Lesson 1 — Why build security in (SDLC + SDL basics)

Must-know concepts

- **SDLC** = Software Development Life Cycle (the structured process to produce software).
- **SDL** = Security Development Life Cycle (standardizes security best practices within the SDLC).
- **Software security** = building security into software *through an SDL within an SDLC* (not bolting it on later).
- **Security triad (CIA)**:
    - **Confidentiality** (keep data secret)
    - **Integrity** (keep data accurate/unchanged)
    - **Availability** (keep services accessible)
- **Threat modeling + attack surface validation throughout SDL** reduces vulnerabilities.
- Align software + hardware choices to maximize overall security.

SDLC phases (memorize all 8)

1. Planning
2. Requirements
3. Design
4. Implementation
5. Testing
6. Deployment
7. Maintenance
8. End of Life

Key terms (quick definitions)

- **Hardware**: physical components.
- **Software**: programs + operating systems.

- **Secure code**: coding with best-practice safeguards against vulnerabilities.
- **Threat modeling**: structured process to protect against vulnerabilities.
- Phase meanings (high-level):
    - Planning: vision + next steps
    - Requirements: determine needed requirements
    - Design: turn requirements into technical design
    - Implementation: build using known resources/components
    - Testing: verify functions in known environment
    - Deployment: release/push out (incl. security rollout)
    - Maintenance: ongoing security monitoring
    - End of Life: remove/retire software properly

Lesson 2 — SDL programs and maturity models (BSIMM, OWASP, NIST, CVE)

Must-know concepts

- Implementing SDL means security is **built-in**, not an afterthought.
- Common SDL-related models/resources:
    - **BSIMM**
    - **SSDL Touchpoints**
    - **OWASP Code Review Guide**
    - **Cisco SDL**

    - **Microsoft Trustworthy Computing SDL**

- **BSIMM**: studies real-world software security initiatives; helps you assess where you are and how to mature.
    - Has **12 best practices** (know that “12” is a fact).
- **OWASP SAMM**: flexible, prescriptive framework to build security into development orgs.
- **NIST**: provides research/info/tools for gov and corporate security.
- **DHS Software Assurance Program** exists (government program).
- **CVE**: common names/list for publicly known vulnerabilities.
- **SDL must map to your current SDLC** (integration is key).
- **Security metrics** support risk decisions, budgeting, and customer assurance.
- **Application security (AppSec)**: developing/adding/testing security features to prevent vulnerabilities.

Key terms

- **Static analysis**: analyze code without executing.
- **Dynamic analysis**: analyze software while executing.
- **Fuzz testing**: feed invalid/unexpected/random inputs to find failures.
- **Metric model**: measures effectiveness of security controls.
- **Measurement model**: data security methods devs use to protect against vulnerabilities.

Lesson 3 — SDLC approaches (Waterfall, V-model, Agile, Scrum, XP)

Must-know comparisons

Waterfall

- Linear phases; output of one becomes input to next.
- ✅ Advantage: clear staged control over development.
- ❌ Disadvantage: little room for reflection/revision.

V-model

- Waterfall variation: after coding, process turns upward (forms a “V”) emphasizing validation/verification.

Agile

- Mixes traditional + modern practices.
- Uses collaboration by self-organizing, cross-functional teams.
- Has **4 core values + 12 principles**.
- ✅ Advantage: rapid, continuous delivery → customer satisfaction.
- ❌ Disadvantage: hard to estimate effort early in SDL.

Scrum

- Agile framework for flexible, holistic teamwork toward a common goal.

Extreme Programming (XP)

- Agile method focused on improving software quality + responsiveness.

Key terms

- **Waterfall methodology**: sequential requirements-to-delivery process.
- **V-model**: waterfall variant with upward validation after coding.
- **Agile methodology**: collaborative iterative approach.

- **Scrum**: flexible team strategy to reach a goal.
- **XP**: improve quality + responsiveness.

Lesson 4 — SDL Phase A1: Security Assessment + Requirements

What happens in A1 (Security Assessment)

- **First SDL phase**.
- Build initial outline of **security milestones** and integrate into project schedule.
- Stakeholders align on **security + privacy implications/requirements**.
- Include software security team in SDLC kickoffs.
- **Privacy impact assessment** includes: legislation summary, required steps, technologies/techniques, extra resources.
- Define **success criteria** and **phase deliverables** (helps postmortems).
- Identify **metrics** to measure in each phase.

