**Day 27**





**“CLOUD SECURITY”**

**Secure Design and Development in Cloud SSDLC**

**Impact of Cloud on SSDLC**

* Cloud apps must follow security activities through development → deployment → operations.
* Shared responsibility model: customers depend on CSPs for some security aspects (varies by IaaS, PaaS, SaaS).
* Visibility & control: limited in public clouds; varies by provider.
* Security planning must align with CSP’s services, features, and controls.
* Metastructure & management plane become part of application security scope.

**SSDLC Frameworks Used:**

* Microsoft SDL
* NIST 800-64
* ISO/IEC 27034
* OWASP
* CSA SSDLC Model (meta-phases across all frameworks).

**CSA SSDLC Model Meta-Phases:**

1. **Secure Design & Development**

* Developer training, coding standards, secure coding, code review, security testing.

1. **Secure Deployment**

* Security checks when moving from **dev → production**.
* Activities: vulnerability scanning, patching, configuration hardening.

1. **Secure Operations**

* Post-deployment protections.
* Activities: Web Application Firewalls (WAFs), log monitoring, vulnerability assessments, continuous patching.

**Cloud SSDLC Phases**

Cloud SSDLC integrates security throughout the software lifecycle, adapting traditional SDLC for cloud-specific considerations.

**1. Training**

* Staff in development, security, and operations get cloud security fundamentals and platform-specific technical training.
* Ensures teams can securely architect, deploy, and manage cloud apps.

**2. Define**

* Establish security standards, architectures, and functional requirements.
* Decide data classification, cloud service usage, pre-approved configurations, tools, and initial entitlements.

**3. Design**

* Focus on cloud architecture, provider capabilities, and security automation.
* Integrate provider’s built-in security features into application architecture for robust protection.

**4. Develop**

* Developers configure services, networks, and PaaS components in a safe development environment (no production access).
* Secure CI/CD pipelines, integrate logging, and leverage cloud security controls.

**5. Test**

* Integrate security testing into pipelines: SAST, DAST, functional tests, unit tests, vulnerability assessments.
* Automate testing to cover both application and infrastructure (IaaS templates).
* Include flagging features for sensitive functions like encryption/authentication.

**Cloud Application Architecture: Microservices**

Goal of cloud application design: Agility, statelessness, and enhanced user experience.  
Organizations must select architecture based on business needs.

**Microservice Architecture**

* Divides a single application into small, independently deployable services.
* Services may use different storage systems and programming languages.
* Communication via lightweight protocols.
* Supported frameworks: Sidecar, Ambassador, Adapter.

**Best Practices for Microservices Security**

1. Secure API Gateway
   * Acts as a single entry point for all services.
   * Manages external requests, prevents attacks, and protects microservices.
2. Authentication & Access Control
   * Use OAuth/OAuth2 and MFA for endpoint security.
   * Blocks malicious access and alerts intruders.
3. Scan Dependencies
   * Track third-party/open-source components.
   * Identify and remediate vulnerabilities in tangled dependencies.
4. Application Security Testing
   * Integrate SAST, DAST, RASP, SCA in DevSecOps pipelines.
   * Ensure security across development and deployment stages.
5. Container Security
   * Secure cloud-native environments where microservices run.
   * Covers container images, registry, orchestration.
   * Automated DevSecOps tools enhance container-level protection.

**Cloud-Based Architecture**

**Features:**

* Uses load balancers, app servers, databases, API gateway, serverless functions
* Supports multiple languages (Java, Python, PHP, Node.js, Docker, etc.)
* Provides elasticity, auto-provisioning, high availability, and scalability

**Best Practices:**

1. Endpoint Security: Protect devices from threats
2. Intrusion Detection/Prevention: Monitor and block attacks
3. Cloud Usage Policies: Control user activities and access

**Event-Driven Serverless Architecture (EDA)**

* Purpose: Decoupled, automatic response to events; scalable and cost-efficient.
* Security Best Practices:
  + Store configuration in environment variables to reduce data exposure.
  + Use strict network access controls (API gateways, ACLs).
  + Apply one-role-per-function via IAM to minimize privilege misuse.

**Cloud-Native Architecture**

* Purpose: Builds and runs apps optimized for cloud; uses containers, microservices, DevOps, CI/CD.
* Security Best Practices:
  + Secure containers and microservices directly.
  + Automate security checks and feedback in CI/CD pipelines.
  + Apply security policies continuously throughout development.

--The End--