Information security manual

Guidelines for software development

Last updated: March 2026

Software development fundamentals

Introduction to software development

This section applies to software development activities for traditional applications (including user applications and server applications), artificial intelligence applications, mobile applications and web applications. Additional sections of these guidelines should also be consulted depending on the type of software development being undertaken. For example, the 'Web application development' section of these guidelines should be consulted for additional controls applicable to web applications.

Development, testing, staging and production environments

Segregating development, testing, staging and production environments, and their associated data, can minimise the likelihood of faulty or malicious content being introduced into a production environment. Furthermore, protecting the authoritative source for software is critical to preventing malicious content being surreptitiously introduced into software.

Control: ISM-0400; Revision: 6; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Development, testing, staging and production environments are segregated.

Control: ISM-1419; Revision: 1; Updated: Sep-18; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Development and modification of software only takes place in development environments.

Control: ISM-1420; Revision: 5; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Data from production environments is not used in non-production environments unless the non-production environment is secured to at least the same level as the production environment.

Authoritative source for software

Software developers need to ensure that they are using a secure authoritative source for software as part of their development environment, as doing so can reduce the security risks related to unauthorised access to source code, source code tampering and other possible cyber supply chain attacks on software artefacts. In doing so, the authoritative source for software should be able to provide sufficient access control and event logging for access to, and modification of, any source code and software artefacts.

Control: ISM-2023; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
An authoritative source for software is established and maintained.

Control: ISM-2024; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The authoritative source for software is used for all software development activities.

Control: ISM-1422; Revision: 3; Updated: Sep-18; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Unauthorised access to the authoritative source for software is prevented.

Control: ISM-1816; Revision: 0; Updated: Dec-22; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Unauthorised modification of the authoritative source for software is prevented.

Issue tracking

To providing visibility into source code changes, and support traceability, software developers should be able to link all changes, including security updates, change or feature updates, or bug fixes to a request or decision that is documented within an issue tracking solution.

Control: ISM-2025; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
An issue tracking solution is used to link software development tasks to security issues and decisions, change or feature requests, programming issues, or bug fixes.

Software artefacts

Software artefacts can include compiled code, third-party libraries, configuration files and any other file consumed by the software development process. Software artefacts, both internally and externally developed, could include malicious content or weaknesses designed to be exploited at a later stage. To assist in preventing the introduction of malicious content or weak software artefacts into an authoritative source for software, all software artefacts should be scanned for malicious content and undergo security testing (where appropriate) before being stored in the authoritative source for software.

Control: ISM-2026; Revision: 1; Updated: Mar-26; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All software artefacts are scanned for malicious content before being imported into the authoritative source for software.

Control: ISM-2027; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All software artefacts are verified by a digital signature, or a secure hash provided over a secure channel, before being imported into the authoritative source for software.

Control: ISM-2028; Revision: 1; Updated: Mar-26; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All software artefacts are tested to detect known weaknesses using static application security testing (SAST), dynamic application security testing (DAST) or software composition analysis (SCA), depending on the software artefact type, before being imported into the authoritative source for software.

Control: ISM-2102; Revision: 0; Updated: Mar-26; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Existing software artefacts in the authoritative source for software are periodically tested to detect known weaknesses using SAST, DAST or SCA, depending on the software artefact type, throughout the software development life cycle.

Control: ISM-2029; Revision: 1; Updated: Sep-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The authoritative source for software restricts the use and import of third-party libraries and software components to trustworthy sources.

Control: ISM-2030; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Scanning is used during commits to identify plain text or encoded secrets and keys, which are then blocked from being stored in the authoritative source for software.

Build solution

The build process for software can unintentionally allow vulnerabilities in software to make it into production environments. This security risk can be reduced by ensuring the build solution uses all available security features and that all automated tested is completed as part of the build process.

Control: ISM-2031; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Compilers, interpreters and build tools (including pipelines) that provide security features to improve executable file security are implemented and such security features are used.

