FLOWE S.p.A. SB

Policy di Sviluppo Sicuro del Software

Consiglio di Amministrazione del 26/07/2023

INDICE

1 PREMESSA

Scopo del presente documento consiste nel presentare i principi adottati da Flowe S.p.A. (di seguito anche "Flowe" o la "Società") per definire i controlli di sicurezza che dovrebbero essere implementati nelle varie fasi del ciclo di vita di Sviluppo del software.

1.1 Contesto di riferimento

La direzione della Società ha riconosciuto l'importanza di integrare al processo di Change management quello di Sviluppo sicuro del software. Tale indirizzo è motivato dal fatto che i punti deboli delle applicazioni e le vulnerabilità del software continuano a essere il mezzo più comune con cui i criminali informatici compiono attacchi. Tali violazioni causano danni economici all'azienda comportando perdite di fatturato, di clienti, di opportunità, nonché il dover sostenere costi aggiuntivi non previsti.

L'adozione di una governance e di politiche formalizzate per lo sviluppo sicuro del software è funzionale ad assicurare che siano correttamente concepiti, progettati, implementati, testati, installati, documentati e gestiti adeguati controlli finalizzati a ridurre l'esposizione al rischio di sicurezza e a rilevare possibili criticità dal punto di vista della sicurezza di Flowe.
Al fine di mitigare i rischi derivanti dagli attacchi informatici in aumento negli ultimi anni, è stata quindi formulata la presente Politica, inserita all'interno del sistema delle fonti della normativa interna, come di seguito rappresentato. Tale Politica prevede lo sviluppo di applicazioni sicure sia by design, quindi progettate per essere sicure, che by default, ovvero le impostazioni predefinite sono le più sicure possibili, come di seguito definito.

1.2 Ambito di riferimento

Con riferimento alla "*Policy sulle modalità di redazione, approvazione, diffusione ed aggiornamento della normativa interna"* del Gruppo, il documento si colloca al primo livello (di vertice) della piramide documentale richiamata nello schema seguente.

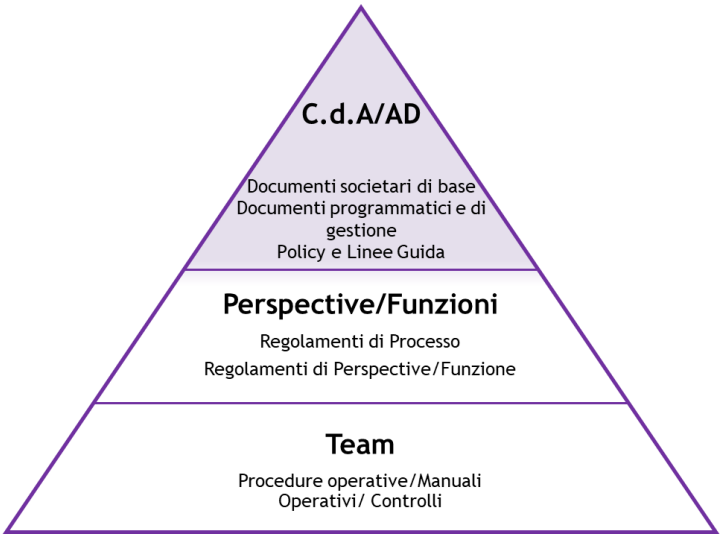

Figura 1: Modello della normativa interna di riferimento

2 APPLICABILITÀ

2.1 Destinatari del documento

Il presente documento è approvato dal Consiglio di Amministrazione di Flowe S.p.A. e viene diffuso, per quanto di competenza, a tutte le strutture organizzative (di seguito anche *Perspective*) della Società.

Le unità organizzative competenti per materia applicano nello svolgimento delle proprie attività i principi e le regole comportamentali definite nella presente *Policy.*

2.2 Responsabilità del documento

L'aggiornamento e la revisione del presente documento sono di responsabilità dell'unità **IT** Operation Security & Governance della Perspective Augmented Intelligence.

3 DEFINIZIONI

DAST- Dynamic Application Security Testing: sistemi che permettono di osservare in dettaglio come si comporta l'applicazione quando è in funzione per scovarne imperfezioni o vulnerabilità prima che si prosegua con lo step di sviluppo successivo.

Pratica per lo sviluppo e la gestione del software: la pratica con cui viene assicurato che le applicazioni soddisfino le esigenze delle parti interessate in termini di funzionalità, affidabilità, manutenibilità, conformità e verificabilità.

SAST- Static Application Security Testing: tool che esaminano il codice binario e il codice di programmazione delle applicazioni senza 'mandare in esecuzione' l'applicazione (ossia senza la necessità di farla girare sui sistemi nei processi di testing).

