
Security Standard - Software Development (SS-003)



Department
for Work &
Pensions

Chief Security Office

Date: 30/07/2026

This Software Development Security Standard is part of a suite of standards, designed to promote consistency across the Department for Work and Pensions (DWP), and supplier base with regards to the implementation and management of security controls. For the purposes of this standard, the term DWP and Department are used interchangeably.

Technical security standards form part of the DWP Digital Blueprint which is a living body of security principles, architectural patterns, code of practice, practices and radars, that aim to support Product Delivery Units (PDUs) and suppliers in delivering the DWP and HMG Digital Strategy. Security standards and policies considered appropriate for public viewing are published here:

<https://www.gov.uk/government/publications/dwp-procurement-security-policies-and-standards>

Technical security standards cross-refer to each other where needed, so can be confidently used together. They contain both mandatory and advisory elements, described in consistent language (see table below).

(Important note for screen reader users.) Paragraphs that contain a '**must**' statement, and therefore denote a mandatory requirement, will contain the following statement after the heading:

(Important) this paragraph contains 'must' activities.

Table 1 – Terms

Term	Intention
must	denotes a requirement: a mandatory element.
should	denotes a recommendation: an advisory element.
may	denotes approval.
might	denotes a possibility.
can	denotes both capability and possibility.
is/are	denotes a description.

1. Contents

2. Revision History	4
3. Approval History.....	9
4. Compliance	10
5. Exceptions Process	10
6. Audience	11
7. Accessibility Requirements.....	11
8. Introduction.....	11
9. Purpose	13
10. Scope.....	13
11. Minimum Technical Security Measures.....	15
11.1 General Coding Practices	15
11.2 Input Validation	23
11.3 Output Encoding	25
11.4 Password Management.....	26
11.5 Authentication	28
11.6 API Management Requirements.....	32
11.7 Session Management.....	35
11.8 Access Control.....	38
11.9 Cryptographic Practices.....	40
11.10 Error Handling and Logging	41
11.11 Data Protection	43
11.12 File Management	44
11.13 Application Lifecycle	45
11.14 Use of Artificial Intelligence Coding Assistants (AICAs) & Agentic Workflows	46
12 Appendices	49
Appendix A Security Outcomes	49
Appendix B Internal References	55
Appendix D Abbreviations	57
Appendix E Definition of Terms	58
Appendix F Accessibility artefacts	59

2. Revision History

Version	Author	Description	Date
1.0		First published version	26/06/2017
1.1		Multiple revisions in all sections following feedback from SMEs.	07/10/2017
2.0		Full update in line with current best practices and standards; Updated Intro, purpose, audience, scope; added reference to CIS v8 security controls Added NIST CSF references 11.1.4 'must'; vulnerability assessments 11.1.5 Added ref to open source and ESRM supplier assurance 11.1.8 Integrity checking 11.1.11 Privilege escalation must be logged 11.1.13 Sanitise user data 11.1.14 Remove access 11.1.19 Automated testing 11.1.22 Do not use 'referrer' header 11.1.23 Secrets management 11.1.24 Code signing 11.2.2 Removed threat modelling; trusted/untrusted 11.2.3 UTF8 by default, EBCDIC by exception 11.2.7 Do not pass user supplied data for processing directly	14/06/2023

		11.2.9 Allowlist 11.2.10 Moved redirects from previous entry 11.2.11 Added RFC3986 11.3.1 Performed 11.4.12 NCSC password guidance 11.4.13 Change password reset limits 11.5.2 OAuth 11.5.3 standardised and federated 11.5.6 Added ref to Privileged User Access standard 11.5.14 Authenticate files before uploading 11.6 Sessions must be encrypted 11.6.10 Geo-locations 11.6.11 Secure Cookie Attribute; HttpOnly attribute; added ref to Use of Crypto standard 11.6.12 Separation of duties 11.6.14 Added ref to Use of Crypto standard 11.7.13 Re-apply permissions on re-enabled accounts 11.7.14 Non-human, automated accounts 11.7.15 ACP templates may be used 11.8.1 Examples added 11.8.5 FIPS140 validation	
--	--	--	--

		11.8.6 Added ref to PKI standard 11.9 STDOUT and STDERR definition 11.9.1 SIEM tooling compatibility – JSON, XML, CSV. 11.9.2 Success and failure logging; must include 11.9.7 Engineering users 11.10 Added ref to Secure Coding Guidelines 11.11.3 Allowlist 11.11.5 Transformation / Conversion 11.12 Added ref to Supply Chain Levels for Software Artifacts framework 11.12.3 All environments	
2.1		All NIST references reviewed and updated to reflect NIST 2.0 All security measures reviewed in line with risk and threat assessments Approval history - Review period changed to up to 2 years Compliance - Policy as Code Scope – Legacy systems 11.1.3 Expand security stories 11.1.5 Software testing 11.1.7 Reference added to section 11.6 API management 11.1.8 typosquatting	30/07/2026

		11.1.9 Every code commit 11.1.0 Where variables are not automatically initialised 11.1.13 Input data sanitisation 11.1.15 Must; 11.1.18 Iterative threat modelling 11.1.19 Testing 11.1.20 Application logic and business rules 11.1.23 Secrets management 11.1.24 Secure Delivery 11.1.25 Patching 11.1.26 SBOM 11.1.27 Infrastructure as Code 11.1.28 Code vulnerability detection 11.2.2 Must 11.2.6 Must; homographs / homoglyphs 11.2.7 Must not 11.2.10 Regular expressions 11.2.11 Homographs / homoglyphs 11.3.2 Where available 11.3.4 Code injection 11.3.5 File path and symlink plug-in validation 11.4.1 Approved hashing algorithms 11.4.2 Password generation and transmission 11.4.4, 11.4.7, 11.4.8, 11.4.10, 11.4.11, 11.4.13, 11.4.14 Must	
--	--	---	--

		11.4.6 Escalating time interval 11.5.11 at rest and in transit 11.5.12 HTTPS 11.5.14 Must; public code repositories 11.5.15 Sensitive code 11.5.16 Credential security 11.5.17 No hard-coded credentials 11.5.18 credential lifetime 11.5.19 Secrets mgmt. controls 11.6 New section added for API management 11.7.3 & 11.7.7 Must 11.7.10 Must not 11.8.1 Must 11.8.2 Must; where possible 11.8.6 Must 11.8.16 Environment access controls 11.9.4 RNGs derived from approved crypto algorithms 11.10.2 Security gates 11.10.6 Must 11.11.1 Must 11.12.1, 11.12.2, 11.12.3 Must 11.13.5 Build and deployment integrity 11.14 AI Coding Assistants and Agentic Workflows Internal refs – Patching standard; Approved algorithms External References – NCSC - Securing HTTP-based APIs	
--	--	---	--

		Definitions – Homographs/homoglyphs; NCSC Password Mgmt	
--	--	---	--

3. Approval History

Version	Name	Role	Date
1.0		Chief Security Officer	26/06/2017
1.1		Chief Security Officer	07/10/2018
2.0		Chief Security Officer	14/06/2023
2.1		Chief Security Officer	30/07/2026

This document is continually reviewed to ensure it is updated in line with risk, business requirements, and technology changes, and will be updated at least every 2 years - the current published version remains valid until superseded.

