Secure Application Development with IS-3

Christopher Thielen

Enterprise Applications Supervisor, IET EIS

Disclaimer

This presentation represents the presenter's reading of the Secure Software Development Standard as published 3/12/2018.

This presentation is not to be interpreted as an official guide nor set of recommendations.

Please refer to the document “University of California – Policy BFB-IS-3” for policy text and direct inquiries through the appropriate channel.

The Nature of Application Security

- Productivity and security are seen as competing forces.
  - Vendor samples are designed to get you started quickly. Security is often an extra step.
- But security is a requirement from stakeholders and the University of California.
- How do we treat security?
  - Is security a product feature?
  - Is security an architectural concern?
  - Is security a development practice?
- Inverted design thinking: security requires consideration of byproduct capabilities – that which your application can do that you never intended.

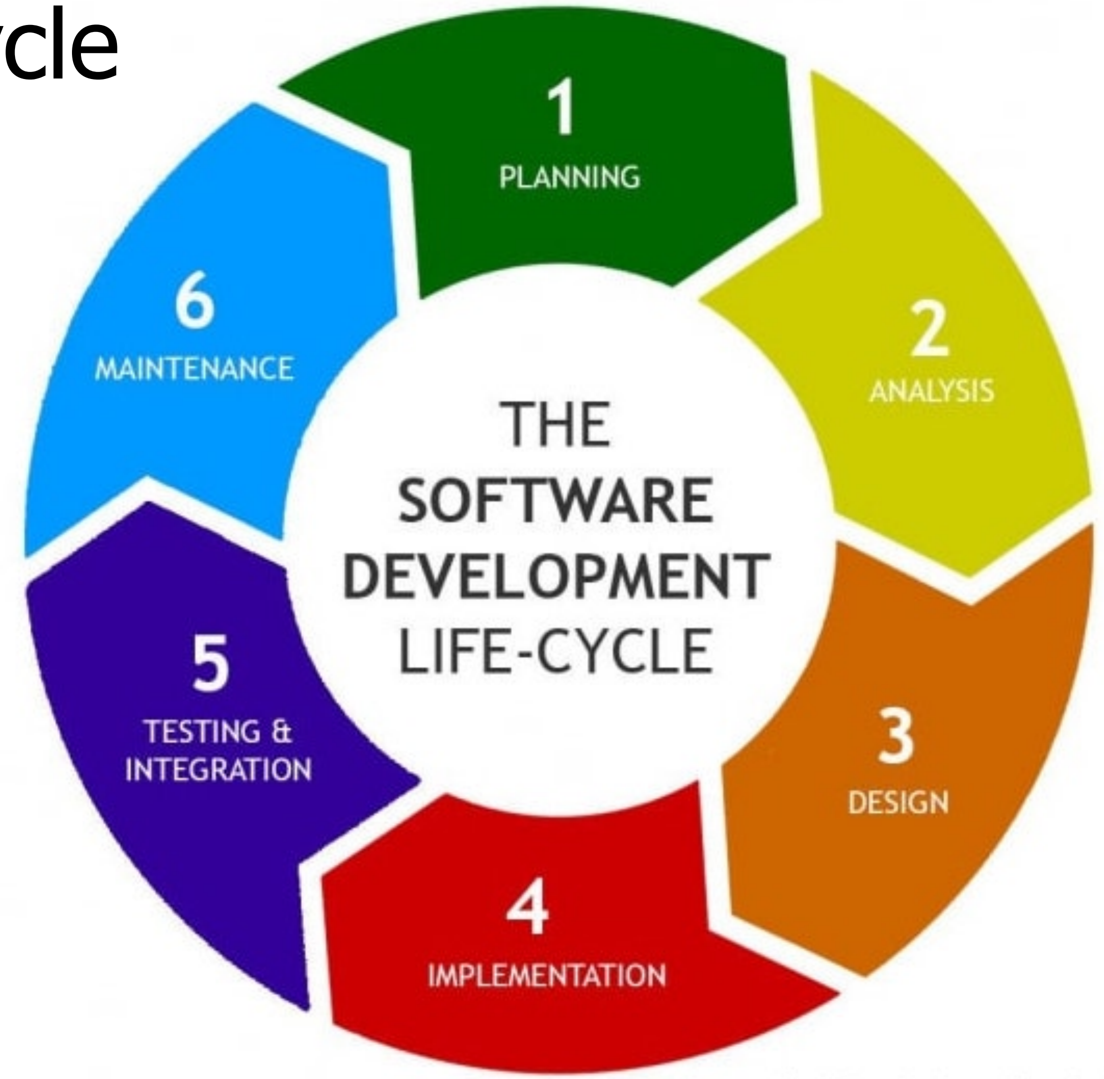
UCDAVIS

Software Development Lifecycle

The Software Development Lifecycle (SDLC) is a framework to describe software development. You adhere to the SDLC whether you realize it or not as every step must be completed, even if implicitly.

Secure thinking can be integrated at every step:

1. **Planning** (what are we thinking of doing?): What information will we be storing? Who should have access?
2. **Analysis** (what specifically will we build?): What risks exist based on the requirements?
3. **Design** (what is our technical architecture?): How secure is our architecture?
4. **Implementation** (coding)
5. **Testing & Integration**: Test your security. Automate common penetration tests.
6. **Maintenance**: Patches, bug fixes, adherence to future standards



Credit: https://arkbauer.com/blog/software-development-life-cycle-sdlc/

UCDAVIS

Key Concept
Protection and Availability Levels

UCDAVIS

Image credit: Christoph Scholz (**CC BY-SA 2.0**)

Protection and Availability Levels

- **Protection Levels and Availability Levels** classify services based on severity.
