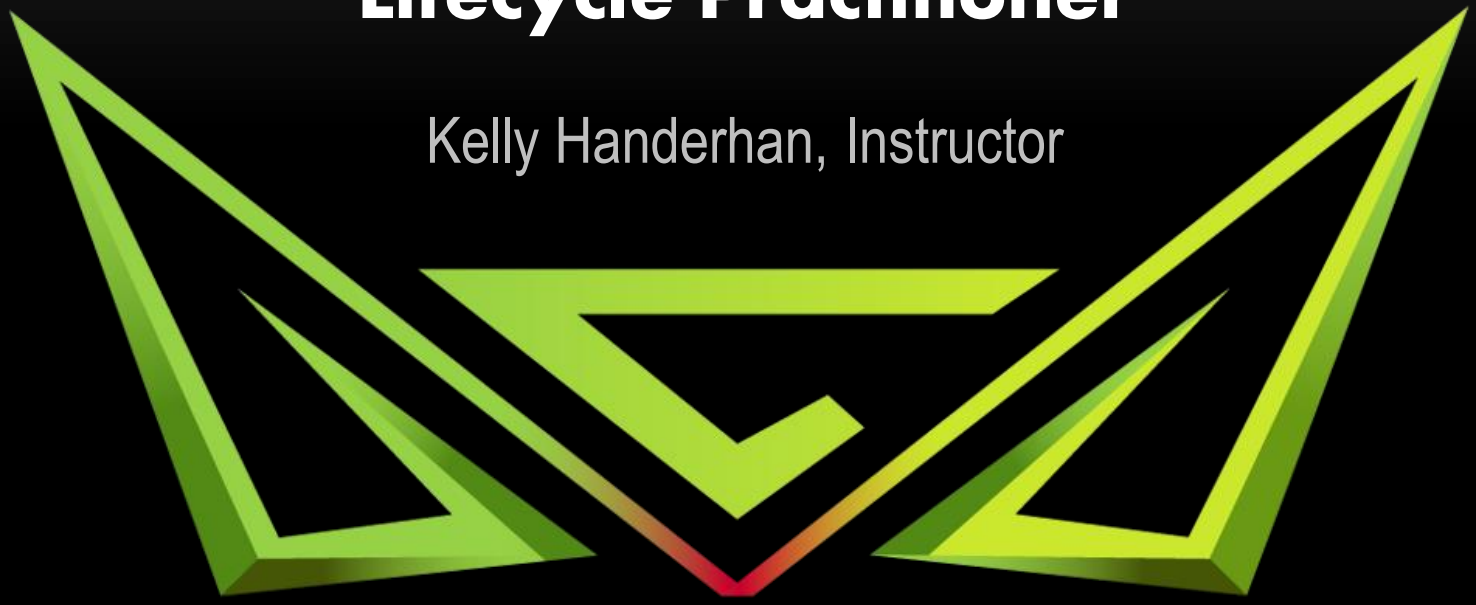
CSSLP
Certified Secure Software Lifecycle Practitioner

Kelly Handerhan, Instructor



CYBRARY

CSSLP AGENDA

- Part I Secure Software Concepts
  - General Security Concepts
  - Risk Management
  - Security Policies and Regulations
  - Software Development Methodologies
- Part II Secure Software Requirements
  - Policy Decomposition
  - Data Classification and Categorization
  - Requirements

CSSLP AGENDA CONTINUED

- **Part III Secure Software Design**
  - Design Process
  - Design Considerations
  - Securing Commonly Used Architectures
  - Technologies
- **Part IV Secure Software Implementation/Coding**
  - Common Software Vulnerabilities and Countermeasures
  - Defensive Coding
  - Secure Software Coding Operations

CSSLP AGENDA CONTINUED

- **Part V**
  - Security Quality Assurance Testing
  - Security Testing
- **Part VI Secure Software Acceptance**
  - Secure Software Acceptance
- **Part VII Secure Software Installation, Deployment, Operations, Maintenance and Disposal**
  - Secure Software Installation and Deployment
  - Secure Software Operations and Maintenance
  - Supply Chain and Software Acquisition

EXAM SPECIFICS

- 175 Questions
- 4 hours to complete the exam
- You can mark questions for review
- You will be provided with 1 laminated sheet 8x11 and a pen.
- You will also have access to an on-screen calculator if necessary
- Many test centers provide earplugs or noise cancelling head phones. Call your center ahead of time to verify
- Questions are weighted (Remember…security transcends technology)

THE CSSLP MINDSET

- The Exam is not about the code. It is about the process of writing the code.
- The better the process, the better the product.
- Do NOT fix Problems. Fix the process.
- How much security is enough?
- All decisions start with risk management. Risk management starts with Identifying/Valuating your assets.
- “Security Transcends Technology”
- Incorporate security into the design, as opposed to adding it on later
- Layered Defense!

TEST TAKING TIPS

- If you haven't already, SCHEDULE THE TEST!!!
- Start with the question mark. Often the beginning of the scenario is a distraction
- Choose an answer for EVERY question. Even those you mark for review, just in case you run out of time.
- Be cautious about changing answers. Your first instinct is often right. Trust yourself and your knowledge and what we do in class. Don't second guess!
- Take Breaks as needed. Plan on 50 questions per hour.

PART I

Secure Software Concepts

Part I General Security Concepts

SECURITY BASICS

SECURITY BASICS AGENDA

- General Security Concepts
  - CIA Triad
  - IAAA of Access Control
- Security Tenets
- Security Models
- Access Control Models
- Adversaries

CONFIDENTIALITY

- **Prevent unauthorized disclosure**
- **Threats against confidentiality:**
  - Social Engineering
    - Training, Separation of Duties, Enforce Policies and Conduct Vulnerability Assessments
  - Media Reuse
    - Proper Sanitization Strategies
  - Eavesdropping
    - Encrypt
    - Keep sensitive information off the network
  - Password Cracking
  - Strong Hashing Algorithms
  - Salting Passwords

INTEGRITY

- **Detect modification of information**
- **Corruption**
- **Intentional or Malicious Modification**
  - Message Digest (Hash)
  - MAC
  - Digital Signatures

AVAILABILITY

- **Provide Timely and reliable access to resources**
  - Redundancy, redundancy, redundancy
  - Prevent single point of failure
  - Comprehensive fault tolerance (Data, Hard Drives, Servers, Network Links, etc)

AUDITING

Logging and reviewing accesses to objects.

- What is the purpose of auditing?
- Auditing is a detective control

TENETS OF SECURE ARCHITECTURE AND DESIGN

- How Much Security is enough?
- Defense in Depth
- Fail-safe
- Economy of Mechanism (The K.I.S.S principle)
- Completeness of Design
- Least Common Mechanism
- Open Design
- Consider the Weakest Link
- Redundancy
- Psychological Acceptability
- Separation of Duties (SOD)
- Mandatory Vacations
- Job rotation
- Least privilege
- Need to know
- Dual control

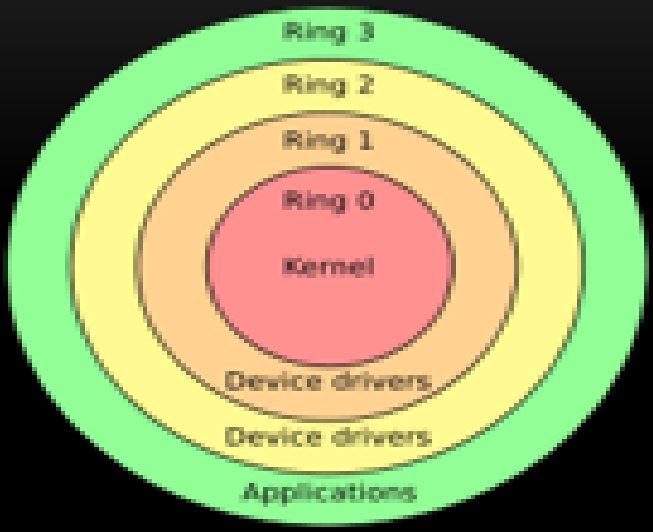
REQUIREMENTS OF SYSTEM ARCHITECTURE:

- An information system's architecture must satisfy the defined business and security requirements.
- Security should be built into an information system by design.
- When designing system architecture, security and business requirements needs to be carefully balanced.
- Tradeoffs are involved in reaching a balance between security and business requirements.

LAYERED DESIGN

CPU Modes & Protection rings

- Protection Rings provide a security mechanism fo an operating system by creating boundaries between the various processes operating on a system and also ensures that processes do not affect each other or harm critical system components.
- Ring 0 – Operating system kernel (supervisor /privilege mode)
- Ring 1 – Remaining parts of the operating system (OS)
- Ring 2 – Operating system and I/O drivers and OS utilities
- Ring 3 – Applications (Programs) and user activity



SECURITY ARCHITECTURE VS. SECURITY MODELS

- The security requirements of an information system are driven by the security policy of the organization that will use the security model lays out the framework and mathematical models that act as security-related specifications for a system architecture.
- The system architecture, in turn, is the overall design of the components - such as hardware, operating systems, applications, and networks – of an information system. This design should meet the specifications provided by the security model.

ELEMENTS OF SYSTEM ARCHITECTURE

- **TCB (Trusted Computer Base)**
  - Originated from the Orange Book and deals with the protection mechanisms within a computer. It addresses hardware, software, and firmware.
  - Security Perimeter
    - It delineates the trusted and the untrusted components within a computer system.
  - Reference Monitor
    - The reference monitor is an abstract machine concept that mediates all access between subjects and objects.
  - Security Kernel
    - The Security kernel enforces the reference monitor concept.
    - Must facilitate isolation of processes
    - Must be invoked at every access attempt.
    - Must be small enough to be tested and verified in a comprehensive manner.

SECURITY MODELS

- Bell-LaPadula
- Biba
- Clark-Wilson
- Brewer-Nash
- Take-Grant

BELL-LAPADULA

- Has 3 rules:
- Simple Security Property – “no read up”
  - A subject cannot read data from a security level higher than subject’s security level.
- *_Security Property – “no write down”
  - A subject cannot write data to a security level lower than the subject’s security level.
- Strong * Property – “no read/write up or down”.
  - A subject with read/write privilege can perform read/write functions only at the subject’s security levels.

BIBA INTEGRITY MODEL

- **The Rules:**
- Simple integrity axiom – "no read down" – A Subject cannot read data from an object of lower integrity level.
- * Integrity axiom – "no write up" – A Subject cannot write data to an object at a higher integrity level.
- Invocation property – A subject cannot invoke (call upon) subjects at a higher integrity level.

CLARK-WILSON

- Integrity models – Clark-Wilson Model
- Model Characteristics:
  - Clark Wilson enforces well-formed transactions through the use of the access triple:
  - User➔Transformation Procedure➔CDI (Constrained Data Item)
- SEPARATION of DUTIES
- Deals with all three integrity goals
  - Prevents unauthorized users from making modifications
  - Prevents authorized users from making improper modifications
  - Maintain internal and external consistency – reinforces separation of duties

BREWER-NASH

- **Brewer-Nash Model – a.k.a. Chinese Wall**
  - Developed to combat conflict of interest in databases housing competitor information
  - Publish in 1989 to ensure fair competition
  - Defines a wall and a set of rules to ensure that no subject accesses objects on the other side of the wall
  - Way of separating competitors data within the same integrated database
-

TAKE-GRANT

- Model that is used for analysis rather than design
- Used to examine a security model and determine whether or not information can leak through the assignment of privileges across boundaries

ACCESS CONTROL MODELS

The models we are about to discuss are

From the TCSEC(Trusted Computer System Evaluation Criteria—Orange Book)

- DAC (Discretionary Access Control)
- MAC (Mandatory Access Control)

- **Established Later**
  - RBAC (Role based Access Control)

DAC

Discretionary Access Control

- Security of an object is at the owner's discretion
- Access is granted through anACL (Access Control List)
- Commonly implemented in commercial products and all client based systems
- Identity Based

MAC

Mandatory Access Control

- **Data owners cannot grant access!**
- **OS makes the decision based on a security label system**
- **Subject's label must dominate the object's label**
- **Users and Data are given a clearance level (confidential, secret, top secret etc)***
- **Rules for access are configured by the security officer and enforced by the OS.**

MAC

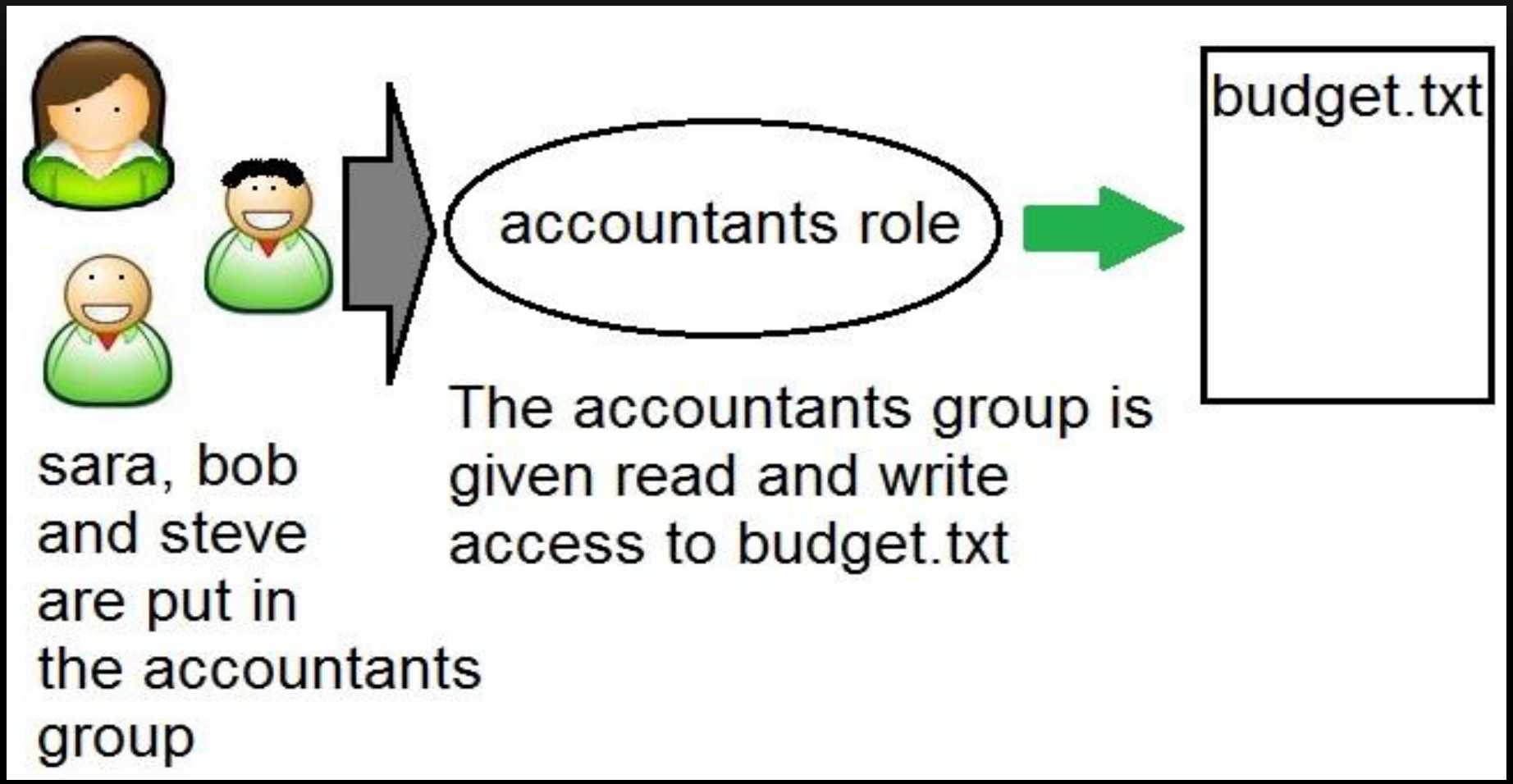
MAC is used where classification and confidentiality is of utmost importance… military.