4. Compliance

Compliance with this standard will be verified through various methods, including but not limited to;

- controls tests performed by 1st line teams and by 2nd line activities (e.g. security testing teams)
- security assurance activities to ensure that Architectural Design and delivery are appropriate and aligned to applicable Authority Security Standards. [See Security Assurance Strategy – Ref. K].
- independent external audit
- Policy as Code (PaC) may be utilised to codify and enforce security defaults, check configuration conformance, identify configuration drift and automate remediation.

Results of these will be fed back to the appropriate Authority Risk and System Owners.

5. Exceptions Process

(Important) this paragraph contains ‘must’ activities.

In this document the term “**must**” is used in bold letters to indicate a mandatory security measure. Any exceptions to the application of this standard, or where specific security measures cannot be adhered to, **must** be presented to the Authority. This **must** be carried out prior to deployment and managed through the design caveats or exception process.

Such exception requests will invoke the Risk Management process to clarify the potential impact of any deviation to the configuration detailed in this standard.

Exceptions to the standard **must** be maintained on a risk register for accountability, traceability, and security governance reporting to senior management.

6. Audience

This document is intended for, but not necessarily limited to, technical architects, engineers, developers, security teams, project teams, including suppliers engaged in the design, development, implementation and operation of systems, services and applications.

7. Accessibility Requirements

(Important) this paragraph contains 'must' activities.

Users of this standard **must** consider accessibility design requirements as appropriate. Further information on accessibility standards can be found in Appendix F.

8. Introduction

(Important) this paragraph contains 'must' activities.

This Software Development Security Standard defines the minimum technical security measures that **must** be implemented when developing and maintaining software for use within the Department for Work and Pensions (DWP). These requirements **must** also be considered against code that may not have been developed by the Authority (e.g. taken on from third party suppliers, not open source third party libraries or repositories). This document should be read in conjunction with the DWP Information Security Policy, Acceptable Use Policy and DWP Open Source Policy.

This set of requirements will support the iterative agile development lifecycle and will be matured and refined as projects and development efforts feedback information to mature it.

There is a risk from threat actors using malicious code to exploit vulnerabilities in Authority systems, applications and business practices. This presents a significant risk to the Confidentiality, Integrity and Availability of Authority Systems and the information they process, store and exchange.

All areas of the Authority should assume that, where there is a business requirement to exchange data with business partners or import content from other information sources including open source, information technology systems and information assets are highly likely to be affected by attacks via malicious code of a direct and also indirect nature.

As this standard only provides minimum measures, they **should** be exceeded as appropriate depending on the threats and risks that need to be addressed, the sensitivity of the data, and in keeping with latest security enhancements.

The security measures are derived from industry best practice i.e. guidance published by NIST, CIS and OWASP (see Appendix C for full list external references) and support the implementation of appropriate security controls as selected by the Authority or our third party providers, such as the CIS Critical Security Controls set. [see Appendix C External References]

Every effort has been made to ensure the security measures are vendor and technology agnostic as far as possible; this is to ensure greater applicability of the standard regardless of the technologies used. The security measures **may** be implemented in different ways, depending on the technology choices and business requirements in question.

The aim of this standard is to:

- ensure security controls that are applicable to software development are implemented consistently across the Authority and by third party providers where applicable.
- mitigate risks from common threats and vulnerabilities associated with software development, to an acceptable level for operation.
- support the achievement of security outcomes described in Appendix A.

Technical security standards ultimately support the achievement of security outcomes sought by the Authority. They set the expectations for what needs to be done to achieve them and why, and provide an objective, measurable statement of the Authority's existing security posture in a number of important areas. The outcomes are based on the official NIST sub-categories where possible to ensure close alignment with the NIST Cyber Security Framework (CSF), and are enabled by the implementation of controls from the CIS Critical Security Controls set. [see Appendix C External References]. Those relevant to the subject of each standard can be found in Appendix A of every technical security standard.

9. Purpose

The purpose of this standard is to ensure that Authority systems and services are designed, configured, deployed, and managed consistently to protect against typical threats at the OFFICIAL tier.

This standard also serves to provide a baseline in which assurance and compliance activities can be carried out, so that the Authority can be assured that security obligations are being met or exceeded.

10. Scope

(Important) this paragraph contains 'must' activities.

This standard applies to all software development across all systems and services within the Authority and supplier base (contracted third party providers), for the purposes of delivering applications and services that handle Authority data.

This includes 'legacy' systems which may contain outdated software components used in development environments and remain of interest to threat actors for exploitation during software integration or testing phases.

With the advent of AI tools, specifically AI Coding Assistants, the software development lifecycle may be fundamentally altered. Whilst they may increase engineering velocity, they may also introduce novel vulnerabilities, including "package hallucination" (where AI suggests malicious, non-existent libraries), the exfiltration of proprietary workspace context to external inference engines, and the deployment of insecure code patterns.

To align with the UK National Cyber Security Centre (NCSC) guidance on secure AI development and the OWASP Top 10 for Large Language Models 2025, the Authority **must** shift from implicit trust in developer tools to explicit, programmatic verification. A number of new requirements pertaining to the use of AI tools have been added to this standard.

Any queries regarding the security measures laid out in this standard **should** be sent to the Authority.

11. Minimum Technical Security Measures

The following section defines the minimum security measures that **must** be implemented to achieve the security outcomes described in Appendix A. For ease of reference, the official NIST sub-category ID is provided against each security measure e.g. PR.PT-3, to indicate which outcome(s) it contributes towards. Refer to Appendix A for full description of outcomes. (Important) this paragraph contains 'must' activities.

11.1 General Coding Practices

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.1.1	The Software Development Security Standard (i.e. this document) must be in place, made available to the development team and shared in accessible area.	GV.PO-01 GV.PO-02
11.1.2	Rules for the development of software and systems must be established, documented and applied to developments within the organisation. This will be made available to projects and other parties with a need to know.	GV.PO-01 GV.PO-02

11.1.3	Projects delivered through an agile project delivery must have Security Stories – these are agile backlog items used to plan, implement, and track security requirements. They ensure security is built into the system from the start rather than applied later on in the development lifecycle. To ensure that security considerations are addressed throughout the development lifecycle, Security stories and relevant acceptance criteria must be added to development teams’ backlogs upon project initiation. These may be identified during Discovery and Alpha, but must be implemented by the time the development moves into Beta.	ID.IM-03
11.1.4	Tested and approved managed code must be used rather than creating new unmanaged code for common tasks. Code analysis and vulnerability scanning tools are available for this purpose. Vulnerability assessments must be completed on a regular basis in line with the Technical Vulnerability Management Policy [Ref. L].	PR.PR-06 ID.RA-01
11.1.5	Supply Chain Agreements must be in place that third parties have been assessed via the ESRM Supplier Security Assurance process where applicable. Please note this may not be possible for open source products, but a commercial wrapper must be in place if available. Open source and third party software must be tested for bugs and vulnerabilities in a suitable test environment prior to being deployed organisation wide.	GV.SC-03 GV.SC-05 GV.SC-07