Control: ISM-2032; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The build solution ensures that all automated testing is completed without warnings, alerts or errors before building software artefacts.

Secure software development

The use of Secure by Design principles and practices, including secure programming practices and either memory-safe programming languages (such as C#, Go, Java, Ruby, Rust and Swift) or less preferably memory-safe programming practices, along with threat modelling and mitigation of common security risks, is an important part of secure software development. In addition, providing mechanisms to assist in determining the authenticity and integrity of software, while configuring it in a secure manner, can assist with cyber supply chain security activities.

As part of secure software development, software should be designed and built to be Secure by Default, that is, the software is secure 'out of the box' with little to no additional setup or configuration required to achieve an adequate level of security. Importantly, all built-in security measures, such as multi-factor authentication and event logging, are included in the base product at no extra cost to consumers.

Control: ISM-2033; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All software security requirements are documented, stored securely and maintained throughout the software development life cycle.

Control: ISM-2034; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Security design decisions are documented and reviewed throughout the software development cycle.

Control: ISM-2035; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Security roles, responsibilities and knowledge requirements required to support the software development life cycle are identified and documented.

Control: ISM-2036; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Security responsibilities for software developers are identified and documented.

Control: ISM-2037; Revision: 1; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software developers that lack sufficient cyber security knowledge and skills required for their projects or tasks undertake suitable training on secure software development and programming practices.

Control: ISM-2038; Revision: 1; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A software developer cyber security knowledge and skills register is implemented and maintained.

Control: ISM-0401; Revision: 8; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Secure by Design principles and practices are followed throughout the software development life cycle.

Control: ISM-1238; Revision: 6; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Threat modelling is used in support of the software development life cycle.

Control: ISM-2039; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The software threat model is reviewed throughout the software development life cycle to ensure it reflects the as-built software and any changes to the threat environment.

Control: ISM-2040; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Secure programming practices for the chosen programming language are used for software development.

Control: ISM-2041; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Memory-safe programming languages, or less preferably memory-safe programming practices, are used for software development.

Control: ISM-2042; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Secure by Default principles and practices are followed throughout the software development life cycle, including by ensuring that all built-in security measures are included and enabled in the base product at no extra cost to consumers.

Control: ISM-1780; Revision: 1; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
SecDevOps practices are used for software development.

Control: ISM-2043; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software is architected and structured to support readability and maintainability.

Control: ISM-1796; Revision: 1; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Files containing executable content are digitally signed by a certificate with a verifiable chain of trust as part of software development.

Control: ISM-1797; Revision: 1; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Installers, patches and updates are digitally signed or provided with cryptographic checksums as part of software development.

Control: ISM-2044; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software has no default credentials; however, if credentials are required, they are created on first install by the installing organisation.

Control: ISM-2045; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Application backwards compatibility does not compromise any security measures or features.

Control: ISM-2046; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Where software allows user impersonation, sensitive data is not logged and appropriate permissions are set.

Control: ISM-2047; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Where software allows an authentication factor to be reset, the user is notified of the reset through a secondary channel.

Control: ISM-2048; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Where software supports multiple user roles, non-administrative users are prevented from altering their profile permissions or privileges.

Control: ISM-2049; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
When user permissions or credentials are changed, software forces all impacted users to re-authenticate.

Control: ISM-2050; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
When digital signatures are processed by software, they are validated against a certificate trust chain and checked for revocation using a Certificate Revocation List or with the Online Certificate Status Protocol.

Control: ISM-2051; Revision: 1; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software generates sufficient event logs to support the detection of cyber security events.

Control: ISM-2052; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Event logs produced by software ensure that any sensitive data is protected.

Control: ISM-1798; Revision: 2; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Secure configuration guidance, in the form of a hardening guide or loosening guide, is produced and made available to consumers as part of software development.