- **Generally you have to buy a specific MAC system, DAC systems don't do MAC**
  - SELinux
  - Trusted Solaris (now called Solaris with Trusted Extensions)

MAC SENSITIVITY LABELS

- All objects in a MAC system have a security label*
- Security labels can be defined the organization.
- They also have categories to support “need to know” at a certain level.
- Categories can be defined by the organization

ROLE BASED ACCESS CONTROL



ROLE BASED ACCESS CONTROL

- **Uses a set of controls to determine how subjects and objects interact.**
- **Don't give rights to users directly. Instead create "roles" which are given rights. Assign users to roles rather than providing users directly with privileges.**

- **Advantages:**
  - This scales better than DAC methods
  - Fights "authorization creep"*

ROLE BASED ACCESS CONTROL

When to use*

- **If you need centralized access**
- **If you DON'T need MAC**
- **If you have high turnover**

ADVERSARIES

- Script Kiddies
- Hackers
- Elite
- Non-Structured
- Structured
- Highly Structured
- Nation State
- Who's the Target?

SECURITY BASICS REVIEW

- General Security Concepts
  - CIA Triad
  - IAAA of Access Control
- Security Tenets
- Security Architecture
- Security Models
- Access Control Models
- Adversaries

Part I General Security Concepts

RISK MANAGEMENT

RISK MANAGEMENT AGENDA

- Definitions and Terms
- Types of Risk
- Governance and Compliance
- Risk Management Models
- Risk Options

RISK RELATED DEFINITIONS

- **Risk: Likelihood that a threat will exploit a vulnerability in an asset**
- **Threat: Has the potential to harm an asset**
- **Vulnerability: A weakness; a lack of a safeguard**
- **Exploit: Instance of compromise**
- **Controls: Protective mechanisms to secure vulnerabilities**
  - Safeguards: Proactive
  - Countermeasures: Reactive mechanism
- **Secondary Risk: Risk event that comes as a result of another risk response**
- **Residual Risk: The amount of risk left over after a risk response**
- **Fallback Plan: "Plan B"**
- **Workaround: Unplanned Response (for unidentified risk or when other responses don't work**

RISK MANAGEMENT

- Risk Assessment: Identify Assets, Threats, Vulnerabilities
- Risk Analysis: Value of Potential Risks
- Risk Mitigation: Responding to Risk
- Risk Monitoring: Risk is FOREVER!

ASSESSMENT

- **Identify and Valuate Assets**
- **Identify Threats and Vulnerabilities**
- **Methodologies:**
  - OCTAVE: an approach where analysts identify asses and their criticality, identify vulnerabilities and threats and base the protection strategy to reduce risk
  - FRAP: Facilitated Risk Analysis Process. Qualitative analysis used to determine whether or not to proceed with a quantitative analysis. If likelihood or impact is too low, the quantiative analysis if foregone.
  - NIST 800-30: Risk management Guide for Information Technology systems

NIST 800-30

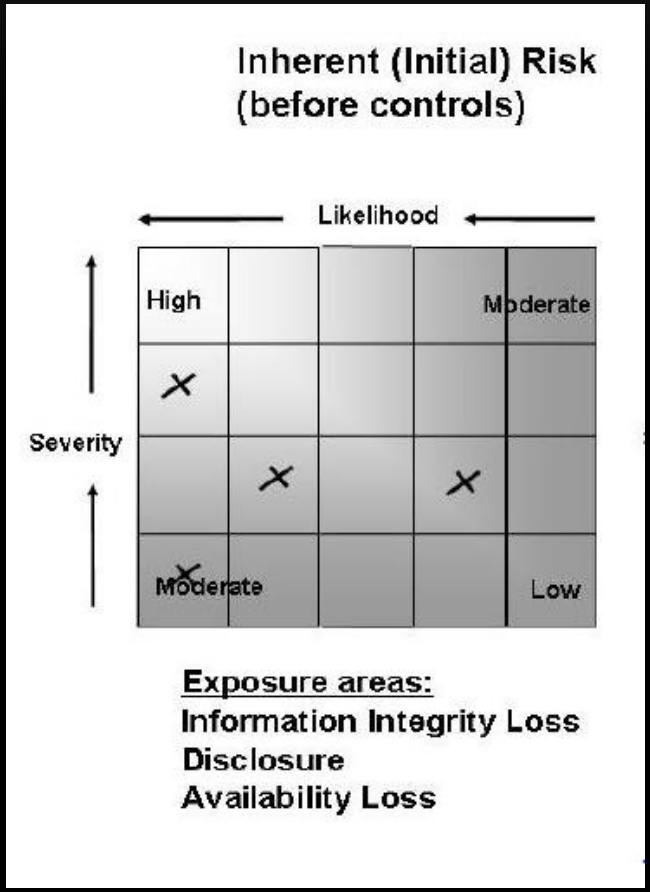
- **9 Step Process:**
  - System characterization
  - Threat identification
  - Vulnerability identification
  - Control analysis
  - Likelihood Determination
  - Impact Analysis
  - Risk Determination
  - Control Recommendations
  - Results Documentation

RISK ANALYSIS

- **Qualitative**
  - Subjective analysis to help prioritize probability and impact of risk events.
  - May use Delphi Technique
- **Quantitative:**
  - Providing a dollar value to a particular risk event.
  - Much more sophisticated in nature, a quantitative analysis if much more difficult and requires a special skill set
  - Business decisions are made on a quantitative analysis
  - Can't exist on its own. Quantitative analysis depends on qualitative information

QUALITATIVE ANALYSIS

- Subjective in Nature
- Uses words like “high” “medium” “low” to describe likelihood and severity (or probability and impact) of a threat exposing a vulnerability
- Delphi technique is often used to solicit objective opinions



CYBRARY.IT

QUANTITATIVE ANALYSIS

- More experience required than with Qualitative
- Involves calculations to determine a dollar value associated with each risk event
- Business Decisions are made on this type of analysis
- Goal is to the dollar value of a risk and use that amount to determine what the best control is for a particular asset
- Necessary for a cost/benefit analysis

QUANTITATIVE ANALYSIS FORMULAS AND DEFINITIONS

- (AV) Asset Value: Dollar figure that represents what the asset is worth to the organization
- (EF) Exposure Factor: The percentage of loss that is expected to result in the manifestation of a particular risk event.
- (SLE) Single Loss Expectancy: Dollar figure that represents the cost of a single occurrence of a threat instance
- (ARO) Annual Rate of Occurrence: How often the threat is expected to materialize
- (ALE) Annual Loss Expectancy: Cost per year as a result of the threat
- (TCO) Total Cost of Ownership is the total cost of implementing a safeguard. Often in addition to initial costs, there are ongoing maintenance fees as well.
- (ROI) Return on Investment: Amount of money saved by implementation of a safeguard. Sometimes referred to as the value of the safeguard/control.

QUANTITATIVE ANALYSIS FORMULAS AND DEFINITIONS CONTINUED

- SLE = AV * EF
- ALE = SLE * ARO

TCO = Initial Cost of Control + Yearly fees

Return on Investment:

ALE (before implementing control)

– ALE (after implementing control)

– cost of control

= ROI (Value of Control)

RISK MITIGATION

- Quantitative Analysis leads to the proper risk Mitigation strategy.
- Reduce
- Accept
- Transfer
- Avoidance
- Rejection

ADDITIONAL RISK TERMS

- **Total Risk: The risk that exists before any control is implemented**
- **Residual Risk: Leftover risk after applying a control**
- **Secondary Risk: When one risk response triggers another risk event**

Risk Calculations

Threats * Vulnerability * Asset Value = Total Risk

Total Risk * Controls Gap = Residual Risk

CYBRARY.IT

RISK MANAGEMENT PROCESS REVIEW

- **Risk Assessment**
  - usually the most difficult to accomplish
  - Many unknowns
  - Necessary effort of gathering the right data
- **Risk Analysis:**
  - can be done qualitatively and/or quantitatively
- **Risk Mitigation**
  - Take steps to reduce risk to acceptable level
- **Maintain that risk level**

*****Remember - Risk must be managed, since it cannot be totally eliminated**

RISK MANAGEMENT REVIEW

- Definitions and Terms
- Types of Risk
- Governance and Compliance
- Risk Management Models
- Risk Options

Part I General Security Concepts

SECURITY POLICIES AND REGULATIONS

SECURITY POLICIES AND REGULATIONS AGENDA

- Definitions
- FISMA
- Legislation
- PCI-DSS
- PII
- Intellectual Property
- Organizations That Promote Standards
- Federal Computers
  - FIPS
  - NIST
- Secure Architecture

ADMINISTRATIVE CONTROLS

- **Policies**
  - High level statement from senior/executive management. Usually driven by laws, industry standards, liability considerations, or other business objectives
- **Standards**
  - Define the specifics of policy
- **Procedures**
  - Step-by-step instructions—"how to"
- **Guidelines**
  - Suggested best practices

FISMA

- Federal Information Security Management Act
- Each federal agency must implement an agency-wide information security management program.
- NIST was designated as the agency to design guides for implementation
- NIST published RMF for the purpose of compliance

SARBANES-OXLEY

- Designed as a response to the corporate misdeeds of the Nineties (Enron, World Com, Arthur Anderson, etc)
- Emphasis is on corporate accountability through internal controls and audits
- Stresses the need to control the integrity of final information so that confidence can be maintained

GRAMM-LEACH-BLILEY

- Governs the collection and disclosure of Personal Financial Information (PFI)
- Covers the design, implementation, and maintenance of the safeguards to protect PFI
- Prohibits the use of pretexting to gain PFI data

HIPAA AND HITECH

- **Healthcare Insurance Portability and Accountability Act**
- **Emphasizes the need for privacy on Personal Healthcare Information (PHI)**
  - This information is frequently sought after by cybercriminals as it contains insurance information and payment information
  - Contains enough PII for identity theft
- **HITECH (Health Information Technology for Economic and Clinical Health Act**

PCI DSS (PAYMENT CARD INDUSTRY DATA SECURITY STANDARD)

- Not a legal mandate
- Payment Card Industry self-regulates its own security standards
- Applies to any business worldwide that transmits, processes or stores payment card transactions to conduct business with customers
- Compliance is enforced by the payment card vendor (Visa, MasterCard, American Express, etc)
- Compliance requirements are dictated by the number of transactions, as well as any previous security issues

PCI DSS (PAYMENT CARD INDUSTRY DATA SECURITY STANDARD) CONTINUED

- **Six Core Principles:**
  - Build and maintain a secure network
  - Protect card holder data
  - Maintain a vulnerability management program
  - Implement strong access control measures
  - Regularly monitor and test the networks
  - Maintain an Information security policy

SAFE HARBOR PRINCIPLES

- Notice: Customers must be informed of what PII is collected and how it will be used
- Choice: Customers must be able to opt out
- Onward Transfer: Transfer to 3rd parties is permissible only when sufficient controls are in place
- Security: Reasonable efforts must be in place to prevent loss of information
- Integrity: Data must be reliable and relevant for the purpose for which it was collected
- Access: Customers have to be able to access information about them and have a means of correcting or deleting if the info is inaccurate
- Enforcement: There must be effective means of enforcing these ru

PERSONALLY IDENTIFIABLE INFORMATION

- Any information that can lead to locating and contacting an individual and identifying that individual uniquely
- Full Name, Mother's Maiden Name
- Social Security Number
- Address, Phone number
- Vehicle Registration Number
- Biometrics
- Other uniquely identifying characteristics

