CISSP® 2015

Domain 8: Software Development Security

V1

Domain 8: Software Development Security

A. Understand and apply security in SDLC

- **Overall**
  - Functionality First
  - Function Complexity
  - Programming itself are not easy to be understood by others
  - Web Programming
  - Changes in Hacking purpose due to Importance of Information
  - Easy tools
  - Mistakenly rely on perimeter devices, instead of software
  - Configuration security: use default installation of OS, Application etc., default admin, quest accounts…; does not patch when 1st installation
- **Failure States**
  - application should return to a safe state for any unpredictable fails
  - that is why **“blue screen” or “restart”**

System Development

- **System Development includes security**
  - early address security, better and effective
- **Life-Cycle Phases**
  - **Project Initiation:** Including Risk Management and Risk Analysis
  - **Requirement gathering:** should include security, such as system/data classification; backup requirement etc.

SDLC

- **System Design Specification:**
  - Consider access control, encryption and integrity mechanism etc.
- **Software Development (& testing):**
  - include programming and **testing**;
  - consider different types of attacks
  - *Separation of Duties and environment*: development, testing and production
- **Installation / Implementation**: configuration, manuel etc.
- **Operational / Maintenance**: continually conducting vulnerability test, new risk analysis for major changes
- **Disposal**: securely, say backup, destroy data, overwriting, degaussing sensitive information.