Control: ISM-2053; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
End of life procedures for software, covering how to remove the software and how to archive or destroy any user accounts and data, are produced and made available to consumers.

Software bill of materials

A software bill of materials is a list of open source and commercial software components used in software development. This can assist software developers in ensuring they are not using software components with known vulnerabilities. It can also assist in providing greater cyber supply chain transparency for consumers of software by allowing for easier identification and management of security risks associated with individual software components used by software.

Control: ISM-2054; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
If a software bill of materials is available for imported third-party software components, it is used during software development to ensure such software components have no known vulnerabilities.

Control: ISM-1730; Revision: 0; Updated: Dec-21; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A software bill of materials is produced and made available to consumers of software.

Cryptographic bill of materials

A cryptographic bill of materials acts as an extension of a software bill of materials by communicating information about relevant cryptographic dependencies and implementations that are relied upon by software. At a high level, a cryptographic bill of materials will map the use of cryptography to the security function it performs and where it performs it. In practice, a cryptographic bill of materials will capture details on cryptographic libraries, algorithms, protocols, parameters and configurations. This can be used to ensure software components are providing support for standardised implementations of cryptographic algorithms that are approved by the Australian Signals Directorate (ASD).

While the format for a cryptographic bill of materials is not yet subject to formal standardisation, there are emerging tools to assist with their generation and management. This is particularly important for software developers in managing their migration to post-quantum cryptographic standards as part of software development activities.

Control: ISM-2082; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
If a cryptographic bill of materials is available for imported third-party software components, it is used

during software development to ensure such software components provide support for standardised implementations of ASD-Approved Cryptographic Algorithms.

Control: ISM-2083; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A cryptographic bill of materials is produced and made available to consumers of software.

Software build provenance

A software build provenance is a manifest that describes the software components used in the build process, the set of commands executed during the build process, any relevant environmental context for the build process and the materials that were generated by the build process. A software build provenance assists in providing greater cyber supply chain transparency for consumers by allowing for the build process to be verified and, if needed, recreated.

Control: ISM-2055; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
If a software build provenance is available for imported third-party software components, it is used during software development to ensure such software components are built to an appropriate standard.

Control: ISM-2056; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A software build provenance is produced and made available to consumers of software.

Network application programming interfaces

Network application programming interfaces (APIs) can facilitate the exchange of data between computing devices. As such, common security risks associated with their use should be mitigated during their development, especially for network APIs that are accessible over the internet. In particular, this includes mitigating poorly secured network APIs that facilitate unauthorised modification of data or access to data not authorised for release into the public domain. In such cases, ensuring authentication and authorisation of clients is performed when clients call network APIs can assist in mitigating unauthorised modification of, or access to, data. Finally, centrally logging and analysing network API use can assist in detecting malicious behaviour and contributing to investigations following cyber security incidents.

Control: ISM-1818; Revision: 2; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Authentication and authorisation of clients is performed when clients call network APIs that facilitate modification of data and are accessible over the internet.

Control: ISM-2013; Revision: 0; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Authentication and authorisation of clients is performed when clients call network APIs that facilitate modification of data but are not accessible over the internet.

Control: ISM-1817; Revision: 2; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Authentication and authorisation of clients is performed when clients call network APIs that facilitate access to data not authorised for release into the public domain and are accessible over the internet.

Control: ISM-2014; Revision: 0; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Authentication and authorisation of clients is performed when clients call network APIs that facilitate access to data not authorised for release into the public domain but are not accessible over the internet.

Control: ISM-1910; Revision: 1; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Network API calls that facilitate modification of data, or access to data not authorised for release into the public domain, and are accessible over the internet, are centrally logged.

Control: ISM-2015; Revision: 0; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Network API calls that facilitate modification of data, or access to data not authorised for release into the public domain, but are not accessible over the internet, are centrally logged.