PERSONALLY IDENTIFIABLE INFORMATION

Distinguishability	Look for data that by itself can identify a unique individual.
Aggregation	Look for two or more pieces of data that when combined can identify a unique individual.
How PII is stored, transmitted, used	• Frequently transmitted over networks • Stored redundantly on servers or portable devices • Used by many people in the organization
Compliance	Your organization must comply with regulations and standards for protecting PII. Which ones will depend where you are based and scope of work. However these may include: • **Payment Card Industry Data Security Standards** (PCI DSS) (International) – setting out requirements for data security when handling card payments • **Data Directive** (EU) – requiring the safe storage using data loss prevention technology of data generated in connection with public electronic communication • **HIPAA and HITECH ACT** (U.S.) – enabling fines of up to $1.5 million per year for a breach of healthcare records • **Criminal Justice and Immigration Act** (UK) – giving the Information Commissioner power to levy fines of up to £500,000 for data breaches There are also a large number of data security regulations applicable at regional or state level. If you work in a geography covered by such legislation you should understand the implications for your organization.
Ease of access	Decide if the PII: • Is easily accessed by any employee • Can be copied, sent and saved without restriction • Is available for use by HR for employee management or by staff • Is not protected by PINs or passwords before being accessible by staff

INTELLECTUAL PROPERTY

- **Intellectual Property Law**
  - Protecting products of the mind
  - Company must take steps to protect resources covered by these laws or these laws may not protect them
- **Main international organization run by the UN is the World Intellectual Property Organization (WIPO)**
- **Licensing is the most prevalent violation, followed by plagiarism, piracy and corporate espionage**

ISO 27000 SERIES

- Provides a common lexicon and approach to Information Security
- 27001: Specifies the requirements for all elements of an ISMS including formulation of requirements in alignment with business goals, selection of controls, ongoing monitoring and communications
- 27002: Provides best practices for the oversight of an ISMS (Information Security Management System)
- How to apply Deming's PDCA (Plan-Do-Check-Act model)
- 27005: Addresses information security risk management
- 27006: Requirements for audit and certification of an ISMS

COMMON CRITERIA ISO 15408

Common Criteria (CC)

- Protection Profile: Requirements from customer
- Target of evaluation: System from vendor designed to meet the requirements of the Protection Profile
- Security target: Documentation from vendor describing how the ToE meets the Protection Profile
- Evaluation Assurance Level (EAL 1-7) Assigned by auditor
- Evaluation packages: Additional add-ons

COMMON CRITERIA EAL RATINGS

- EAL 1 – Functionally tested
- EAL 2 – Structurally tested
- EAL 3 – Methodically tested and checked
- EAL 4 – Methodically designed, tested, and reviewed
- EAL 5 – Semi formally designed and tested
- EAL 6 – Semi-formally verified designed and tested
- EAL 7 – Formally verified designed and tested

SEI-CMMI (SOFTWARE ENGINEERING INSTITUTE – CAPABILITY MATURITY MODEL INTEGRATED)

- **Developed by the Software Engineering Institute of The Carnegie Mellon University in Pittsburgh**
- **Describes the procedures, principles, and practices in better software development processes. Has five maturity models:**
  - Initial
    - Development based on Ad Hoc effort. No procedures in place and there is no assurance of consistency; thereby affecting software quality.
  - Repeatable
    - A formal structure has been developed including quality assurance. However, no formal process models have been defined.
  - Defined
    - Formal procedures and defined processes have been put in place for projects.
  - Managed
    - Formal processes have been put in place to allow for qualitative data analysis. Metrics are defined for process improvement. Quantitative understanding of quality
  - Optimized
    - Integrated plans for continuous process improvement

OASIS

- Organization for the Advancement of Structured Information Standards
  - Application Vulnerability Description Language (AVDL)
  - Security Assertion Markup Language (SAML)
  - Extensible Access Control Markup Language (XACML)
  - Key Management Interoperability Protocol (KMIP) Specification
  - Universal Description, Discovery and Integration (UDDI)
  - Web Services

OWASP

- Worldwide free/open community with a focus on web-based application security
- Publishes a top-ten list of application security risks
- Publishes numerous guides on secure practices
  - Development guides
  - Code Review guides
  - Testing guides

ITIL

- **Information Technology Infrastructure Library (ITIL) is the de facto standard for best practices for IT service managmenet**
- **5 Service Management Publications:**
  - Strategy
  - Design
  - Transition
  - Operation
  - Continual Improvement

****While the Publications of ITIL are not testable, it's purpose and comprehensive approach are testable. It provides best practices for organization and the means in which to implement those practices**

NIST STANDARDS

- **800-12 An Introduction to Computer Security: Broad overview of elements of secure computing (HW, SW, Information)**
- **800-14 Generally Accepted Principles and Practices for Security of IT Systems**
- **800-27 Engineering Principles for Information Technology Security**
  - Let security be the foundation of the design
  - Reduce risk to an acceptable level
  - Strive for simplicity
  - Use open design and standards when possible

NIST STANDARDS CONTINUED

- **800-30 Risk Management Guide for IT**
  - Critical success factors for risk management programs
  - Integrates risk management into development process
  - Cost/benefit Analysis
  - Residual risk evaluation
  - Risk mitigation options

NIST STANDARDS CONTINUED

- 800-61 Computer Security Incident Handling Guide
- 800-64 Security Considerations in the Information Systems Development Life Cycle
  - Maximize ROSI (Return on Security Investment)
  - ID security vulnerabilities early
  - Exam design issues if security requirements change
  - ID shared security services to reduce duplication of effort
  - Manage risk and mitigation strategies (Reduce, Accept, Transfer
- 800-100 Information Security Handbook: A Guide for Managers

FISMA (FEDERAL INFORMATION SECURITY MANAGEMENT ACT)

- Federal law that requires every federal agency (and their contractors) to implement an agency-wide Information Security program including
- Inventory of systems
- Categorize information and systems according to risk level
- Security controls
- Certification and accreditation of systems (including risk assessment and system security plans)
- Training
- All accredited systems are supposed to have a set of monitored security controls to provide a level of continuous monitoring

FIPS (FEDERAL INFORMATION PROCESSING STANDARDS)

- **FIPS 199: Standards for Security Categorization of Federal Information Systems**
- **FIPS 200: Minimum Security Requirements for Federal Information and Information Systems**
- **FIPS 197: Advanced Encryption System**
- **FIPS 186-3: Secure Hash Standard**
- **FIPS 190-4: Security Requirements for Cryptographic Modules**
- **FIPS 140 Series: A Profile for US Federal Cryptographic Key Management Systems (CKMS)**

SECURITY POLICIES AND REGULATIONS: AGENDA

- Definitions
- FISMA
- Legislation
- PCI-DSS
- PII
- Intellectual Property
- Organizations That Promote Standards
- Federal Computers
  - FIPS
  - NIST
- Secure Architecture

PART II

Secure Software Requirements

REQUIREMENT GATHERING REQUIREMENTS: AGENDA

- **SMART Requirements**
- **Types of Requirements**
  - Core Security
  - General
  - Operational
  - Other
- **Information Gathering Techniques**
  - Brainstorming, Facilitated Workshops, Surveys, Questionnaires
  - Policy Composition/Requirements Traceability
  - PNE (Protection Needs Elicitation)
  - Use and Misuse Modeling
- **Data Classification**

SMART REQUIREMENTS

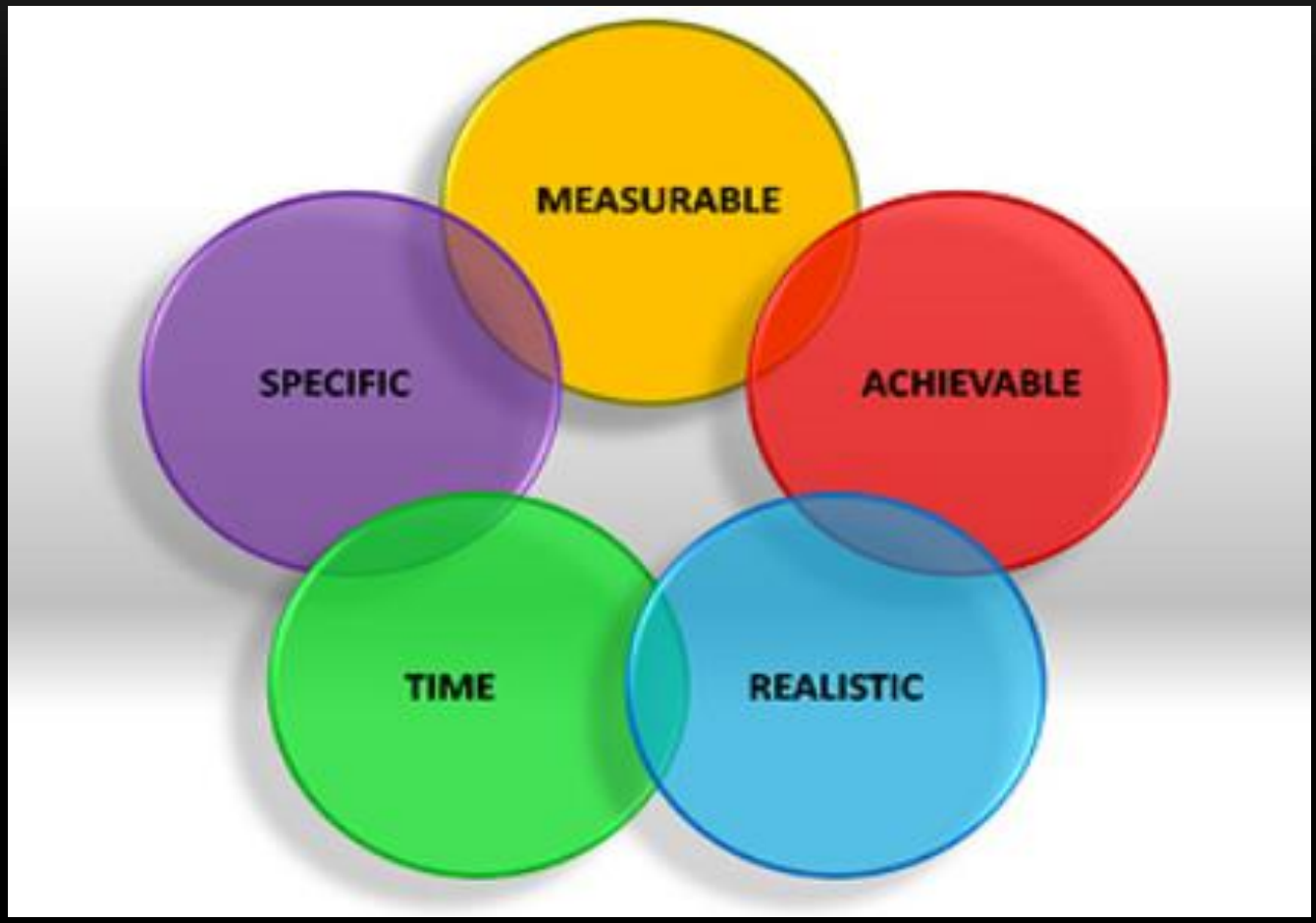


Image Source: http://www.smiletemplates.com/powerpoint-diagrams-charts/smart-objectives/02485/

SPECIFIC

- Specific: Non-generic, not open to misinterpretation
- Weak Requirement: All important sales data should be included on the monthly report
  - Avoid words like all, never, always, and other similar adjectives.
  - What if what you consider important doesn't match what the customer considers important?
- Strong Requirement: The monthly report shall contain the following fields: Total Sales, Avg Retail Price, Cost of Product, Total Sold, Remaining inventory
  - This leaves little room for interpretation of what will be covered

MEASUREABLE

- **What are the critical success factors that must be achieved? How will I know when I have achieved them?**
- **Be wary of undertaking any project with requirements that cannot be verified as complete.**
- **Weak requirement: The application will improve customer service**
  - How much of an improvement will be expected? How will that be monitored, tracked and verified?
- **Strong requirement: The application will improve customer satisfaction as measured by a 2% decrease in hold times and an improvement in customer service feedback scores no less that .5%.**

ATTAINABLE

- Also referred to as achievable, actionable, or appropriate
- Ensure that the requirement is physically able to be achieved given existing circumstances.
- Weak Requirement: The completion of printing and the shipping of books will take place on the first day of each month
  - After books are printed, there may be a verification process that takes two days to complete
- Strong Requirement: The completion of printing will be completed on the first of the month. The shipping must be completed no later than the 5th day of each month

REALISTIC

- Makes sure that the requirement is realistic to deliver when considering other constraints of the project and requirements
- Weak requirement: A customer may request that all work be completed by April 15th, however, based on other constraints/risks associated with the project, this may not be a reasonable requirement
- Strong Requirement: Work will be completed by April 15th, assuming that the submitted budget is approved, that resources will be available as documented and that project team members will be devoted to the project and not removed to perform work on other projects

TIMELY

- Should be time bound, if possible
- Weak requirement: Work should be completed as soon as possible
- Strong requirement: Work should be completed by June 1st.