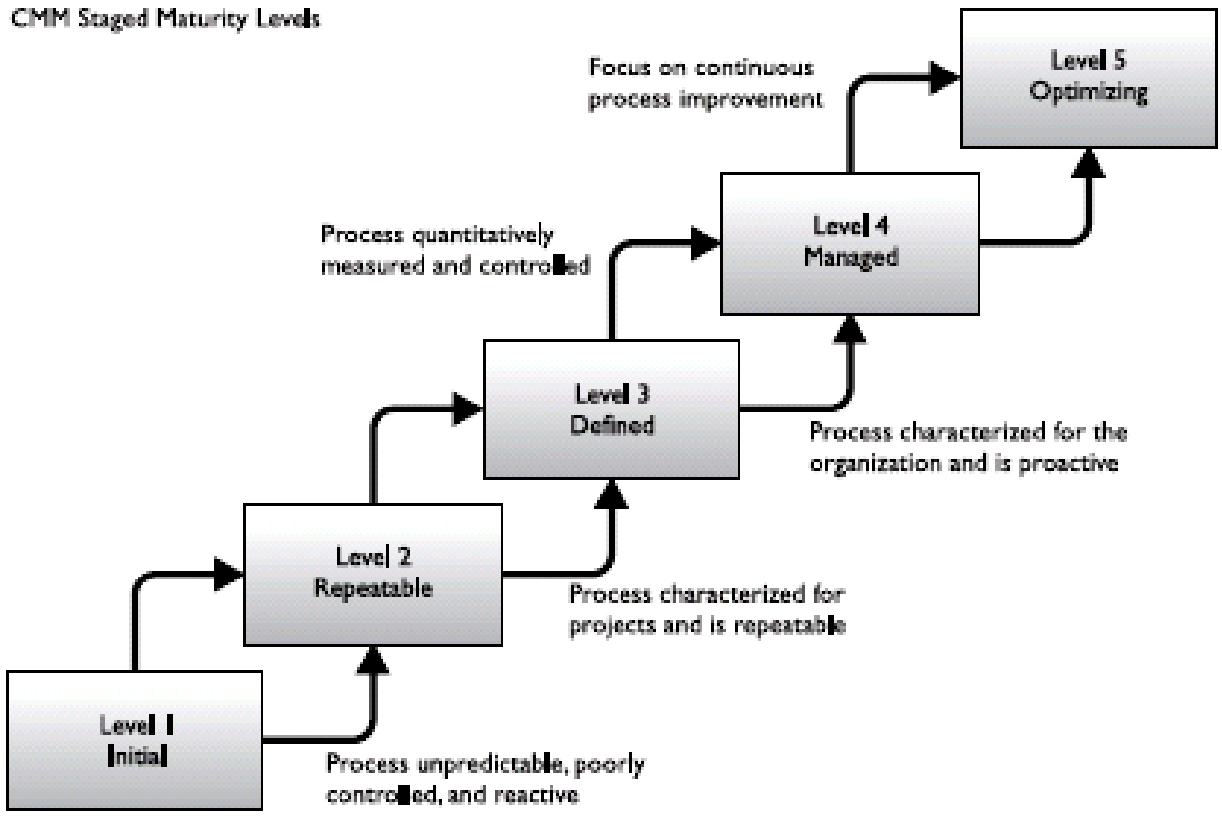
Testing

- **Testing Types**
  - **Unit** testing
  - **Integration** testing
  - **Acceptance** testing: Ensure to meet customer requirement
  - **Regression** testing: Partially retesting a modified program
- **Postmortem Review**
  - improvement, Lesson Learning

A2. Capability Maturity Model (CMM)

- describe the **maturity** of software development process
- Define the maturity by considering the policy, procedure, guideline and best practice to develop a standard approach
- Five maturity levels
  - **Initial (1)**: ad hoc; recognize issue; no standardized process; case-by-case; individual
  - **Repeatable (2)**: formal structure, change control and QA; no formal training & communication
  - **Defined (3)**: formal procedure; documented, communicated, trained; no deviation detection
  - **Managed (4)**: formal process for metrics, improvement etc.
  - **Optimizing (5)**: budgeted and integrated plan for continuous process improvement

The Capability Maturity Model (CMM)



Question

When should security first be addressed in a project?

A. During requirements development

B. During integration testing

C. During design specifications

D. During implementation

Escrow

- **Software Escrow**
  - third party company to keep vendor's source code
  - source code will be released to customer if vendor is out of business
  - a protection

A4. Change Management

- **Change Management**
  - Request
  - Impact assessment
  - Approval/Disapproval
  - Build and test
  - Notification: notify users about proposed change and schedule of deployment
  - Implementation
  - Validation

A5. Integrated product team

- Also called “DevOps”
- An Integrated Product Team (IPT) is a multidisciplinary **group of people** (users, customers, management, developers, contractors) who are collectively responsible for delivering a defined product or process.
- IPTs are used in **complex development** programs/projects for review and decision making.

Waterfall vs. Prototype

- **Waterfall**: phase by phase
  - **Structured Programming Development:** traditional
  - **Spiral**: similar waterfall with PDCA (Plan-Do-Check-Act) in each stage
  - **Cleanroom**: structured and formal method developing and testing to ensure error free
- **Iterative Development**
  - **Prototyping:** During requirement gathering, create model to show customer, easier to discuss and gather exact requirement.
  - **Modified Prototype Model (MPM):** even quick timeframe, Ideal for Web application development
  - **Rapid Application Development (RAD**): quickly to satisfy immediate needs, beware poor design
  - **Joint analysis development (JAD):** team with users, developer, technical expert.

Other tools

- **Computer-Aided Software Engineering (CASE)**
  - tools to create and manage software
  - example: translator, compilers, assemblers…. Debugger, code analyzer, version control
- **Component-Based Development**
  - Using standardized building blocks to assemble
- **Reuse Model:** objects can be exported reused and modified
- **Extreme Programming:** programming in pairs

B. Enforce security controls in dev. env.
B1. Security of the software environments

- **Segregation of environments**: Production, UAT, DEV, DR etc.
- **Segregation of duties**: Developer are not allowed to production.
- In release, both source and object are released to production.
- Once UAT signoff, do not allow Developer or Vendor to modify any objects (last minute change).

B. Enforce security controls in dev. env.
B1. Security of the software environments

- **Open Source**
  - Keep secrecy of proprietary code vs. open-source code
  - Many successful model.
  - Security concerns:
    - **Agree**: help to find vulnerabilities, prevent “security by obscurity”
    - **Disagree**: hacker may not disclose the vulnerabilities for their own benefit
  - Full or Partial Disclosure
- **Citizen Programmers**
  - Or called “Casual Programmer”
  - Little or no training in security issue
  - Example, VB programmer, Excel Macro etc.
  - **Risk**: lack of version control; access to macro; incomplete testing; little documentation,