Software input handling

Many vulnerabilities in software are caused by a lack of secure input handling. As such, it is essential that software does not trust any input, such as website addresses and their parameters, Hypertext Markup Language (HTML) form data, cookie values, or request headers, without performing validation or sanitisation. 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 and ensuring Unicode input is handled appropriately.

Control: ISM-1240; Revision: 5; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Validation and sanitisation are performed on all input received over the internet by software.

Control: ISM-2016; Revision: 1; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Validation and sanitisation are performed on all input received over a local network by software.

Control: ISM-2057; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All input validation rules are documented, matched in code and tested with both positive and negative unit testing or integration testing.

Control: ISM-2058; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Data sources and serialised data inputs are validated before being deserialised.

Control: ISM-2059; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
File uploads or input are restricted to specific file types, with malicious content scanning occurring prior to file access, file execution or file storage.

Software interaction with databases

Structured Query Language (SQL) injection attacks, facilitated by the use of dynamically generated queries, are a significant threat to the confidentiality, integrity and availability of database contents. Specifically, SQL injection attacks can allow malicious actors to steal database contents, modify database contents, delete an entire database or even in some circumstances gain control of the underlying database server. Furthermore, when database queries from software fail, they may display detailed error information about the structure of databases. This can be used by malicious actors to further tailor their SQL injection attacks.

Finally, centrally logging and analysing all queries to databases from software that are initiated by users can assist in monitoring the security posture of databases, detecting malicious behaviour and contributing to investigations following cyber security incidents.

Control: ISM-1275; Revision: 2; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All queries to databases from software are filtered for legitimate content and correct syntax.

Control: ISM-1276; Revision: 5; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Parameterised queries or stored procedures, instead of dynamically generated queries, are used by software for database interactions.

Control: ISM-1278; Revision: 5; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software is designed or configured to provide as little error information as possible about the structure of databases.

Control: ISM-1536; Revision: 3; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All queries to databases from software that are initiated by users, and any resulting crash or error messages, are centrally logged.

Software security testing

Software security testing can assist software developers in identifying vulnerabilities in their software. In doing so, testing should be designed to be repeatable and scalable while ensuring vulnerabilities in software are identified and remediated as early as possible. Such software security testing should include SAST, DAST and SCA in order to achieve comprehensive test coverage. As part of software security testing, software developers may also wish to replicate testing within typical consumer environments. Furthermore, software developers may choose to use independent testing to assist with removing any potential for bias that might occur when they test their own software. Finally, software needs to be comprehensively tested for vulnerabilities during development, prior to all releases and periodically in order to attempt to identify any previously unidentified vulnerabilities.

Control: ISM-0402; Revision: 9; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software is comprehensively tested for vulnerabilities, using SAST, DAST and SCA prior to its initial release, any subsequent releases and periodically in order to attempt to identify any previously unidentified vulnerabilities.

Control: ISM-2060; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Code reviews are utilised to ensure software meets Secure by Design principles and practices as well as secure programming practices.

Control: ISM-2061; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Software developer-supported security-focused peer reviews are conducted on all critical and security-focused software components.

Control: ISM-2062; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Unit testing and integration testing, covering both positive and negative use cases, are used to ensure code quality and security.

Vulnerability disclosure program

Implementing a vulnerability disclosure program, based on responsible disclosure, can assist an organisation to improve the security of their products and services as it provides a way for internal and external parties to responsibly notify the organisation of vulnerabilities in a coordinated manner. Furthermore, following the resolution of reported vulnerabilities, it can assist an organisation in notifying their consumers of vulnerabilities that have been discovered in their products and services, and any patches, updates or vendor mitigations that should be applied.

A vulnerability disclosure program should include processes and procedures for receiving, identifying, documenting, validating and prioritising vulnerabilities disclosed by internal and external parties. In support of this, a vulnerability disclosure policy should be made publicly available that covers:

- the purpose of the vulnerability disclosure program
- types of security research that are and are not allowed
- how to report any vulnerabilities
- actions, and associated timeframes, upon notification of vulnerabilities

- expectations regarding the public disclosure of vulnerabilities
- any recognition or reward for finders of vulnerabilities.