- PL and AL thresholds may dictate which policy items to follow, i.e. public data may be treated differently than highly sensitive, proprietary data.

- Protection is concerned with **data privacy and integrity**.
  - Compromised data may violate privacy, expose intellectual property, and more.
- Availability is concerned with **service efficiency and disruption**.
  - Service disruptions are expensive.

UCDAVIS

Protection Levels

- **P1 (Minimal, Public)**: Information readily available to the public. Primary concern is unauthorized modification.
- **P2 (Low, Internal)**: Information not intended for public use, but not covered by statue, regulations, etc. Compromise may result in minor damage, small financial loss, or have minor privacy impacts.
- **P3 (Moderate, Proprietary)**: Compromise may result in moderate fines, civil actions. Privacy violations are moderate, financial loss is moderate, legal action may result. This includes lower risk items that, when combined, represent increased risk.
- **P4 (High, Statutory)**: Significant harm in all forms, including reputation. May include civil or criminal violations.

UCDAVIS

Availability Levels

- **A1 (Minimal)**: <u>Minimal</u> impact or minor financial loss.
- **A2 (Low)**: <u>Minor</u> losses or inefficiencies.
- **A3 (Moderate)**: <u>Moderate</u> financial losses and/or reduced customer service.
- **A4 (High)**: <u>Significant</u> financial loss, major impairment of the Location. IT Resources required by statutory, regulatory, or legal obligations are major drivers for this risk level.

Requirements
For Secure Software Development

UCDAVIS

Image credit: Yuri Samoilov (**CC BY 2.0**)

Secure Software Requirements

1. Software Development Process
2. Input Validation
3. Exception and Error Handling
4. Cross Site Scripting (XSS) and Invalidated Redirects/Forwards
5. Insecure Direct Object References
6. Logging
7. TLS and Secure APIs
8. Credentials/Passphrases
9. Session and Logout
10. Federated Authentication/SAML/Shibboleth
11. File Management
12. Secure Configuration
13. Documentation
14. Version Control

What follows are highlighted items from each section, not the entirety of the section itself.