CORE SECURITY REQUIREMENTS: CONFIDENTIALITY

- Overt: Cryptography and Masking
- Covert: Steganography
- States of Data
    - At rest
    - In process
    - In transit
- Examples of Confidentiality Requirements:
    - PII Must be protected against disclosure using approved algorithms
    - Password and sensitive fields should be masked
    - Passwords at rest must not be stored in clear text
    - TLS or SSL must be used for all transmittal of sensitive information
    - The use of unsecure transmission protocols (like FTP, etc) shall not be allowed
    - Log files shall not store sensitive information

CORE SECURITY REQUIREMENTS: INTEGRITY

- **System Integrity: Protection against system or software modification: System should perform as expected**
  - Code injection can modify the database
  - Input validation is a mitigation technique
- **Data Integrity: Ensuring the accuracy and reliability of data**
  - CRCs, Checksums, Message Digests, Hashes, MACs
  - Internal and External Consistency
- **Some examples of Integrity Requirements:**
  - Input Validation should be used in all forms to ensure that data control language is not entered, and field size and data types are enforced
  - Published software should provide the user with a message digest so the user can validate the accuracy and completeness of the software
  - Subjects should be prevented from modifying data, unless explicitly allowed

CORE SECURITY REQUIREMENTS: AVAILABILITY

- Providing Timely Access to Resources
- Metrics Used:
  - MTD/RTO/RPO
  - SLAs
  - MTBF/MTTR
- Examples of Availability Requirements:
  - Software shall meet availability requirements of 99.999%, as specified in the SLA
  - Software should support access to up to 200 users simultaneously
  - Software must support replication and provide load balancing
  - Mission critical functionality of the software should be restored to normal operations within 30 minutes

CORE SECURITY REQUIREMENTS: AUTHENTICITY

- Validation of an entity's identity claim
  - Anonymous: public access
  - Basic: User supplied password transmitted in the clear
  - Digest: Challenge/Response
  - Integrated: Directory Services authentication
  - Certificate-based: X.509 v4 certificates used
  - Forms: An internet form prompts user to enter credentials which are validated
  - Token-based: Allows SSO
  - Smart Cards: Requires EAP and is often integrated with a PKI
  - Biometric Authentication Credentials inherently bound to a subject
    - Static vs. Dynamic
    - FAR, FRR, CER

CORE SECURITY REQUIREMENTS: AUTHENTICITY

- Examples of Authenticity Requirements
  - User must provide authentication information at login, but shall not have to provide this information for subsequent access to intranet resources
  - For access to financially sensitive information, subjects shall be required to authenticate via a smart card and a PIN
  - Internal and External users should be able to access the software
  - Mutual Authentication will be supported through the use of certificates

CORE SECURITY REQUIREMENTS: AUTHORIZATION

- Confirms that an authenticated entity has the privileges and permissions necessary
- CRUD Operations (Create, Read, Update, Delete)
- Access Control Models
  - DAC: Discretionary Access Control
  - MAC: Mandatory Access Control
  - RBAC: Role Based Access Control
  - RuBAC: Rule Based Access Control

CORE SECURITY REQUIREMENTS: AUTHORIZATION

- **Examples of Authorization Requirements**
  - Access to highly sensitive information will be restricted to users with Secret or Top Secret clearance
  - Unauthenticated users will only have read permission to public access pages
  - Only those with administrative credentials will be able to modify files

CORE SECURITY REQUIREMENTS: ACCOUNTABILITY

- **Tracing an action to a subject--also known as auditing**
- **Must include the following:**
  - Identity of subject
  - The Action
  - Object on which the action was performed
  - Timestamp
  - Examples of Accountability Requirements:
    - All failed logon attempts will be logged with Timestamp and source IP address
    - Audit logs should not overwrite previous events. They should append to previous entry and alert admin when space becomes limited
    - Audit logs must be retained for one year.

CORE SECURITY REQUIREMENTS: AUTHORIZATION

- **Examples of Authorization Requirements**
  - Access to highly sensitive information will be restricted to users with Secret or Top Secret clearance
  - Unauthenticated users will only have read permission to public access pages
  - Only those with administrative credentials will be able to modify files

GENERAL REQUIREMENTS

- **Session Management: Sessions allow state tracking and keep users from having to re-authenticate for each access**
  - Each user activity will need to be uniquely tracked
  - Sessions must be terminated when a user logs off or closes browser window
  - Session ID and related information must be encrypted
- **Error & Exception Management: Can potentially disclose information about software design/architecture**
  - Error message visible to end users must reveal only information necessary without revealing internal system info
  - Security exception details must be audited and monitored periodically
- **Configuration Parameters: Typical configuration items include initialization parameters, connection strings, keys, and other associated variables.**
  - Web application's configuration files must encrypt sensitive data
  - Passwords should not be hard-coded in line code
  - Initialization and disposal of global variables must be carefully monitored

OPERATIONAL REQUIREMENTS

- **Deployment Environment**
  - Software will be deployed on internal intranet only
  - Software will use a proprietary protocol on port 2249
- **Archiving**
  - Data collected will be archived automatically at 3 months
  - Archives will be accessed through company's SAN
  - Archives will be retained for 7 years
- **Anti-piracy**
  - Software must be digitally signed to against tampering
  - License keys must not be hard-coded in the software
  - Dynamic license verification checking should be supported

OTHER REQUIREMENTS

- **Sequencing and Timing Requirements**
  - Race Conditions should be prohibited
  - Infinite loops that keep a program from returning to the normal flow of logic
- **International Requirements**
  - To support countries with import restrictions, software must be backwards compatible to support 40-bit encryption
  - Language inputs of English, French, Spanish must be supported
- **Procurement Requirements**
  - Contracts and SLAs should be well-written to encourage vendor compliance.
  - Right-to-audit may be necessary to provide accountability for the vendor

INFORMATION GATHERING TECHNIQUES

- **Brainstorming**
- **Affinity Mapping**
- **Facilitated Workshops**
- **Surveys and Questionnaires**
  - Delphi Technique
- **Policy Decomposition**
- **Requirements Traceability**
- **PNE**
- **Use and Misuse Case**

INTRODUCTION

- **Policy decomposition involves mapping high level policies or goals to more specific "workable" requirements**

Image Source: https://blog.nvisium.com/2014/05/a-more-secure-development-lifecycle-iii.html

CYBRARY.IT

REQUIREMENTS TRACEABILITY MATRIX

Business Requirement	Funtional Requirement	Testing Requirement	Security Requirement
Password fields should be protected from shoulder surfing	All password fields will be set as masked fields	Enter passwords in all password fields to verify that they are masked	The programming language should have masked field as a option
Credit card information should be protected in transit	TLS should be used for all transport of credit card information	Verify that all credit card information is tranported using TLS	TLS should be used on all pages using TLS 2.0 or higher
Passwords must not be stored in clear text in the database	Passwords should be stored as a hash value	Verify that all passwords in the database are hashed	

Image Source: https://blog.nvisium.com/2014/05/a-more-secure-development-lifecycle-iii.html

PNE (PROTECTION NEEDS ELICITATION)

- Purpose is to “draw out” the security requirements from the customer
  - 7 procedures used for PNE
  - Approaching/Engaging the Customer
  - Acquiring the IMM (Information Management Model)
  - Identify Least Privilege applications
  - Threat Analysis
  - Prioritize based on needs of the customer
  - Preparing the IPP (Information Protection Profile)
  - Customer Buy-In

USE AND MISUSE CASE

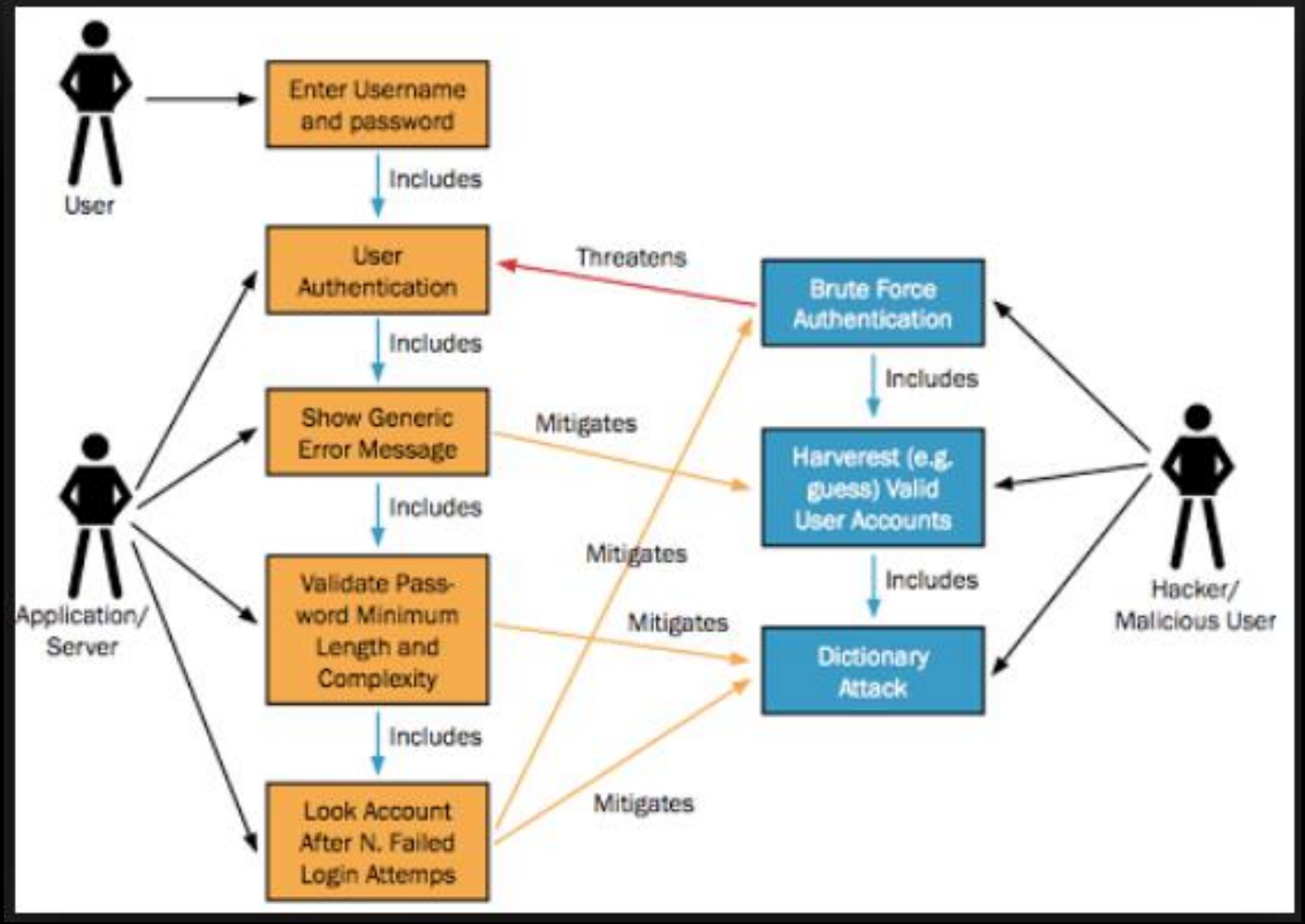


Image source: http://clarotesting.com/page15.htm

DATA CLASSIFICATION

- **Types of data**
  - Structured--databases
  - Unstructured—images, videos
- **Labeling**
  - MAC environments uses labels on subjects and objects
  - Subject's label must dominate the object's label
- **Data Owner determines the classification of the data, and defines the authorized list of users and access criteria**
- **Data Custodian maintains and enforces the controls relevant to the data's classification**

PART III

Secure Software Design

PART III SECURE SOFTWARE DESIGN

- Design Process
- Design Considerations
- Security Common Used Architecture
- Secure Technologies

Part III Secure Software Design

DESIGN PROCESSES

DESIGN PROCESSES

- Reduce the Attack Surface
- Threat Modeling
- Risks in Design
- Controls Evaluation

REDUCING THE ATTACK SURFACE

- **Evaluate the attack surface of the product**
  - User Input Fields
  - Protocols/Services/Interfaces/Processes
  - Resource files
  - Open named pipes/open sockets
  - How many items are accessible
  - Dynamic web pages (ASP, etc)
  - Guest accounts enabled
  - ACL configuration

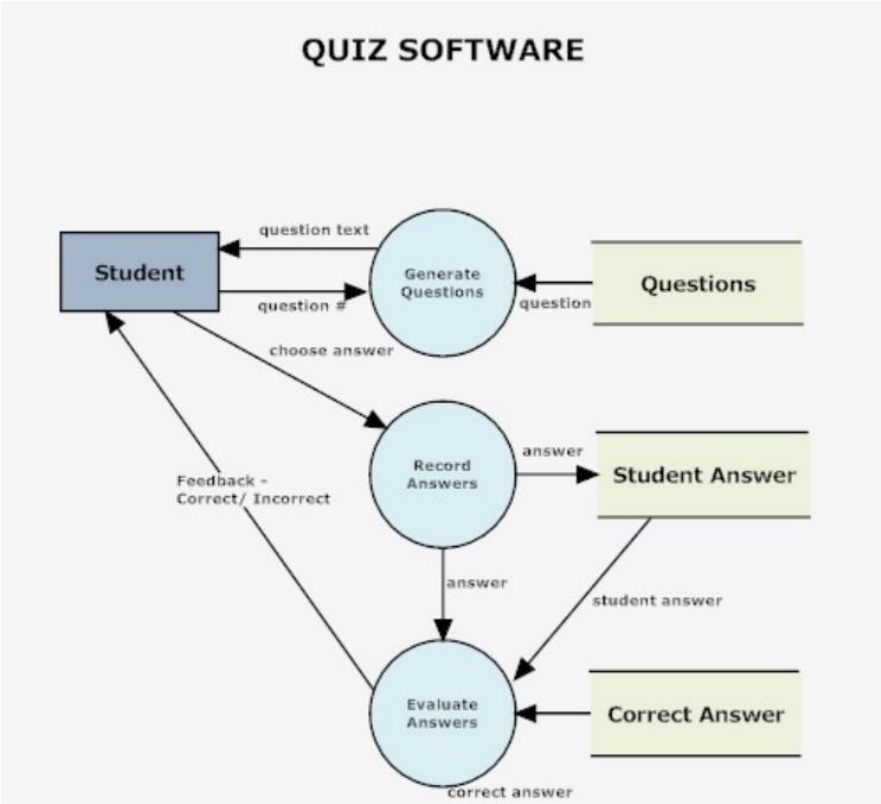
THREAT MODELING

- **Identify Security Objectives**
  - Legislative Drivers
  - Contractual Requirements
  - Alignment with Business Objectives
- **CIA Triad**
- **Tools for Threat Modeling**
  - Data Flow Diagrams
  - Use/Misuse Cases

