CYBERGLASS SECURITY STANDARDS

Secure Development and Mobile Application Security Standard

Requirements for building, testing, releasing and maintaining [Company Name]'s web, API, mobile and AI software securely, including open-source components, secrets, the build pipeline and vulnerability disclosure.

Organization	[Company Name]
Document ID	STD-06
Document owner	[Name, title]
Approved by	[Name, title]
Version	1.0 [draft / approved]
Effective date	[YYYY-MM-DD]
Next review	[YYYY-MM-DD] (at least every 12 months)
Classification	Internal

PREVIEW

About this template (delete this box before publishing)

This document is a template to help [Company Name] write its own policy. It is a guideline based on common industry practice and publicly available frameworks, not a finished policy, legal advice or a guarantee of compliance. Every organization is different: review each requirement, remove what does not apply, add what is missing, and set values your organization can actually meet. Have the final version reviewed by people who know your systems, obligations and risks.

- Set each product's risk tier first (section 4); it decides how much of the rest applies.
- Turn on the built-in protections of your code platform before buying tools: branch protection, secret scanning and dependency alerts cover sections 6–8 for most small teams.
- Use Appendix A as your release checklist and keep the completed copies; they are the evidence customers and auditors ask for.
- Replace every highlighted **[placeholder]** with your organization's details, then remove the highlighting.
- Grey **Guidance** boxes explain what auditors, insurers and clients look for. Read them, then delete them before publishing.

Preview pages · full editable Word file at cyberglass.ca/shop

Mappings are CyberGlass guidance that cite reference numbers only; they are not official crosswalks published by the standards bodies. Use them alongside your own licensed copy of each standard.

Executive summary

IN PLAIN WORDS *for leaders, clients and procurement teams (keep and adapt)*

What it does: Builds security into how software is planned, coded, tested, released and fixed: web, APIs, mobile apps and AI features.

How it protects clients: Client data is protected by authorization on every request, no secrets in apps or code, tested releases and fast fixes for vulnerabilities.

What clients can ask for: The release checklist (Appendix A), an SBOM for the version they use, penetration test summaries, the vulnerability disclosure policy and fix-time measures (section 18).

Measured by: Releases with checklist and SBOM, Critical and High fixed on time, secrets revoked within 24 hours, branch protection coverage.

Real-world context: why this document matters

When	What happened	What it means for this document
Mar 2024	A contributor who had spent about two years gaining maintainer trust slipped a backdoor into a widely used open-source compression library. It was caught only because an engineer noticed slow remote logins, shortly before it reached the stable releases of major Linux distributions.	Section 8: know every dependency, prefer well-maintained components and keep the ability to roll back quickly.
Dec 2020	Attackers broke into a software vendor's build system and inserted malicious code into signed updates of its network management product. About 18,000 customers installed the trojanized update, and several government agencies were then breached through it.	Section 8: hardened, ephemeral build runners, signed artifacts and recorded provenance make a tampered build detectable.
Dec 2021	A critical flaw in a very widely used open-source Java logging library let attackers run code by sending a crafted text string. Mass exploitation began within days, and many organizations spent weeks just finding where the library was used.	Section 8: an SBOM for every release answers 'are we affected?' in minutes instead of weeks.
Sep 2022	Researchers found 1,859 public Android and iOS apps containing hard-coded cloud credentials; about three-quarters held valid tokens giving access to private cloud services, and some exposed millions of records, including users' biometric data used by banking apps.	Sections 7 and 10: no long-lived cloud keys in apps; apps get short-lived, user-scoped tokens from the back end.

Summarized by CyberGlass from public reporting: Open-source maintainers' advisories and CISA alert, Mar–Apr 2024; US government advisories and company filings, Dec 2020–2021; CISA and national cyber centre advisories, Dec 2021; Symantec research reported by The Hacker News and Dark Reading, Sep 2022. Organizations are described, not named. Figures are as publicly reported at the time.