UCDAVIS

Software Development Process

- Change Management
  - For smaller teams, talk to someone besides yourself to ensure you haven't forgotten any details about a change you're about to make.
- Requirements for P3 or higher, A3 or higher:
  - Code reviews which include cyber-security risks
  - **Check for common security mistakes**
  - Reviews should be performed by a senior developer, or preferably an independent one
  - Use automated tools for static and dynamic analysis

UCDAVIS

Input Validation

- **Validate all input** (data type, string length, foreign key existence – assume you'll receive anything and everything)
- Expect GET parameters and form elements to be manipulated in ways Javascript or browsers cannot protect against
- Use **parameterized SQL statements** or equivalents
- Set **autocomplete=off** to prevent caching of sensitive information
- **URL/URI must never include**:
    - Credentials
    - Access tokens, serial numbers, record numbers
    - PII (names, SSNs, birthdates)
    - PHI (Medical Record Number, diagnosis, condition)

Exception and Error Handling

- **Do not reveal native frameworks**
- Set up custom error pages
- Ensure all errors and exceptions provide users/administrative staff enough information to diagnose the issue
- Never use empty catch blocks
- **Log all exceptions and errors**

UCDAVIS

Cross Site Scripting (XSS) and Invalidated Redirects/Forwards

- **Do not send untrusted data data or code to the browser.**
- **Test for common XSS attacks**
  - **https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md**

Insecure Direct Object References

- Use access control checks for any object, file, database key, etc.

Logging

- Log:
  - **Authentication success/failure**
  - **Use of privileged functions**
  - Administrative activities
  - System start/stop
- Log security-related events to a separate log file, if possible
- **Do not log sensitive information**
- Use log file naming conventions (rotation, location)
- **Monitor your log files in a central location**
- Ensure sufficient space is available for logs

UCDAVIS

TLS and Secure APIs

- **Force HTTPS and disable HTTP**
- **Use TLS 1.2 or later for machine-to-machine connections (including database connection)**
- Encrypt and authenticate APIs, extract tools, service busses
- **Separate public and private application APIs**
- Configure CORS to restrict invocation rights

UCDAVIS

Credentials/Passphrases

- Never store user passphrases
- **Never hardcode credentials**
- Protect service accounts with established tools and techniques
- **Implement lockout after 5 failed authentication attempts**
- Use secure protocols for credential/secrets exchange

Session and Logout

- **Ensure session timeout is implemented**
- Make sure session tokens are random and long
- Change session tokens at each logon and privilege escalation
- Delete, remove, and invalidate tokens on logout
- **Provide a logout function**, prominently displayed throughout the application
- Properly close records, delete temporary objects, and close operations on the server:
  - As part of logout
  - After a set period of inactivity
  - After a browser window close

UCDAVIS

Federated Authentication

- **Only use approved SSO providers**
- **Assign permissions to user groups, not individual users**
- Assign users to user groups
- Ensure a user group with no/minimal privileges exists
  - Assign new users to this group and make privilege escalation explicit
- Log all actions that grant users or groups permissions
- **Re-check authorization at every request**

File Management

- **Prevent/disable directory traversal/directory listing**
- **Filter web requests to block access to files with non-approved extensions** (e.g. .dll, .config, .java, .cs, etc.)
- For AL3 or higher, set a directory quota or similar control
- Ensure shared volumes/file shares have appropriate access restrictions limited to the application service account

UCDAVIS

Secure Configuration

- Use secure configuration libraries, tools
- Perform vulnerability scans
- **Run components with the least privileges possible**
- **Execute SQL with the least privilege possible**
- Define, implement, and maintain secure settings (i.e. do not rely on default settings)
- **Secure the configuration used by the application**
- Set Header and Content-Security-Policy when possible (helps protect against XSS)

UCDAVIS

Documentation

(Applies to P3/A3 or higher)

- Record the steps taken to protect confidentiality concerns, integrity concerns, availability concerns
- Track defects and fixes to defects
- Note/record testing processes and how testing processes mitigate cyber risk
- List components used and the security state of the components

Version Control

(Applies to P3/A3 or higher)

- **Use a software version control system or repository**
- Deploy test and production applications from the version control system or repository
- **Tag/label versions so they can be extracted at a later time**
- Configure version control to prevent/detect unauthorized changes

References

- IS-3: https://policy.ucop.edu/doc/7000543/BFB-IS-3
- IS-3 Secure Software Development Standard: https://security.ucop.edu/files/documents/guides/secure-software-development-standard.pdf
- UC Davis IS-3 information page: https://iet.ucdavis.edu/learn-more

UCDAVIS

Thank you

UCDAVIS