11.1.6	Outsourced development (including suppliers and external third parties) activities must follow the DWP Software Development security standard (i.e. this document).	GV.PO-01 GV.PO-02 GV.SC-05
11.1.7	Utilise task specific built-in APIs to conduct operating system tasks. Do not allow the application to issue commands directly to the Operating System, especially through the use of application initiated command shells. See section 11.6 API Management for more information on APIs.	PR.PR-06
11.1.8	The integrity of interpreted code, libraries, executable files, and configuration files must be checked, using automated tooling where available. Care must be taken when selecting these to ensure that the names reference legitimate components and not similar looking packages containing malware ('typosquatting').	PR.DS-01 PR.PR-06
11.1.9	Ensure that every code commit is thread-safe.	PR.PR-06
11.1.10	Where software being used does not automatically perform it, explicitly initialise all variables and other data stores, either during declaration or just before the first usage.	PR.PR-06
11.1.11	In cases where the applications or functions must run with elevated privileges, raise privileges as late as possible, and drop them as soon as possible. These must be logged.	PR.AA-05

11.1.12	Avoid calculation errors by understanding the programming language's underlying representation and how it interacts with numeric calculation. Pay close attention to byte size discrepancies, precision, signed/unsigned distinctions, truncation, conversion and casting between types, "not-a-number" calculations, and how the language handles numbers that are too large or too small for its underlying representation.	PR.PR-06
11.1.13	All input data must be sanitised before use.	PR.PR-06 DE.CM-09
11.1.14	Restrictions must be in place, and access removed, to prevent unauthorised users from generating new code or altering existing code in Authority code repositories. Staff leaving the Authority (including contractors) must have their permissions removed in a timely manner.	PR.AA-05 PR.DS-01
11.1.15	Review all secondary applications, third party code and libraries to determine business necessity and validate safe functionality, as these can introduce new vulnerabilities and potential back doors. This activity must also be performed periodically after the initial review.	ID.RA-01 ID.RA-04 DE.CM-01
11.1.16	Safe updating must be implemented, if the production application will utilise automatic updates, using cryptography with verification of signatures.	PR.DS-02 PR.PR-06
11.1.17	Systems updates must use encrypted channels to transfer the code from the host server, where appropriate (mandatory for write operations).	PR.DS-02

11.1.18	Threat Modelling must be carried out as part of the design phase to include all application functionality and APIs, and then reassessed iteratively as code / products evolve.	ID.RA-03
11.1.19	**There must be a planned Security Test prior to go live – this must include automated testing at minimum covering Dynamic Application Security Testing (DAST), fuzzing, interactive testing, and runtime instrumentation plus additional testing where necessary.	ID.IM-02
11.1.20	The application logic must implement the business rules and the code must be reviewed and applied in line with the original ticket/story and acceptance criteria.	ID.AM-03
11.1.21	Limit the number of transactions a single user or device can perform in a given period of time. The transactions/time must be above the actual business requirement, but low enough to deter automated attacks.	PR.IR-01 PR.IR-03
11.1.22	The "referrer header" must not be used as an authorisation check, as it can be spoofed.	PR.AA-05
11.1.23	Secrets such as keys, certificates, or credentials must utilise the Authority-approved secrets management system in accordance with SS-007 Use of Cryptography Security Standard [Ref. A] as applicable.	PR.DS-01
11.1.24	All artifacts must be served via secure channels with cryptographic validation (signed artifacts, signed containers, https/TLS etc.)	PR.DS-02

11.1.25	All developer tools and environments must be patched up to date in line with SS-033 Patching & Exposure Management Standard [Ref. M] to mitigate against vulnerabilities, including; • Development and build servers; • integrated development environments; • version control systems; • software development infrastructure • cloud environments • network components • Vulnerabilities must be disclosed upon discovery to System Owners and affected end users.	ID.AM-08 PR.PS-02
---------	--	----------------------

11.1.26	A complete Software Bill of Materials (SBoM) must be generated, maintained, and made available upon request for all bespoke applications, microservices, and integrated third-party components. For modern environments (utilising automated CI/CD pipelines), SBoM generation must be fully automated during the build phase using standard machine-readable formats (e.g., CycloneDX or SPDX) and continuously monitored via Software Composition Analysis (SCA) tooling. For legacy environments or suppliers where automated pipelines are not technically feasible, an SBoM must still be maintained. This may be generated via standalone scanning tools or maintained as a managed baseline document, but it must be comprehensively reviewed and updated prior to every major and minor software release. Regardless of environment, all identified third-party dependencies must be assessed against known vulnerability feeds (e.g., NCSC, CISA KEV etc.) to inform patching priorities in line with SS-033 Patching & Exposure Management standard [Ref. M].	ID.AM-02
---------	--	----------

11.1.27	Infrastructure as Code (IaC) Security: This clause applies exclusively where application infrastructure is provisioned programmatically. IaC templates must be treated as application source code and subjected to the same secure development lifecycle controls, including version control, peer review, and adherence to the principle of least privilege. Where automated deployment pipelines exist, IaC repositories must undergo automated static analysis (linting and security scanning) prior to deployment to detect misconfigurations (e.g., publicly exposed storage). Where deployment is manual, IaC templates must be subjected to a documented security peer review against SS-023 Cloud Computing Security Standard [Ref. C] prior to execution.	PR.PS-06
11.1.28	Where supported by the repository architecture, teams should implement local pre-commit hooks to automate the "shift-left" detection of plaintext credentials and static code vulnerabilities before they are committed to the local index	PR.PS-06

* For the full security control set on cloud related issues please refer to SS-023 Cloud Computing Security Standard [Ref. C].

** For full details of the required security testing please refer to SS-027 Security Testing Standard [Ref. D].

11.2 Input Validation

(Important) This section contains 'must' statements.

Reference	Minimum Technical Security Measures	NIST ID
11.2.1	All data inputs must be validated on a trusted system (e.g., the server, not the client).	ID.RA-01
11.2.2	Data from all data sources must be validated, whether trusted or untrusted.	ID.RA-01
11.2.3	Character sets must use UTF-8 by default, for all sources of input by default. Other character sets (such as EBCDIC) may be used by exception only, and must be recorded.	PR.PS-06
11.2.4	Data must be encoded to a common character set before being validated.	PR.PS-06
11.2.5	All validation failures must result in input rejection.	PR.PS-06
11.2.6	Where UTF-8 extended character sets are used, validation must be carried out after UTF-8 decoding is completed, to ensure no homographs/homoglyphs are present.	PR.PS-06
11.2.7	User supplied data must not be passed directly to any dynamic include function i.e. prevent malicious code from affecting the logic of the function. Validation for all client provided data before processing, including all parameters, URLs and HTTP header content (e.g. Cookie names and values) must be in place. This is to include automated post backs from embedded code.	PR.PS-05 PR.PS-06

11.2.8	Header values in both requests and responses containing only ASCII characters must be verified.	PR.PS-06
11.2.9	All input must be validated against an "allow list" of allowed characters.	PR.PS-06
11.2.10	Validation must be in place for; • expected data types • data range • data length • redirects • regular expressions	PR.PS-06
11.2.11	If any potentially hazardous characters must be allowed as input, you must ensure that you implement additional controls like URI encoding, secure task specific APIs and accounting for the utilisation of that data throughout the application. If your standard validation routine cannot address the following inputs for example, then they must be checked where appropriate; - Check for null bytes (%00) - Check for new line characters (%0d, %0a, \r, \n) - Check for "dot-dot-slash" (../ or ..\) path alterations characters. In cases where UTF-8 extended character set encoding is supported, address alternate representation like: %c0%ae%c0%ae/ - Check for homographs/homoglyphs (Utilise canonicalisation to address double encoding or other forms of obfuscation attacks). Please refer to RFC3986 [see External References] for further information.	PR.PS-06

