SSD CT Study Checklist

Week 1 - Module Intro Slide Deck –

o Secure Software Development Life Cycle (SSDLC) based on the following slides:

* Module Sypnosis
* Aims of the module

Type of questions asked: Given a scenario suggest how SSDLC can help a company?

Ans: A company will be able to build secure software by gathering security requirement, implement secure software design, code, testing and deployment, which will protect the company’s software from hacker and user failure.

# Week 2 - Security Concepts

o Holistic Security (Refer to example)

o Challenges | Week 2 Research Point (In Week 3 notes)

-> Iron Triangle

-> Security vs Usability

-> Security as an afterthought

Type of question: Given a scenario, identify and explain what type of challenge(s) is being faced by the company.

Week 2 Research Point (in Week 3 Folder Week2ResearchPoint.pdf)

o Attacker vs Defender

o Auditing & Accountability

* Accountability and auditing (who did what when and where) important to achieve non-repudiation

o Non-Repudiation

o Data Privacy / Anonymization Techniques (Focus)

* Replacement
* Suppression
* Generalization
* Pertubation

Type of Question: Given a scenario, identify which of the above points are either applied or not applied.

# Week 3 - Security Requirements

o Quality Attributes of Secure Software

-> Reliability

-> Resiliency

-> Recoverability (restore software operations)

Type of question: Match a given scenario to the above quality attributes

o Confidentiality

-> Encryption and Decryption (Data in rest/stop), HTTPS (Data in transit)

o Integrity

-> Hashing (Salted and unsalted hash)

-> Benefits of Salted Hash

-> Example of Salted Hash

o Availability

-> Service Level Agreement 99.999% (5'9s)

-> Time frame given to recover the software from time of disruption

o Authentication

-> Forms (HTTPS)

o Authorization

-> Role Based Access Control

o Accountability

-> Logging (WHO did WHAT, WHERE and WHEN)?

o Session Management

-> Session ID (Cookie)

o Errors and Exception Management

-> Non-Verbose error message (Not too detailed)

o Explicit vs High-Level

-> How the instructions towards the software developer are detailed

Type of question: Given a scenario identify and explain which of the above security requirements are applicable to the given scenario OR how the above security requirements can be applied to improve the given scenario.

# Week 4 - Gathering Secure Software Requirements

o Brainstorming

-> Issues

- Finding Available time to come together and brainstorming

- People may be introverted

- Takes a long time (not all details can be captured)

- Talk about things that do not relate to the business security (waste of time)

o Surveys and Questionnaires

-> Ask questions related to the security requirement in a nontechnical manner (descriptive manner)

o Policy Decomposition

-> Extracting security requirements from existing policies

o Data Classification

-> To identify which data to protect or make publicly available (refer to the labelling Week 4 Slide 9)

Type of question: Match a given scenario to the above PNE / Secure Software Requirements Gathering Techniques

o Subject-Object Matrix

-> 2-dimensional matrix that identifies the subjects and objects to perform CRUD (Create, Read, Update, Delete)

o Use Case and Misuse Case Modelling

-> Use case - Actions that can be performed by users

-> Misuse case - Actions that unauthorised users can perform

Type of question: Given a scenario come up with Subject-Object Matrix and Use Case and Misuse Case diagrams

# Week 5 / Week 6 - Secure Software Design

o Benefits of Secure Design

-> Minimise time required to identify and patch security issues

-> Minimise the redesigning of software

o Encryption

-> Symmetric Key Cryptosystem

- No Complex Mathematical Formulas

- Fast

- Transposition, Bitwise XOR function, Substitution

-> Asymmetric Key Cryptosystem

- Exponentiation

-> Combining Symmetric Key Cryptosystem and Asymmetric Key Cryptosystem (HTTPS)

o Integrity

-> Hashing

- Salted Hash

-> Frustrate Hackers (make them have to crack the hash again even if its the same password)

-> Defends against Dictionary Attack and Rainbow Attacks

- Unsalted Hash

-> Same passwords will produce the same hash and the Hackers does not require to crack it again