Requirements focus areas (3)

1. Gathering software requirements
2. Data classification
3. Managing data protection requirements

Requirement types (know the difference)

- **Functional requirements**: what the system does (core purpose).

- **Non-functional requirements**: constraints/restrictions (performance, security constraints, etc.).
- **Operational requirements**: how system functions in its operating environment.
- **Compliance requirements**: **legal**, **financial**, **industry standards**.

Key terms

- **Security Assessment (A1)**: identify product risks + security milestones.
- **Product risk profile**: determines actual cost of product from different perspectives.
- **Threat profile**: operating environment + potential threats.
- **Requirement traceability matrix (RTM)**: table listing all security requirements.
- **Privacy impact assessment**: evaluates privacy issues/risk rating for PII.

Lesson 5 — SDL Phase A2: Architecture + Threat Modeling

A2 big picture

- Second SDL phase: bring security into SDLC design/architecture.
- **Software security policy**: what must be protected and how.
- Collaboration with privacy, centralized groups, and/or legal counsel is best practice.

Threat modeling (why + steps)

- Proactive: prepares before threats are discovered.
- Helps prioritize remediation by identifying threats/vulnerabilities.

Five steps (memorize order)

1. Identify security objectives
2. Survey the application
3. Decompose it
4. Identify threats
5. Identify vulnerabilities

Visual tool

- **Data Flow Diagrams (DFDs)**: visual representation of process/flow.

Threat categories: STRIDE (memorize)

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

Ranking risk: DREAD (memorize)

- **D**amage potential
- **R**eproducibility
- **E**xploitability
- **A**ffected users
- **D**iscoverability

Other frameworks mentioned

- **PASTA**: Process for Attack Simulation and Threat Analysis (repeatable framework).
- **Web Application Security Frame**: organizes common web app vulnerabilities.
- **Trike**: security auditing from a risk management perspective.

Key terms (quick)

- **Application-centric**: start from the application design.
- **Asset-centric**: start from assets senior management wants protected.
- **Threat source**: attacker/entity.
- **Threat vector**: attacker's path to exploit.
- **Vulnerability**: weakness that can be exploited.
- **Third-party codes**: externally developed reusable software.

Lesson 6 — SDL Phase A3: Design & Development testing foundations

A3 big picture

- Review policies outside SDL policy; security group + centralized infosec collaborate.
- Testing validates security *before release* (cheaper than post-deployment fixes).
- Test environment should mimic real execution environment.
- Security testing is **ongoing**, including in the true operational environment.

Testing approaches (know the trio)

- **White box**: internal perspective, full knowledge.
- **Gray box**: partial knowledge; uses code understanding to design tests.
- **Black box**: external perspective, no prior knowledge.

Alpha vs Beta (easy exam points)

- **Alpha testing**: by developers.
- **Beta testing**: by people not familiar with development.

Key terms

- **Security test cases**: find security issues at the lowest level.
- **Scanning**: identifying deficiencies across/around the system.
- **System test**: test system + interactions with other systems.
- **Scripts**: detailed step-by-step instructions.
- **Secure testing scripts** vs **functional testing scripts**: security-focused vs scenario/function-focused.
- **Internal resources** vs **external resources**: in-org vs temporary/hired.

Lesson 7 — SDL Phase A4: Code review + AppSec tooling

A4 big picture

- Review policies outside SDL policy domain.
- QA testing occurs throughout the SDLC.
- Test type categories:

1. **Benchmarks**
2. **Scheduled tests**
3. **Exploratory tests**

Code review (why it matters)

- Finds/fixes many security issues **before** testing/shipping → reduces cost.
- Four basic techniques:
    1. Automated scanning
    2. Manual penetration testing
    3. Static analysis
    4. Manual code review

AppSec reality

- AppSec = finding, fixing, preventing application vulnerabilities.
- Hard to scale in large orgs.

Tools/techniques mentioned

- **OWASP ZAP (Zed Attack Proxy)**: open-source web security testing tool.
    - **Spider**: finds inputs and feeds scanners.
    - **Passive scanner**: silently analyzes HTTP requests/responses.
    - **Active scanner**: modifies HTTPS inputs and analyzes responses.
- **SonarQube**: static code analysis platform (bugs, code smells, vulnerabilities, hotspots).

Key terms (quick)

- **AST (abstract syntax tree)**: basis for later metrics/issues.
- **Control flow analysis**: step through logical conditions.
- **Data flow analysis**: trace data from input to output.
- **Pull request**: request to merge code into another branch.
- **Documentation**: guides/support for ongoing use.