11.3 Output Encoding

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.3.1	All encoding must be performed on a trusted system (e.g., the server, not the client).	PR.PS-06
11.3.2	Where available, a standard, tested routine for each type of outbound encoding must be used by the delivery team.	PR.PS-06
11.3.3	Contextual output encoding: encode output data returned to the client that originated outside the application's trust boundary. HTML entity encoding is one example, but there are other use cases like SQL queries, XML, LDAP.	PR.PS-06
11.3.4	Software must be coded to minimise the risk of code injection.	PR.PS-05 PR.PS-06
11.3.5	File paths and symbolic links ('symlinks') must be validated anywhere that file descriptors are used e.g. confirm existence of the file, that its actually a file etc.	PR.PS-06

11.4 Password Management

(Please refer to NCSC guidance on password management, and also SS-001 pt.1 Access and Authentication Security Standard [Ref. E]).

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.4.1	Password hashing must be implemented on a trusted system (e.g., the server, not the client) and utilise a hashing algorithm in line with the Authority Approved Cryptographic Algorithms document [Ref. N].	PR.AA-04
11.4.2	Where available, an Authority-approved secrets manager must be used to generate passwords, but if passwords need to be transmitted, they must be sent over an encrypted connection or as encrypted data, in line with SS-007 Use of Cryptography Security Standard [Ref. A].	PR.DS-02
11.4.3	Enforcement of password complexity requirements established by policy or regulation must be adhered to, in line with SS-001 pt.1 Access and Authentication Security Standard [Ref. E]. Authentication credentials must be sufficient to withstand attacks that are typical of the threats in the deployed environment. (e.g., requiring the use of alphabetic as well as numeric and/or special characters).	PR.AA-01 PR.AA-03
11.4.4	Password length requirements must be enforced as established by policy or regulation in line with SS-001 pt.1 Access and Authentication Security Standard [Ref. E]. Eight characters is commonly used, but 16 is better or consider the use of multi-word pass phrases.	PR.AA-01 PR.AA-03

1.4.5	Password entry must be obscured on the user's screen. (e.g., on web forms use the input type "password").	PR.AA-04
1.4.6	Enforce account disabling after 3 invalid login attempts. The account must be disabled for an escalating time interval, minimum time of 15 minutes, then doubling each time before accounts can be unlocked, sufficient to discourage brute force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed.	PR.AA-02 PR.AA-03
1.4.7	Password reset and changing operations must require the same level of controls as account creation and authentication.	PR.AA-01
1.4.8	If using email based resets, these must only send email to a pre-registered address with a temporary link/password.	PR.AA-01
1.4.9	Temporary passwords and links must have a short expiration time e.g. 10 minutes.	PR.AA-01
1.4.10	The changing of temporary passwords must be enforced on the next use.	PR.AA-01
1.4.11	Users must be notified when a password is changed.	PR.AA-01
1.4.12	Password changes and prevention of password re-use must follow NCSC guidance on password management, see External References.	PR.AA-01
1.4.13	Passwords must only be reset up to 3 times a day before the account is locked, to prevent attacks on password re-use.	PR.AA-01

1.4.14	Monitoring must be implemented to identify attacks against multiple user accounts, utilising the same password. This attack pattern is used to bypass standard lockouts, when user IDs can be harvested or guessed.	PR.AA-01 DE.CM-09
--------	--	----------------------

11.5 Authentication

(SS-007 Use of Cryptography Security Standard can be referred to for specifics on encryption [Ref. A])

(Important) this paragraph contains ‘must’ activities.

Reference	Minimum Technical Security Measures	NIST ID
11.5.1	All authentication controls must be enforced on a trusted system (e.g., the server, not the client).	PR.AA-03
11.5.2	Applications must establish and utilise standard, tested, authentication protocols or mechanisms e.g. API gateway, OAuth.	PR.AA-03
11.5.3	A standardised and federated approach for all authentication controls, including libraries that call external authentication services must be used where possible and available. Authentication logic must be segregated from the resource being requested and use redirection to and from the centralised authentication control.	PR.AA-03
11.5.4	All authentication controls must fail securely.	PR.AA-03
11.5.6	All administrative and account management functions must be in line with SS-001 pt.2 Privileged User Access Security Standard [Ref. F].	PR.AA-05

11.5.7	If the application manages a credential store, it must be managed in accordance with the SS-007 Use of Cryptography Security Standard [Ref. A] and SS-001 pt.1 Access and Authentication Security Standard [Ref. E].	PR.AA-01 PR.DS-01
11.5.8	Authentication data must be validated only on completion of all data input, especially for sequential authentication implementations.	PR.AA-03
11.5.9	Authentication failure responses must not indicate which part of the authentication data was incorrect. (For example, instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both).	PR.AA-03
11.5.10	Authentication must be used for connections to external systems that involve sensitive information or functions.	PR.AA-03
11.5.11	Authentication credentials for accessing services external to application/s must be encrypted both at rest and in transit, and stored in a protected location on a trusted system (e.g., the server, not the client). An Authority-approved secrets management system must be utilised for this, in line with SS-007 Use of Cryptography security standard [Ref. A]. The source code is NOT a secure location.	PR.AA-03 PR.DS-01 PR.DS-02
11.5.12	HTTPS POST (or PUT) must be used for requests to transmit authentication credentials.	PR.AA-03 PR.DS-02

11.5.13	The last use (successful or unsuccessful) of a user account must be reported to the user at their next successful login, for both internal and external facing web applications.	PR.AA-02 PR.AA-03
11.5.14	Where possible, authentication must be required before allowing a file to be uploaded.	PR.AA-03
11.5.15	Sensitive code (e.g. configuration data, business logic, calculation logic etc.) must not be uploaded to public repositories.	PR.DS-01
11.5.16	Credentials must be generated in a secure environment and not exported from where they have been generated. This is especially the case for credentials based on public key cryptography. If a credential needs to be exported (normally the case if symmetric keys are used), then this must be done using secure transit, and the process must be automated wherever possible.	PR.AA-01 PR.DS-02
11.5.17	Credentials must not be hard-coded directly into source code.	PR.AA-01
11.5.18	The lifetime of a credential must be minimised and set to the amount of time appropriate to the use case and threat, with defined expiry limits. The use of long-lived credentials must be avoided.	PR.AA-01

11.5.19	To enforce the prohibition of hard-coded credentials (as per 11.5.17), development teams and suppliers must implement active preventative and detective controls to ensure secrets do not enter version control; Primary Control (where CI/CD or modern VCS is used): Automated secret scanning must be implemented within the pipeline to fail builds or block commits if unencrypted credentials, API keys, or private cryptographic keys are detected. Compensating Control (for legacy environments lacking automated CI/CD): Where automated scanning is not technically supported, development teams must evidence a strict, documented manual peer-review gate (e.g., mandatory four-eyes principle) dedicated specifically to checking for embedded secrets prior to code merge, supplemented by periodic static codebase scanning using standalone tools.	PR.AA-04
---------	--	----------