- Characteristics

-> Referential Integrity

-> Resource Locking

- Two updates on the same resource are stopped

- Poor design results in a Deadlock

o Availability

-> Failover

- Automatic switching from active system (Primary System) to a standby system

- Potential Single Points of failure are addressed during the designing of the

solution

-> Replication

- Active / Active

- Master (Primary Node) and Slaves (Secondary Node) get updated at

the same time

- Active / Passive

- Master (Primary Node) gets updated first, Slaves (Secondary Node) at

a later time

-> Scalability

- Able to handle an increase traffic load

- Vertical Scaling

-> Scaling up

-> Increasing memory and storage resources

- Horizontal Scaling

-> Scaling out

-> Installing additional copies of software

o Authentication

-> Multi-Factor Authentication

- Example: OTP, Authenticator Code

- What you have, what you are and What you know?

-> Single Sign On (SSO)

- Only needs to verify once

- Sign into multiple services

- Example: Microsoft Office, Google

- Poor design can result in major security breaches

-> Federation

- Does not require registration of another service

- Example: LinkedIn can be logged into by using a Google Account

Type of question: Given a scenario identify and explain which of the above security design techniques are applicable to the given scenario or how the above security requirements can be applied to improve the given scenario.

o Key Secure Design Principles

-> Least Privilege

- Right amount of privilege to perform the task

- Example: Modular Programming

-> Execute Smaller code to perform actions specific to role of users

-> Separation of Duties

- Two or more conditions must be satisfied

- Example: Programmer should not be allowed to review their own code

-> Defence-in-Depth

- Layering Security Controls

- Input Validation to prevent against malicious code

- Input Validation to prevent dynamic query constructions

- Security Zones (separate different levels of access)

-> Fail Secure

- Account lockout

- Error Management

- Example: Too many failed password entry

- Example: General Error Messages

-> Economy of Mechanism

- Unnecessary Functionalities should be avoided

-> Complete Mediation

- Parameter values when changed leads to an appropriate action

- Example: changing username in the parameter results in the site going to the

login page

-> Open Design

- Security of software is not dependent on the secrecy of the design

-> Least Common Mechanism

- Code is segregated for the different users

- 2 features should not have a common code and function

-> Psychological Acceptability

- Easy to use

- Don't affect usability

- Transparent to users

-> Weakest Link

-> Leveraging Existing Components

- Use security management well reviewed instead of creating your own

- Increase Productivity by reducing time needed to develop the existing

Components

Type of question: Match a given scenario the above Secure Software Design Principles

# Week 7 - Secure Software Design

o Threat Modelling

-> Extremely crucial for developing hack-resilient software

-> Should address common threats

-> Identify entry and exit point

-> Threat Modelling Process

o Model Application Architecture

* Deployment structure (Servers)
* Ports / Protocol
* Identities and Authentication
* Actors (End users and staff)
* Subject object matrix
* External Dependencies (3rd party services, webserver)

o Identify Threats

* Entry Points
  + Any Page that requires updating by user
  + Preferences that are selected by user stored to the database
* Exit Points
  + Any Page that shows the result which calls on the database
  + Preferences of the page is stored on the database
  + Any page that changes based on the user’s action
* Apply STRIDE
  + Spoofing
  + Tampering
  + Repudiation
  + Information Disclosure
  + Denial of Service
  + Elevation of Privilege

o Identify, Prioritize, and implement controls

* DREAD
  + Damage Potential (D)
  + Reproducibility (R)
  + Exploitability (E)
  + Affected Users (A)
  + Discoverability (Di)
* Risk = Probability (R+E+Di) X Impact (D+A)

Type of question: Given a scenario identify the Entry and Exit points, and apply STRIDE and Calculate DREAD risk based on the given values and propose security controls to the top threats

Tips: Understand the example don't memorize

Example Questions:

Which does the following refer to?

A. Integrity

B. Availability

C. Authentication

D. Accountability