptance Testing)

**Deployment/transition into operations**
System/Software Development Life Cycle (SDLC)

Integrated System/Security Engineering in RAD

Major Activities
- Concurrent risk and opportunity-driven growth of system understanding and definition
  - Initial scoping
  - Concept definition
  - Investment analysis
- System life cycle architecture and CONOPS
  - Build to increment plans and specifications
- Develop Increment III prototype
  - Exercise Increment III prototype
  - Rebaseline system features & capabilities
- Develop Increment IV prototype
  - Exercise Increment IV prototype
  - Rebaseline system features & capabilities (if necessary)
- Develop Increment V prototype
  - Exercise Increment V prototype
  - Transition into operations
  - Plan for future release (if necessary)

Evaluation of evidence of feasibility to proceed
- OTBR Package
  - Draft PIP
- Updated PIP
  - CONOPS
  - Conceptual Architecture
  - System Reqs. & Functional Specs.
- Prototype
  - Updated PIP
  - Updated CONOPS
  - System Architecture
  - Updated System Reqs. & Functional Specs.
- System Reqs. & Functional Specs.

Stakeholder review & commitment
- High, but addressable
- Acceptable
- Too high, unaddressable
- Negligible

Risk?
Adjust scope, priorities, or discontinue

From preceding phase
- Objectives
  - System Requirements
- Requirements
  - System Architecture
- Functions
  - System Design
- System model
  - Prototype
To next phase

1. Requirements analysis
2. Functional definition
3. Physical definition
4. Design validation
Questions:

• What are the relationships between SDLC models and SSE-CMM models?
  – SDLC describes… to a system acquisition project
  – SSE-CMM describes…

• What are the relationships between security controls models (NIST SP800-53, DoDI 8500.2, ISO/IEC 27001, etc.) and CMM/SSE-CMM models?
  – Security assurance requirements provide measurement of…
  – CMM utilizes the measurement metrics from security control models to measure…
Answers:

• What are the relationships between SDLC models and SSE-CMM models?
  – SDLC describes the key engineering process to a system acquisition project
  – SSE-CMM describes the key security and management processes to a security engineering practice

• What are the relationships between security controls models (NIST SP800-53, DoDI 8500.2, ISO/IEC 27001, etc.) and CMM/SSE-CMM models?
  – Security assurance requirements provide measurements of management, operational, and technical controls
  – CMM utilizes the measurement metrics from security control models to measure practice maturity
Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats
Software Environment and Security Controls

Review of Computer Operations Architecture Model

• **Reference monitor** is a conceptual abstraction of a “machine”, system, or software that mediates access of objects by subjects.

• **Trusted computing base** is a system of security controls that meets the confidentiality and integrity security objectives.

• **Secure kernel** is a part of the trusted computing base that implements reference monitor concept.

Reference Monitor

- **Reference monitor** is performed by a **reference validation mechanism**.
- **Reference validation mechanism** is a system composed of hardware and software.
- **Operating condition principles:**
  - The reference validation mechanism must be **tamper proof**.
  - The reference validation mechanism must **always be invoked**.
  - The reference validation mechanism must be small enough to be **subject to analysis and tests** to assure that it is correct.
- **OS shall be evaluated at TCSEC B2** (i.e. structured protection) and above.
The Trusted Computing Base (TCB) is the totality of protection mechanisms within a computing system – hardware, firmware, software, processes, transports.

The TCB maintains the confidentiality and integrity of each domain and monitors four basic functions:
- Process activation
- Execution domain switching
- Memory protection
- Input/output operation

Secure Kernel

- **Secure kernel** is an implementation of a reference monitoring mechanism responsible for enforcing security policy.

- It meets the following three (3) conditions:
  - **Completeness**. All accesses to information must go through the kernel.
  - **Isolation**. The kernel itself must be protected from any type of unauthorized access.
  - **Verifiability**. The kernel must be proven to meet design specifications.

In the kernel model, the inside layer controls basic OS services, such as:
- memory management,
- security,
- I/O,
- request management, etc.

User applications, environment subsystems, and subsystem DLLs exist on the outer layers.
Processor Privilege States

- **Processor privilege states** protect the **processor** and the **activities** that it performs.
- **Privileged levels** are called **rings**.

For example: Intel x86 has 4 privilege ring levels
- **Ring 0** contains **kernel** functions of the OS.
- **Ring 1** contains the **OS**.
- **Ring 2** contains the **OS utilities**.
- **Ring 3** contains the **applications**.

![Diagram showing rings 0 to 3 with labels for OS Kernel and Applications]
Software Environment and Security Controls

Example of Processor Privilege States

VMware ESX

- Hypervisor operates at **Ring 0**
- Guest OS kernel and OS now moved to **Ring 1**
- OS utilities in **Ring 2**
- Application in **Ring 3**

Same principles, but different technology thus different attacks

- Reference monitoring principles is consistent even with virtualization: Violation of privilege
  - Hypervisor vulnerabilities. Attack of kernel (Ring 0)
  - Hypervisor escape vulnerabilities. Violation of isolation of guest VMs (Ring 0)
  - Administrative VM vulnerabilities
    - Management server vulnerabilities. Exploitation of virtualized system configuration. (Ring 0)
    - Management console vulnerabilities. Attacks of privileged state (Entire TCB)
  - Guest VM vulnerabilities. Exploitation of OS vulnerabilities, but can potentially provide an attack vector to administrative VM, hypervisor, then other guest VMs (Ring 3/Ring 2 \(\rightarrow\) Ring 1 \(\rightarrow\) Ring 0)

Reference: Virtual Reality (http://blogs.vmware.com/virtualreality/2008/06/)
Example of Processor Privilege States – Many-to-Many

- VMware vSphere further abstracts the hardware layer
  - Virtual Machine File System (VMFS) for abstraction of data storage
  - vNetwork Distributed Switch (vDS) for abstraction of network layer
  - vMotion for distribution of processing power and high availability

Software Environment and Security Controls

More complexity, more attack surfaces - Examples

- **Hypervisor vulnerability:**
  - CVE-2010-2070: Xen IA-64 architecture, allows local user to modify processor status register that can cause DoS. (CVSS: 4.9 [Medium])

- **Hypervisor escape vulnerability:**
  - CVE-2009-1244: VM display function in VMware allows guest OS user to execute arbitrary code in hypervisor. (CVSS: 6.8 [Medium])

- **Administrative VM vulnerabilities:**
  - CVE-2008-2097: Buffer overflow in VMware ESX management service that allows remote authenticated users to gain root privileges. (CVSS: 9.0 [High])
  - CVE-2008-4281: Directory traversal in VMware ESXi that allows VM administrators to gain elevated privileges. (CVSS: 9.3 [High])
  - CVE-2009-2277: Cross-site scripting (XSS) vulnerability in WebAccess in VMware VirtualCenter that allows remote attacker to inject arbitrary web script to steal “context data” such as authentication credentials (CVSS: 4.3 [Medium])