11.6 API Management Requirements

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.6.1	High level designs must be threat modelled to understand any potential threats early during the design process. API controls must be implemented, such as; • Rate limiting / throttling • Schema validation at gateway / mesh • Anomaly detection • Circuit breakers • Protection against Broken Object Level Authorisation / Insecure Direct Object References APIs must be security tested in line with the threat model. See NCSC's 'Securing HTTP-based APIs' [External References].	ID.RA-03 ID.RA-04
11.6.2	APIs must be well documented, recorded in an asset register, and version controlled.	ID.AM-02
11.6.3	Every API endpoint must be authenticated and must use replay-resistant credentials, and all data validated before any processing. This may be achieved by using public key cryptography in line with SS-002 PKI & Key Management security standard [Ref. G], or by using signed JSON Web Tokens. Basic authentication must not be used.	PR.AA-03

11.6.4	Access policies must be defined and provisioned to an access server or API gateway capable of fine-grained control, enforcing least privilege permissions, denying access by default, and validating permissions on every request. These policies must implement robust authorisation mechanisms (e.g., OAuth 2.0 with appropriate scope validation, attribute-based access control) and include checks designed to prevent common authorisation vulnerabilities such as Broken Object Level Authorisation (BOLA) and Broken Function Level Authorisation (BFLA), as highlighted in the OWASP API Security Top 10.	PR.AA-01 PR.AA-05
11.6.5	Each service must have a totally unique API key for calling another service. This key must comprise a unique Service ID and a User ID at minimum. API keys must never be uploaded to code repositories.	PR.PS-06
11.6.6	API flow control must be implemented, for example via throttling or rate limiting, to limit the effects of excessive requests but must allow for spikes in usage from legitimate users. Repeated API authentication failures may be an indicator of excessive requests and must be alerted on.	PR.PS-06 PR.IR-03 PR.IR-04
11.6.7	Services must only be able to access messaging channels required for their function; Access to any given messaging channel must be limited to functionality required (such as read only, write, etc).	PR.AA-05

11.6.8	All data received via APIs (including parameters, headers, payloads) must undergo rigorous input validation against a defined schema to prevent injection attacks, Server-Side Request Forgery (SSRF), and other payload-based attacks.	PR.PS-05 PR.PS-06
11.6.9	APIs providing access to Authority data or services must undergo regular, specific security testing (using appropriate SAST, DAST, IAST tools and manual penetration testing) throughout the development lifecycle and before deployment, focusing on vulnerabilities outlined in the OWASP API Security Top 10. Testing requirements are further detailed in SS-027 Security Testing Standard [Ref. D].	ID.RA-01
11.6.10	API session tokens or access tokens must adhere to the session security principles outlined in section 11.7, including appropriate expiry, secure storage, and consideration for token binding where feasible.	PR.PS-06
11.6.11	Exposure to endpoints must be limited, for example by; • Removing unused endpoints; • Restricting access to privileged endpoints; • Block known malicious IP addresses / use an allowlist; • Minimising API responses so that they do not return excessive or unnecessary data, such as internal identifiers or debug details.	PR.PS-06

11.6.12	Centralised API logging and monitoring must be implemented to maintain visibility into the operations and interactions within the API ecosystem. At a minimum, the following types of events must be logged and alerted upon; • Login attempts; • Permission changes and privilege escalations; • Failed requests; • Password changes; • Account modifications	PR.PS-04 DE.CM-09 DE.AE-02
---------	---	---

11.7 Session Management

(Sessions must be encrypted in line with SS-007 Use of Cryptography Security Standard [Ref. A]).

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.7.1	Use the server or framework's session management controls. The application must only recognise these session identifiers as valid.	PR.PS-06
11.7.2	Session identifier creation must always be done on a trusted system (e.g., the server, not the client), and configured in the correct mode.	PR.PS-06
11.7.3	The domain and path for cookies containing authenticated session identifiers must be set to an appropriately restricted value for the site.	PR.AA-03 PR.PS-06
11.7.4	Logout functionality must fully terminate the associated session or connection.	PR.PS-06

11.7.5	Logout functionality must be available from all pages protected by authorisation.	PR.PS-06
11.7.6	Establish a session inactivity timeout that is as short as possible, based on balancing risk and business functional requirements. It must be no more than two hours for external services, and 6 hours for internal services.	PR.PS-06
11.7.7	Persistent logins must be disallowed and periodic session terminations must be enforced, even when the session is active. Especially for applications supporting rich network connections or connecting to critical systems. Termination times must support business requirements and the user must receive sufficient notification to mitigate negative impacts.	PR.AA-04 PR.PS-06
11.7.8	If a session was established before login, close that session and establish a new session after a successful login.	PR.AA-04 PR.PS-06
11.7.9	Generate a new session identifier on any re-authentication.	PR.AA-03 PR.PS-06
11.7.10	Concurrent logins with different geo-locations must not be allowed. If a session is still active, and the user is trying to login, logout the existing userid and allow the user to login.	PR.AA-04 PR.PS-06

11.7.11	Do not expose session identifiers in URLs, error messages or logs. Session identifiers must only be located in the HTTP cookie header. Cookies must set a secure attribute (see External References https://owasp.org/www-community/controls/SecureCookieAttribute) and must be transmitted over a secure connection in line with SS-007 Use of Cryptography Security Standard [Ref. A]. Set cookies with the HttpOnly attribute, unless you specifically require client-side scripts within your application to read or set a cookie's value (see External References https://owasp.org/www-community/HttpOnly).	PR.DS-02 PR.PS-06
11.7.12	Protect server side session data from unauthorised access, by other users of the server, by implementing appropriate access controls on the server, maintaining separation of duties.	PR.AA-02 PR.DS-01
11.7.13	Generate a new session identifier and deactivate the old one periodically. (This can mitigate certain session hijacking scenarios where the original identifier was compromised).	PR.PS-06
11.7.14	Generate and transmit new session identifiers in line with SS-007 Use of Cryptography Security Standard [Ref. A].	PR.DS-02
11.7.15	Supplement standard session management for sensitive server-side operations, like account management, by utilising per-session strong random tokens or parameters. This method can be used to prevent Cross Site Request Forgery attacks.	PR.PS-06

11.7.16	Supplement standard session management for highly sensitive or critical operations by utilising per-request, as opposed to per-session, strong random tokens or parameters.	PR.PS-06
---------	---	----------

11.8 Access Control

(SS-001 pt.1 Access and Authentication Security Standard [Ref. E] can be referred to for specifics)

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.8.1	Trusted system objects, e.g. server side session objects, must be used for making access authorisation decisions.	PR.AA-04
11.8.2	A single site-wide component must be used to check access authorisation where possible. This includes libraries that call external authorisation services.	PR.AA-04
11.8.3	Access controls must fail securely.	PR.AA-04
11.8.4	Deny all access if the application cannot access its security configuration information.	PR.AA-04
11.8.5	Enforce authorisation controls on every request, including those made by server side scripts, "includes" and requests from rich client-side technologies.	PR.AA-04
11.8.6	Privileged logic must be segregated from other application code to ensure that privileges granted to one cannot be used for the other.	PR.AA-05 PR.PS-06
11.8.7	Restrict access to files or other resources, including those outside the application's direct control, to only authorised users.	PR.AA-05

