Australian Government
Australian Signals Directorate

ACSC Australian Cyber Security Centre

Australian Government Information Security Manual

JUNE 2021

Guidelines for Software Development

Application development

Types of application development

These guidelines are applicable to both traditional application development activities as well as mobile application development activities.

Development environments

Segregating development, testing and production environments can limit the spread of malicious code and minimises the likelihood of faulty code in a production environment.

Security Control: 0400; Revision: 5; Updated: Aug-20; Applicability: O, P, S, TS
Development, testing and production environments are segregated.

Security Control: 1419; Revision: 1; Updated: Sep-18; Applicability: O, P, S, TS
Development and modification of software only takes place in development environments.

Security Control: 1420; Revision: 3; Updated: Jun-21; Applicability: O, P, S, TS
Data in production environments is not used in testing or development environments unless the testing or development environments are secured to the same level as the production environments.

Security Control: 1422; Revision: 3; Updated: Sep-18; Applicability: O, P, S, TS
Unauthorised access to the authoritative source for software is prevented.

Secure software design

Threat modelling is an important part of secure software design. Threat modelling identifies at risk components of software, enabling security controls to be identified to reduce security risks.

Security Control: 1238; Revision: 3; Updated: Sep-18; Applicability: O, P, S, TS
Threat modelling and other secure design techniques are used to ensure that threats to software and mitigations to those threats are identified and accounted for.

Secure programming practices

Once a secure software design has been identified, secure programming practices should be followed during software development activities.

Security Control: 0401; Revision: 4; Updated: Oct-19; Applicability: O, P, S, TS
Platform-specific secure programming practices are used when developing software, including using the lowest privilege needed to achieve a task, checking return values of all system calls, validating all inputs and encrypting all communications.

Software testing

Software testing can lessen the risk of security vulnerabilities in software being introduced into a production environment. Software testing can be performed using both static testing, such as code analysis, as well as dynamic testing, such as input validation and fuzzing. Vulnerability scanning tools can also assist in the detection of known security vulnerabilities, such as out of date or vulnerable dependencies. Using an independent party for software testing will remove any bias that can occur when a software developer tests their own software.

Security Control: 0402; Revision: 3; Updated: Sep-18; Applicability: O, P, S, TS
Software is tested for security vulnerabilities by software developers, as well as an independent party, before it is used in a production environment.

Vulnerability disclosure program

Implementing a vulnerability disclosure program, based on responsible/coordinated disclosure, can assist organisations, vendors and service providers to improve the security of their products and services as it provides a way for security researchers, customers and members of the public to responsibly notify them of potential security vulnerabilities in a coordinated manner. Furthermore, following the verification and resolution of a reported security vulnerability, it can assist organisations, vendors and service providers in notifying their customers of any security vulnerabilities that have been discovered in their products and services and any recommended security patches, updates or mitigations.

A vulnerability disclosure program should include processes to receive, verify, resolve and report on security vulnerabilities disclosed by both internal and external sources. In support of this, a vulnerability disclosure policy should be made publicly available that covers:

- the purpose of the vulnerability disclosure program
- the types of security research that are allowed
- the types of security research that are not allowed
- how to report potential security vulnerabilities
- the actions that will be taken on receiving notification of potential security vulnerabilities and indicative timeframes for these actions
- any expectations regarding the public disclosure of verified security vulnerabilities
- any recognition finders of verified security vulnerabilities will receive.

Finally, the Australian Cyber Security Centre (ACSC) encourages security researchers, customers and members of the public to responsibility report security vulnerabilities directly with organisations, vendors and service providers. However, the ACSC recognises that this is not always practical, initial attempts at communication may be unsuccessful or the person making the report may not wish to do so directly. In such cases, security vulnerabilities can be reported to the ACSC as an independent coordinator at https://www.cyber.gov.au/acsc/report.

Security Control: 1616; Revision: 0; Updated: Aug-20; Applicability: O, P, S, TS
A vulnerability disclosure program is implemented to assist with the secure development and maintenance of products and services.

Further information

Further information on a secure development life cycle model, known as the Trustworthy Computing Security Development Lifecycle, is available at https://docs.microsoft.com/en-au/previous-versions/ms995349(v=msdn.10).

Further information on secure coding practices is available at https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=21274.