B2 Security weaknesses and vulnerabilities at the source-code level

- **Security Issues of Programming Languages**
  - **Low-level vs. High-level languages:** more standardization, easier to understand, more functionalities
  - **Object-Oriented Technology and Programming:** Reusability, Encapsulation (Data Hiding inside class) and Polymorphism (Dog Go WOOF and Cat Go MEOW)
  - **Libraries & Toolsets**: Increased Dependability, Effective Use of Specialists, Standards Compliance, Accelerated Development; any downside?
  - **Integrated Development Environments (IDEs) & Runtime**: include source code editor, build automation tools and debugger, hierarchy diagram

Security Issues in Source Code

- **Buffer Overflows**
  - Taking too much data are accepted as input to application or OS
  - Attack purpose: make a mess of memory segment or accomplish a specific task (such as open a command shell)
  - Developer did not code the program properly, should use “bounds checking”
  - If attacker knows the size of the buffer and address allocation, he can control the value of “return pointer” of memory, to point to the malicious code
  - Countermeasure: (1) if buffer overflow is identified, apply patch, hot fix etc. (2) use proper programming, such bounds checking (3) Some language (such as C) are more susceptible to buffer overflow than others, programmer should understand these issues.

Malicious Software (Malware)

- spread by email, sharing media, document, program, download, purposely inserted by attacker
- email address book
- **Viruses**
  - Small software; to **reproduce**; **requires host action** to spread; may cause destruction
  - **Macro virus**: written in macro language; common and easy to write
  - **Boot Sector Virus**: Some part in boot sector, some part in harddisk marked as bad
  - **Compression virus**: append to executable and compress them
  - **Script virus**: written in script, such as VB script, java script.
- **Worms**
  - Getting smaller different from virus (getting similar to virus)
  - Reproduce on their own without a host application
  - ILOVEYOU is worm as user executes email attachment and will spam to all address. ILOVEYOU is virus by using e-mail client.

Malicious Software (Malware)

- **Logic Bombs:** activate when certain condition or date or time
- **Trojan Horses**
  - pretend to another program (say notepad.exe) to capture information and send to attacker
  - can be detected by size
- **Remote Access Trojan (RAT):** allow intruder to access and use a system remotely; attacker can download or upload files, send command, install zombie…
- **Botnets**
  - “bot” is short for “robot”, is a zombie
  - Carry out powerful DDoS attacks
- **Spyware & Adware**
- **Hoaxes**: False Information about Malware

Malware Protection

- **Policy:** email attachment, awareness, more than one scanners
- **Scanners:** Signature, to search string/code, Known virus only
- **Heuristic Scanners:** intelligent analysis of program structure for suspicious, such as modify own code..
- **Activity Monitors**: a form of automatic auditing, promote alter or delete program file, may cause many warnings for valid operations
- Change Detection: use Cyclic Redundancy Check (CRC) to detect the change of system, programs…
- Reputation Scoring: mainly on website reputation
- Zero-Day/Zero-Hour: New website is Suspicious

Question

3. An application is downloaded from the Internet to perform disk cleanup and to delete unnecessary temporary files. The application is also recording network login data and sending it to another party. This application is best described as which of the following?