Finally, ASD encourages security researchers and other members of the public to responsibly report vulnerabilities directly to an organisation. However, ASD 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, vulnerabilities can be reported to ASD as an independent coordinator. Note, under ASD's limited use obligation, information voluntarily provided to ASD about vulnerabilities cannot be used for regulatory purposes.

Control: ISM-1616; Revision: 0; Updated: Aug-20; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A vulnerability disclosure program is implemented to assist with the secure development and maintenance of products and services.

Control: ISM-1755; Revision: 1; Updated: Dec-22; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A vulnerability disclosure policy is developed, implemented and maintained.

Control: ISM-1756; Revision: 1; Updated: Dec-22; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Vulnerability disclosure processes, and supporting vulnerability disclosure procedures, are developed, implemented and maintained.

Control: ISM-1717; Revision: 3; Updated: Sep-24; Applicable: NC, OS, P, S, TS; Essential 8: N/A
A 'security.txt' file is hosted for each of an organisation's internet-facing website domains to assist in the responsible disclosure of vulnerabilities in the organisation's products and services.

Reporting and resolving vulnerabilities

Following the identification of vulnerabilities in software, either via internal software security testing or external security researchers, it is important that they are publicly disclosed in a responsible and timely manner. This allows users of impacted software to determine their risk exposure as part of their own cyber supply chain risk management activities. In doing so, such vulnerabilities should be resolved in a timely manner. In support of this, root cause analysis should be performed and, to the greatest extent possible, entire vulnerability classes should be remediated.

If vulnerabilities cannot be resolved in a timely manner via patches or updates, advice should be provided on how, to the greatest extent possible, the likelihood of vulnerabilities being exploited can be reduced, the impact of vulnerabilities being exploited can be reduced or both.

Control: ISM-1908; Revision: 2; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Vulnerabilities identified in software are publicly disclosed in a responsible and timely manner, including with Common Weakness Enumeration and Common Platform Enumeration information.

Control: ISM-1754; Revision: 4; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Vulnerabilities identified in software are resolved in a timely manner.

Control: ISM-1909; Revision: 1; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
In resolving vulnerabilities, root cause analysis is performed and, to the greatest extent possible, entire vulnerability classes are remediated.

Software event logging

Centrally logging and analysing security-relevant usage, error messages and crashes for software can assist in monitoring the security posture of software, detecting malicious behaviour and contributing to investigations following cyber security incidents.

Control: ISM-1911; Revision: 3; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Security-relevant usage, error messages and crashes for software are centrally logged.

Further information

Further information on a secure software development framework can be found in National Institute of Standards and Technology Special Publication 800-218, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*.

Further information on Secure by Design and Secure by Default principles and practices can be found in the following publications:

- ASD's *Secure by Design foundations*
- ASD's *IoT Secure by Design guidance for manufacturers*
- United Kingdom's National Cyber Security Centre's *Secure development and deployment guidance*
- United Kingdom's Central Digital and Data Office's *Secure by Design Principles* and *Secure by Design Activities*
- United States' Cybersecurity & Infrastructure Security Agency's *Safe Software Deployment: How Software Manufacturers Can Ensure Reliability for Customers*
- United States' Cybersecurity & Infrastructure Security Agency's *Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software*.

Further information on secure programming practices is available from the Carnegie Mellon University's Software Engineering Institute.

Further information on the need for memory-safe programming languages can be found the following publications:

- United States' Cybersecurity & Infrastructure Security Agency's *The Case for Memory Safe Roadmaps: Why Both C-Suite Executives and Technical Experts Need to Take Memory Safe Coding Seriously*
- United States' Cybersecurity & Infrastructure Security Agency's *Exploring Memory Safety in Critical Open Source Projects*
- United States' National Security Agency's *Software Memory Safety*
- United States' National Security Agency and Cybersecurity & Infrastructure Security Agency's *Memory Safe Languages: Reducing Vulnerabilities in Modern Software Development*.