- **Guest VM vulnerabilities:**
  - CVE-2011-2145: VMware Host Guest File System (HGFS) allows Solaris or FreeBSD guest OS users to modify guest OS files. (CVSS: 6.3 [Medium])
  - CVE-2011-2217: ActiveX controls in Internet Explorer allows remote attacker to execute arbitrary code or corrupt memory in VMware Infrastructure. (CVSS: 9.3 [High])

Reference:
- NVD (http://web.nvd.nist.gov/view/vuln/search)
Security Controls for Software Environment

- For CISSP Exam, countermeasures are also called “security controls”…
  - Security Controls for Buffer Overflows
  - Memory Protection
  - Covert Channel Controls
  - Cryptography
  - Password Protection Techniques
  - Inadequate Granularity of Controls
  - Control and Separation of Environments
  - Time of Check/Time of Use (TOC/TOU)
  - Social Engineering
  - Backup Controls
  - Malicious Code/Malware Controls
  - Virus Protection Controls
  - Mobile Code Controls
  - Sandbox
  - Programming Language Support
  - Access Controls

Software Environment and Security Controls

Security Controls for Buffer Overflow

• One of the **oldest** and **most common** problems to software.

• A buffer overflow occurs when a program or process tries to **store more data in a buffer** (temporary data storage area) **than it was intended to hold**.

• Vulnerability is caused by lack of parameter checking or enforcement for **accuracy** and **consistency** by the software application or OS.

• Countermeasure:
  – Practice good SDLC process (**code inspection & walkthrough**).
  – Programmer implementing **parameter checks** and enforce data rules.
  – Apply **patches** for OS & applications.
  – If available, implement **hardware states and controls for memory protection**.
  – Buffer management for OS.
Software Environment and Security Controls

Memory Protection

- **Memory protection** is enforcement of access control and privilege level to prevent unauthorized access to OS memory.

- **Countermeasures:**
  - Ensure all system-wide data structures and memory pools used by *kernel-mode* system components can only be accessed while in kernel mode.
  - Separate software processes, protect private address space from other processes.
  - Hardware-controlled memory protection
  - Use **Access Control List (ACL)** to protect shared memory objects.
Covert Channel Controls*

- **Covert channel** is an un-controlled information flow (or unauthorized information transfer) through hidden communication path(s).
  - **Storage** channel
  - **Timing** channel

- **Countermeasure steps:**
  - **Identify** potential covert channel(s)
  - **Verify** and validate existence of covert channel(s)
  - **Close** the covert channel by install patch or packet-filtering security mechanism.

*Note: While the definition of covert channel may be old, it is considered as “fundamental” in CISSP CBK.

Reference: NCSC-TG-30, *A Guide To Understanding Covert Channel Analysis of Trusted System*
Covert Channel Controls*

- Countermeasure for covert channel:
  - Information Flow Model is a variation of access control matrix
  - Information Flow Model is based on Object Security Levels.
  - Object-to-object information flow is constrained in accordance with object’s security attributes.

<table>
<thead>
<tr>
<th>Object</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>N/A</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>X</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>N/A</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
</tbody>
</table>
Cryptography

- Cryptography provides **confidentiality, integrity, authentication**, and **non-repudiation** in information operations.
  - **Asymmetric Key Cryptography**
    - Because of slow cipher operation speed, it is mostly used for key management function.
  - **Symmetric Key Cryptography**
    - Because of speed, symmetric-key cryptosystems are used for crypto. operations. E.g. SSL/TLS at Transport-level (communication path), e-mail & SOAP messages at message-level.
  - **Hash Function**
    - Message Digest
    - Message Authentication Code (MAC)
    - Key-hashed MAC (HMAC)
  - **Digital Signature**
Software Environment and Security Controls

Security Controls: Password Protection Techniques

• **Password Structure**
  – Password length
  – Password complexity: a mix of upper/lowercase letters, numbers, special characters
  – Not using common words found in dictionary

• **Password Maintenance**
  Set password lifetime limits & policy…
  – Password change in <90> days
  – Password can not be reused within <10> password changes
  – <One> change to <every 24 hr.>
  – Password file must be encrypted and access controlled.
Granularity of Controls

- **Separation of duties** means that a process is designed so that separate steps must be performed by different people (i.e. force collusion)
  - Define elements of a process or work function.
  - Divide elements among different functions

- **Least privilege** is a policy that limits both the system’s user and processes to access only those resources necessary to perform assigned functions.
  - Limit users and system processes to access only resources necessary to perform assigned functions.

- **Separation of system environments.**
  - Development environment.
  - QA/test environment.
  - Production or operational environment.
Software Environment and Security Controls

Other Security Controls

• **Social Engineering**
  – Countermeasure: User **security awareness training**.

• **Backup, Malicious Code/Malware, Virus Protection Controls**
  – Countermeasures:
    • Install & use **anti-virus system**, **H-IDS**.
    • Enable **access control to critical system files**.
    • Tape backups, **access control of media**.
    • **Encrypt sensitive information** for confidentiality & integrity.

• **Mobile Code Controls**
  – Install **Sandbox** for access control of mobile codes.
  – Example: **Java “containers”** or Java Virtual Machine (**JVM**).
    • **Java applets** running in Web browser.
    • Applications using Java Remote Method Invocation (RMI) to run **Java Beans**.
Software Environment and Security Controls

Security Controls – Access Controls

- Discretionary access control (**DAC**)
  - **Information owner determines** who has **access** & what **privileges** they have.

- Mandatory access control (**MAC**)
  - **Information classification & system determine access.**
  - Access decision based on **privilege (clearance) of subject** & **sensitivity (classification) of object (file).**
  - Requires **labeling** (or data tag)

- Access Control/Capability Matrix
  - Implement through the use of **ACL**.

- View-based Access Control
  - Authorization of specific views by **tables, columns,** and **key sets**.
Questions:

• What are the three operating condition principles for a reference monitor?
  -
  -
  -

• What are the three operating conditions for a secure kernel?
  -
  -
  -
Answers:

• What are the three operating condition principles for a reference monitor?
  – **must be tamper proof**
  – **must always be invoked**
  – **subject to analysis and tests**

• What are the three operating conditions for a secure kernel?
  – Completeness (must always be invoked)
  – Isolation (must be tamper proof)
  – **Verifiability (each operations shall be subject to analysis and tests)**
Questions:

• What causes buffer overflow?