A. A virus

B. A Trojan horse

C. A worm

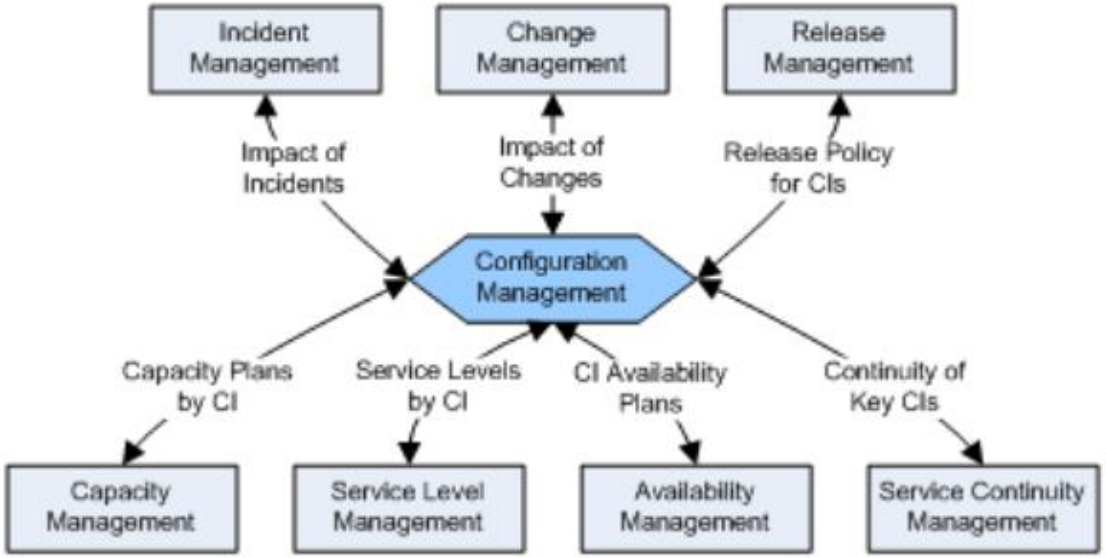
D. A logic bomb

Program Development Protections

- **Inadequate Granularity of Controls**
  - Granularity = Quality of code
  - **Example**:
    - No access to object A, but program has, user can call program to get access right of object A
    - Programmer has access to computer room or program in production
- **Input validation, Output validation**
- **Incomplete parameter check and enforcement**
- **Password protection**
- **Cryptographic:** DB encryption, Hash to detect integrity
- **Separation of Environments (Prod, UAT, Dev, DR…)**
- **Backup Controls:** O/S and Program backup

B3. Configuration Mgt. as an aspect of secure coding

- **Configuration Management (CM)** is the discipline of identifying, recording, evaluating, tracking, coordinating, reporting, and controlling Configuration Items (CI) by performing supporting process activities that maintain the integrity of these items throughout the life cycle of a project, including their versions, constituent components and relationships.
- CI can be Hardware device, applications, programs and source codes etc.
- CM can maintain the program version, ownership, relationship, previous change history etc.



B4. Security of code repositories

- Ensuring safety of code while developing, using, at rest
  - **Physical Security**: Data center, Lock, Fire, CCTV…
  - **System Security**: Hardening, Patching
  - **Operational Security**: Change Management, log review
  - **Software Security**: monitoring, preventing and eliminating attacks
  - **Communication**: Data exchange, HTTPS, SSH…
  - **File system and backups:**
  - **Employee Access:**

B5. Security of application programming interfaces

- **Cohesion**
  - how many different types of tasks in a modular
  - **high cohesion**: only one or several very similar task; good practice; easier modify, simple and reuse
- **Coupling**
  - how much interaction to carry out its task
  - **low coupling**: not too much interaction; good practice; easier understanding

C. Assess the effectiveness of software security
C1. Auditing and logging of changes

- Auditing and Logging are **important** to overall health and security of systems
- Tools:
  - **Information Integrity:** example: compare what is processed vs. supposed
  - **Information Accuracy**: input validation,
  - **Information Auditing**: detect abnormal activities, such as unauthorized access

C2. Risk analysis and mitigation

- Risk Anywhere, including SDLC, bug….
- Corrective Actions:
  - Use a Change Control Process (or **Change Management**)
  - Read All Related **Documentation** before patching
  - **Testing**
  - Have a Working **Backup** and Schedule Production Downtime
  - Always have a Back-Out Plan (or **Fallback**)
  - **Forewarn** Helpdesk and Key User Groups
  - Target **Non-Critical Servers first**