Lesson 8 — SDL Phase A5: Ship (final review + scanning + pen testing)

A5 big picture

- **Ship (A5)**: final security analysis/review before release.
- **Policy compliance analysis**: verifies product meets standards before release.

Vulnerability scanning + pen testing

- Scanners identify weaknesses.
- Pen testing simulates a hacker (authorized).
- Pen testing phases (memorize these 4 as given):
    1. Assess
    2. Identify
    3. Evaluate and plan
    4. Deploy

Lab practice (range)

- A controlled networking lab helps test without needing written authorization/permissions each time (inside the lab scope).
- **Nmap**: popular network scanning/security auditing tool.
- Choose techniques based on what best fits the application/system.

Key terms

- **Authenticated scans**: scanner logs in.
- **External scans**: outside firewall perspective.
- **Internal scans**: inside network perspective.
- **Intrusive target search**: exploit when vulnerability found.
- **Range**: networking lab for vulnerability analysis.
- **SQL injection**: code injection that may destroy software.
- **Virtualization**: creates software services (virtual environments).
- **Target machine**: virtual space to practice attack surface identification.
- **Open-source license compliance** + **open-source software security**: licensing rules + security of in-house OSS use.

Lesson 9 — Post-release support + PSIRT + CVSS + M&A

Core ideas

- Security experts reporting to engineering strengthens collaboration.
- Security quality is built across the entire engineering process.
- Not every company can include every practice—prioritize high-value PRSAs/tools.
- **CVSS** measures vulnerability severity.

- Post-release privacy issues may need additional dev/QA/security resources.
- Third-party reviews may be needed post-release.
- M&A can trigger architectural review to handle changes after consolidation.
- Post-release certifications should be included in requirements *before deployment*.

Key terms

- **Post-Release Support phase**: prepares for vulnerabilities after release.
- **PSIRT**: team that receives/investigates/reports vulnerabilities.
- **Post-Release PSIRT Response**: handles externally discovered post-release vulnerabilities.
- **Legacy code**: old, unsupported code.
- **M&A**: merger and acquisition (companies consolidate).
- **Software Security Champion (SSC)**: promotes security awareness/best practices internally.
- **Software Security Evangelist (SSE)**: promotes awareness of products to wider software community.

Lesson 10 — Modern environments + OpenSAMM + BSIMM categories + STRIDE recap

Modern delivery environments

- Software likely deployed in **Agile**, **DevOps**, **Digital Enterprise**, or combinations.
- **Agile**: deliver value faster.
- **DevOps**: teams collaborate for operations, enhancements, defect removal, optimization.

- **Cloud** changes how apps are built/deployed/used; public cloud increases security challenges.

- **Digital enterprise**: technology enabling/improving business activities.

OpenSAMM (functions + practices)

Business functions

1. Governance
2. Construction
3. Verification
4. Deployment

Practice examples (know what belongs where)

- Governance: **Education & Guidance (EG)**, **Policy & Compliance (PC)**, **Strategy & Metrics (SM)**

- Construction: **Secure Architecture (SA)**, **Security Requirements (SR)**, **Threat Assessment (TA)**

- Verification: **Code Review (CR)**, **Design Review (DR)**, **Security Testing (ST)**

- Deployment: **Environment Hardening (EH)**, **Operational Enablement (OE)**, **Vulnerability Management (VM)**

BSIMM recap

- Study of existing initiatives; used to gather data from large development orgs.

- Four BSIMM category types:

    1. Governance
    2. Intelligence
    3. SSDL touchpoints

4. Deployment

STRIDE recap (again)

- Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege.

High-yield “memorize lists” page

CIA: Confidentiality, Integrity, Availability
SDLC (8): Planning, Requirements, Design, Implementation, Testing, Deployment, Maintenance, End of Life
SDL phases (as used here): A1 Security Assessment → A2 Architecture → A3 Design & Development → A4 Design & Development → A5 Ship → Post-Release Support/PSIRT
STRIDE: Spoofing, Tampering, Repudiation, Information Disclosure, DoS, Elevation of Privilege
DREAD: Damage, Reproducibility, Exploitability, Affected Users, Discoverability
Testing views: White / Gray / Black box
Alpha/Beta: Developers / External users
Test categories: Benchmarks / Scheduled / Exploratory
Pen test phases: Assess / Identify / Evaluate & plan / Deploy
OpenSAMM functions: Governance / Construction / Verification / Deployment
BSIMM categories: Governance / Intelligence / SSDL Touchpoints / Deployment