• Why a good information flow model is a good tool for supporting the identification of covert channel?
Answers:

• What causes buffer overflow?
  – When a program or process that lacks parameter enforcement control tries to store more data in a buffer than it was intended to hold

• Why a good information flow model is a good tool for supporting the identification of covert channel?
  – Information flow model is the system design baseline that illustrates the directional vectors of information flow between objects (e.g., programs or processes)
Questions:

• Program that allows the information owner to determine who has what type of access and privilege is an implementation of what type of access control?

• For mandatory access control (MAC), an access decision is based on privilege of ___ & sensitivity of ___?
Answers:

• Program that allows the information owner to determine who has what type of access and privilege is an implementation of what type of access control?
  – Discretionary access control (DAC)

• For mandatory access control (MAC), an access decision is based on privilege of ____ & sensitivity of ____?
  – Subject
  – Object
Topics

Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats
Programming Languages

• A set of instructions and rules that tell the computer what operations to perform.

• Languages have evolved in “generations”
  – 1st Generation: Machine language
  – 2nd Generation: Assembly language
  – 3rd Generation: High-level language
    • Ada, COBOL, BASIC, FORTRAN, Pascal, C, C+, C++, C#, Java
  – 4th Generation: Very high-level language
    • SQL, JavaScript, Perl, SGML (Standard General Markup Language): HTML, XML, SAML, XACML.
  – 5th Generation: Natural language
    • BPEL (Business Process Execution Language), BQEL (Business Query Language)
Programming Languages

• **Assembler** – program that translates an assembly language program into machine language.
  – **Assembly Language** → **Machine Language**.

• **Compiler** – translates a high-level language into machine language.
  – **High-level Language** (3rd Gen.) → **Machine Language**.

• **Interpreter** – instead of compiling a program at once, the interpreter translates it instruction-by-instruction. It has a fetch and execute cycle.
  – **Very high-level Language** (4th Gen.) → **Interpreter instruction** → **Machine Language**.
Object-Oriented Programming (OOP)

- OOP method that creates an object.
  - The object is a block of pre-assembled code that is a self-contained module.
  - Once written, object can be reused.
  - Objects are encapsulated, thus providing some security.
  - Objects have methods (code with programming interfaces) and attributes (data) encapsulated together.
Object-Oriented Programming (OOP) – Characteristics

- **Object** is an instance of the class.
- **Class** tell the system how to make objects.
- **Encapsulation** is the technique of keeping together data structures and methods (procedures) which act on them.
- **Method** is a procedure or routine associated with one or more classes.
- **Message**: objects perform work by sending messages to other objects.
- **Inheritance** is the ability to derive new classes from existing classes. A derived class (or subclass) inherits the instance variables and methods of the “base-class” (or superclass), and may add new instance variables and methods.
Programming Languages

Object-Oriented Programming (OOP) – Characteristics

- **Polymorphism** describes the process of using an object in different ways for different set of inputs.
- **Polyinstantiation** is creating a new version of an object by replacing variables with other values (or variables).
  - Also used to prevent inference attacks against databases because it allows different versions of the same information to exist at different classification levels.
- **Cohesion** is the ability of a module to execute one function with little interaction from other modules.
- **Coupling** is a measure of the interconnection among modules in an application.
Programming Languages

Distributed Object-Oriented Systems

• Common Object Request Broker Architecture (CORBA)
  – A standard that “wrap” data objects. The object request broker (ORB) component enables heterogeneous applications and computing environment to interoperate.

• Component Object Model (COM) & Distributed Component Object Model (DCOM)
  – COM and DCOM are Microsoft object-oriented system standards for interoperate in a heterogeneous applications within a homogeneous (Microsoft) computing environment. It uses Object Linking & Embedding (OLE) and ActiveX.

• Java
  – Java Platform Standard Edition (Java SE)
  – Java Platform Enterprise Edition (Java EE)
Common Object Request Broker Architecture (CORBA)

- A set of standards that address the need for interoperability between hardware and software.
  - Allows applications to communicate with one another regardless of their location.
  - The Object Request Broker (ORB) establishes a client/server relationship between objects.
  - The ORB enforces the system’s security policy.
• CORBA uses Interface Definition Language (IDL) to describe interface requirements.
• CORBA uses Internet Inter-ORB Protocol (IIOP) to communicate between Object Request Brokers (ORBs).

Source: http://www.omg.org/gettingstarted/corbafaq.htm
Component Object Models

- Component Object Model (COM) architecture
  - An open software architecture from DEC and Microsoft, allowing interoperation between ObjectBroker and OLE. Microsoft evolved COM into DCOM.

- Distributed Component Object Model (DCOM) architecture
  - An extension of COM to support objects distributed across a network.
Component Object Models

Programming Languages

Object Linking & Embedding (OLE)

- OLE allows applications to share functionality by live data exchange and embedded data.
  - **Embedding** – places data in a foreign program.
    For example: Embedding of a Visio diagram inside of a PowerPoint slide.
  
  - **Linking** – capability to call a program.
    For example: Double click on the embedded Visio diagram in a PowerPoint slide and invoke Visio application to edit the diagram.
ActiveX

- A loosely defined set of technologies developed by Microsoft. ActiveX is a set of technologies that enables interactive contents for web.

- Elements of ActiveX technologies:
  - **ActiveX Controls**: interactive objects in a web page that provides user interaction functions.
  - **ActiveX Documents**: enable user to view non-HTML documents (e.g. Word, Excel, or PPT)
  - **ActiveX Scripting Controls**: integrated controls for ActiveX controls and/or Java Applets from web browser or server.
  - **Java Virtual Machine (JVM)**: enables web browser (IE) to run Java applets and integrate with ActiveX controls.
  - **ActiveX Server Framework**: provide web server functions to support the above functions plus objects for database access and online transactions.
Java Platforms

- Java is designed as a standard application “platform” for computing in a networked heterogeneous environment (developed by Sun Microsystems.)
- Java is a high-level programming language. Java source code are compiled into bytecode, which can then be executed by a Java interpreter.
- Java has three platforms:
  - Java SE (Standard Edition)
  - Java EE (Enterprise Edition)
  - Java ME (Micro Edition)

Programming Languages

Java Platform Enterprise Edition

- Java Enterprise Edition (Java EE) uses Java SE as a foundation

- There are Containers are the runtime components for Java EE.
  - Applet
  - Application Client
  - Web
  - EJB

Questions:

• COBOL, FORTRAN, C, C+, C++, C# are what generation programming languages?

• JavaScript, Perl, SQL, SGML are what generation programming languages?