11.8.8	Restrict to authorised users only; • access to protected URLs • access to protected functions • direct object references • access to services • access to application data • access to user and data attributes and policy information used by access controls • access security-relevant configuration information	PR.AA-05
11.8.9	If state data must be stored on the client, use encryption and integrity checking on the server side to catch state tampering.	PR.DS-01
11.8.10	Limit the number of transactions a single user or device can perform in a given period of time. The transactions/time must be above the actual business requirement, but low enough to deter automated attacks.	PR.PS-06 PR.IR-03 PR.IR-04
11.8.11	If user permissions are changed, force a log out and re-authentication.	PR.AA-03
11.8.12	Implement account auditing and enforce the disabling of unused accounts in line with SS-001 pt.1 Access and Authentication Security Standard [Ref. E].	PR.AA-01
11.8.13	The application must support disabling of accounts and terminating sessions when authorisation ceases (e.g., Changes to role, employment status, business process, etc.). If a disabled account is re-enabled, all permissions must be removed and re-applied in line with the role before re-using.	PR.AA-01

11.8.14	Service accounts (i.e. non-human, for automated processes) or accounts supporting connections to or from external systems must have the least privilege possible.	PR.AA-05
11.8.15	Create an Access Control Policy to document an application's business rules, data types and access authorisation criteria and/or processes so that access can be properly provisioned and controlled. This includes identifying access requirements for both the data and system resources. Templates may be used for this purpose.	PR.AA-05
11.8.16	Environment access controls must be deployed that include strong access logging (Role Based Access Controls), audit logging, environment baselining and change control for CI/CD systems, which must be fed into Authority SIEM tooling.	PR.PS-04

11.9 Cryptographic Practices

(SS-007 Use of Cryptography Security Standard [Ref. A] can be referred to for specifics on encryption)

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.9.1	All cryptographic functions used to protect secrets (e.g. API tokens, KMS keys, private certificates, key material etc.) from the application user must be implemented on a trusted system (e.g., the server, not the client).	PR.DS-01
11.9.2	Protect master secrets from unauthorised access.	PR.DS-01
11.9.3	Cryptographic modules must fail securely.	PR.DS-01

11.9.4	All random numbers, random file names, random GUIDs, and random strings must be generated using random number generators derived from the cryptographic modules specified in the DWP Approved Cryptographic Algorithms document [Ref. N].	PR.DS-01 PR.PS-06
11.9.5	Cryptographic modules used by the application must have an active validation to FIPS 140. (See NIST Cryptographic Module Validation Program http://csrc.nist.gov/groups/STM/cmvp/validation.html (See External References)).	PR.DS-01
11.9.6	Cryptographic keys must be managed as per SS-007 Use of Cryptography Security Standard [Ref. A] and SS-002 PKI & Key Management Security Standard [Ref. G]. Any exceptions must be notified.	PR.DS-01

11.10 Error Handling and Logging

For clarity, standard output (stdout) is the stream to which a program writes its output data; standard error (stderr) is another output stream typically used by programs to output error messages or diagnostic data.

(For relevant logging details pertaining to the wider Authority requirements for logging please refer to SS-012 Protective Monitoring Security Standard [Ref. H]).

(Important) this paragraph contains ‘must’ activities.

Reference	Minimum Technical Security Measures	NIST ID
11.10.1	Logging Output Integration must be compatible with existing Authority aggregation and SIEM tooling e.g. JSON, XML, or CSV formats.	PR.PS-04

11.10.2	All logging controls must be implemented on a trusted system (e.g., the server, not the client), and must support both success and failure of specified security events. Security gates must be embedded in security monitoring in all CI/CD pipelines, defined automated rollback and patch response workflows. All outputs must feed into the Authority SIEM tooling.	PR.PS-04
11.10.3	Ensure logs contain important log event data, which must include input and output validation failures, authentication successes and failures, authorisation (access control) failures, session management failures, application errors and system events, use of higher-risk functionality.	PR.PS-04
11.10.4	Where possible, ensure log entries that include un-trusted data will not execute as code in the intended log viewing interface or software. Sensitive information such as personal data must not be written to log files.	PR.PS-04
11.10.5	Restrict access to logs to only authorised individuals.	PR.AA-02
11.10.6	Sensitive information in error responses must not be disclosed, including system details, session identifiers or account information.	PR.PS-06
11.10.7	Use error handlers that do not display debugging or stack trace information to the end user – engineering users should use appropriate outputs (std err and std out and also message output dialogues).	PR.PS-06
11.10.8	Implement generic error messages and use custom error pages.	PR.PS-06

11.11 Data Protection

For more detailed information on secure coding practices, please refer to the Engineering Secure Coding Guidelines (Ref. J].

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.11.1	<u>Security-specific</u> server-side source-code must be protected from being downloaded by a user, for example; • security enforcing functions, • internal security or anti-fraud rules, • the existence, configuration and/or status of sensitive components within our estate such as configuration files, IP addresses and Domain names, • anything that identifies vulnerabilities in existing versions of Open Source code, owned by either Authority or external code, • any legal or commercial issues/or any system/service vulnerabilities or remediation plans, • code which incorporates components of code from a third party (whether open source or proprietary), where the relevant license terms preclude or restrict such sharing, • any code, standing data, sensitive data, staff records or other components that could help a threat actor to access customer records.	PR.DS-01 PR.PS-06

11.12 File Management

(Important) this paragraph contains 'must' activities.

Reference	Minimum Technical Security Measures	NIST ID
11.12.1	User supplied data must not be passed directly to any dynamic include function i.e. prevent malicious code from affecting the logic of the function.	PR.PS-05 PR.PS-06
11.12.2	The type of files that can be uploaded must be limited to only those types that are needed for business purposes. Where possible, validate that uploaded files are of the expected type by checking file headers. Checking for file type by extension alone is not sufficient.	PR.PS-05 PR.PS-06
11.12.3	When referencing existing files, use an allowlist of allowed file names and types must be used. Validate the value of the parameter being passed and if it does not match one of the expected values, either reject it or use a hard coded default file value for the content instead.	PR.PS-05 PR.PS-06
11.12.4	Scan user uploaded files for viruses and malware using up to date dictionaries, rejecting those that may be suspect, in line with SS-015 Malware Protection Security Standard [Ref. I].	DE.CM-09
11.12.5	Data from external or less trusted sources may contain malicious content. If such data cannot be validated, it must be transformed or converted to another format to reduce this risk before being passed on to its destination. [See external references NCSC Secure Design Principles 2.1]	DE.CM-09

11.13 Application Lifecycle

Further information on application lifecycles can be found in the Supply Chain Levels for Software Artifacts framework (See External References).

(Important) this paragraph contains ‘must’ activities.

