Domain 8

Software Development Life Cycle

Software Development Life Cycle (SDLC)

The process of developing software is organized, planned, and managed using the software development methodology.

The management or development team must select the software development methodology that will be most effective for the project at hand if they are to manage the project effectively.

Every methodology differs in its strengths and shortcomings as well as the motivations for its creation.

SDLC Model

- **Waterfall**
  - Changes were difficult to manage – in early days there were minor changes
  - Developed in 1976

Each phase leads into the next phase

Cant go back in previous phase

- **Spiral**
  - Developed in 1988
  - Identify 100 features to develop
  - First 10 features develop and test (PoC)
  - Team feedback and incorporated in development
  - Again 5 features developed and send to Team for
  - feedback
  - This is on going process
  - Cant go in production till all 100 features
  - developed

Agile

- Rapid app development
- Extreme programming
- idea is work on 10% requirements – once developed, test and implemented after that work on next 5 requirements. Once test and implemented go for next phase

- **Scrum**
  - Product owner
  - Team
  - Scrum Master

DevOps

- Integrated Team derived from development and operations
- Software, QA, Operations and Security Team work together – all Teams are integrated

Coordinated b/w Development, QA & Operations

DevSecOps

Development, QA, Operations & Security

Software Capability Maturity Model (SCMM)

- **Initial**
  - Software dev is ad-hoc
- **Repeatable**
  - Define soft dev process
  - Everyone follow the doc
  - This is re-active
  - Once find issue – fix it
- **Defined**
  - Process defined
  - Documents available
  - Pro-active
- **Managed**
  - Monitor software dev stages
  - Monitor objectives – either we are achieving or not
- **Optimized**

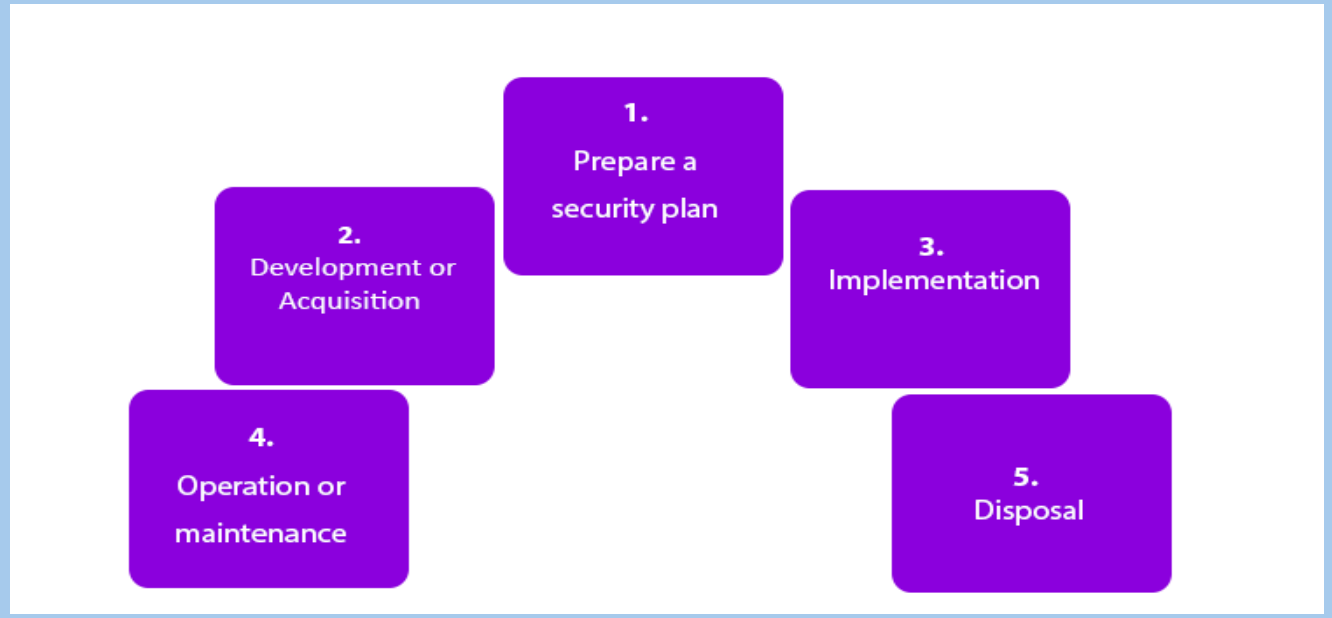
How to improve software Security in organization