• What mechanism translates a high-level language (3rd Generation) into machine language?
Answers:

• COBOL, FORTRAN, C, C+, C++, C# are what generation programming languages?
  – 3\textsuperscript{rd} Generation

• JavaScript, Perl, SQL, SGML are what generation programming languages?
  – 4\textsuperscript{th} Generation

• What mechanism translates a high-level language (3\textsuperscript{rd} Generation) into machine language?
  – Compiler
Questions:

• In object-oriented programming (OOP), what tells the system how to make object(s)?

• In OOP, what is the technique that keeps the data structures and methods (procedures) together?

• In OOP, what is the term that describes the process of using an object in different ways for different set of inputs?
Questions:

• In object-oriented programming (OOP), what tells the system how to make object(s)?
  – **Class**

• In OOP, what is the technique that keeps the data structures and methods (procedures) together?
  – **Encapsulation**

• In OOP, what is the term that describes the process of using an object in different ways for different set of inputs?
  – **Polymorphism**
Topics

Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats
Database and DB Warehousing Vulnerabilities, Threats, and Protections

Database Management System (DBMS)

- Databases are developed to manage information from many sources in one location.
  - Eliminate the need for duplication of information in the system (thus preserves storage space).
  - Prevent inconsistency in data by making changes in one central location.

- DBMS consists of: hardware, software, and databases used to manage large sets of structured data (or information asset).
  - Enables Multiple Users and Applications to access, view, and modify data as Needed.
  - Can enforce control restrictions.
  - Provides data integrity and redundancy.
  - Established procedures for data manipulation.
Database and DB Warehousing Vulnerabilities, Threats, and Protections

**DBMS Models**

- **Hierarchical DBMS**
  - Stores information records (data) in a single table
  - Uses parent/child relationships
  - Limited to a single tree, no links between branches

- **Network DBMS**
  - Relationship of information records are of same type
  - All associations are direct connects, which forms a network

- **Relational DBMS**
  - Information records are structured in tables
  - Columns are the “attributes”, Rows are the “records”

- **Object-oriented DBMS & object relational DBMS**
  - Information records are objects
  - Relationships of objects are dynamic. The association can be made hierarchical, network, or relational
Relational DBMS (RDBMS)

- Information records (data) are structured in database tables.
  - Columns (attributes) represent the variables
  - Rows (records) contain the specific instance of information records

- Atomic relation = Every row/column position has always exactly one data value and never a set of values.

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Port of Entry (POE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456-123456</td>
<td>Smith</td>
<td>John</td>
<td>DCA</td>
</tr>
<tr>
<td>234567-123456</td>
<td>Rogers</td>
<td>Mike</td>
<td>LGA</td>
</tr>
<tr>
<td>345678-123456</td>
<td>Johnson</td>
<td>John</td>
<td>SFO</td>
</tr>
<tr>
<td>456789-123456</td>
<td>Smith</td>
<td>Jack</td>
<td>SAN</td>
</tr>
</tbody>
</table>
Data within the RDBMS...

- Unique ID is the **primary key**. It identifies each row (record or tuple).
- Tuple cannot have a null value in the primary key.
- The **primary key value guarantees that the tuple is unique**.
- **“Foreign key”** is an attribute or combination of attributes in another database table that **matches** the value of **“primary key”** in the first database table.
  - **Referential integrity rule**
    - For any foreign key value, the reference relation to another table must have a tuple with the same value of the other table’s primary key.
    - A null value in the foreign key field prevents a join.
### Relational DBMS (RDBMS) – Primary & Foreign Keys

#### Traveler Manifest Table

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Port of Entry (POE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456-123456</td>
<td>Smith</td>
<td>John</td>
<td>DCA</td>
</tr>
<tr>
<td>234567-123456</td>
<td>Rogers</td>
<td>Mike</td>
<td>LGA</td>
</tr>
<tr>
<td>345678-123456</td>
<td>Johnson</td>
<td>John</td>
<td>SFO</td>
</tr>
<tr>
<td>456789-123456</td>
<td>Smith</td>
<td>Jack</td>
<td>SAN</td>
</tr>
</tbody>
</table>

#### Baggage Manifest Table

<table>
<thead>
<tr>
<th>Unique Tag ID</th>
<th>Airline</th>
<th>Flight Number</th>
<th>Unique ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCA456-123456</td>
<td>AA</td>
<td>AA-456</td>
<td>123456-123456</td>
</tr>
<tr>
<td>LGA567-123456</td>
<td>JetBlue</td>
<td>JB-567</td>
<td>234567-123456</td>
</tr>
<tr>
<td>SFO678-123456</td>
<td>United</td>
<td>UA-678</td>
<td>345678-123456</td>
</tr>
<tr>
<td>SAN89-123456</td>
<td>NW</td>
<td>NW-89</td>
<td>456789-123456</td>
</tr>
</tbody>
</table>

**Primary Key**

- Unique ID in the Traveler Manifest Table

**Foreign Key**

- Unique Tag ID in the Baggage Manifest Table
Relational DBMS (RDBMS) – View & Schema

- **Data dictionary** – Central repository of data elements and their relationships.

- **Schema** – Holds data that describes a database.

- **View** – Virtual relation defined by the database to keep subjects from viewing certain data.
Relational DBMS (RDBMS) – Security Issues

- **Ensure integrity of input data** (check input values, prevent buffer overflow).

- **Access control** ensuring only authorized user are performing authorized activities (“need-to-know”, “least privilege”).

- **Preventing deadlock** (stalemate when 2 or more processes are each waiting for the other to do something before they can proceed).
• **Class** is a set of **objects** which shares a common structure and behavior. The relationship between classes can be hierarchical. (i.e. super-class, and subclass.)

• **Object** is a unique instance of a data structure defined according to the template provided by its class. Each object has its own values for the variables belonging to its class and can respond to the messages (methods) defined by its class.

• **Method** is a procedure or routine associated with one or more classes.
Database and DB Warehousing Vulnerabilities, Threats, and Protections

OODBMS & ORDBMS

• Object-oriented database (OODB) represents a “paradigm-shift” in the traditional database models (hierarchical, network, and relational).
  – Example of OODBMS: Versant.

• Object relations are build dynamically based on “business needs” instead of a series of fixed “business processes”.
  – Currently, the foundational DBMS engine for most of ORDBMS are still RDBMS. Object relations are build:
    • Presentation Layer: User/client level.
    • Business Logic Layer: Accepts commands from the presentation layer and send instructions to the data layer.
    • Data Layer: The database.
  – Example of ORDBMS: Oracle (8i, 9i, 10g), IBM DB2.
Data Warehousing and Mining

- **Data Warehousing**
  - Combines data from multiple databases or data sources into a large database called “data warehouse”.
  - Requires more stringent security because all data is in a central facility.

