PRACTICAL - 2

AIM: Study eliciting security requirements for software projects, prioritize security requirements, and explain their importance.

Solution:

Security Requirements Elicitation

Security requirements specify what protections a software system must provide to defend against threats and unauthorized actions. Unlike functional requirements, security requirements safeguard **confidentiality, integrity, availability (CIA)** and ensure **compliance with standards** like GDPR, HIPAA, OWASP, etc.

Categories of Security Requirements:

- **Authentication:** Confirming a user's identity.
- **Authorization:** Ensuring users have permissions to perform actions.
- **Data Protection:** Securing sensitive information in storage and transit.
- **Input Validation:** Checking and sanitizing user input.
- **Auditing & Logging:** Recording key actions and system events.
- **Error Handling:** Preventing information disclosure through error messages.
- **Session Management:** Controlling and securing user sessions.

Eliciting Security Requirements

1. Stakeholders Identification

The first step is to identify all relevant stakeholders who have an interest in the security of the system. These are the people who either use the system, manage it, develop it or are responsible for legal and regulatory compliance.

Examples of stakeholders include:

- Business owners (set priorities and budget)
- End-users (interact with the system daily)
- Developers and system architects (design and build the system)
- System administrators (maintain operations and access)
- Security experts (define protection strategies)
- Legal and compliance officers (ensure legal standards are met)

2. Techniques to Elicit Security Requirements

Once stakeholders are identified, various techniques can be used to collect security requirements. These techniques help uncover potential threats, risks and specific protections that need to be in place.

01. **Stakeholder Interviews and Questionnaires:** Talk directly to stakeholders to understand their security concerns. Ask about the type of data handled, who needs access, and what risks they foresee.
02. **Misuse and Abuse Cases:** This involves thinking like an attacker, not just how legitimate users would use the system, abuse cases consider how attackers might

misuse it. These scenarios help identify potential threats and guide the design of security controls to prevent them.

Example:
Abuse Case: "An attacker injects malicious SQL to bypass authentication."
Security Requirement: "System must validate and sanitize all user inputs."

03. **Threat Modeling:** Threat modeling is a structured method to analyze the system's architecture and data flows in order to identify vulnerabilities and potential attack vectors. This is often done using threat classification frameworks like STRIDE, which helps developers think from an attacker's perspective.

    STRIDE stands for:
    - **S**poofing (identity forgery)
    - **T**ampering (unauthorized data modification)
    - **R**epudiation (denying actions)
    - **I**nformation Disclosure (data leaks)
    - **D**enial of Service (disrupting services)
    - **E**levation of Privilege (gaining unauthorized access)

04. **Security Checklists:** Checklists based on established security standards are a practical way to ensure that common and well-known threats are addressed. These are often used alongside other techniques to verify that all major risk areas are covered.

    Common sources include:
    - OWASP Top 10 (for web security)
    - NIST 800-53 (government security controls)
    - ISO/IEC 27001 (international information security standard)

05. **Compliance Requirements:** Many industries are governed by laws and regulations that impose specific security requirements. This step involves identifying those requirements and ensuring the system complies with them. Failure to meet these requirements can result in legal penalties and reputational damage.

    Examples of regulations:
    - GDPR – Requires personal data encryption and consent.
    - HIPAA – Protects medical data and ensures audit logging.
    - PCI-DSS – Mandates secure handling of credit card data.

Common Security Requirements

Requirement	Description	Example	Mitigation
Authentication	Enforce strong user authentication for sensitive functions	Admin login requires password + OTP	Multi-factor authentication
Authorization	Restrict access to authorized users only	User can view only their data, not others'	Role-based access control

Input Validation	Validate and sanitize all user-provided data	Prevent SQL Injection, XSS via parameter checks	Server-side validation
Data Encryption	Encrypt sensitive data in storage/transit	Banking app encrypts account numbers in DB	AES, TLS
Auditing	Log administrative and sensitive actions	Log all changes to user profiles	Centralized log management
Session Management	Expire sessions after inactivity	User auto-logout after 10 minutes idle	Session timeout

Prioritization of Security Requirements

Security requirements should be prioritized based on their impact on system confidentiality, integrity and availability, as well as criteria such as **asset sensitivity**, **threat likelihood** and **impact severity**. This helps in effectively allocating security resources and addressing the most critical risks first.