Software Assurance Maturity Model (SAMM)

- Measurable
- Accountable
- Versatile

SDLC Phases

- **Plan**
- **Requirements gathering** -> Features/Sec issue
- **Design** -> Sec Controls
- **Development** -> Code review
- **Testing**



Testing

Static

Code review

Dynamic ->

Fuzz
Unit
Integrate
Regression

Evaluation

Trust -> Are Security features are included in the software

Assurance

Control fulfill the gaps

Accreditation -> Risk on management, they have to accept
PROD -> Software moves into production

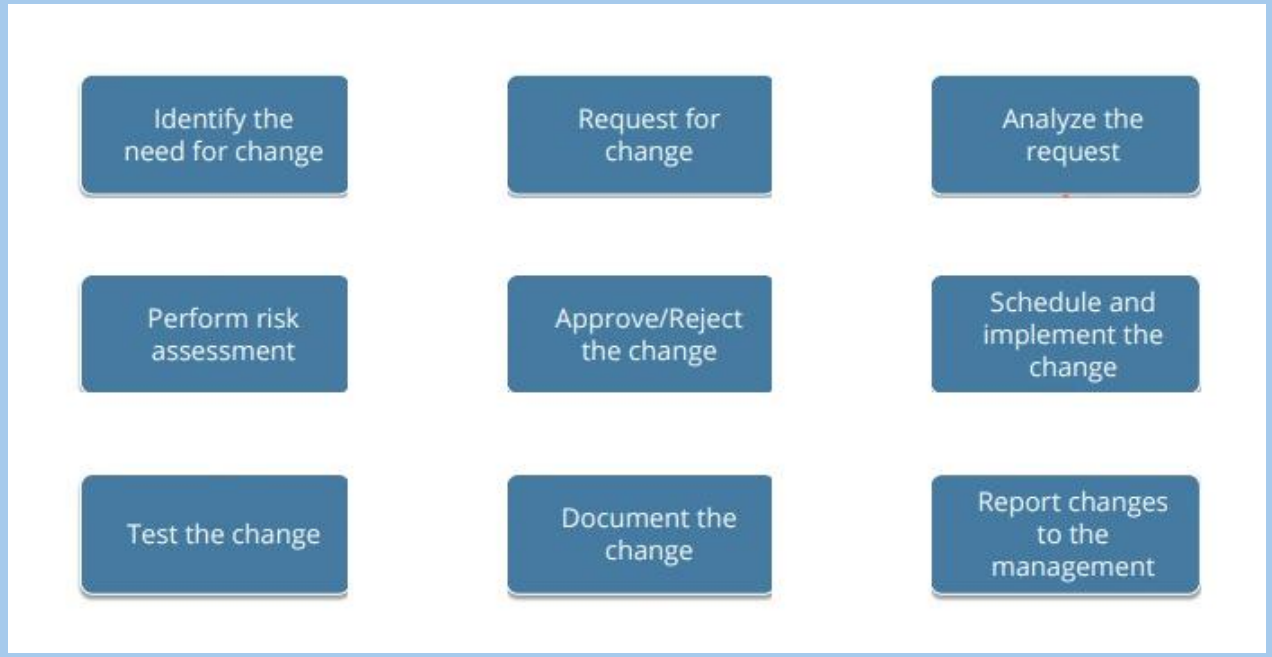
Evaluation -> Evaluate the performance
Trust -> Does software has all Security identified and implemented
Assurance -> Controls implemented - > What is the guarantee that Controls are fulfilling the requirements