Reference	Minimum Technical Security Measures	NIST ID
11.13.1	Code promotion must be controlled and managed at identified significant points within the development lifecycle.	PR.PS-06
11.13.2	Where appropriate, when decommissioning hosts and environments that were used in the development effort, they must be securely wiped or deleted after use.	PR.PS-03
11.13.3	<u>All</u> environments must be logically segregated from each other.	PR.IR-01
11.13.4	Consideration must be given to any integration needs early in the project. New software products and systems must integrate with applicable existing solutions and capabilities where appropriate.	PR.PS-06
11.13.5	Build and Deployment Integrity: To prevent the injection of malicious code during the build and release process, environments used for compiling and deploying software must be strictly controlled; • Baseline Requirement (All Environments): Access to build servers, release management tools, and deployment credentials must be restricted using Role-Based Access Control (RBAC) and Multi-Factor Authentication (MFA). A cryptographic hash (e.g., SHA-256) of the final deployment artifact must be generated at the point of build and verified prior to deployment to ensure no tampering	PR.PS-05 PR.PS-06

	occurred in transit. Advanced Requirement (Critical Modern Environments): For bespoke applications handling highly sensitive Authority data (Criticality Levels 1 & 2) deployed via automated pipelines, the provenance of the build should be cryptographically signed by the pipeline tooling, aligning with Supply Chain Levels for Software Artifacts (SLSA) Level 3 principles, to ensure a non-repudiable chain of custody from code commit to production.	
--	---	--

11.14 Use of Artificial Intelligence Coding Assistants (AICAs) & Agentic Workflows (Important) this paragraph contains ‘must’ activities.

Reference	Minimum Technical Security Measures	NIST ID
11.14.1	Product teams and developers remain 100% responsible for the final versions of any outputs that AI coding assistants produce. All source code generated, suggested, or modified by an AICA must be supported by robust and comprehensive unit/integration tests and subjected to mandatory "human-in-the-loop" peer review prior to merging.	ID.AM-08 PR.PS-06
11.14.2	All AI-generated code must be subjected to automated Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA) within the CI/CD pipeline before being merged into any primary or production branch.	ID.AM-08 PR.PS-06

11.14.3	Developers must strictly verify the existence, provenance, and integrity of all software packages, libraries, and methods suggested by an AICA before integration, to mitigate the risk of "package hallucination" and subsequent supply chain compromise.	ID.RA-09 PR.PS-06
11.14.4	Project teams must utilise internal package registries or mirrors for libraries to prevent package confusion attacks originating from AI suggestions.	ID.RA-09 PR.PS-06
11.14.5	Hardcoded secrets, API keys, and cryptographic materials must not be present in development environments or local workspaces where AICA plugins are active.	ID.PS-01 PR.PS-06
11.14.6	Developers must utilise synthetic data or redacted examples when prompting external AI models. Sensitive production data or personal data must never be used in prompts or handled within AICA-enabled development environments.	ID.DS-10 PR.PS-06
11.14.7	AICAs and agentic extensions must operate under the principle of least privilege. Agentic systems must not be granted autonomous permissions to push code directly to production systems or execute structural environment changes.	PR.AA-05 PR.PS-06
11.14.8	Where agentic workflows are utilised, the system must require explicit, documented human consent for all high-risk or privileged operations	PR.AA-05 PR.PS-06

11.14.9	System prompts and instructions used to guide autonomous agentic workflows must be treated as critical code artifacts. They must be subject to version control, secure storage, and vulnerability scanning equivalent to traditional source code.	ID.AM-08 PR.PS-02 PR.PS-06 DE.CM-09
---------	---	--

12 Appendices

Appendix A Security Outcomes

The minimum security measures defined in this standard contribute to the achievement of security outcomes described in the table below. For consistency, the official NIST Sub-category IDs have been carried through to the standards.

Table 1 – List of Security Outcomes Mapping

NIST Ref	Security Outcome (sub-category)	Related Security measure
GV.PO-01	Policy for managing cybersecurity risks is established based on organisational context, cybersecurity strategy, and priorities and is communicated and enforced	11.1.1, 11.1.2, 11.1.6
GV.PO-02	Policy for managing cybersecurity risks is reviewed, updated, communicated, and enforced to reflect changes in requirements, threats, technology, and organisational mission	11.1.1, 11.1.2, 11.1.6
GV.SC-03	Cybersecurity supply chain risk management is integrated into cybersecurity and enterprise risk management, risk assessment, and improvement processes	11.1.5
GV.SC-05	Requirements to address cybersecurity risks in supply chains are established, prioritised, and integrated into contracts and other types of agreements with suppliers and other relevant third parties	11.1.5, 11.1.6

GV.SC-07	The risks posed by a supplier, their products and services, and other third parties are understood, recorded, prioritised, assessed, responded to, and monitored over the course of the relationship	11.1.5
ID.AM-02	Inventories of software, services, and systems managed by the organisation are maintained	11.1.26, 11.6.2
ID.AM-03	Representations of the organisation's authorised network communication and internal and external network data flows are maintained	11.1.20
ID.AM-08	Systems, hardware, software, services, and data are managed throughout their life cycles	11.1.25, 11.14.1, 11.14.2, 11.14.9
ID.RA-01	Vulnerabilities in assets are identified, validated, and recorded	11.1.4, 11.1.15, 11.2.1, 11.2.2, 11.6.9
ID.RA-03	Internal and external threats to the organisation are identified and recorded	11.1.18, 11.6.1
ID.RA-04	Potential impacts and likelihoods of threats exploiting vulnerabilities are identified and recorded	11.1.15, 11.6.1
ID.RA-09	The authenticity and integrity of hardware and software are assessed prior to acquisition and use	11.14.3, 11.14.4

ID.IM-02	Improvements are identified from security tests and exercises, including those done in coordination with suppliers and relevant third parties	11.1.19
ID.IM-03	Improvements are identified from execution of operational processes, procedures, and activities	11.1.3,
PR.AA-01	Identities and credentials for authorised users, services, and hardware are managed by the organisation	11.4.3, 11.4.4, 11.4.7, 11.4.8, 11.4.9, 11.4.10, 11.4.11, 11.4.12, 11.4.13, 11.4.14, 11.5.7, 11.5.16, 11.5.17, 11.5.18, 11.6.4, 11.8.12, 11.8.13
PR.AA-02	Identities are proofed and bound to credentials based on the context of interactions	11.4.6, 11.5.13, 11.7.12, 11.10.5
PR.AA-03	Users, services, and hardware are authenticated	11.4.3, 11.4.4, 11.4.6, 11.5.1, 11.5.2, 11.5.3, 11.5.4, 11.5.8, 11.5.9, 11.5.10, 11.5.11, 11.5.12, 11.5.13, 11.5.14, 11.6.3, 11.7.3, 11.7.9, 11.8.11
PR.AA-04	Identity assertions are protected, conveyed, and verified	11.4.1, 11.4.5, 11.5.19, 11.7.7, 11.7.8, 11.7.10, 11.8.1, 11.8.2, 11.8.3, 11.8.4, 11.8.5

PR.AA-05	Access permissions, entitlements, and authorisations are defined in a policy, managed, enforced, and reviewed, and incorporate the principles of least privilege and separation of duties	11.1.11, 11.1.14, 11.1.22, 11.5.6, 11.6.4, 11.6.7, 11.8.6, 11.8.7, 11.8.8, 11.8.14, 11.8.15, 11.14.7, 11.14.8
PR.DS-01	The confidentiality, integrity, and availability of data-at-rest are protected	11.1.8, 11.1.14, 11.1.23, 11.5.7, 11.5.11, 11.5.15, 11.6.5, 11.7.12, 11.8.9, 11.9.1, 11.9.2, 11.9.3, 11.9.4, 11.9.5, 11.9.6, 11.11.1
PR.DS-02	The confidentiality, integrity, and availability of data-in-transit are protected	11.1.16, 11.1.17, 11.1.24, 11.4.2, 11.5.11, 11.5.12, 11.5.16, 11.7.11, 11.7.14
PR.DS-10	The confidentiality, integrity, and availability of data-in-use are protected	11.14.6
PR.IR-01	Networks and environments are protected from unauthorised logical access and usage	11.1.21, 11.13.3
PR.IR-03	Mechanisms are implemented to achieve resilience requirements in normal and adverse situations	11.1.21, 11.6.6, 11.8.10
PR.IR-04	Adequate resource capacity to ensure availability is maintained	11.6.6, 11.8.10