C3. Acceptance testing

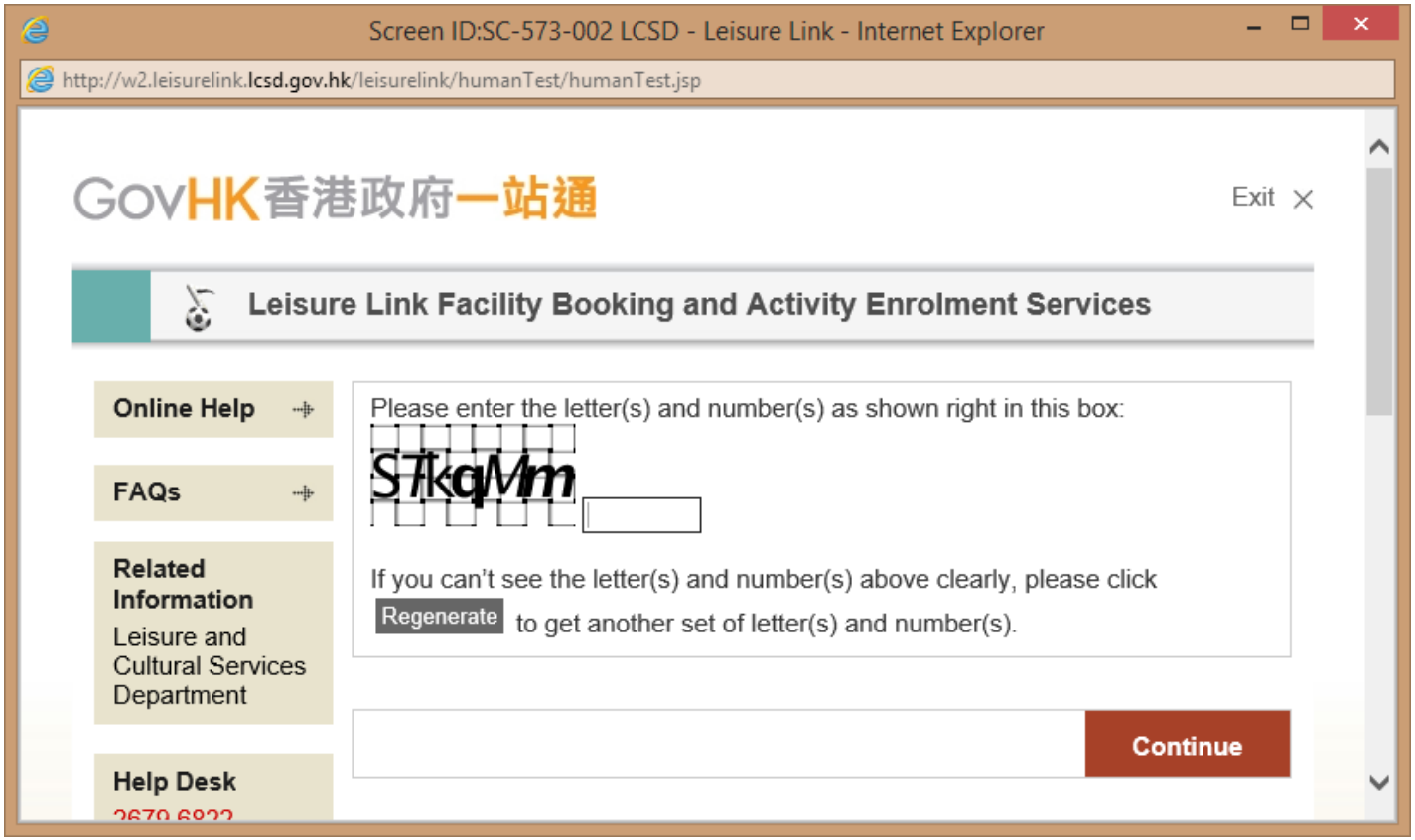
- **Acceptance testing** is a test conducted to determine if the **requirements** of a specification or contract are **met** prior to its delivery.
- **User acceptance testing**: testing done by **users** before the system is moved to Production site.
- **Operational acceptance testing**: to ensure **operational readiness** that processes and procedures are in place to allow the system to be used and maintained.
- **Contract acceptance testing**: test against acceptance criteria as documented in a contract, before the system is accepted.
- **Regulation acceptance testing**: to ensure it meets governmental, legal and safety standards.
- **Alpha testing**: involves testing of the operational system **by internal staff**, before it is released to external customers.
- **Beta testing (or field testing):** involves testing **by a group of customers** who use the system at their own locations and provide feedback, before the system is released to other customers.

D. Assess security impact of acquired software

- There are **risks** if you develop or acquire software for your organization.
- **Major phases:**
  1. **Planning Phase**
  2. **Contracting Phase**
  3. **Monitoring & Acceptance Phase**
  4. **Follow-on:** Continuous improvement, disposal, decommissioning

CAPTCHA

- A CAPTCHA: **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
- CAPTCHA is a type of challenge-response test used in computing to determine whether or not the user is human.



Attacks

- **Distributed Denial of Service**
  - logical extension of DoS
  - use hundreds or thousands to computer (or zombie)
  - countermeasure: restrict unnecessary ICMP and UDP; network-based IDS; hardening; rename admin & strict password; perimeter router;