Exp: Malware ->

Change Management

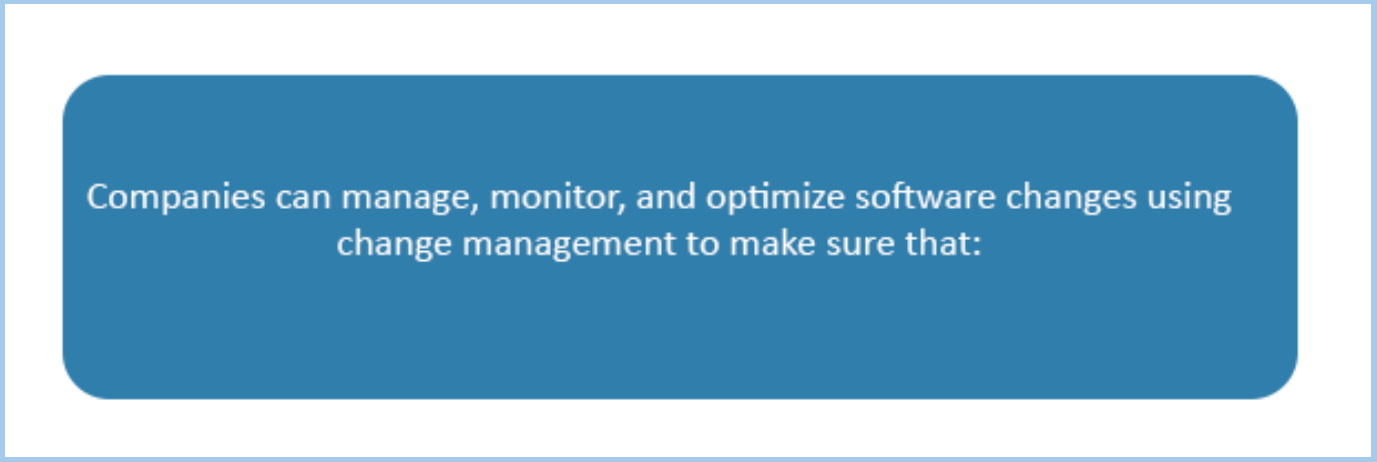
- Identify the need for Change
- Request for Change
- Analyze
- Perform Risk Assessment

- If any unexpected change occurs in system
  - Change requested
  - Change advisory Board (CAB)
  - They evaluate the impact
  - Develop changes
  - Test
  - Move to PROD
  - Reporting



Change Types

- **Standard Change**
  - Low risk/Impact – Don’t need to follow all the stages
- **Normal Change**
  - Change go through all the stages
- **Emergency Change**
  - Address immediately – don’t need to go through all the stages



Buffer Overflow

- App has input area
- Attackers input very large amount of Data in input area
- Buffer or memory overflow
- This is DDoS
- Hackers input scripts in input area they overflow memory

Buffer – temporary storage area to storage Data which program used

When store Data outside memory range

Inject more Data in memory which cant hold by design

When attackers inject malicious code on trusted webSite

- **SQL Injection Attack**

  - Attackers inject scripts into the code
  - Go to web server
  - Once scripts run -> it will generate results goes to code
  - That results users can see on the user page
  - Attackers do same thing for Active Directory
  - Do same thing for XML
  - Attackers bypass authentication
  - For that use input validation
  - Application check the inputs -> if valid -> Accepted
  - If invalid -> Not Accepted

Attackers inject SQL query from the client to web application

Software Configuration Management

The process of methodically managing, organizing, and controlling changes to documents, codes, and other entities across the SDLC is referred to as software configuration management (SCM).

- **Process of SCM**

  1. Configuration Identification
  - Document software features
  - Identified all the software features

  2. Configuration Control

  3. Configuration Status Accounting
  - Verify that required features are implemented
  - Do UAT to verify

  4. Configuration Audit

Social Engineering

- Contact users through
  - Email – Phishing
  - SMS – Smashing
  - Voice – Vishing

Software Security and Assurance

A security policy is enforced by the security kernel.

All information references and authorization changes must go through this tiny section of the operating system.

It adheres strictly to the reference monitor mechanism.

Software Security and Assurance: Cryptography

They are employed to protect the confidentiality and accuracy of data.

The operating system contains a number of particular files that can be encrypted using encryption algorithms.

By encrypting data, cryptographic methods change the information to make it secure.

Password Protection

***Passwords are a handy way for operating systems and application software to authenticate users.**

***Password security include restrictions on**

-How the password is chosen

-How difficult the password is

-Password time limits

-Password length

***Encrypting password files with one-way encryption methods or hashing is the most common fix.**

***Overstrike or password masking is another tool for password security.**

Methods to Assess the Effectiveness of Software Security

SYSTEM AUTHORIZATION:

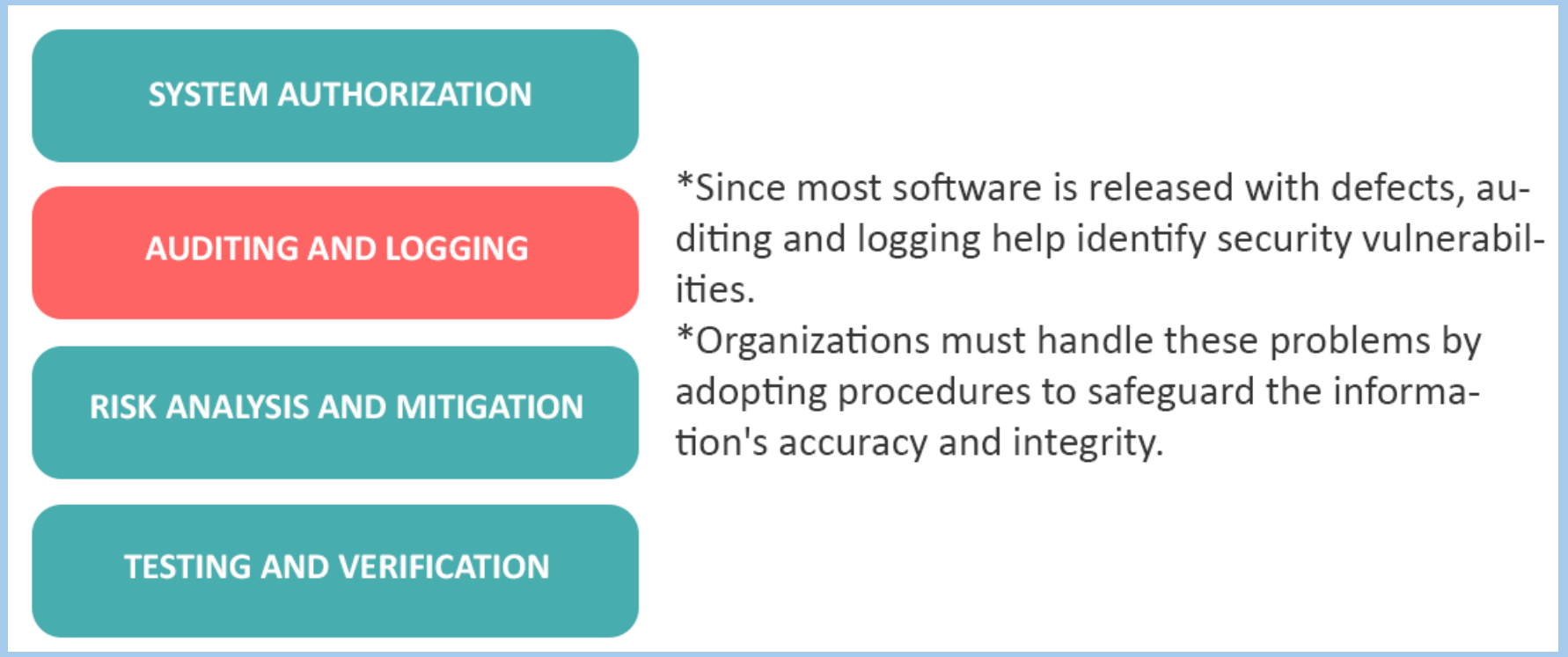
*Systems that process, store, or send information must be certified, accredited, or authorised.

AUDITING AND LOGGING:

*It guarantees that a control framework is chosen and properly enforced throughout the company with the aid of standards.

RISK ANALYSIS AND MITIGATION:

TESTING AND VERIFICATION:


SYSTEM AUTHORIZATION
AUDITING AND LOGGING
RISK ANALYSIS AND MITIGATION
TESTING AND VERIFICATION
*Since most software is released with defects, auditing and logging help identify security vulnerabilities.
*Organizations must handle these problems by adopting procedures to safeguard the information's accuracy and integrity.

SYSTEM AUTHORIZATION:

AUDITING AND LOGGING:

RISK ANALYSIS AND MITIGATION:

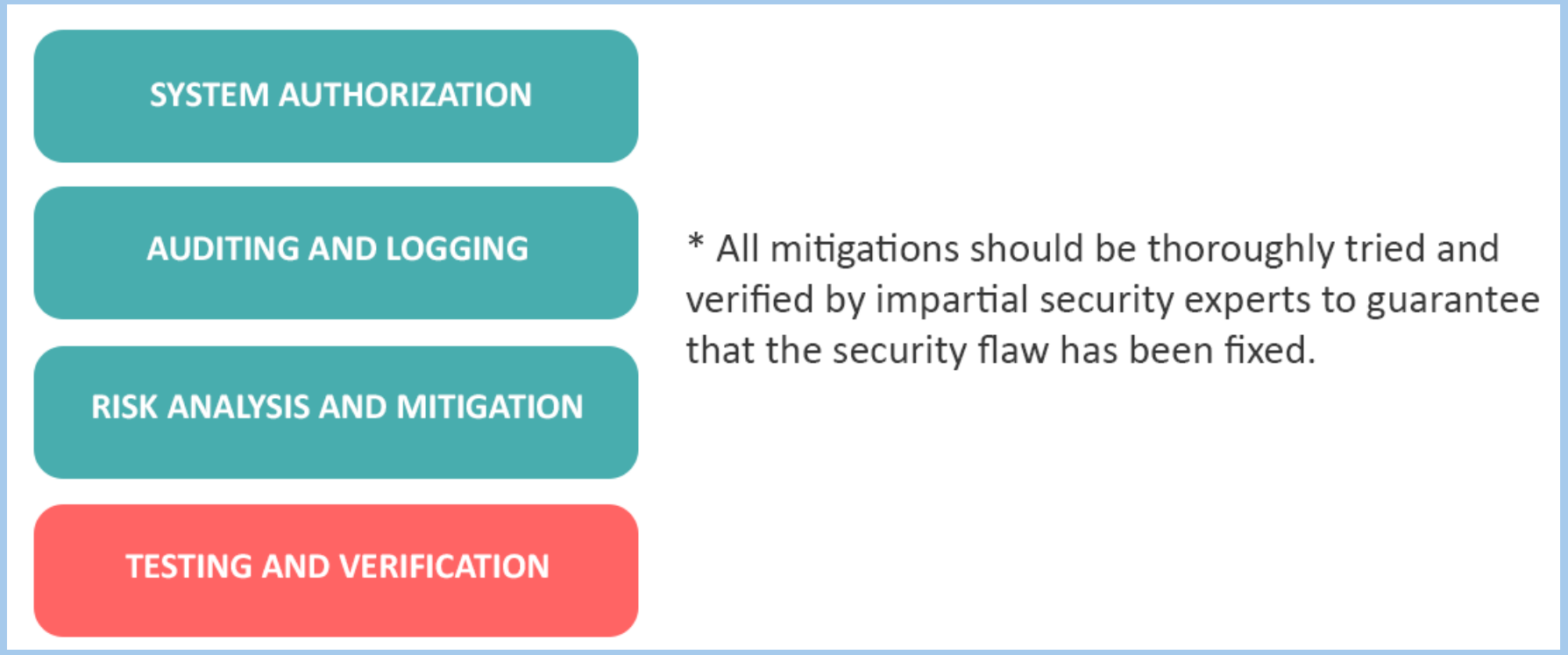
TESTING AND VERIFICATION:

Change management and the SDLC must both include risk assessment and mitigation as continuous processes. It includes:

*Using standardised methods as outlined in models like 1S0 and NIST, evaluating risk and reporting to stakeholders.

*Monitoring and managing vulnerabilities.

*Evaluating the results and assigning a priority to them in order to decide what remedial measures need to be taken.


SYSTEM AUTHORIZATION
AUDITING AND LOGGING
RISK ANALYSIS AND MITIGATION
TESTING AND VERIFICATION
* All mitigations should be thoroughly tried and verified by impartial security experts to guarantee that the security flaw has been fixed.

Certification

*The process may involve the use of safeguard evaluation, risk analysis, verification, testing, and auditing techniques.

*The goal is to confirm that the technique is suitable for the client's requirements.

*Certification, which is frequently an internal verification, is only trusted by those who work for the business.

Accreditation

*The designated approving authority (DAA) officially declares an IT system's permission to operate in a particular security mode with a preset set of protections at an acceptable level of risk as accreditation.

*Management can formally admit that the evaluated system's general security performance is sufficient once accreditation is complete.

Free and Open-Source

FOSS is a term used to refer to software that is freely available for use, modification, and distribution.

As a result, other developers have the chance to add to the growth and evolution of a software like a community.

Free and Open-Source Software

The flexibility to use the program however and for whatever reason you please.

The ability to examine how a software functions and modify it to perform calculations as you see fit

Free software does not necessarily imply that it is also cost-free. Although FOSS is frequently offered without charge, its primary distinction from private software is that it is free in the sense of

The ability to share files freely in order to assist others

Ability to modify the software and distribute your customised copies to others