Security by Desing: un software progettato per essere sicuro e capace di garantire la riservatezza, integrità e disponibilità delle informazioni. Al fine di ottenere tale risultato, vengono anticipati e minimizzati a priori gli impatti delle vulnerabilità che il software potrà manifestare in produzione.

Security by Default: l'insieme di strategie necessarie per sviluppare e configurare, di default appunto, un software al suo massimo livello di sicurezza.

SDLC: Software Development Life Cycle.

4 ATTORI, RUOLI E RESPONSABILITÀ

IT Operation Security & Governance: responsabile della definizione dei principi di Secure Software Development Lifecycle, nonché della revisione degli stessi con frequenza almeno annuale. È inoltre responsabile di verificare la corretta adozione della presente politica all'interno di Flowe.

Dev: personale addetto alle attività di sviluppo IT per una data area (es: P0, Data Platform). All'interno di Flowe, i team che svolgono attività di sviluppo IT sono all'interno delle Perspective:

- Augmented Intelligence
- Digital Product

5 PRINCIPI DI SECURE SDLC

Lo sviluppo sicuro del codice considera e implementa opportune attività di sicurezza nel corso di tutte le fasi del processo SDLC, consentendo di incrementare sensibilmente il livello di sicurezza di ogni singola fase del ciclo di vita del software.

In linea con il principio di Security by design, al fine di garantire che il software sia progettato per essere sicuro, bisognerebbe, ad esempio, ridurre al minimo la superficie d'attacco, garantire la separazione dei ruoli nelle varie fasi del ciclo di vita del software, gestire gli errori in maniera sicura e comprendendo le cause che originano i problemi di sicurezza, stabilire valori predefiniti sicuri e ridurre i "*single point of failure*".

Di seguito, vengono rappresentate le macrofasi che dovrebbero essere presenti nel processo di sviluppo sicuro del software. Per tutte le fasi di seguito rappresentate, si rimanda alle procedure operative per quanto riguarda il dettaglio dei tool utilizzati.

Figura 21 – Fasi del processo di Secure SDLC

5.1 DISEGNO

In questa prima fase dovrebbero essere effettuate le analisi dei requisiti di sicurezza e delle probabilità di impatto delle minacce. I principali attributi di sicurezza che dovrebbero essere garantiti sono:

- Riservatezza: consentire l'accesso solo ai dati per i quali l'utente è autorizzato;
- Integrità: garantire che i dati non vengano manomessi o alterati da utenti non autorizzati;
- Disponibilità: garantire che i sistemi e i dati siano sempre disponibili agli utenti autorizzati quando questi ne hanno bisogno;
- Autenticità: garantire la catena di provenienza di un messaggio, attraverso, ad esempio, l'utilizzo di tecniche di crittografia; rafforza l'attributo dell'integrità;
- non-ripudio: garantire che qualsiasi azione sul sistema non possa essere in seguito rinnegata;
- controllo degli accessi: assicurare un meccanismo per tutelare gli asset informatici

da accessi non autorizzati, da parte di utenti sui quali non sia validata l'identità.

Nella fase di Disegno dovrebbe essere esaminato il sistema tramite un'analisi sia delle minacce che potrebbero impattare i componenti applicativi coinvolti, sia della superficie d'attacco, al fine di individuare le parti del sistema maggiormente esposte e, pertanto, più vulnerabili. La modellazione delle minacce è il processo di valutazione e documentazione degli impatti associati alla sicurezza di un particolare sistema e/o applicazione software. Al fine di analizzare i requisiti, sarebbe opportuno seguire le best practice di settore (ad esempio, Linee Guida Agid per l'adozione di un ciclo di sviluppo software sicuro).

Al fine di garantire, in linea con il principio di Security by design, che il software sia progettato per essere sicuro si dovrebbe stabilire valori predefiniti sicuri nonché adottare principi quali il minimo privilegio, attraverso il quale all'account venga garantito il livello minimo di privilegio richiesto e necessario per poter eseguire i relativi processi di business.

A questo principio si aggiunge la "Difesa in profondità", il quale, a livello applicativo, prevede il raggiungimento di una corretta postura di sicurezza attraverso l'utilizzo coordinato e combinato di molteplici contromisure di sicurezza.

Nello sviluppo sicuro di una applicazione software, è necessario prevedere che, per l'entità che approva un'azione, l'entità che svolge un'azione e l'entità che controlla tale azione, sia garantita la separazione dei ruoli.

Inoltre, è opportuno progettare il software evitando per quanto possibile la scelta dei "*single point of failure*"[1] e definendo gli opportuni requisiti di ridondanza.

I principi dovrebbero essere recepiti nelle Procedure operative al fine di garantire la realizzazione di un software sicuro fin dalla fase di Disegno.