- **Data Mining**
  - A.k.a. Knowledge-discovery in databases (KDD).
  - Practice of automatically searching large stores of data for patterns.
  - Data mining tools are used to find associations and correlations to product Metadata and can show previously unseen relationships.
Database Controls

- **Granularity** - The degree to which access to objects can be restricted.

- Content dependant access control
  - **Permissions by View** combining specific tables, columns, and key sets.
    - Authorizations for specific views having specific attributes, and for actions to perform within those views.
    - DAC, by specific grant to user or group by owner.
    - MAC, by classification level.
  - **Cell Suppression**
    - A technique used to hide or not show specific cells that contain information that could be used in an inference attack.
Database Controls

- **Partitioning** – Involved dividing a database into different parts which makes it harder for an individual to find connecting pieces

- **Noise and perturbation** – A technique of inserting bogus information aimed at misdirecting or confusing an attacker

- **Concurrency** – allowing multiple users to access the data contained within a database at the same time.
  - Making sure the most up to date information is available
  - If concurrent access is not managed by the Database Management System (DBMS) so that simultaneous operations don't interfere with one another problems can occur when various transactions interleave, resulting in an inconsistent database.
Database Controls – Types of Integrity Service

- **Semantic integrity** – Ensures that *structural* and *semantic rules* are enforced. Types of rules include data types, logical values, uniqueness constraints, and operations that could adversely affect the database.

- **Entity integrity** – Ensures that *tuples are uniquely identified by primary key values*.

- **Referential integrity** – Ensures that all *foreign keys reference valid (and existing) primary keys*. The other word, *if a record does not include a primary key it cannot be referenced*. 
Database Controls – Configurable Controls for Integrity

- **Rollback** – is a statement that ends a current transaction and cancels all other changes
  - Occurs when some type of “glitch” is encountered during transaction
- **Commit** – terminates a transaction and executes all changes that were just made by a user.
  - If a user attempts a “commit” and it cannot be completed correctly…a “rollback” is executed to ensure integrity
- **Savepoint(s)** – are used to ensure that if a system failure occurs, or an error is detected, the database can return to a known good state prior to the problem
- **Checkpoint(s)** – (similar to Savepoints) when the database S/W fills to a certain amount of memory, a checkpoint is initiated, which saves the data from the memory segment to a temporary file.
Database Security Controls

• **Polyinstantiation**
  – Allows a relation to contain multiple rows with the same primary key
  – The multiple instances of Primary Keys are distinguished by their security levels
  – Used to prevent inference attacks by inserting “bogus” data at lower security levels

• **Granularity** – The degree to which access to objects can be restricted.
  – Granularity can be applied to both the actions allowable on objects, as well as to the users allowed to perform those actions on the object
Database Security Issues

- **Online Transaction Processing (OLTP)**
  - Usually used when multiple databases are clustered to provide fault tolerance and higher performance.
  - **Transaction logs** are used for synchronization of databases.
  - OLTP transactions occur in real time which usually updates more than one database…which introduces integrity threats. To counteract this **ACID** test should be implemented.
    - **Atomicity** – Divides transactions into units of work and ensures all modifications take effect or none do.
    - **Consistency** – A transaction must follow integrity policy for that specific database and ensure that all data is consistent in the different databases.
    - **Isolation** – Transactions execute in isolation until completed, without interacting with other transactions.
    - **Durability** – Once the transaction is verified as accurate on all systems, it is committed and the databases cannot be rolled back.
Database Threats

- **Aggregation**
  - The act of combining information from separate sources.
  - The combined information has a sensitivity level greater than any of the individual parts.

- **Inference**
  - A user deduces (infers or figures out) the full story from pieces learned through aggregation and other sources.
  - Differs from aggregation in that data not explicitly available is used during the act of deduction (inference or plain figuring it out).

- **Deadlocking**
  - Two processes have locks on separate objects and each process is trying to acquire a lock on the object the other process has.
Questions:

- What are the four types of database management system (DBMS) models?
- In RDBMS, what is the definition for atomic relation?
- In RDBMS, what is a primary key?
Answers:

• What are the four types of database management system (DBMS) models?
  – Hierarchical
  – Network
  – Relational
  – Object-oriented

• In RDBMS, what is the definition for atomic relation?
  – Every row/column position always contains exactly one data value

• In RDBMS, what is a primary key?
  – The attribute that uniquely identifies each record
Questions:

• For RDBMS, how is the relationship between database tables created?

• In an object-oriented relational database (ORDBMS), what are the three layers where the object relations are build?
Answers:

• For RDBMS, how is the relationship between database tables created?
  – When an attribute of a database table is also an attribute of another database table

• In an object-oriented relational database (ORDBMS), what are the three layers where the object relations are build?
  – Presentation Layer: User/client level
  – Business Logic Layer: Accepts commands from the presentation layer and send instructions to the data layer
  – Data Layer: The database
Questions:

• For granularity access control, what are the two content dependent access control implementations for a DBMS?
  -
  -

• For DBMS, what is the term used that describes multiple users accessing data contained within a database at the same time?
  -

• What is the act of combining information from different sources?
  -
Answers:

• For granularity access control, what are the two content dependent access control implementations for a DBMS?
  – Permissions by view
  – Cell suppression

• For DBMS, what is the term used that describes multiple users accessing data contained within a database at the same time?
  – Concurrency

• What is the act of combining information from different sources?
  – Aggregation
Software Development Security Domain

• Governance & Management
• System Life Cycle and Security
• Software Environment and Security Controls
• Programming Languages
• Database and DB Warehousing Vulnerabilities, Threats, and Protections

Software Vulnerabilities and Threats
Vulnerabilities & Threats

Relationship between Threat, Risk, and Countermeasure

- **Threat source.** Entity that may acts on a vulnerability.
- **Threat.** Any potential danger to information life cycle.
- **Vulnerability.** A system has weakness or flaw that may provide an opportunity to a threat source.
- **Risk.** The likelihood of a threat source take advantage of a vulnerability.
- **Exposure.** An instance of being compromised by Threat Source.
- **Countermeasure / safeguard.** An administrative, operational, or logical mitigation against potential risk(s).
Vulnerabilities & Threats

Structural Defects, Weaknesses, Bugs, and Vulnerabilities

- **Vulnerabilities** are weaknesses that allow attackers to compromise the security objectives of information and/or information systems.
- **Defects** can be design flaws and/or implementation weaknesses.
- **Bugs** are implementation-level weaknesses.

