**1. Key Lessons about Web Application Security and the CIA Triad**

* **Confidentiality**: Learned about securing sensitive data through encryption (e.g., HTTPS, TLS). Exposure of sensitive information compromises user trust.
* **Integrity**: Techniques like hashing and input validation help protect against tampering (e.g., SQL injection). A lack of integrity can lead to altered or maliciously modified data.
* **Availability**: Ensured by using protections against DDoS attacks, monitoring, and redundant servers. Downtime or unavailability disrupts user experience and operations.  
  These principles form the foundation for designing secure web applications.

**2. Impact of Vulnerabilities and Exploits**

* **Effects**: Vulnerabilities like XSS, CSRF, and SQL Injection allow attackers to steal data, deface applications, or gain unauthorized access.
* **Defence**: Use secure coding practices, automated scanners, and security-focused frameworks. Example: Validate user input and apply parameterized queries for database interactions.

**3. Role of Layers in Web Security and Associated Threats**

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Role in Web Security** | **Threats** | **Mitigation Strategies** |
| **User** | Users interact with the web application through a browser or app. | Social engineering (phishing), weak passwords, or unawareness of malicious links. | User education, multi-factor authentication (MFA), password strength enforcement, and awareness campaigns. |
| **Client** | End-user systems and browsers render web applications and handle inputs. | Cross-Site Scripting (XSS), clickjacking, malicious ads (malvertising). | Implement Content Security Policy (CSP), secure cookies, input sanitization, and use security headers (e.g., X-Frame-Options). |
| **Network** | Transports data between the client and server over HTTP/HTTPS protocols. | Man-in-the-Middle (MITM) attacks, DDoS attacks, DNS spoofing, and packet sniffing. | Use HTTPS (TLS encryption), Web Application Firewalls (WAF), Intrusion Detection/Prevention Systems (IDS/IPS), and DDoS mitigation tools. |
| **Server** | Processes user requests, hosts business logic, and delivers content to the client. | Server misconfigurations, unauthorized access, privilege escalation, or code injection (e.g., RCE, SQL injection). | Patch regularly, configure least privilege access, use firewalls and intrusion prevention, and validate inputs on the server-side. |
| **Database** | Stores application data, including user information, records, or transactions. | SQL injection, insecure storage of sensitive data, and privilege mismanagement. | Apply access controls, use encrypted connections (e.g., TLS), parameterized queries (prepared statements), and secure storage mechanisms like hashing (e.g., bcrypt). |

**4. Failures in Configuration, Policy, or Assumptions**

**The Case of WannaCry**

The WannaCry ransomware attack (2017) provides a concrete example of how web application security can fail due to misconfigurations, weak policies, faulty mechanisms, and dangerous assumptions.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Explanation** | **What Went Wrong in WannaCry** |
| **Configuration** | Improper configurations in systems and applications can leave them exposed to attacks. | Many organizations had not applied the Microsoft patch for the EternalBlue vulnerability (MS17-010), leaving systems vulnerable to exploitation. |
| **Policy** | Weak or poorly enforced policies fail to establish safeguards for secure operations and maintenance. | Policies on timely patch management and network segmentation were either absent or not strictly enforced, allowing rapid malware propagation. |
| **Mechanism** | Faulty or outdated security mechanisms can fail to provide adequate protection or containment during an attack. | Legacy systems running unsupported versions of Windows lacked modern security mechanisms like automated patch updates and advanced threat protection. |
| **Assumptions** | Misplaced assumptions, like "our systems won't be targeted" or "we're safe behind firewalls," increase risk. | Organizations assumed their internal networks and firewalls would prevent ransomware from spreading, underestimating how fast it could propagate laterally. |

**Insights from WannaCry**

1. **Failure in Configuration**: Neglecting to apply critical patches exposes systems to known vulnerabilities. In WannaCry, unpatched SMB services (Server Message Block protocol) were exploited, which could have been avoided.
2. **Policy Oversight**: Weak governance over software updates left millions of systems vulnerable globally. Many organizations lacked explicit, enforceable patching schedules.
3. **Mechanism Shortcomings**: Reliance on outdated systems (e.g., Windows XP) without robust fallback measures allowed the attack to succeed even after Microsoft issued emergency patches.
4. **Dangerous Assumptions**: The belief that firewalls or antivirus software alone were sufficient to stop a sophisticated ransomware campaign was a critical error.

**5. Risk and Impact Evaluation**

* **Risk**: Depends on exposure, attack sophistication, and the value of the assets targeted.
* **Impact**: Measured in terms of financial loss, reputation damage, and regulatory penalties.
* Evaluate risks by prioritizing high-risk vulnerabilities using frameworks like OWASP Top 10. Examples include categorizing XSS as medium risk (most websites) or high risk (if impacting sensitive operations like banking).

**6. Effective Prevention Strategies**

* Adopting secure development practices like **Threat Modeling** and **Test-Driven Security (TDS)**.
* Implementing **strong authentication mechanisms** (MFA).
* Regular updates and patches for software and dependencies.
* Conducting **penetration tests** and incorporating **automated security scanning** in CI/CD pipelines.