5.2 IMPLEMENTAZIONE

Nella fase di Implementazione dovrebbe essere realizzato il sistema attraverso la stesura del codice sicuro da parte del team di sviluppatori, in linea con quanto individuato nelle fasi precedenti.

A seguito della scrittura, in tale fase dovrebbe essere svolta l'analisi del codice. La code review è funzionale ad individuare la presenza di bug (ad esempio, errori di programmazione, errori di sintassi) che potrebbero portare a malfunzionamenti e/o vulnerabilità sfruttabili da un attaccante.

Durante la fase di Implementazione, dovrebbero essere previsti dei Security Gate per la verifica del codice in fase di Sviluppo. Questi dovrebbero includere test di sicurezza basati sull'analisi delle minacce e l'analisi statica del codice sorgente (es. SAST) per verificare la presenza di vulnerabilità ("security flaws").

In linea con i principi di Security by design, in tale fase dovrebbe essere garantita la separazione delle funzioni. Pertanto, dovrebbe essere previsto che la figurata incaricata

[1] Componente critico del sistema che ha la capacità di arrestare le operazioni del sistema stesso in caso di guasto.

della stesura del codice, sia differente da quella identificata per la revisione ed analisi dello stesso.

Di seguito si riportano, a titolo esemplificativo e non esaustivo, alcune *best practices* che consentono e facilitano lo sviluppo sicuro del software.

Gestione degli errori **e** **Attività di logging**	Mostrare all'utente finale messaggi di errore di carattere generico, in modo tale da non rivelare dettagli sullo stato interno dell'applicazione
	Gestire tutte le eccezioni nel codice sorgente in modo tale che non si verifichi un'eccezione non gestita attraverso i linguaggi e i framework impiegati nello sviluppo di applicazioni web
	Nascondere, bloccare o gestire gli errori generati da eventuali framework o software di terze parti in uso
	Registrare su log: • tutte le attività di autenticazione e validazione; • tutte le attività o le occorrenze in cui il livello di privilegi di un utente subisce un cambiamento; • tutte le attività amministrative; • qualsiasi accesso ai dati sensibili.
	Non registrare dati inappropriati, ad esempio i dati sensibili in forma non cifrata, o non necessari
	Conservare e mantenere i log in modo appropriato
	Limitare e registrare tutti gli accessi al codice e agli strumenti di sviluppo
	Utilizzare il software di controllo del codice sorgente per gestire le modifiche software durante lo sviluppo
	Mantenere una copia master limitata del codice e degli strumenti condivisi separata dal sistema condiviso
	Attivare e monitorare regolarmente la registrazione su log degli accessi attraverso API
Gestione ambiente/rete	Separare ambienti di sviluppo, test e operativi per ridurre il rischio di accesso non autorizzato o change al sistema operativo
	Considerare l'utilizzo di componenti di rete basti su provider cloud per segmentare il traffico in modo ponderato
	Garantire l'integrità dell'ambiente operativo

	Proteggere la connettività di rete per limitare l'esposizione agli attacchi di rete
Controllo Accessi	Limitare l'accesso al codice sorgente, alle utilità di sistema, ai privilegi degli sviluppatori e ai manuali per sviluppatori
	Conservare in modo sicuro le credenziali degli sviluppatori, per garantire che solo le persone e i casi d'uso autorizzati abbiano accesso all'ambiente di sviluppo
	Controllare le configurazioni del provider cloud dei criteri di default di controllo degli accessi
Protezione dei dati	Garantire la privacy delle password di sviluppo conservandole in un luogo diverso dal sistema di sviluppo in applicazioni, file o workstation non protetti
	Utilizzare in tutti i punti di accesso delle applicazioni web l'HTTPS verificando che quest'ultimi siano rilasciati da una autorità certificante con buona reputazione
	Disattivare l'accesso tramite HTTP a tutte le risorse protette
	Disattivare su tutti i server eventuali schemi di cifratura considerati deboli
	Utilizzare l'intestazione "Strict-Transport-Security" al fine di garantire che il browser non parli con il server tramite http
	Memorizzare le password utente utilizzando tecniche di hashing sicuro con algoritmi forti come PBKDF2, bcrypt o SHA-512
	Scambiare in modo sicuro le chiavi crittografiche
	Definire e configurare un processo sicuro di gestione delle chiavi crittografiche
	Disabilitare il data caching attraverso l'uso dell'intestazione di controllo della cache e disabilitare anche l'auto completamento
	Crittografare i dati sensibili o critici prima che questi vengano archiviati
	Limitare l'utilizzo e la memorizzazione dei dati sensibili
	Assicurare che i dati di produzione non siano utilizzati nell'ambiente di sviluppo senza adeguate tecniche di approvazione o anonimizzazione
Configurazione	Proteggere l'ambiente fisico dei server e delle workstation