<table>
<thead>
<tr>
<th>Information Systems Security Engineering (ISSE) Life Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover Information Protection Needs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Development: Rational Unified Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
</tr>
<tr>
<td>Business Modeling</td>
</tr>
</tbody>
</table>

**McGraw’s Software Security Touch Points**

- Requirements and Use Cases
- Architecture & Design
- Test Plans
- Code
- Test & Test Results
- Feedback From The Fields

Focus on software structural defects (flaws)  Focus on software weaknesses (bugs)
Vulnerabilities & Threats

Common Weakness Enumeration (CWE)

- CWE is an online dictionary of software weaknesses.
### 2011 CWE/SANS Top 25 Most Dangerous Programming Errors

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>93.8</td>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>[2]</td>
<td>83.3</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>[3]</td>
<td>79.0</td>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>[4]</td>
<td>77.7</td>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>[6]</td>
<td>76.8</td>
<td>CWE-862</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>[7]</td>
<td>75.0</td>
<td>CWE-798</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>[8]</td>
<td>75.0</td>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>[9]</td>
<td>74.0</td>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>[10]</td>
<td>73.8</td>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>[11]</td>
<td>73.1</td>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>[12]</td>
<td>70.1</td>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>[13]</td>
<td>69.3</td>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td>
</tr>
<tr>
<td>[14]</td>
<td>68.5</td>
<td>CWE-494</td>
<td>Download of Code Without Integrity Check</td>
</tr>
<tr>
<td>[16]</td>
<td>66.0</td>
<td>CWE-829</td>
<td>Inclusion of Functionality from Untrusted Control Sphere</td>
</tr>
<tr>
<td>[17]</td>
<td>65.5</td>
<td>CWE-732</td>
<td>Incorrect Permission Assignment for Critical Resource</td>
</tr>
<tr>
<td>[18]</td>
<td>64.6</td>
<td>CWE-676</td>
<td>Use of Potentially Dangerous Function</td>
</tr>
<tr>
<td>[19]</td>
<td>64.1</td>
<td>CWE-327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
</tr>
<tr>
<td>[20]</td>
<td>62.4</td>
<td>CWE-131</td>
<td>Incorrect Calculation of Buffer Size</td>
</tr>
<tr>
<td>[21]</td>
<td>61.5</td>
<td>CWE-307</td>
<td>Improper Restriction of Excessive Authentication Attempts</td>
</tr>
<tr>
<td>[22]</td>
<td>61.1</td>
<td>CWE-601</td>
<td>URL Redirection to Untrusted Site ('Open Redirect')</td>
</tr>
<tr>
<td>[23]</td>
<td>61.0</td>
<td>CWE-134</td>
<td>Uncontrolled Format String</td>
</tr>
<tr>
<td>[24]</td>
<td>60.3</td>
<td>CWE-190</td>
<td>Integer Overflow or Wraparound</td>
</tr>
<tr>
<td>[25]</td>
<td>59.9</td>
<td>CWE-759</td>
<td>Use of a One-Way Hash without a Salt</td>
</tr>
</tbody>
</table>

Vulnerabilities & Threats

Categories of Software Weaknesses

- **Insecure interaction** between components
  - “Weaknesses related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threats, or systems.”

- **Risky resource management**
  - “Weaknesses related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.”

- **Porous defenses**
  - “Weaknesses related to defensive techniques that are often misused, abused, or just plain ignored.”

Reduce / Eliminate Software Vulnerabilities

• Addressing **structural/design flaws**
  – Understand the information protection needs
  – Develop use/abuse cases
  – Define system security requirements
  – Design system architecture
  – Develop detailed system design & security controls

• Addressing **software bugs (weaknesses)**
  – Develop detailed software design & specifications
  – Implement code reviews
    • Static code analyzers
  – Perform tests
    • Unit, subsystems, system, acceptance tests
    • Vulnerability scanners

---

**Information Systems Security Engineering (ISSE) Life Cycle**

<table>
<thead>
<tr>
<th>Discover Information Protection Needs</th>
<th>Define Requirements</th>
<th>Design System Architecture</th>
<th>Develop Detailed System Design &amp; Security Controls</th>
<th>Implement System &amp; Security Controls</th>
<th>Continuous Monitoring</th>
</tr>
</thead>
</table>

**McGraw’s Software Security Touch Points**

<table>
<thead>
<tr>
<th>Requirements and Use Cases</th>
<th>Architecture &amp; Design</th>
<th>Test Plans</th>
<th>Code</th>
<th>Test &amp; Test Results</th>
<th>Feedback From The Fields</th>
</tr>
</thead>
</table>

**Focus on software structural defects (flaws)**

**Focus on software weaknesses (bugs)**
Vulnerabilities & Threats

Threats to Software – Buffer Overflow ...(1/2)

• One of the **oldest** and **most common** problems to software.
  – Wagner et. al. estimated over 50% of all vulnerabilities are due to buffer overflow.*

• No. 3 in 2011 CWE/SANS Top 25.

• A buffer overflow occurs **when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.**

• In buffer overflow attacks, the extra data may contain codes designed to trigger specific actions, in effect sending new instructions to the attacked computer that could, for example, damage the user's files, change data, or disclose confidential information.

Reference:
**Vulnerabilities & Threats**

**Threats to Software – Buffer Overflow ...(2/2)**

Recommended countermeasure to prevent buffer overflow attacks:

- Patch, patch, and patch
- Always check for inputs. Enforce controls at the interfaces
- Ensure applications are not exposed to faulty components
- Use language and frameworks that are relatively “immune” to buffer overflows:

<table>
<thead>
<tr>
<th>Language/Environment</th>
<th>Compiled / Interpreted</th>
<th>Strongly Typed</th>
<th>Direct Memory Access</th>
<th>Safe/ Unsafe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java, JVM</td>
<td>Both</td>
<td>Yes</td>
<td>No</td>
<td>Safe</td>
</tr>
<tr>
<td>.NET</td>
<td>Both</td>
<td>Yes</td>
<td>No</td>
<td>Safe</td>
</tr>
<tr>
<td>Perl</td>
<td>Both</td>
<td>Yes</td>
<td>No</td>
<td>Safe</td>
</tr>
<tr>
<td>Python</td>
<td>Interpreted</td>
<td>Yes</td>
<td>No</td>
<td>Safe</td>
</tr>
<tr>
<td>Ruby</td>
<td>Interpreted</td>
<td>Yes</td>
<td>No</td>
<td>Safe</td>
</tr>
<tr>
<td>C/C++</td>
<td>Compiled</td>
<td>No</td>
<td>Yes</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Assembly</td>
<td>Compiled</td>
<td>No</td>
<td>Yes</td>
<td>Unsafe</td>
</tr>
<tr>
<td>COBOL</td>
<td>Compiled</td>
<td>Yes</td>
<td>No</td>
<td>Safe</td>
</tr>
</tbody>
</table>

Vulnerabilities & Threats

Threats to Software – Cross-site Scripting (XSS) ...(1/2)

- XSS is one of the most prevalent web application (web app) security flaw.
- No. 4 in 2011 CWE/SANS Top 25 and OWASP Top 10.
  - XSS occurs when a web app in web browser accepts “untrusted data” and sends it to a web app server without proper validation. Attackers can then execute scripts in a victim’s web browser to hijack user sessions, deface web sites, insert malicious content, redirect users, etc.
  - These “untrusted data” could be JavaScript, or other browser-executable RIA contents such as Active X, Flash, Silverlight, etc.