How this document connects

Document or register	How records and decisions flow
IT-04 Information Security Policy	Policy basis: secure development, cryptography, logging
IT-06 Incident Response Plan	Exposed secrets and exploited product vulnerabilities become incidents INC-YYYY-NNN

PREVIEW

Preview pages · full editable Word file at cyberglass.ca/shop

3. Scope

This Standard applies to:

a) all software that [Company Name] develops, configures or has developed for it, including web applications, APIs and microservices, mobile apps, desktop software, browser extensions, firmware and scripts used in production

b) AI features in products, such as chat assistants, retrieval over customer data and AI agents (agents are also covered by STD-04)

c) infrastructure as code, container images, build and deployment pipelines, and the source code repositories that hold them

d) open-source libraries, software development kits (SDKs), third-party services and code generated by AI coding assistants

e) employees, contractors, outsourced development teams and agencies that write or release code for [Company Name]

4. Secure Development Lifecycle and Risk Tiers

CONTROL ID **STD-06-4**
Maps to NIST CSF 2.0: PR.PS-06, ID.AM-02, GV.PO-01
Maps to ISO/IEC 27001:2022: A.8.25, A.5.8
Maps to NIST SP 800-53 Rev. 5: SA-3, SA-8
Maps to NIST SP 800-218 (SSDF): PO.1, PO.5

Each product or service shall be assigned a risk tier by its product owner with the Security lead. The tier sets how much assurance each phase needs.

Tier	Typical products	Minimum assurance
Tier 1 (high)	Handles payments, health, financial or children's data; authentication services; admin consoles; AI agents that act on systems; apps with more than [100,000] users	Threat model for every significant change, security sign-off before release, yearly independent penetration test, all section 9 checks blocking
Tier 2 (medium)	Business applications with personal or confidential customer data; public APIs; mobile apps that sign users in	Threat model per major feature, automated checks blocking for Critical and High findings, penetration test every [2] years or after major redesign
Tier 3 (low)	Marketing sites, internal tools without sensitive data, prototypes not exposed to customers	Section 6 and 7 controls, dependency scanning, basic security checklist before release

Phase	Required activities	Evidence kept
Plan	Assign tier; record security and privacy requirements (section 5); decide whether the feature uses personal data or AI	Tier in the product register; requirements in the ticket or design
Design	Threat model for Tier 1–2 changes; architecture review for new data flows, integrations and AI features	Threat model record; review decisions
Build	Secure coding, peer review, secret scanning, dependency checks (sections 6–8)	Pull request history; scan results

PREVIEW

Preview pages · full editable Word file at cyberglass.ca/shop

Appendix A. Release Security Checklist (worked example)

Example: release 3.4 of a Tier 1 mobile banking-style app and its API, adding instant payments to saved payees. Replace with your own release.

Check	Result	Evidence or note
Tier confirmed; threat model updated for new payment flow	Done	TM-2026-014; new threat: payee change by stolen session; mitigation: step-up authentication
Security requirements (ASVS level 2; MASVS testing profiles L2 and R) traced to tickets	Done	12 requirements, all linked to tickets
SAST, dependency and secret scans clean of Critical and High	Done	One High in an image library upgraded before release
API authorization tested for every new endpoint and object	Done	Automated tests for payee and payment IDs belonging to another user
Mobile package inspected: no secrets, secure storage, no clear-text traffic	Done	Build 3.4.0 (412) checked
Penetration test of payment flow; findings retested	Done	2 Medium open with due dates within 30 days
SBOM generated and stored; artifacts signed	Done	SBOM attached to release record
Privacy disclosures and in-app consent match new data collected	Done	No new personal data; payee name already disclosed
Rollback or feature flag ready; monitoring alerts for payment errors	Done	Flag 'instant_payments' default off; staged rollout 5% → 100%
Approvals: product owner, engineering lead, security lead	Done	Recorded in release ticket

PREVIEW

Appendix B. Definitions

Term	Definition
Secure development lifecycle	The set of security activities built into each phase of making and maintaining software
Threat model	A structured look at what could go wrong with a system and what will be done about it
SAST / DAST	Static application security testing analyses source code; dynamic testing exercises the running application
Software composition analysis (SCA)	Automated identification of open-source components and their known vulnerabilities and licences
Software bill of materials (SBOM)	A machine-readable list of the components in a piece of software
Provenance	Verifiable information about how, where and from what source an artifact was built

Preview pages · full editable Word file at cyberglass.ca/shop