Further information on implementing a vulnerability disclosure program can be found in:

- Google's ***Starting a Vulnerability Disclosure Program*** at https://developers.google.com/android/play-protect/starting-a-vdp
- European Union Agency for Cybersecurity's ***Good Practice Guide on Vulnerability Disclosure*** at https://www.enisa.europa.eu/publications/vulnerability-disclosure
- Netherland's National Cyber Security Centre's ***Coordinated Vulnerability Disclosure: The Guideline*** at https://english.ncsc.nl/publications/publications/2019/juni/01/coordinated-vulnerability-disclosure-the-guideline
- Carnegie Mellon University's ***The CERT Guide to Coordinated Vulnerability Disclosure*** at https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=503330
- International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 29147:2018, ***Information technology – Security techniques – Vulnerability disclosure***, at https://www.iso.org/standard/72311.html
- ISO/IEC 30111:2019, ***Information technology – Security techniques – Vulnerability handling processes***, at https://www.iso.org/standard/69725.html.

Web application development

Protecting web applications

Even when a web application only contains public data, there remains a need to protect the integrity and availability of the data processed by the web application and the system it is hosted on.

Web application frameworks

Web application frameworks can be leveraged by software developers to enhance the security of a web application while decreasing development time. These resources can assist software developers to securely implement complex components such as session management, input handling and cryptographic operations.

Security Control: 1239; Revision: 3; Updated: Sep-18; Applicability: O, P, S, TS
Robust web application frameworks are used to aid in the development of secure web applications.

Web application interactions

Hypertext Transfer Protocol Secure (HTTPS) is Hypertext Transfer Protocol (HTTP) using Transport Layer Security (TLS) encryption. The use of HTTPS for web applications ensures that not only are individuals' interactions with web applications kept confidential, but the integrity of their interactions are also maintained.

Security Control: 1552; Revision: 0; Updated: Oct-19; Applicability: O, P, S, TS
All web application content is offered exclusively using HTTPS.

Web application input handling

Most web application security vulnerabilities are caused by the lack of secure input handling. It is essential that web applications do not trust any input such as the website address and its parameters, Hypertext Markup Language (HTML) form data, cookie values and request headers without validating or sanitising it. Examples of validation and sanitisation include:

- ensuring a telephone form field contains only numerals
- ensuring data used in a Structured Query Language query is sanitised properly
- ensuring Unicode input is handled appropriately.

Security Control: 1240; Revision: 2; Updated: Sep-18; Applicability: O, P, S, TS
Validation and/or sanitisation is performed on all input handled by a web application.

Web application output encoding

The likelihood of cross-site scripting and other content injection attacks can be reduced through the use of contextual output encoding. The most common example of output encoding is the use of HTML entities. Performing HTML entity encoding causes potentially dangerous HTML characters such as '<', '>' and '&' to be converted into their encoded equivalents '<', '>' and '&'.

Output encoding is particularly useful where external data sources, which may not be subject to the same level of input filtering, are output to users.

Security Control: 1241; Revision: 3; Updated: Sep-18; Applicability: O, P, S, TS
Output encoding is performed on all output produced by a web application.

Web browser-based security controls

Web browser-based security controls such as Content-Security-Policy, HTTP Strict Transport Security (HSTS) and X-Frame-Options can be leveraged by web applications to help protect themselves and their users. This is achieved via the use of security policy in response headers which users' web browsers apply according to the defined security policy. Since the security controls are applied via response headers, it makes it possible to apply the security controls to legacy or proprietary web applications where changes to the source code are impractical.

Security Control: 1424; Revision: 3; Updated: Oct-19; Applicability: O, P, S, TS
Web applications implement Content-Security-Policy, HSTS and X-Frame-Options response headers.

Open Web Application Security Project

The Open Web Application Security Project (OWASP) provides a comprehensive resource to consult when developing web applications.

Security Control: 0971; Revision: 7; Updated: Apr-19; Applicability: O, P, S, TS
The OWASP Application Security Verification Standard is followed when developing web applications.

Further information

Further information on auditing of web applications can be found in the event logging and auditing section of the ***Guidelines for System Monitoring***.

Further information on implementing TLS can be found in the Transport Layer Security section of the ***Guidelines for Cryptography***.

Further information on web application security can be found in the following ACSC publications:

- ***Implementing Certificates, TLS and HTTPS*** at https://www.cyber.gov.au/acsc/view-all-content/publications/implementing-certificates-tls-and-https
- ***Protecting Web Applications and Users*** at https://www.cyber.gov.au/acsc/view-all-content/publications/protecting-web-applications-and-users
- ***Securing Content Management Systems (CMS)*** at https://www.cyber.gov.au/acsc/view-all-content/publications/securing-content-management-systems.

Further information on web application security is available in the OWASP ***Application Security Verification Standard*** at https://wiki.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project.

Further information on common web application frameworks for different programming languages, including a comparison of their functionality, is available at https://en.wikipedia.org/wiki/Comparison_of_web_frameworks.