Vulnerabilities & Threats

Threats to Software – Cross-site Scripting (XSS) ...(2/2)

- Recommended countermeasures to prevent XSS attacks:
  - Never insert untrusted data except in allowed locations.
  - Use “escaping” (a.k.a. output encoding) technique.
  - Use an HTML policy engine to validate or clean user-driven HTML in an outbound way.
  - Prevent DOM-based XSS.
  - Use “HTTPOnly” cookie flag

Reference:
- XSS (Cross Site Scripting Prevention Cheat Sheet – OWASP (https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)) (5/14/12)
In 2011, SQL Injection is No.1 in both CWE/SANS Top 25 and OWASP Top 10.

SQL injection occurs when an application sends “untrusted data” to an interpreter as a part of command or query.
- These “untrusted data” can be in SQL queries, LDAP queries, Xpath queries, etc.

Attackers can:
- Alter the logic of SQL queries to bypass security (e.g., authentication, authorization, etc.) to gain unauthorized access to data (e.g., steal, corrupt, or change data.)
- Trick the interpreter to execute unintended commands

Threats to Software – SQL Injection ...(2/3)

Reference: Exploits of a Mom http://xkcd.com/327/ (5/14/12)
Vulnerabilities & Threats

Threats to Software – SQL Injection ...(3/3)

• Recommended countermeasures to prevent SQL injection attacks:
  – Use “prepared statements” (/ parameterized queries) such as:
    • Java EE – use PreparedStatement() with bind variables
    • .NET – use parameterized queries like SqlCommand() or OleDbCommand() with binding variables
    • PHP – use PDO with strongly typed parameterized queries (use binParam())
  – Use stored procedures
  – Escaping all “user supplied” inputs. Treat user inputs as untrusted data, don’t insert them directly as a part of a SQL query

OS Command Injection (a.k.a. Shall Injection) is #2 in 2011 Top 25 CWE.

- Attacker injects and execute unwanted system commands through vulnerable applications that lacks proper input data validation (e.g., forms, cookies, HTTP headers etc.)
- As with SQL Injection, it is a variant of Code Injection attack, except it utilizes applications running as “system” instead of “user”.

Recommended countermeasure:

- Validate inputs
- Use application provided API
- Run automated code analysis tools

Vulnerabilities & Threats

Use of Automated Analysis Tools

- For detection of structural flaws ("defects")
  - Tool integration frameworks (a.k.a. IDEs)
    - Software engineering management, architecture/design modeling (MBSE), requirements traceability, design patterns
  - Code quality review tools

- For detection of software weakness ("bugs")
  - Static code analysis tools
    - Source code security analyzers, byte code scanners, binary code scanners
  - Dynamic analysis tools
    - Web application vulnerability scanners, database vulnerability scanners
  - Network vulnerability scanners
  - SCAP-compatible security configuration scanners
Vulnerabilities & Threats

Malicious Code / Malware

• Malicious code / malware (MALicious softWARE)

• For CISSP, there are many types of “malware”:
  – Viruses
  – Worms
  – Trojan horses
  – Rootkits
  – Spyware
  – Some cookies…

Vulnerabilities & Threats

Malware as a Threat to Information Operations ...(1/3)

• Operations are getting better at addressing insider threats…

» VZ (Verizon)
» USSS (United States Secret Service)

• The fact is that most of threats are still from external threat agents.

Most of data breaches are from hacking and malware...

Majority of malware are installed remotely...

Advanced Persistent Threat (APT) is very real
- Malware is now a tool for hackers
- They are stealing data...

Malicious code / malware (MALicious softWARE)

- **Virus** – A program or piece of code that is loaded onto your computer without your knowledge and runs against your wishes. Viruses can also replicate themselves. A simple virus that can make a copy of itself over and over again is relatively easy to produce.

- **Polymorphic virus** – A virus that changes its virus signature (i.e., its binary pattern) every time it replicates and infects a new file in order to keep from being detected by an antivirus program.
Threats to Software – Malicious Code / Malware

- **Boot sector virus** – A boot sector virus is a common type of virus that replaces the boot sector with its own code. Since the boot sector executes every time a computer is started, this type of virus is extremely dangerous.

- **Macro virus** – A type of computer virus that is encoded as a macro embedded in a document. Many applications, such as Microsoft Word and Excel, support powerful macro languages. These applications allow you to embed a macro in a document, and have the macro execute each time the document is opened.

  - According to some estimates, 75% of all viruses today are macro viruses. Once a macro virus gets onto your machine, it can embed itself in all future documents you create with the application.
Vulnerability & Threats

Threats to Software – Malicious Code / Malware

- **Worm** – A program or algorithm that replicates itself over a computer network and usually performs malicious actions. Differ from viruses in that they are self contained and do not need a host application to reproduce.

- **Logic bomb** – Also called slag code, programming code (typically malicious) added to the software of an application or operating system that lies dormant until a predetermined period of time or event occurs, triggering the code into action.
Vulnerability & Threats

Threats to Software – Malicious Code / Malware

• **Trojan horse** – A destructive program that masquerades as a benign application. Unlike viruses, Trojan horses do not replicate themselves but they can be just as destructive. One of the most insidious types of Trojan horse is a program that claims to rid your computer of viruses but instead introduces viruses onto your computer.

• **Data diddler** – refers to the payload in a Trojan or virus that deliberately corrupts data, generally by small increments over time.

• **Hoax** – usually warnings about viruses that do not exist, generally carry a directive to the user to forward the warning to all addresses available to them.