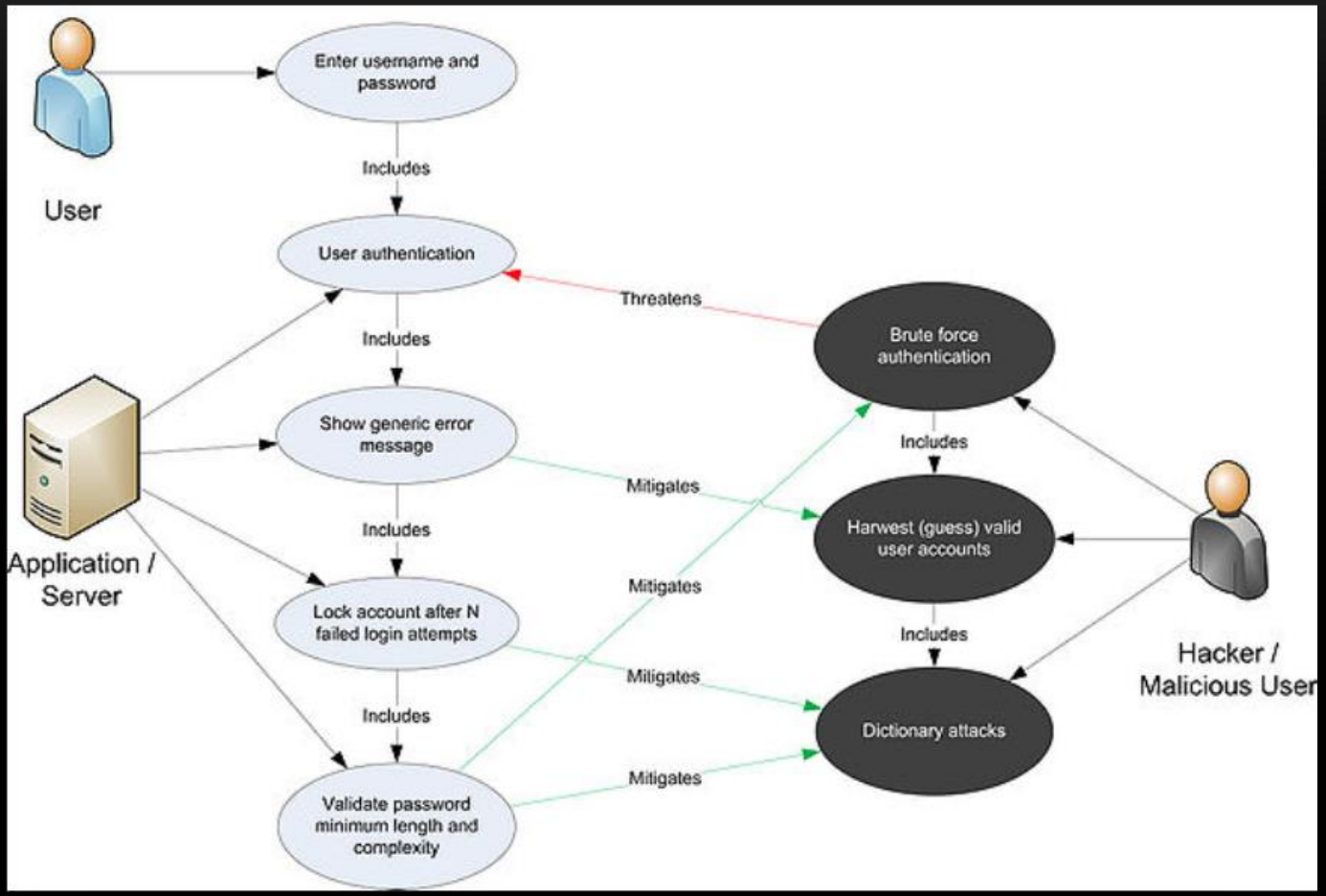
THREAT MODELING: DATA FLOW DIAGRAMS

Blue circles are data transformations (processing). Arrows are data flows. The "Student database" is a data store

http://www.smartdraw.com/examples/preview/index.aspx?example=Quiz_Software



USE/MISUSE CASES



https://www.owasp.org/index.php/Application_Threat_Modeling

CYBRARY.IT

THREAT MODELING: STRIDE

Threat	Mitigation
Spoofing	Authentication
Tampering	Integrity Verification (Message Digests/CRCs)
Repudiation	Non-Repudiation (Digital Signatures, Keys)
Information Disclosure	Confidentiality Through Encryption
Denial of Service	High Availability/Redundancy/Fault Tolerance
Escalation of Privilege	Authorization

RISKS IN DESIGN

- **Code Reuse**
- **Flaws vs. Bugs**
  - Flaw: Inherent fault with the design of code
  - Bug: Implementation fault
- **Open vs. Closed Design**

CONTROLS EVALUATION

- Efficacy of Controls
- Economy of Mechanism
- Cost/Benefit Analysis
- Psychological Acceptability

CONSIDERATIONS FOR DESIGN

- **C-I-A**
- **AAA**
- **Secure Design Principles**

SECURE DESIGN PRINCIPLES

Secure Design Principles

The following are secure design principles that are employed to achieve application security:

- Good Enough Security
- Least Privilege
- Defense in Depth
- Separation of Duties
- Fail Safe
- Complete Mediation
- Open Design
- Economy of Mechanism
- Least Common Mechanism
- Psychological Acceptability
- Leverage Existing Components
- Weakest Link
- Single Point of Failure

Part III Secure Software Design

SECURITY AND COMMON ARCHITECTURES

SOFTWARE DEVELOPMENT METHODOLOGIES

- Waterfall
- Prototype
- Spiral
- Agile

WATERFALL

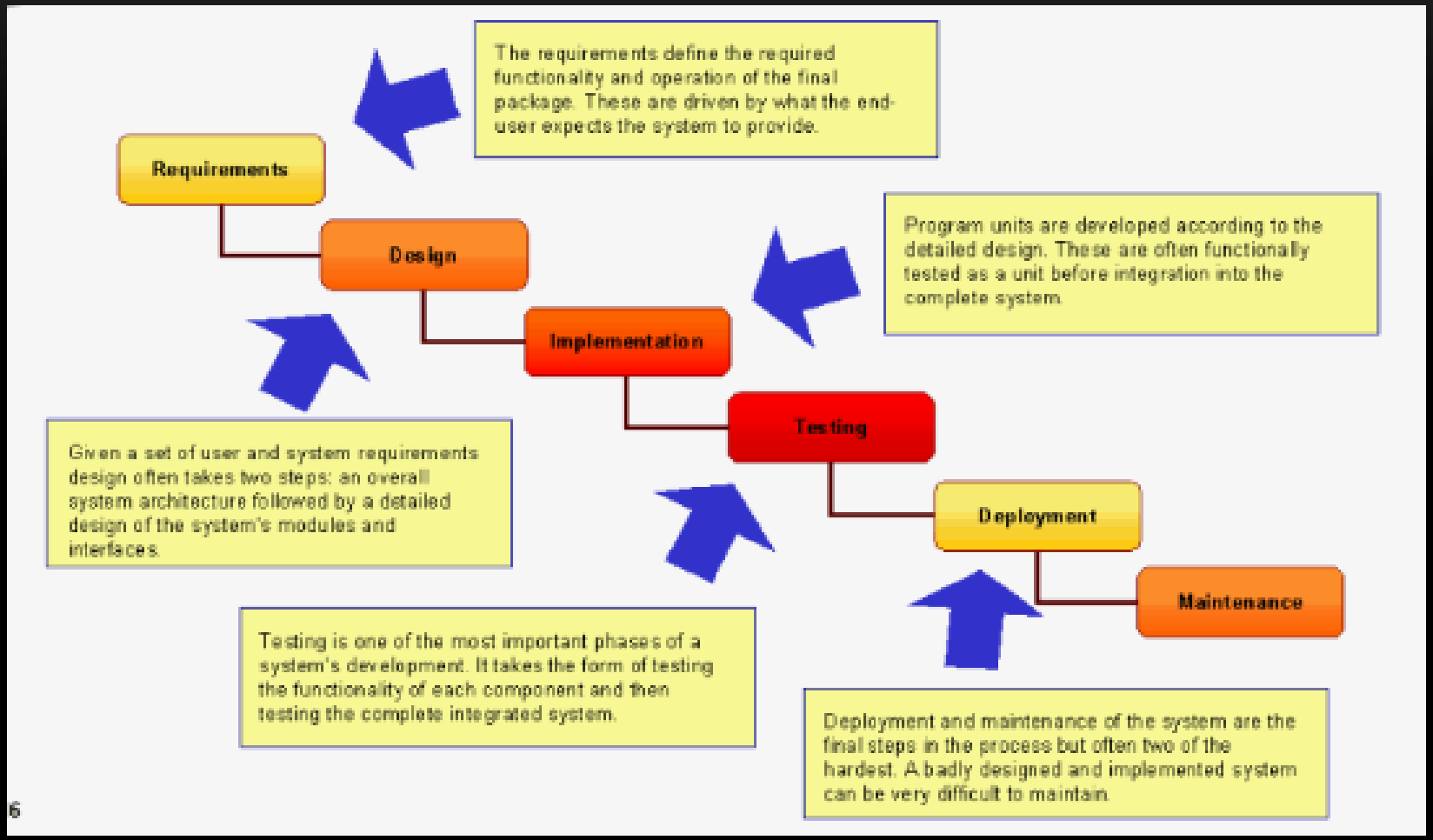


Image Source: http://www.softeng.rl.ac.uk/st/archive/SoftEng/SESP/Presentations/SoftwareEngineeringforCSED/sld016.htm

CYBRARY.IT

PROS AND CONS WITH THE WATERFALL

- Pros
  - Each phase has specific deliverables and a review process.
  - Phases are processed and completed one at a time.
  - Best for small projects where requirements are very well understood.
  - It reinforces "define before design" and "design before code".
- Cons
  - Adjusting scope during the life cycle can kill a project
  - No working software is produced until late during the life cycle.
  - High amounts of risk and uncertainty.
  - Poor model for long and ongoing projects.
  - Poor model if there is a high probability of change
  - The end of the project can be far removed from the beginning in which the initial requirements were specified

PROTOTYPING

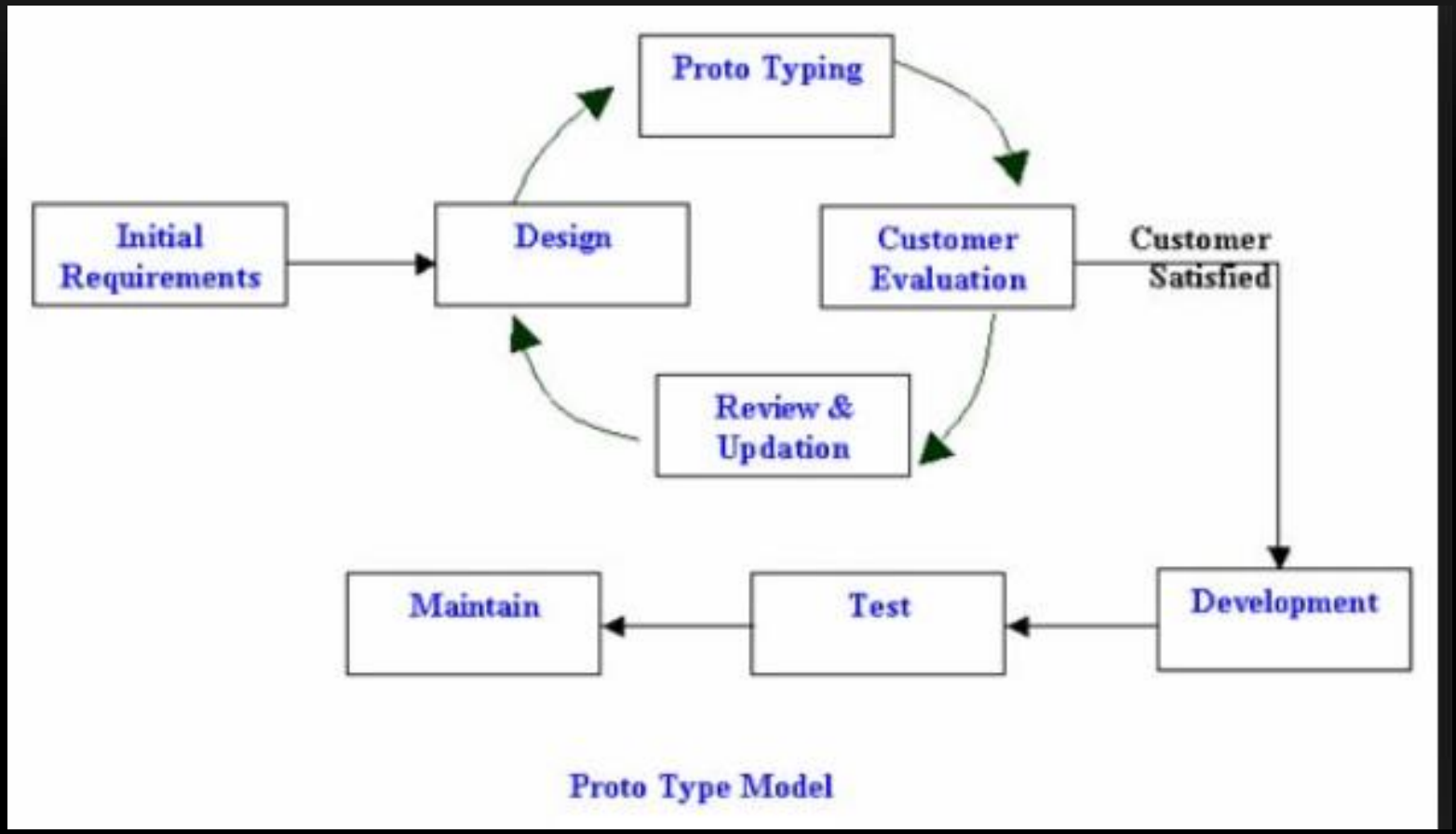
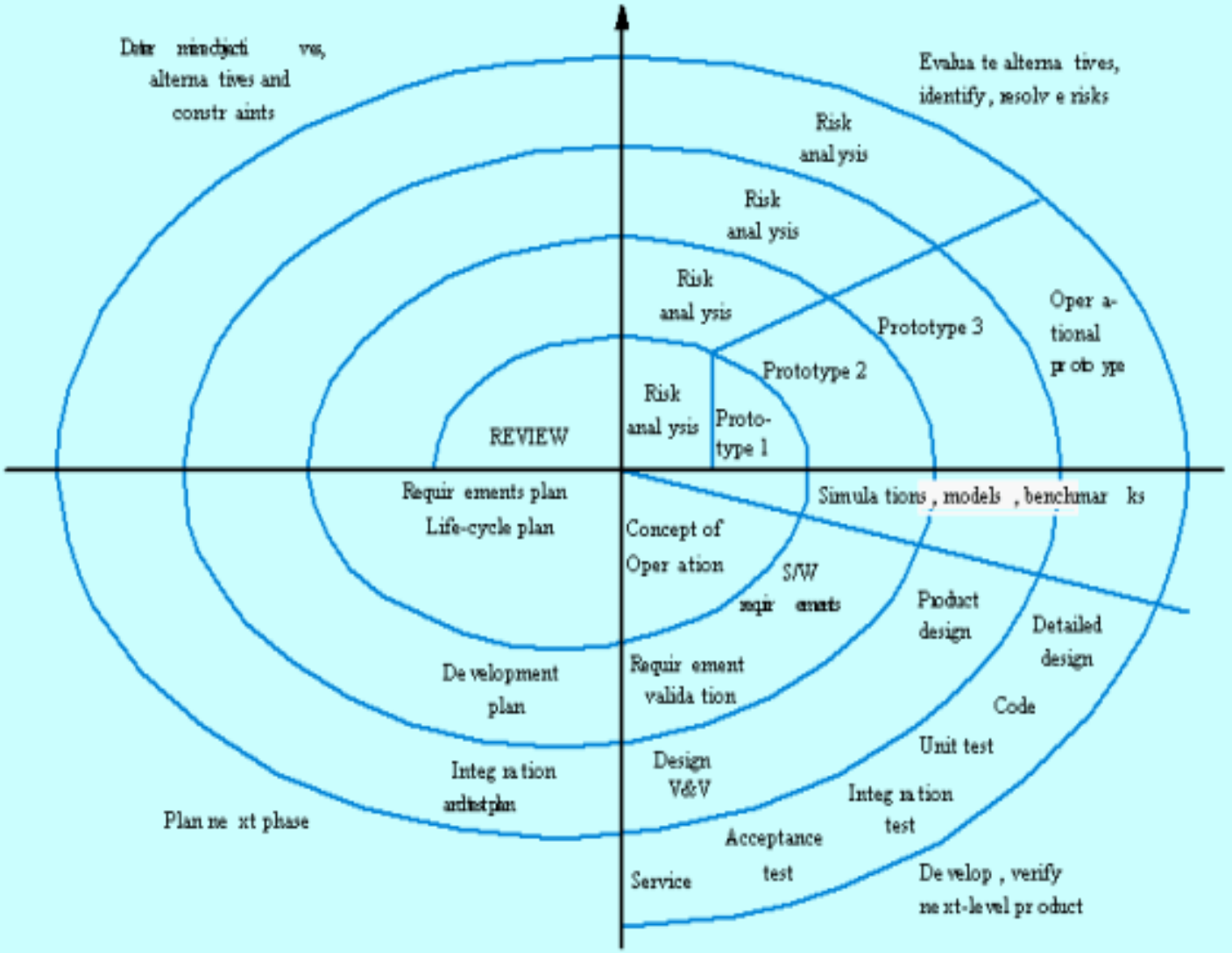


