# Software Development Security

## Understand and integrate security in the Software Development Life Cycle (SDLC)

1. Feasibility
2. Project initialization
3. Requirements
4. Design
5. Develop
6. Test / Assess
7. Operate / Maintain
8. Dispose / Retire
9. Development methodologies - Waterfall, Sashim, Agile Software Dev
10. Maturity models - evaluating and improving the software development process
11. Operation and maintenance - the system is modified by the addition of hardware and software and by other events
12. Change management - tracks changes across an entire software development program.
13. Integrated product team - customer-facing group that focuses on the entire lifecyle of a product

## Identify and apply security controls in development environments

- Security of the software environments - contractors must be DFARS compliant. There is a Application Security Assessment STIG to ensure developers are developing with secure methods. Development environment is highly compromisable so there must be monitoring, anti-virus, patching, and security controls/hardening
- Configuration management as an aspect of secure coding - ensure automated builds with static code analysis on code base
- Security of code repositories - must have stric permissions for accessing all of source code. have data loss prevention so users can't take all code that has been worked on

## Assess the effectiveness of software security

- Auditing and logging of changes - must have proper version history and changes documented
- Risk analysis and mitigation - Perform risk analysis. how often are vulnerabilities and bugs being commited. how long do they take to fix when in production?

## Assess security impact of acquired software

There could be vulnerabilities and flaws in software that is acquired. There could even be malware or backdoors in them. There are various procurement tools to analyze the security of libraries or other products used

## Define and apply secure coding guidelines and standards

- Security weaknesses and vulnerabilities at the source-code level - there should be static analysis done to test for vulnerabilities. Fixing pentesting and fuzzing findings are much more expensive than fixing a

finding before commiting it back to master stream.
- Security of application programming interfaces - test interfaces for abuse cases/fuzzing
- Secure coding practices - ASD STIG, CERT SEI, OWASP Top 10, SANS Top 25

Home Page