1. CIA-Based Prioritization

The CIA Triad - Confidentiality, Integrity and Availability, is a core model used to assess the criticality of a security requirement. By evaluating how much each requirement affects these three aspects, we can assign an overall priority level to guide implementation.

Requirement	Confidentiality	Integrity	Availability	Priority
Password Encryption	High	Medium	Low	Critical
Backup System	Low	Medium	High	High
Input Validation	Medium	High	Medium	High

2. MoSCoW Method

The MoSCoW technique is a prioritization method that categorizes requirements into four groups: Must Have, Should Have, Could Have and Won't Have (for now). This method helps teams manage scope while focusing on essential security features first.

- **Must Have**: Items are essential for system security, such as encryption and access control.
- **Should Have**: Features improve protection but can be added after core functionality.
- **Could Have**: Items are beneficial enhancements, like biometrics.
- **Won't Have (Now)**: Refers to lower-priority features that are not feasible in the current phase.

3. Risk Matrix

A Risk Matrix evaluates threats based on their likelihood and impact, allowing teams to assign priority levels accordingly. This visual tool helps in deciding which threats need urgent mitigation and what controls should be implemented.

Threat	Likelihood	Impact	Priority	Control
SQL Injection	High	Critical	Urgent	Input validation, WAF
XSS	Medium	High	High	Output encoding, CSP
Weak Authentication	High	High	Critical	MFA, strong password policy

Case Study Analysis: Zoom's Security Missteps (2020)

Background: In 2020, Zoom experienced explosive growth due to the COVID-19 pandemic, becoming a critical tool for remote work, education and social communication. However, this rapid adoption exposed several security and privacy flaws. While the platform prioritized usability and scalability, it initially fell short in meeting core security requirements, especially under the scrutiny of millions of new users worldwide.

- **Issues:**
  - Failure to patch the Apache Struts CVE-2017-5638 vulnerability.
  - Poor access control led to widespread incidents of "Zoom-bombing" (unauthorized intrusions into meetings).
- **Missed Security Requirements:**
  - Data encryption during transmission and at rest.
  - Strong authentication and better meeting access control mechanisms (e.g., waiting rooms, passcodes, user authentication).
- **Impact:**
  - Exposure of sensitive meetings, including corporate, educational and governmental sessions.
  - Loss of user trust and public scrutiny.
  - Legal and reputational consequences, forcing Zoom to initiate a 90-day security plan and overhaul its security posture.
- **Lessons Learned:**
  - Security cannot be an afterthought in feature-rich systems.
  - Security requirements must evolve with product scale and user base.
  - Rapid growth must be supported by scalable and proactive security practices.

Importance of Security Requirements

Security requirements are essential for protecting software from threats, ensuring that it operates safely and as intended. They help in identifying critical risks early in the development process and reduce the chance of vulnerabilities being exploited.

1. Prevents Security Breaches: Well-defined security requirements help identify and mitigate threats early, reducing the risk of attacks such as data breaches, injection attacks or privilege escalations.

- Prevents exploitation of known vulnerabilities
- Minimizes unauthorized access to sensitive data

2. Reduces Long-Term Costs: Fixing security issues after deployment is significantly more expensive than resolving them during design or development. Early prioritization saves cost and effort.

- Security flaws found in production can cost up to 30 times more to fix than if addressed during development.
- Reduces time spent on post-deployment patching and incident response

3. Ensures Regulatory Compliance: Modern applications must adhere to security and privacy standards like GDPR, HIPAA, PCI-DSS, etc. Security requirements ensure compliance is built into the system.

- Avoids legal penalties and audits
- Aligns with national and international data protection laws

4. Protects User Trust and Business Reputation: Users expect their personal information and activities to remain private and secure. Breaches can permanently damage a company's reputation.

- Builds trust through secure-by-design applications
- Prevents customer churn caused by public security incidents

5. Supports Secure System Architecture: Security requirements guide architectural decisions, helping developers adopt secure patterns such as encryption, access control and secure communication protocols.

- Enables use of secure APIs and frameworks
- Promotes defense-in-depth and layered security design