Image Source: https://qastation.wordpress.com/tag/sdlc/page/2/

PROS AND CONS OF PROTOTYPING

- Pros
- The software designer and implementer can obtain feedback from the users early in the project
- The client and the contractor can compare if the software made matches the software specification, according to which the software program is built.
- It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met.
- Cons
  - Clients rarely understand all the ramifications of proposed changes
  - Developers may use shortcuts to create the prototype and sometimes do not formalize their processes for the actual product

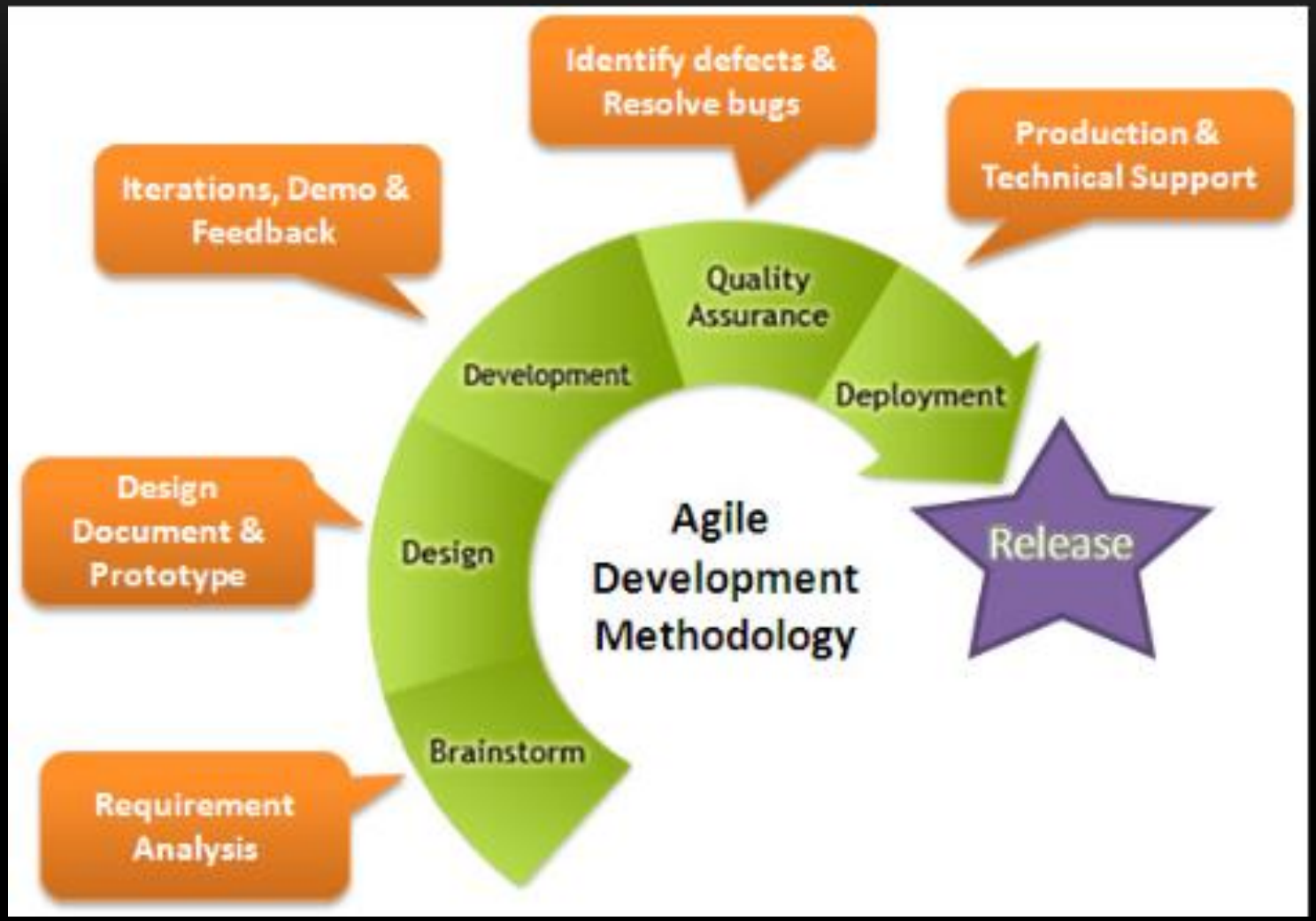
SPIRAL


Determine objectives, alternatives and constraints
Evaluate alternatives, identify, resolve risks
Risk analysis
Risk analysis
Risk analysis
Prototype 3
Operational prototype
Prototype 2
Risk analysis
Prototype 1
REVIEW
Requirements plan
Life-cycle plan
Simulations, models, benchmarks
Concept of Operation
S/W requirements
Product design
Detailed design
Development plan
Requirement validation
Code
Unit test
Integration and test plan
Design V&V
Integration test
Plan next phase
Acceptance test
Service
Develop, verify next-level product

PROS AND CONS OF SPIRAL

- Pros
  - High amount of risk analysis
  - Good for large and mission-critical projects.
  - Software is produced early in the software life cycle
- Cons
  - Can be a costly model to use.
  - Risk analysis requires highly specific expertise.
  - Project's success is highly dependent on the risk analysis phase.
  - Doesn't work well for smaller projects.

AGILE



SECURITY AND COMMON ARCHITECTURES

- Distributed Computing
- Service Oriented Architecture
- Rich Internet Application
- Ubiquitous Computing
- Cloud Architecture

DISTRIBUTED COMPUTING

- **Client-Server**
  - Thin vs. Fat Clients
  - Scalability
  - Availability
  - Maintainability
  - Security
- **Peer-to-Peer (P2P)**
  - Frequently used for file sharing
  - Channel Security –transport protocols
  - Data Confidentiality and Integrity—encryption and hashing
  - Securing the call Stack/Flow—Validation and authorization checks at various points of the call/flow

SERVICE ORIENTED ARCHITECTURE

- SOA is an architecture and a vision on how heterogeneous applications should be developed and integrated in the enterprise.
- Share a formal contract
- Loosely coupled
- Abstraction
- Composable
- Reusable
- Autonomous
- Stateless
- Discoverable

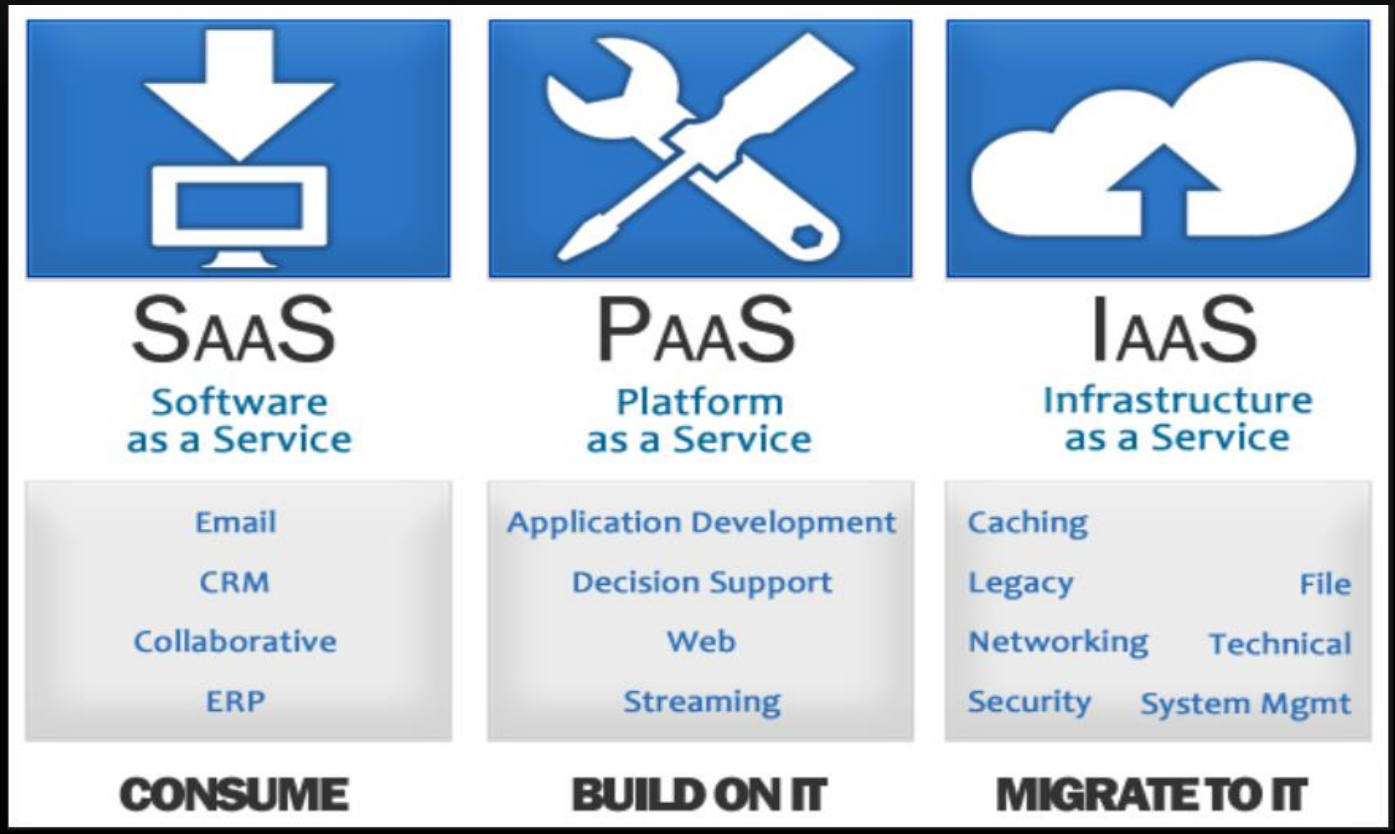
RICH INTERNET APPLICATIONS

- Client Side Threats
  - XSS
  - CSRF
- Server Side Threats
  - Code Injection
    - Validate Input
  - Aggregation and Inference
    - Masking
    - Polyinstatiation

UBIQUITOUS COMPUTING

- Wireless Networking
- RFID (Radio Frequency ID)
- NFC (Near Field Communications)
- LBS (Location Based Services)

CLOUD ARCHITECTURE



https://www.pinterest.com/backboneforbigd/sme/

CYBRARY.IT

Part III Secure Software Design

SECURE TECHNOLOGIES

AUTHENTICATION AND IDENTITY MANAGEMENT

- **Authentication and Identity Management**
  - Identification: Making a claim
  - Authentication allows users to support the claim of their identity
  - Identity and Access Management
    - Services/policies/procedures for managing a digital identity/provisioning
- **Security controls (Including Management) are audited annually under Sarbanes-Oxley (SOX)**

CREDENTIAL MANAGEMENT

- **Exploits**
  - MITM and Traffic Hijacking
  - Unauthorized Access
  - Privilege Escalation
- **Solutions**
  - Certificates
  - Single Sign on

TRAFFIC FLOW CONTROL

- **Proxies**
- **Firewalls**
- **Middleware**
- **Logging**
- **Data Loss Prevention**
  - Exfiltration of Data
- **Virtualization**

TRUSTED COMPUTING

- **Trusted Computing Base (TCB)**
  - Reference Monitor
  - Security Kernel
- **Trusted Platform Module**
- **Secure State Model**
  - Root kits
- **Privilege Management**
- **Database Integrity**

PART IV

Secure Software Coding

PART IV SECURE SOFTWARE CODING AGENDA

- System Architecture
- Common Software Vulnerabilities and Countermeasures
- Defensive Coding Practices/Secure Software Coding

COMPUTER ARCHITECTURE

- Computer Architecture
- Central Processing Unit
- Arithmetic Logic Unit (ALU)
- Control Unit (CU)
- Memory
- Primary storage
- Secondary storage
- Volatile storage
- Nonvolatile storage
- Cache storage

CPU CYCLES

- Fetch
- Decode
- Execute
- Store

FETCH

- Fetching
- The control unit gets the instruction from system memory. The location of each instruction and data in system memory is identified by a unique address and the control unit uses the memory address to get the program instruction. The instruction pointer is used by the processor to keep track of which instruction codes have been processed and which ones are to be processed subsequently. The data pointer keeps track of where the data area in stored in the computer memory, i.e., it points to the memory address.