• **Trapdoor/backdoor** – can also be called a maintenance hook; it’s a hidden mechanism that bypasses access control measures.
Validation Time... 😊

1. Classroom Exercise

2. Review Answers
## Classroom Exercise: Constructing a Security Engineering Project... (1/5)

<table>
<thead>
<tr>
<th>Systems Engineering (SE) Activities</th>
<th>Security Engineering (ISSE) Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discover Mission/Business Needs</strong></td>
<td><strong>Discover Information Protection Needs</strong></td>
</tr>
<tr>
<td>The SE helps the customer understand and document the information</td>
<td>The ISSE facilitates the system owners, architects, and engineers in assessing the information</td>
</tr>
<tr>
<td>management needs that support the business or mission. Statements</td>
<td>protection needs by performing risk assessment, capturing the information management model (IMM),</td>
</tr>
<tr>
<td>about information needs may be captured in an information management</td>
<td>defining the information protection policy (IPP) and compile them into a comprehensive information</td>
</tr>
<tr>
<td>model (IMM).</td>
<td>management plan (IMP).</td>
</tr>
<tr>
<td><strong>Define System Requirements</strong></td>
<td><strong>Define System Security Requirements</strong></td>
</tr>
<tr>
<td>The SE allocates identified needs to systems. A system context is</td>
<td>The ISSE allocates the information protection needs in accordance with the information management</td>
</tr>
<tr>
<td>developed to identify the system environment and to show the allocation</td>
<td>plan (IMP) that aligns with a preliminary system security concept of operations (CONOPS) and</td>
</tr>
<tr>
<td>of system functions to that environment. A preliminary system Concept</td>
<td>generates a set of baseline security requirements in accordance with FIPS 200.</td>
</tr>
<tr>
<td>of Operations (CONOPS) is written to describe operational aspects of</td>
<td><strong>Design System Security Architecture</strong></td>
</tr>
<tr>
<td>the candidate system (or systems). Baseline requirements are established.</td>
<td>The ISSE works in conjunction with system architect and engineers in defining a system architecture</td>
</tr>
<tr>
<td><strong>Design System Architecture</strong></td>
<td>using the designated system architecture framework to explain the system architecture at the</td>
</tr>
<tr>
<td>The SE performs functional analysis and allocate by analyzing</td>
<td>conceptual and logic levels in meeting the defined baseline security requirements.</td>
</tr>
<tr>
<td>candidate architectures, allocating requirements, and selecting</td>
<td></td>
</tr>
<tr>
<td>mechanisms. The system engineer identifies components or elements,</td>
<td></td>
</tr>
<tr>
<td>allocates functions to those elements, and describes the</td>
<td></td>
</tr>
<tr>
<td>relationships between the elements.</td>
<td></td>
</tr>
</tbody>
</table>

(1/5)
<table>
<thead>
<tr>
<th>Systems Engineering (SE) Activities</th>
<th>Security Engineering (ISSE) Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Develop Detailed System Design</strong></td>
<td><strong>Develop Detailed Security Design</strong></td>
</tr>
<tr>
<td>The SE analyzes design constraints, analyzes trade-offs, does detailed system design, and considers life-cycle support. The systems engineer traces all of the system requirements to the elements until all are addressed. The final detailed design results in component and interface specifications that provide sufficient information for acquisition where the system is implemented.</td>
<td>The ISSE analyzes the design constraints, trade-offs from the system architecture and begin to work with system architect and engineers to define detailed system design.</td>
</tr>
<tr>
<td><strong>Implement System</strong></td>
<td><strong>Implement System Security</strong></td>
</tr>
<tr>
<td>The SE moves the system from specifications to the tangible. The main activities are acquisition, integration, configuration, testing, documentation, and training. Components are tested and evaluated to ensure that they must meet the specifications. After successful testing, the individual components – hardware, software, and firmware – are integrated, properly configured, and tested as a system.</td>
<td>The ISSE works with SE in implementing the baseline detailed system design. The information systems security engineer provide inputs to the certification and accreditation (C&amp;A) process and verify the implemented system design meets the defined baseline security requirements against the identified threats.</td>
</tr>
<tr>
<td><strong>Assess System Effectiveness</strong></td>
<td><strong>Assess System Security Effectiveness</strong></td>
</tr>
<tr>
<td>The results of each activity are evaluated to ensure that the system will meet the user’s needs by performing the required functions to the required quality standard in the intended environment. The systems engineer examines how well the system meets the needs of the mission.</td>
<td>The ISSE focuses on the effectiveness of the implemented security controls and countermeasures, and validates them against the defined information management plan (IMP).</td>
</tr>
</tbody>
</table>
1. Discovering the Information Protection Needs
   1.1 Collect & analyze system information: Business/Mission Needs, high-level concept of information operations, data sensitivity, mode of operations, etc.
   1.2 Perform Risk Assessment of the “to-be” information system
   1.3 Generate Information Management Model (IMM)
   1.4 Generate Information Protection Policy (IPP)
   1.5 Assemble Information Management Plan

2. Defining the System Security Requirements
   2.1 Define security context description (i.e. scope)
   2.2 Generate system security requirements: functional & assurance
3. Designing the System Security Architecture
   3.1 Describe the Conceptual Security Architecture
   3.2 Describe the Logical Security Architecture
   3.3 Describe the Physical Security Architecture

4. Developing the Detailed System Security Design
   4.1 Describe the Security Architecture at the components level
      4.1.1 Defending the Network & Infrastructure
      4.1.2 Defending the Enclave Boundary
      4.1.3 Defending the Computing Environment
      4.1.4 Supporting the IT Infrastructure
5. Implementing the System Security
   5.1 Implement system design for defending the network infrastructure
   5.2 Implement system design for defending the enclave boundary
   5.3 Implement system design for defending the computing environment
   5.4 Implement system design for supporting the IT Infrastructure

6. Assessing the Security Effectiveness
   6.1 Perform analysis on Security Requirements Traceability matrix (S-RTM)
   6.2 Verify conformance of system design to S-RTM
   6.3 Validate security implementation to S-RTM
   6.4 Support Security Certification & Accreditation (C&A) Team
Group 1: Waterfall SDLC Model

- **Requirements**
  - ISSE Phase 1: Discovering Information Protection Needs
- **Design**
  - ISSE Phase 2: Define Security Requirements
  - ISSE Phase 3: Design System Security Architecture
- **Implementation**
  - ISSE Phase 4: Develop Detailed Security Design
  - ISSE Phase 5: Implement System Security
- **Verification**
  - ISSE Phase 6: Assess Security Effectiveness
- **Maintenance**
Group 1: Waterfall SDLC Model
Group 2: DoD-STD-2167A (V-Model)

ISSE Phase 1: Discovering Information Protection Needs

ISSE Phase 2: Define Security Requirements

ISSE Phase 3: Design System Security Architecture

ISSE Phase 4: Develop Detailed Security Design

ISSE Phase 5: Implement System Security

ISSE Phase 6: Assess Security Effectiveness
Group 2: DoD-STD-2167A (V-Model)
Group 3: Boehm’s Spiral SDLC Model
Group 3: Boehm’s Spiral SDLC Model
Reference

ANSWERS
Group 1: Waterfall SDLC Model
Group 2: DoD-STD-2167A (V-Model)
Group 3: Boehm’s Spiral SDLC