PR.PS-01	Configuration management practices are established and applied	11.14.5
PR.PS-02	Software is maintained, replaced, and removed commensurate with risk	11.1.25, 11.14.9
PR.PS-03	Hardware is maintained, replaced, and removed commensurate with risk	11.13.2
PR.PS-04	Log records are generated and made available for continuous monitoring	11.6.12, 11.8.16, 11.10.1, 11.10.2, 11.10.3, 11.10.4
PR.PS-05	Installation and execution of unauthorised software are prevented	11.2.7, 11.3.4, 11.6.8, 11.12.1, 11.12.2, 11.12.3, 11.13.5
PR.PS-06	Secure software development practices are integrated, and their performance is monitored throughout the software development life cycle	11.1.4, 11.1.7, 11.1.8, 11.1.9, 11.1.10, 11.1.12, 11.1.13, 11.1.16, 11.1.27, 11.1.28, 11.2.3, 11.2.4, 11.2.5, 11.2.6, 11.2.7, 11.2.8, 11.2.9, 11.2.10, 11.2.11, 11.3.1, 11.3.2, 11.3.3, 11.3.4, 11.3.5, 11.6.5, 11.6.6, 11.6.8, 11.6.10, 11.6.11, 11.7.1, 11.7.2, 11.7.3, 11.7.4, 11.7.5, 11.7.6, 11.7.7, 11.7.8, 11.7.9, 11.7.10, 11.7.11, 11.7.13, 11.7.15, 11.7.16, 11.8.6, 11.8.10, 11.9.4,

		11.10.6, 11.10.7, 11.10.8, 11.11.1, 11.12.1, 11.12.2, 11.12.3, 11.13.1, 11.13.4, 11.13.5, 11.14.1, 11.14.2, 11.14.3, 11.14.4, 11.14.5, 11.14.6, 11.14.7, 11.14.8, 11.14.9
DE.CM-01	Networks and network services are monitored to find potentially adverse events	11.1.15
DE.CM-09	Computing hardware and software, runtime environments, and their data are monitored to find potentially adverse events	11.1.13, 11.4.14, 11.6.12, 11.12.4, 11.12.5, 11.14.9
DE.AE-02	Potentially adverse events are analysed to better understand associated activities	11.6.12

Appendix B Internal References

Below, is a list of internal documents that **should** be read in conjunction with this standard.

Table 2 – Internal References

Ref	Document	Publicly Available*
A	SS-007 Use of Cryptography Security Standard	Yes
B	Information Management Policy	Yes
C	SS-023 Cloud Computing Security Standard.	Yes
D	SS-027 Security Testing Security Standard	No
E	SS-001 pt.1 Access and Authentication Security Standard	Yes
F	SS-001 pt.2 Privileged User Access Security Standard	Yes
G	SS-002 PKI & Key Management Security Standard	Yes
H	SS-012 Protective Monitoring Security Standard	Yes
I	SS-015 Malware Protection Security Standard	Yes
J	Engineering Secure Coding Guidelines	No
K	Security Assurance Strategy	No
L	Technical Vulnerability Management Policy	Yes
M	SS-033 Security Patching & Exposure Mgmt Standard	Yes
N	DWP Approved Cryptographic Algorithms	No

Requests to access non-publicly available documents **should be made to the Authority.*

Appendix C External References

The following publications and guidance were considered in the development of this standard and **should** be referred to for further guidance.

Table 3 – External References

External Documents List
CIS Critical Security Controls set
UK General Data Protection Regulation (UK GDPR)
SEI CERT Secure Coding Standards - www.securecoding.cert.org/
NIST Cryptographic Module Validation Program http://csrc.nist.gov/groups/STM/cmvp/validation.html
RFC 3986: Uniform Resource Identifier (URI): Generic Syntax
https://owasp.org/www.community/controls/SecureCookieAttribute
https://owasp.org/www-community/HttpOnly
Supply Chain Levels for Software Artifacts framework - https://slsa.dev/spec/v0.1/levels
DWP Open Source Policy
NCSC Secure Design Principles - https://www.ncsc.gov.uk/collection/cyber-security-design-principles/making-compromise-difficult
NCSC - Securing HTTP-based APIs
NCSC Guidelines for secure AI system development
OWASP Top 10 for Large Language Model Applications
NCSC Password Management - https://www.ncsc.gov.uk/collection/top-tips-for-staying-secure-online/password-managers

Appendix D Abbreviations

Table 4 – Abbreviations

Abbreviation	Definition
AES	Advanced Encryption Standard – defined in FIPS 197 in NIST. Different modes of operation are covered in different documents.
CIS	Centre for Internet Security
DA	Design Authority (DA)
DAST	Dynamic Application Security Testing
DDoS	Distributed Denial of Service
DWP	Department of Work and Pensions.
FIPS 140-2	Federal Information Processing Standard 140-2 is a U.S. government computer security standard used to approve cryptographic modules
GCSP	Government Security Classification Policy
IDE	Integrated Development Environment
ITHC	IT Security Health Check
LDAP	Lightweight Directory Access Protocol
NIST	National Institute of Standards and Technology
NIST – CSF	National Institute of Standards and Technology – Cyber Security Framework
OWASP	Open Web Application Security Project
SAST	Static Application Security Testing
SQL	Structured Query Language
TLS	Transport Layer Security
XFS	‘X’ File System
XML	Extensible Markup Language

Appendix E Definition of Terms

Table 5 – Glossary

Term	Definition
Security Wrapper	A concept that contains security controls and architectures that are complimentary and which can include some or all of the following; Identity and Access management, security event and incident management, anti-malware, cryptographic key management, trust management, resilience management, threat and vulnerability management, fraud management (anti-fraud)
Cryptographic Key Material	Any parameter passed to an encryption cipher which influences the output of the algorithm (with the exception of the message itself).
Security Story	Similar to a user story, a definition of expectation with respect to security; developed from an initial security review, examining functionality, policies, legal, regulatory and contractual requirements, etc. These may then be better defined with the development team and prioritised for handling. When the development cycle/sprint begins, the development team can pick up the security stories as well as the functional stories to begin the process of agile security development.
Homographs / homoglyphs	Visually identical characters from different scripts (e.g., Cyrillic 'a' vs. Latin 'a') that can be used to create deceptive code or instructions.

Appendix F Accessibility artefacts

A variety of accessibility guidance is available from the below URL, that includes:

DWP Digital Accessibility Policy | DWP Intranet

<https://accessibility-manual.dwp.gov.uk/>

<https://www.gov.uk/guidance/guidance-and-tools-for-digital-accessibility>

<https://www.gov.uk/guidance/accessibility-requirements-for-public-sector-websites-and-apps>