Further information on cyber supply chain transparency, and recommended content for a software bill of materials, can be found in the United States' National Telecommunications and Information Administration's *The Minimum Elements For a Software Bill of Materials (SBOM)* publication.

Further information on software bill of materials can also be found in the United States' Cybersecurity & Infrastructure Security Agency's *A Shared Vision of Software Bill of Materials (SBOM) for Cybersecurity* publication.

Further information on strong authentication can be found in the 'Authentication hardening' section of the *Guidelines for system hardening*.

Further information on software security testing can be found on the Open Worldwide Application Security Project's (OWASP) DevSecOps Guidelines and Source Code Analysis Tools websites.

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

- Google's *Starting a Vulnerability Disclosure Program*
- Carnegie Mellon University's Software Engineering Institute's *The CERT Guide to Coordinated Vulnerability Disclosure*
- International Organization for Standardization/International Electrotechnical Commission 29147:2018, *Information technology – Security techniques – Vulnerability disclosure*
- International Organization for Standardization/International Electrotechnical Commission 30111:2019, *Information technology – Security techniques – Vulnerability handling processes*.

Further information on developing a vulnerability disclosure policy is available from the disclose.io project to assist an organisation with their implementation.

Further information on recommended contents for a 'security.txt' file is available to assist an organisation with their implementation.

Further information on reporting vulnerabilities to ASD as an independent coordinator, including ASD's limited use obligation, is available from ASD.

Further information on event logging can be found in the 'Event logging and monitoring' section of the *Guidelines for system monitoring*.

Artificial intelligence application development

Introduction to artificial intelligence application development

This section describes the controls applicable to artificial intelligence application development and extends upon the 'Software development fundamentals' section of these guidelines.

Secure artificial intelligence application development

Secure coding resources for software developers should be followed when developing artificial intelligence applications. In addition, artificial intelligence applications should be designed to reduce their attack surface with artificial intelligence models being stored in a non-executable file format that does not allow arbitrary code execution.

Control: ISM-2084; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Artificial intelligence-specific documentation, including model and system cards (or equivalent artefacts), is used to document model characteristics, system architectures, use cases and security risks.

Control: ISM-2072; Revision: 1; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Artificial intelligence models are stored in a non-executable file format that does not allow arbitrary code execution.

Control: ISM-2085; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The exposure of exact artificial intelligence model confidence scores in API responses or user interfaces is prevented.

Control: ISM-2103; Revision: 0; Updated: Mar-26; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Organisational data generated, collected or processed by artificial intelligence applications is not used for training, fine-tuning or improving artificial intelligence models unless informed and explicit consent has been obtained from data owners in advance.

Artificial intelligence model poisoning

Artificial intelligence applications, especially when embedded in other applications, risk exposing sensitive data, proprietary algorithms or confidential details through their output. This can result in unauthorised data access, privacy violations and intellectual property breaches.

Control: ISM-2086; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The source and integrity of artificial intelligence models, structures and weights are verified.

Control: ISM-2087; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The source and integrity of training data for artificial intelligence models is verified.

Control: ISM-2088; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Data validation and verification techniques are used to ensure the reliability and accuracy of training data used by artificial intelligence models.

Unbounded consumption

Cyber attacks designed to disrupt services, deplete a target's financial resources or steal intellectual property by cloning an artificial intelligence model's behaviour depend on unbound consumption vulnerabilities in order to succeed.

Control: ISM-2089; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Artificial intelligence model performance metrics are monitored and anomalies are investigated.

Control: ISM-2090; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Rate limiting is applied to inference queries for artificial intelligence models.