e operatività	Automatizzare il processo di distribuzione del software garantendo che le modifiche vengano apportate in modo coerente e ripetibile in tutti gli ambienti
	Definire e attuare un processo rigoroso di Change Management
	Definire gli opportuni requisiti di sicurezza di un software
	Condurre una Desing Review: un'analisi dei rischi con i professionisti della sicurezza modellando l'applicazione per identificare i rischi maggiormente rilevanti
	Eseguire una code review al fine di individuare i bug di sicurezza
	Condurre dei test di sicurezza sia durante che dopo lo sviluppo al fine di garantire che l'applicazione soddisfi gli standard di sicurezza imposti
	Configurare tutti i componenti dell'infrastruttura a supporto dell'applicazione secondo le migliori pratiche di sicurezza e le linee guida di hardening
	Definire e verificare regolarmente un piano di gestione degli incidenti
	Formare il team sugli aspetti di sicurezza dell'applicazione software

5.3 VERIFICA

Una volta conclusa la fase di Implementazione e prima della fase di rilascio definitivo del software, i team che rispettano i principi di sviluppo sicuro del software dovrebbero effettuare una verifica del codice elaborato mediante test funzionali e di sicurezza. In merito ai primi, sono test che hanno lo scopo di verificare che il codice rispetti le specifiche formali che realizzano i requisiti funzionali. Per quanto riguarda i test di sicurezza, essi mirano a controllare la vulnerabilità della superficie di attacco, in modo da agire in via preventiva alla correzione di eventuali problemi che potrebbero verificarsi in fase di rilascio. In tale fase si dovrebbero analizzare gli aspetti di sicurezza del sistema in esecuzione in un ambiente controllato impiegando tecniche e strumenti di analisi dinamica (es. DAST).

I test dei requisiti di sicurezza vengono eseguiti al fine di cercare vulnerabilità di sistemi, applicazioni e dati, tra cui, a titolo esemplificativo ma non esaustivo:

- accesso non autorizzato al sistema operativo sottostante;
- accesso non autorizzato all'applicazione e alle risorse dell'applicazione;
- accesso non autorizzato ai dati di controllo o autenticazione;
- accesso non autorizzato alle risorse di rete;
- attacchi Denial of Service;
- violazione delle politiche sulla privacy;

- qualsiasi altra violazione dei requisiti di sicurezza (ad esempio, viene eseguita una registrazione appropriata per tentativi di accesso non autorizzati).

Tutte le vulnerabilità scoperte dovrebbero essere risolte e il sistema dovrebbe essere nuovamente testato per verificarne l'effettiva risoluzione. Inoltre, le attività di correzione di una vulnerabilità possono esporre il sistema a nuove minacce, pertanto, dovrebbero essere eseguiti ulteriori test di verifica per individuare eventuali nuove vulnerabilità.

5.4 VALIDAZIONE E APPROVAZIONE

Durante la fase di Validazione, immediatamente prima del rilascio, dovrebbe essere effettuata una *final security review* per verificare se il software soddisfa tutti i requisiti di sicurezza individuati nella fase iniziale del progetto. In questa fase ci si dovrebbe accertare, inoltre, che gli eventuali bug di sicurezza precedentemente identificati siano stati corretti e che il software sia sufficientemente robusto di fronte a nuove vulnerabilità.

In linea con il principio della separazione delle funzioni sopra menzionato, la figura individuata per l'approvazione del rilascio deve essere diversa dalla figura che ha scritto il codice.

Il rilascio del software dovrebbe avvenire solamente dopo il completamento delle attività sopra elencate.

5.5 SUPPORTO

La fase conclusiva riguarda la manutenzione e l'assistenza post rilascio. Al fine di adeguarsi all'evoluzione delle vulnerabilità del software, dovrebbero essere individuate le contromisure necessarie a garantire una "sicurezza continua".

Il *Team* dovrebbe analizzare le nuove minacce e rilasciare aggiornamenti laddove valutato come necessario. Dato che tali minacce potrebbero ridurre l'efficacia dei controlli esistenti o avere un impatto sulla riservatezza, sulla disponibilità o sull'integrità del sistema, dovrebbero essere eseguite valutazioni periodiche degli impatti che potrebbero costituire.

6 NORMATIVA DI RIFERIMENTO

6.1 NORMATIVA INTERNA

- *Policy di Change Management;*
- *Policy di Conglomerato sulle modalità di redazione, aggiornamento, approvazione e diffusione della Normativa Interna.*

6.2 NORMATIVA ESTERNA

- Linee guida per l'adozione di un ciclo di sviluppo di software sicuro – AgID;

- Linee guida per la modellazione delle minacce ed individuazione delle azioni di mitigazione conformi ai principi del secure/privacy by design – AgID;
- OWASP Code Review Guide v2.0 e successive versioni;
- NIST Special Publication 800-218. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities e successive versioni.