DECODE

- **The control unit deciphers the instruction and directs the needed data to be moved from system memory onto the ALU.**

EXECUTION

- The ALU and the ALU performs the mathematical or logical operation on the data

STORING

- **The ALU stores the result of the operation in memory or in a register. The control unit finally directs the memory to release the result to an output device or a secondary storage device.**

EXECUTION TYPES

- **Multiprogramming**
- **Multitasking**
  - Cooperative
  - Preemptive (true multitasking)
- **Multithreading**
- **Multiprocessing**
  - Asymmetric
  - Symmetric
- **Multi-core processors**

CPU MODES

- User (Problem State)
- Privileged (Kernel Mode)

MEMORY

- Random-Access Memory (RAM)
  - Dynamic/Static
  - Cache
- Read-Only Memory (ROM)
- Programmable Read-Only Memory
- (PROM)
- Erasable Programmable Read-Only Memory (EPROM)
- Electronically Erasable Programmable Read Only Memory (EEPROM)

WHY IS SOFTWARE UNSECURE?

- Lack of training
- Lack of funding
- No prioritization of security
- Security as an afterthought

VULNERABILITY DATABASES AND RESOURCES

- OWASP (Open Web Application Security Project) Top Ten
- CVE (Common Vulnerabilities and Exposures)
- CWE (Common Weakness Enumeration)
- NVD (National Vulnerability Database)
- US CERT (Computer Emergency Response Team) Vulnerability Database

OWASP (OPEN WEB APPLICATION SECURITY PROJECT) TOP TEN

- **OWASP is an international non-profit organization**
- **OWASP (Open Web Application Security Project) Top Ten**
- **Offers a broad consensus on the most common security flaws/exploits**
- **Designed to raise awareness and the stress the need for security in web-based applications**

https://www.owasp.org/index.php/About_OWASP

OWASP TOP TEN 2013

OWASP Top 10 – 2013 (New)
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013

1. CODE INJECTION

- Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization

2. BROKEN AUTHENTICATION & SESSION MANAGEMENT

- Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities

3. XSS (CROSS SITE SCRIPTING)

- XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites

4. INSECURE DIRECT OBJECT REFERENCES

- Defined as an unauthorized user or process which can invoke the internal functionality of the software by manipulating parameters and other object values that directly reference this functionality. Issues resulting include:
  - Data disclosure
  - Privilege escalation
  - Authentication and authorization checks bypass
  - Restricted resource access

5. SECURITY MISCONFIGURATIONS

- Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date

6. SENSITIVE DATA EXPOSURE

- Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser
- **Primary reasons for sensitive data exposure:**
  - Insufficient data-in-transit protection
  - Insufficient data-at-rest protection and
  - Electronic social engineering

7. MISSING FUNCTION LEVEL ACCESS CONTROL

- Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization
- Failure to restrict access to privileged functionalities or URLs. Web pages that provide administrative functionality are the primary targets for such brute force attacks Mitigation: Role Based Access Control (RBAC) of functions and URLs that denies access by default

8. CROSS SITE REQUEST FORGERY (CSRF)

- A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim

CSRF MITIGATION STRATEGIES

- Do not save username/password in the browser.
- Do not check the “remember me” option in websites.
- Do not use the same browser to surf the Internet and access sensitive websites at the same time, if you are accessing both from the same machine.
- Read standard emails in plain text.
- Explicitly log off after using a web application.
- Use client-side browser extensions that mitigate CSRF attacks.

DEVELOPER STRATEGIES TO MITIGATE CSRF

- Implement the software to use a unique session specific token (called a nonce) that is generated in a random, non-predictable, non-guessable and/or sequential manner.
- CAPTCHAs can be used to establish specific token identifiers per session.
- The uniqueness of session tokens is to be validated on the server side and not be solely dependent on client based validation.
- Use POST methods instead of GET requests for sensitive data transactions and privileged and state change transactions, along with randomized session identifier generation and usage

9. KNOWN VULNERABLE COMPONENT USAGE

- Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts
- Deprecated, insecure and banned APIs

10. NON VALIDATED REDIRECTS AND FORWARDS

- Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages

WHAT IS DEFENSIVE CODING

- Defensive coding is a form of proactive, secure coding intended to ensure the continuing function the software under unforeseen circumstances. Defensive programming techniques are used especially when a piece of software is likely to be misused.
- Benefits of defense coding generally improve:
- General quality - reducing the number of bugs and flaws associated with the software
- Making the source code comprehensible - the source code should be readable and understandable so it is approved in a code review
- Making the software behave in a predictable manner despite unexpected inputs or user actions.

DEFENSIVE CODING

- **Input Validation**
  - Correct data type and format
  - Falls within the expected and allowed range of values
  - Is not interpreted as code when it should not be

DEFENSIVE CODING

- **Sanitization**
- **Convert something that is considered dangerous into its safe form. Both inputs and outputs can be sanitized.**
- **Input Sanitation types:**
  - Stripping: Removing harmful characters from user supplied input
  - Substitution: Replacing user supplied input with safer alternatives

DEFENSIVE CODING

- **Output Sanitization Output sanitization is usually performed by encoding (sometimes referred to as encoding) the data before it is presented to the client.**
- **Methods of Output Sanitation in Web Apps:**
  - HTML entity encoding
  - URL encoding

DEFENSIVE CODING

- **Error Handling The error messages must be:**
  - non verbose and explicitly specified in the software.
  - Use an index of the value or reference map.
  - Redirect errors and exceptions to a custom and default error handling location and depending on the context of where the user has logged in (remote or local), appropriate message detail can be displayed

SAFE APIs

- Identify APIs as potential entry points.
- Avoid banned and deprecated APIs that are susceptible to security breaches
- Use proper authentication and audit all API usage

CONCURRENCY

- **Defined as simultaneous operations**
- **Avoid race windows**
  - Fix in code or logic before coding
- **Atomic operations**
  - Single threaded operation
- **Mutual Exclusion (Mutex)**
  - Resource Locking which also provides integrity

TOKENIZING

- **The process of replacing sensitive data with unique identification symbols that still retain the needed information about the data, without compromising its security.**
- **Usually done to support external standards and compliance requirements**
  - PCI-DSS
  - PII

SANDBOXING

- The security mechanism that prevents software running on a system from accessing the host operating system.
- Creates a separation from the host operating system so that untested, untrusted and unverified code and programs, especially those that are published by third parties can be run.
- Principle of Least Privilege in action

ANTI-TAMPERING

- **Integrity Criteria**
- **Techniques include**
  - Obfuscation
  - Protection against reverse engineering
  - Code signing – most common in web-based code snippets such as Java Applets and Active-X components.

SECURE PROCESSES FOR SOFTWARE

- Version Control
- Code analysis
- Code/Peer review

VERSION CONTROL

- Provides that:
- Correct version of code is used
- Rollback capabilities are available
- Track ownership of code
- Track Changes to code

CODE ANALYSIS

- Inspect code for quality and
- weaknesses
- 2 types:
  - Static code analysis involves the inspection of the code without executing the code (or software program).
  - Dynamic code analysis is the inspection of the code when it is being executed (run as a program).

CODE REVIEW

- **Code Reviews**
- **It is a systematic evaluation of the source code with the goal of finding out syntax issues and weaknesses in the code that can impact the performance and security of the software.**
- **Inspect for:**
  - Insecure code
  - Inefficient code

WHAT TO LOOK FOR IN CODE REVIEWS

- Injection Flaws
- Non-Repudiation Mechanisms
- Spoofing Attacks
- Errors and Exception Handling
- Cryptographic Strength
- Unsafe & Unused Functions
- Reversible Code
- Privilege Code
- Maintenance Hooks
- Logic Bombs
- Timing & Synchronization
- Implementations
- Cyclomatic Complexity

SECURE BUILDS

- Physically securing access to the systems that build code.
- Using access control lists (ACLs) that prevent access to unauthorized users.
- Using version control software to assure that the code built is of the right version.
- Build automation is the process of scripting or automating the tasks that are involved in the build process.

Part V Secure Software Testing

SECURE SOFTWARE TESTING

- Quality Assurance
- Testing Artifacts
- Types of Testing
- Impact Assessment and Corrective Action

QUALITY ASSURANCE

- **QA of software tests:**
  - Reliability: Does the software function as expected?
  - Recoverability: Can the software restore itself to a functioning state after downtime (accidental or intentional)
  - Resiliency: Can the software withstand attacks?
  - Interoperability: Can the software function in disparate environments
  - Privacy: Are the various forms of PII, PHI, and PFI protected appropriately?

TESTING ARTIFACTS

- In software development life cycle (SDLC), "**artifacts"** refers to elements that are produced by people involved in the process.
- Test Strategy
- Test Plan
- Test Case
- Test Script
- Test Suite
- Test Harness

TEST STRATEGY

- Outlines the testing approach that will be undertaken
- Main instrument that is used to communicate issues with testing to the software development team and other members of the project
- Includes the testing goals, methods, time requirements, environment configuration information and necessary resources
- High level in nature

TEST PLAN

- More granular than strategy
- Documents the testing requirements
- Details the testing approach systematically
- Identifies the workflow a tester would perform

TEST CASE

- Takes the test requirements from the test plan and defines specific measurable conditions to validate that the requirements are being met
- Generally contains a unique identifier, reference to the requirement being validated, any preconditions that need to be met, actions, inputs and expected results

TEST SCRIPT

- Details how the testing is to be performed
- What the step-by-step actions of the tester will be
- For each test case, one or more test scripts need to be created

TEST HARNESS

- **All of the necessary tools required to complete the software testing process.**
  - Testing tool
  - Test data
  - Testing configurations
  - Test cases
  - Test Scripts

TYPES OF TESTING

- Functional
- Non-Functional
- Other
- Security

TYPES OF TESTING: FUNCTIONAL TESTING

- Unit Testing—conducted by developers during the implementation phase of SDLC
  - Breaks the functionality of software down into smaller parts and tests each part separate from the other parts for build and compilation and logic errors
- Logic Testing—validates the logic of the code—Is it well written?
- Integration Testing—Tests the "sum of its parts"
- Regression Testing—Validates that the software doesn't break previous functionality or security

TYPES OF TESTING: NON-FUNCTIONAL

- Performance Testing: Will it meet the objectives of the business and satisfy requirements of the SLA?
- Load Testing: What volume of tasks or users can the software handle?
- Stress Testing: What is the breaking point of the software? There are two primary objectives:
  - Does the software fail securely?
  - Can the software recover gracefully?
- Scalability Testing: Similar to load testing and helps identify performance bottlenecks
- Environment Testing: Validates the security of the environment in which the software will operate
- Interoperability Testing: Are the interfaces between disparate environments working?
- Disaster Recovery Testing: Can the critical services be restored within documented time constraints in the event of a disaster
- Simulation Testing: Will the software function in the production environment? Requires the lab environment to be configured as much like production as possible

OTHER TESTING

- Privacy Testing: Validates that sensitive information is protected appropriately
- UAT: User Acceptance Testing: End user needs to be assured that the application will meet their specified requirements. Must happen before software is considered ready for release
- All other testing should be completed (Unit, integration, regression, etc)
- Real-world usage scenarios of the software are identified and test cases are created to cover these scenarios

SECURITY TESTING

- **White Box (aka Structural Analysis)—full access to :**
  - Source Code
  - Design Documents
  - Use and Misuse Cases
  - Configuration Files

SECURITY TESTING

- **Black Box Testing: No knowledge of the code**
- **Also known as zero knowledge assessment, as the testers have no access to supporting documentation or the internal working of the code**
  - Fuzzing
  - Scanning
  - Penetration Testing

FUZZING

- Also known as fault injection testing
- Brute force type of testing in which faults are injected into the software and the behavior is observed
- Verifies the effectiveness of input validation
- Also used to find coding defects and security bugs
- Ideally prevents issues with buffer overflows, remote code execution, logic faults, etc

SCANNING

- **Scanning is used to**
  - Map the environment
  - Identify server versions, open ports and running services
  - Inventory and validate asset management databases
  - Identify patch levels
  - Prove due care and due diligence for compliance issues

TYPES OF SCANNING

- Vulnerability Scanning is performed with the goal of providing detection and identification of security flaws and weaknesses in the software/system
- Content Scanning analyzes the actual contents of the document (web pages, files, etc) for malicious content in macros, embedded scripts, etc
- Privacy Scanning : Performed to detect violations of privacy policies

PENETRATION TESTING (PEN-TESTS)

- **Where as scanning is passive, pen-testing looks to actively exploit a weakness. Usually follows the steps:**
  - Reconnaissance (Enumeration and Discovery) which allows learning and listing information about the network, often from publicly available sources like the internet
  - Resiliency Attack: Attempt to exploit the potential vulnerabilities from the reconnaissance
  - Removal of Evidence: Clean up any evidence of the compromise
  - Reporting and Recommendations: Should include technical vulnerabilities as well as non-compliance with organizational processes and policies

