California Department of Technology
Office of Information Security

(916) 445-5239
HTTPS://CDT.CA.GOV/SECURITY/

SECURE SOFTWARE
INFORMATION SHEET NO. 2
MAY 2017 (REVISED)

SOFTWARE SECURITY CHECKLISTS

Checklists are essential tools for the development of secure software. They frame a structured review and analyses process, and form the standard of excellence expected for the software product. Software security considerations supported with the use of checklists include:

- **Completeness:** Checking focuses on traceability among software product objects of various types including requirements, specifications, designs, code, and test procedures. Completeness analysis may be assisted by tools that trace the components of a product object of one type to the components of another type. Completeness analysis of predecessor and successor objects reveals what sections are missing and what fragments may be extra. A byproduct of the completeness analysis is a clear view of the relationship of requirements to the code product: straightforward (one to one), simple analysis (many to one), and complex (one to many).

- **Correctness:** Checking focuses on reasoning about programs by answering informal verification and correctness questions derived from the prime constructs of structured programming and their composite use in proper programs. Input domain and output range are analyzed for all legal values and all possible values. State data is similarly analyzed. Adherence to project-specified disciplined data structures is analyzed. Asynchronous processes and their interaction and communication are analyzed.

- **Style:** Checking is based on project-specified style guidance. This guidance is expected to call for block structured templates. Naming conventions and commentary are checked for consistency of use along with alignment, highlighting, and case. More advanced style guidance may call for templates for repeating patterns and semantic correspondence among software product artifacts of various types.

- **Rules of construction:** Checking focuses on the software system's architecture and the specific protocols, templates, and conventions used to carry it out. For example, these include interprocess communication protocols, tasking and concurrent operations, program unit construction, and data representation.

- **Multiple views:** Checking focuses on the various perspectives and view points required to be reflected in the software product. During execution many factors must operate harmoniously as intended including initialization, timing of processes, memory management, input and output, and finite word effects. In building the software product, packaging considerations must be coordinated including program unit construction, program generation process, and target machine operations. Product construction disciplines of systematic design and structured programming must be followed as well as interactions with the user, operating system, and physical hardware.

The following publicly-available programming and application security checklists are relatively widely used for checking for one or more of these characteristics. These checklists should help developers assess the key security aspect of their software at various stages of its life cycle.

- Department of Defense, Defense Information Systems Agency's *SRG/STIG Applicability Guide and Collection Tool* *http://iase.disa.mil/stigs/agct/Pages/index.aspx*

- Australian Computer Emergency Response Team, *AusCERT UNIX and Linux Security Checklist v3.0* (publicly released, ver. 25 July 2007) http://www.auscert.org.au/render.html?it=7289

- Open Web Application Security Project (OWASP), *OWASP Testing Guide v2* http://www.owasp.org/index.php/OWASP_Testing_Guide_v2_Table_of_Contents

- NASA, *Software Security Checklists* https://www.nasa.gov/accessibility/resources/res_checklist.html

- Visa U.S.A, *CISP Payment Application Best Practices Checklist* http://www.911software.com/CISP_Payment_Application_Best_Practices.pdf

- NIST, *National Checklist Program Repository* http://checklists.nist.gov/

CREDITS AND ACKNOWLEDGEMENTS:
Goertzel, Karen Mercedes, *et al, Security in the Software Lifecycle: Making Software Development Processes—and the Software Produced by Them—More Secure,* Draft Version 1.2 (August 2006), U.S. Department of Homeland Security