Control: ISM-2091; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Resource limits are enforced for artificial intelligence models.

Excessive agency

Excessive agency vulnerabilities allow for damaging actions to be performed in response to unexpected, ambiguous or manipulated outputs from artificial intelligence applications, regardless of what is causing an artificial intelligence application to malfunction.

Control: ISM-2092; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Access control policies are implemented to enforce fine-grained permissions for artificial intelligence applications.

Control: ISM-2093; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Role-based access controls are implemented for artificial intelligence applications to restrict access to sensitive data.

Prompt injection

Prompt injection vulnerabilities exist in how artificial intelligence applications process user prompts, and how such input may force artificial intelligence models to incorrectly pass prompt data to other parts of the model, potentially causing them to violate guidelines, generate harmful content, enable unauthorised access or influence critical decisions.

Control: ISM-1924; Revision: 1; Updated: Sep-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Generative artificial intelligence applications evaluate user prompts to detect and mitigate adversarial inputs or suffixes designed to elicit unintended behaviour or assist in the generation of sensitive or harmful content.

Sensitive data exposure and improper output

Artificial intelligence applications, especially when embedded in other applications, risk exposing sensitive data, proprietary algorithms or confidential details through their output. This can result in unauthorised data access, privacy violations and intellectual property breaches. In addition, improper output handling vulnerabilities occur when insufficient validation, sanitisation and handling of outputs generated by artificial intelligence applications occur before they are passed downstream to other applications.

Control: ISM-2094; Revision: 0; Updated: Dec-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Content filtering is implemented by artificial intelligence applications to detect and block sensitive data exposure and improper output.

Further information

Further information on large language model application security risks can be found in the *OWASP Top 10 for Large Language Model Applications version 2025* publication.

Further information on artificial intelligence security risks can be found in the following ASD publications:

- *An introduction to artificial intelligence*
- *Artificial intelligence and machine learning: Supply chain risks and mitigations*
- *Engaging with artificial intelligence*.

Further information on artificial intelligence security risks can also be found in the following publications:

- MITRE's *Adversarial Threat Landscape for Artificial-Intelligence Systems*
- National Institute of Standards and Technology AI 100-2 E2023, *Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations*
- United Kingdom's National Cyber Security Centre and United States' Cybersecurity & Infrastructure Security Agency's *Guidelines for secure AI system development*
- United States' National Security Agency's *AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems*

- United States' National Security Agency's *Deploying AI Systems Securely: Best Practices for Deploying Secure and Resilient AI Systems*.

Mobile application development

Introduction to mobile application development

This section describes the controls applicable to mobile application development and extends upon the 'Software development fundamentals' section of these guidelines.

Secure mobile application development

OWASP provides comprehensive resources for software developers that should be followed when developing mobile applications.

Control: ISM-1922; Revision: 0; Updated: Jun-24; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The OWASP Mobile Application Security Verification Standard is used in the development of mobile applications.

Further information

Further information on mobile application security can be found in the *OWASP Mobile Application Security Verification Standard version 2.1.0* publication.

Web application development

Introduction to web application development

This section describes the controls applicable to web application development and extends upon the 'Software development fundamentals' section of these guidelines.

Secure web application design and development

Web application frameworks can be leveraged by software developers to enhance the security of web applications while decreasing development time. These resources can assist in securely implementing complex software functions, such as session management, input handling and cryptographic operations. In addition, OWASP provides comprehensive resources for software developers that should be followed when developing web applications.

Control: ISM-1239; Revision: 4; Updated: Mar-22; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Robust web application frameworks are used in the development of web applications.

Control: ISM-0971; Revision: 8; Updated: Mar-23; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The OWASP Application Security Verification Standard is used in the development of web applications.

Control: ISM-1849; Revision: 0; Updated: Mar-23; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The OWASP Top 10 Proactive Controls are used in the development of web applications.