IMPACT ASSESSMENT AND CORRECTIVE ACTIONS

- **The defect report should indicate urgency and severity levels of vulnerability**
- **Corrective actions can dictates risk mitigation strategy**
  - Mitigate risk—fix the flaw
  - Transfer risk—postpone the inclusion of the function (not the fix) to a later version
  - Avoid Risk—replace the software

ADDRESSING DEFECTS

- Correct the defects in the development environment
- Verify the solution in the testing environment
- Verify the software's function in UAT
- Then release and monitor software in production

SECURE SOFTWARE TESTING

- Quality Assurance
- Testing Artifacts
- Types of Testing
- Impact Assessment and Corrective Action

Part VI Secure Software Acceptance

SECURE SOFTWARE ACCEPTANCE

- Introduction to Software Acceptance
- Pre-release activities
- Post-release activities

INTRODUCTION TO SOFTWARE ACCEPTANCE

- The purpose of software acceptance phase of the lifecycle is to determine whether or not the product has met the delivery criteria (pre-defined) as specified in the contract
- The support this assessment comes from the tests, structured reviews and audits,
- Software Qualification Testing is the formal analysis that is done to determine whether a system or software product satisfies its acceptance criteria. These tests are conducted by the customer to ensure their requirements have been met

ELEMENTS OF A QUALIFICATION TESTING PLAN

- The required features to be tested
- Required load limits
- Number and type of stress tests
- All necessary risk mitigation and security tests
- Requisite performance levels
- Tested Interfaces

PRE-RELEASE ACTIVITIES: COMPLETION CRITERIA

- **Established by the project's contract**
- **Per ISO 9126 the six generic criteria for judging a product's suitability are:**
  - Functionality: Does it meet requirements
  - Reliability: Are there fault tolerance elements and how often are there failures
  - Usability: Ease of Use
  - Efficiency: response and processing time
  - Maintainability: Change management
  - Portability: is the product adaptable to a new environment
- **Other measures can be included as part of completion criteria as well.**

RISK ACCEPTANCE

- Formal risk acceptance procedures must be addressed and well documented.
- Risk assessment requires a detailed knowledge of the risks and consequences associated with the software under consideration. This information is contained in a properly executed threat model, which is created as part of the development process.

RISK ACCEPTANCE

- Risk Acceptance
- Formal LOB executive acceptance of risk associated with software Part of overall Risk Management Strategy
- Acceptance documented
- Documentation Format variable but
- includes:
- Risks
- Actions
- Issues
- Decisions
- Document Templates recommended

SOFTWARE DOCUMENTATION

Document Type	Assurance Aspect
RTM	Are functionality and security aspects traceable to customer requirements and specifications?
Threat Model	Is the threat model comprehensively representative of the security profile and addressing all applicable threats?
Risk Acceptance Document	Is the risk appropriately mitigated, transferred or avoided? Is the residual risk below the acceptable level? Has the risk been accepted by the AO with signatory authority?
Exception Policy Document	Is there an exception to policy and if so is it documented? Is there a contingency plan in place to address risks that do not comply with the security policy?
Change Requests	Is there a process to formally request changes to the software and is this documented and tracked? Is there a control mechanism defined for the software so that only changes that are approved at the appropriate level can be deployed to production environments.
Approvals	Are approvals (risk, design and architecture review, change, exception to policy, etc.) documented and verifiable? Are appropriate approvals in place when existing documents like BCP, DRP, etc. need to be redrafted?
BCP or DRP	Is the software incorporated into the organizational BCP or DRP? Does the DRP not only include the software but also the hardware on which it runs? Is the BCP/DRP updated to include security procedures that need to be followed in the event of a disaster?
Incident Response Plan (IRP)	Is there a process and plan defined for responding to incidents (security violations) because of the software?
Installation Guide	Are steps and configuration settings predefined to ensure that the software can be installed without compromising the secure state of the computing ecosystem?
User Training Guide/Manual	Is there a manual to inform users how they will use the software?

VERIFICATION

- Verification
- Does the software meet the developer's description? Does the software satisfy the requirements?

VALIDATION

- **Does the software solve the problem that it was supposed to solve. Does it meet a real-world need?**

VERIFICATION AND VALIDATION CHECKS

- Check for the presence of security protection mechanisms to ensure confidentiality, integrity of data and system, availability, authentication, authorization, auditing, secure session management, proper exception handling and configuration management

CERTIFICATION

- The technical evaluation of the security features of a software product. Does the product provide the appropriate needs for security in a particular environment? Is it technically secure? Completed by independent testers or QA

ACCREDITATION

- **Management's acceptance (risk acceptance) of the product and their decision to implement the software in their environment**

POST-ACCEPTANCE

- Ongoing updates, patches and changes reviewed and applied while software is in O&M phase. Reporting of each update to external and internal organizations Any significant issue or bug is identified, tracked and repaired Final retirement of software – End of Life – event is conducted

Part VII Secure Deployment, Operations, Maintenance and Disposal

SECURE INSTALLATION AND DEPLOYMENT

- Hardening
- Environment Configuration
- Release Management
- Bootstrapping and Secure Startup

HARDENING

- **What should be hardened? Everything! Operating systems, applications, hardware, etc**
- **Operating systems and Software:**
  - MSB (Minimum Security Baseline should be created and approved to ensure compliance with organizational security policy. All operating systems should conform to MSB
  - Applications must be kept up to date
    - Hotfixes
    - Patches
    - Service Packs.
- Remove all maintenance hooks
- Removal of debugging flags in code
- Removal of unnecessary comments that contain sensitive information

COMMON MISCONFIGURATIONS OF SETTINGS

- Hard coding credentials and cryptographic keys inline code or in configuration files in cleartext
- Not disabling the listing of directories and files in a web server.
- Installation of software with default accounts and settings.
- Installation of the administrative console with default configuration settings.

Installation or configuration of unneeded services, ports and protocols, unused pages, and unprotected files and directories.

Missing software patches

Lack of perimeter and host defensive controls such as firewalls, filters, etc.

Enabling tracing and debugging can lead to attacks on confidentiality assurance.

ENVIRONMENT CONSIDERATIONS

- Does the test environment mimic the production environment?
  - Communications Ports,
  - Interfaces,
  - Privileges and Rights of Software itself
- Are there vulnerabilities inherent to the environment that were not previously considered?
- Test and default accounts need to be turned off.
- Unnecessary and unused services need to be removed in all environments.
- Access rights need to be denied by default and granted explicitly even in
- development and test environments just as they would be managed in the
- deployed production environment.

RELEASE MANAGEMENT

- Release management is the process of ensuring that all changes that are made to the computing environment are planned, documented, thoroughly tested and deployed with least privilege, without negatively impacting any existing business operations, customers, end-users or user support teams.

RELEASE PLANNING

- Planning a release involves:
- Gaining consensus on the release's contents
- Agreeing to the phasing over time and by geographical location, business unit and Customers
- Producing a high-level release schedule
- Planning resource levels (including staff overtime)
- Agreeing on roles and responsibilities
- Producing back-out plans
- Developing a quality plan for the release
- Planning acceptance of support groups and the customer

BOOTSTRAPPING AND SECURE STARTUP

- Host system start – Sequence of events and processes that self-start the system to a preset state is referred to as booting or bootstrapping.
- Mainframe Environments – this is called IPL (Initial Program Load)
- Criteria is to maintain Security during events

SECURE STARTUP INCLUDES

- Secure Startup Focus Areas
- POST – Power On Self Test
- BIOS – Basic Input Output System
- TCB – Trusted Computing Base of system to be securely maintained

OPERATIONS AND MAINTENANCE

- **Software operations should provide processing that is:**
  - Reliable
  - Resilient
  - Recoverable
- **Software must be monitored and maintained as part of ongoing risk management**
- **Even with software that is secure upon installation, as the environment and the threat landscape change, additional risks can materialize**

OPERATIONS SECURITY (OPSEC)

- Software OPSEC is the assurance that the software will continue to function as is expected to in a reliable fashion for the business, without compromising its state of security by monitoring, managing and applying the needed controls to protect resources (assets).

OPSEC

- **Requires monitoring of four basic elements**
  - Hardware
  - Software
  - Media
  - People

OPSEC: HARDWARE

- **Network Components: Switches, routers, firewalls, etc.**
- **Communication devices: Phones, fax, PDA, VoIP, etc.**
- **Computing Components Servers, workstations, desktops, laptops, etc.**
- **Many of these mechanisms have default passwords, administrative accounts, and general configurations designed for ease of use rather than security**
  - Username: Admin Password: Password
  - No security (or weak security) on Wireless Access points
  - Defaulting to full access, as opposed to defaulting to no access
  - Default shares created

OPSEC: SOFTWARE

- Various SW components which affect Security
- In-house developed software
- External third party software
- Operating system software and Data (includes stored and transactional data)
- Often default port numbers are used
- Services are often loaded even when they are not necessary
- Data May not be protected at rest/in transit

OPSEC: MEDIA

- USB
- Tapes
- hard drives (both internal and external)
- optical CD/DVD
- NIST SP 800-88 covers aspects of Media Security and Sanitation
- Media Reuse has a huge impact on Confidentiality
- Degaussing exposes media to a strong magnet to wipe out the cylinders, tracks, sectors of a magnetic drive
- Zeroization overwrites 0s to media, over and over again
- Physical Destruction is the only guaranteed way to guarantee no remnants of secure data
- Deleting files simple removes the pointer to the file. Formatting is a process that is easily undone

OPSEC PEOPLE

- People can be both our greatest strength and weakness in a secured environment. They can notice things that don't feel or seem right. They can take advantage of human judgement. However, 85 percent of fraud comes from an internal source
- Social Engineering takes advantage of the fact that most people want to help. It may be possible for an attacker to trick someone within the organization to disclose information to an untrusted source.
- Training and accountability are essential elements to helping mitigate the risks associated with my employees

ACCESS CONTROL TYPES

- **Preventative**
  - Controls used to prevent undesirable events from taking place
- **Detective**
  - Controls used to identify undesirable events that have occurred
- **Corrective**
  - Controls used to correct the effects of undesirable events
- **Deterrent**
  - Controls used to discourage security violations
- **Recovery**
  - Controls used to restore resources and capabilities
- **Compensation**
  - Controls used to provide alternative solutions
- **Directive**
  - An employee handbook for instance will provide directions on how to maintain compliance with security policy

MONITORING

- Validate compliance to regulations and other governance requirements.
- Demonstrate due diligence and due care on the part of the organization towards its stakeholders.
- Provide evidence for audit defense.
- Assist in forensics investigations by collecting and providing the requested evidence if tracked and audited.
- Determine that the security settings in the environment are not below the levels prescribed in the minimum security baselines.

MONITORING CONTINUED

- Ensure that the confidentiality, integrity and availability aspects of software
- assurance are not impacted adversely.
- Detect insider and external threats that are orchestrated against the organization.
- Validate that the appropriate controls are in place and working effectively.
- Identify new threats such as rogue devices and access points that are being introduced into the organization's computing environment.
- Validate the overall state of security.

MONITORING

- **Characteristics of good metrics include:**
  - Consistency: The results from the same data set must be the same or equivalent
  - Quantitative: Precise, objective, numeric values
  - Objectivity: Unbiased
  - Relevance: should have a direct bearing on a decision or judgement
  - Inexpensive: Should be cost-effective

AUDITING

- **Audits are important detective controls and can be used to correlate information after an event.**
- **Audits can be used to:**
  - Ensure policies are being followed/are effective
  - Make sure that individual user accounts aren't unintentionally being allowed to accumulate rights/permissions
  - Check the accuracy and completeness of transactions that are authorized
  - Privileged actions are restricted to authorized personnel

INCIDENT MANAGEMENT

- **Events: an observable change in state**
- **Alerts: Flagged events that may require further investigation to determine if an incident has taken place**
- **Incidents: Adverse impact to the system or network**

TYPES OF INCIDENTS

- DoS or DDoS: Attacks the availability of the system
- Malicious Code: virus, worms, logic bombs, etc
- Unauthorized access: A subject gains access to a restricted object
- Inappropriate usage: Violation of the acceptable use of a system

INCIDENT RESPONSE

- **Should be consistent and well controlled. There is a four step process**
  - Preparation
  - Detection and analysis
  - Containment, eradication, and recovery
  - Post-incident review

PROBLEM MANAGEMENT

- **An incident with an unknown cause is referred to as a problem.**
  - Incident notification
  - Root cause analysis
  - Solution determination
  - Request for change
  - Implement solution
  - Monitor and report

CHANGE MANAGEMENT

-
- When change is determined to be a necessity upon undertaking problem management activities, the change management processes and protocols should be followed as published by the organization.
- Multiple standards and techniques available
- Depends on industry and standards

CHANGE MANAGEMENT

- Procedural
- Scheduling
- Documentation
- Awareness / Training
- Back out plans / fall backs
- Change Management Database (CMDB)
- What / When / Who
- Vendor contact / support info

PATCH MANAGEMENT

- Patches are additional pieces of code developed to address problems (commonly called “bugs”) in software.
- Patches enable additional functionality or address security flaws within a program.
- Not all vulnerabilities have related patches; thus, system administrators must not only be aware of applicable vulnerabilities and available patches, but also other methods of remediation (e.g., device or network configuration changes, employee training) that limit the exposure of systems to vulnerabilities.

DISPOSAL

- **Sun Setting Criteria**
  - The software has reached its end of vendor support.
  - The software is no longer compatible with the architecture of the hardware.
  - Software that can provide the same functionality but in a more secure fashion is available as new products, upgrades or versions releases.

DISPOSAL OF SOFTWARE

- Disposal of Software
- Archiving/Backed up
- Escrow
- Discarded
- Overwritten
- Physically destroyed

DECOMMISSIONING SOFTWARE

- System Strategy for Data/Information handling
- Coordination with Other Systems
- Media Sanitation
- Support Agreements termination process
- Archiving of retained data
- Disposal of assets

PART VII

- Deployment
- Operations
- Maintenance
- Disposal