Control: ISM-1850; Revision: 0; Updated: Mar-23; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The OWASP Top 10 are mitigated in the development of web applications.

Control: ISM-2063; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
If supported, web application session cookies set the HttpOnly flag, Secure flag and the SameSite flag by default.

Control: ISM-2064; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Web application session cookies contain only digitally signed opaque bearer tokens.

Control: ISM-2065; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Web application session cookies using opaque bearer tokens that are not digitally signed use non-sequential random identifiers with a minimum of 128 bits of entropy, preferably 256 bits of entropy.

Control: ISM-2066; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Web application sessions are centrally managed server side.

Control: ISM-2067; Revision: 0; Updated: Jun-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Web applications that support Single Sign On equally support Single Logout.

Web security policy response headers

Web security policy response header measures, such as Content-Security-Policy, Hypertext Transfer Protocol Strict Transport Security and X-Frame-Options, can be applied by web browsers to help protect themselves. This is achieved by web server software specifying security policy in response headers which web browsers then apply.

Control: ISM-1424; Revision: 5; Updated: Mar-25; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Content-Security-Policy, Hypertext Transfer Protocol Strict Transport Security and X-Frame-Options are specified by web server software via security policy in response headers.

Web application interactions

Hypertext Transfer Protocol Secure (HTTPS) is the Hypertext Transfer Protocol secured by Transport Layer Security (TLS) encryption. The use of HTTPS for web applications ensures that interactions with web applications are confidential and that the integrity of such interactions are also maintained.

Control: ISM-1552; Revision: 0; Updated: Oct-19; Applicable: NC, OS, P, S, TS; Essential 8: N/A
All web application content is offered exclusively using HTTPS.

Web application programming interfaces

Web APIs can facilitate the exchange of data between computing devices. As such, common security risks associated with their use should be mitigated during their development.

Control: ISM-1851; Revision: 0; Updated: Mar-23; Applicable: NC, OS, P, S, TS; Essential 8: N/A
The OWASP API Security Top 10 are mitigated in the development of web APIs.

Web application output encoding

The likelihood of cross-site scripting and other content injection attacks can be reduced through the use of output encoding. In particular, output encoding is useful when external data sources, which may not be subject to the same level of input filtering, are output to users. The most common example of output encoding is the conversion of potentially dangerous HTML characters into their encoded equivalents, such as ‘<’, ‘>’ and ‘&’ into ‘<’, ‘>’ and ‘&’.

Control: ISM-1241; Revision: 4; Updated: Mar-22; Applicable: NC, OS, P, S, TS; Essential 8: N/A
Output encoding is performed on all output produced by web applications.

Further information

Further information on web application security can be found in the *OWASP Application Security Verification Standard 5.0.0* and *OWASP Top 10 Proactive Controls 2024* publications.

Further information on web application security risks can be found in the *OWASP Top 10:2025* publication.

Further information on implementing HTTPS can be found in ASD's *Implementing certificates, TLS, HTTPS and opportunistic TLS* publication.

Further information on using TLS in HTTPS can be found in the 'Transport Layer Security' section of the *Guidelines for cryptography*.

Further information on web API security can be found in the *OWASP API Security Top 10 2023* publication.

Disclaimer

The material in this guide is of a general nature and should not be regarded as legal advice or relied on for assistance in any particular circumstance or emergency situation. In any important matter, you should seek appropriate independent professional advice in relation to your own circumstances.

The Commonwealth accepts no responsibility or liability for any damage, loss or expense incurred as a result of the reliance on information contained in this guide.

Copyright

Use of the Coat of Arms

The terms under which the Coat of Arms can be used are detailed on the Department of the Prime Minister and Cabinet website (https://www.pmc.gov.au/resources/commonwealth-coat-arms-information-and-guidelines).

Australian Government
Australian Signals Directorate

ASD AUSTRALIAN SIGNALS DIRECTORATE
ACSC Australian Cyber Security Centre