8 SECURE SOFTWARE DEVELOPMENT LIFE CYCLE (SSDLC): ANALISI DELLE METODOLOGIE E DEI PROCESSI

8.1 Life Cycle & Maturity Models

8.1.1 Software Assurance Maturity Model (SAMML)

SAMM è un framework aperto per aiutare le organizzazioni a formulare e attuare una strategia di sicurezza software, che più si adatti ai rischi specifici della particolare organizzazione. Il progetto OpenSAMM, un'attività di OWASP, mantiene e aggiorna la documentazione SAMM.

References	**www.owasp.org/index.php/Category:Software_Assurance_Maturity_ModelOWASP SAMM Project** **www.opensamm.orgOpenSAMM**

Le risorse fornite da SAMM attraverso il sito web aiutano a:

- Valutare le pratiche di sicurezza software esistenti di un'organizzazione
- Costruire un programma software security assurance in iterazioni ben definite
- Dimostrare miglioramenti concreti al programma di security assurance
- Definire e misurare le attività relative alla sicurezza in tutta l'organizzazione

Essendo un progetto Open, i contenuti SAMM sono liberamente fruibili. Il modello si basa su 4 funzioni aziendali (Governance, Construction, Verification e Deployment) di sviluppo software e di 12 procedure di sicurezza. Ogni funzione all'interno dello sviluppo del software prevede tre pratiche di sicurezza:

- Governance
  - Strategy & Metrics
  - Education & Guidance
  - Policy & Compliance
- Construction
  - Security Requirements
  - Threat Assessment
  - Secure Architecture
- Verification
  - Design Review
  - Security Testing
  - Code Review
- Deployment
  - Environment Hardening
  - Vulnerability Management
  - Operational Enablement

Figura 14 - SAMM Structure

Per ogni security practice, tre Maturity Levels sono definiti in termini di specifiche attività e metriche che un'organizzazione potrebbe adottare al fine di ridurre i rischi per la sicurezza e aumentare l'affidabilità del software.

Risultati più rilevanti:

Maturity Model: SAMM version 1.0	Il modello è disponibile in formato XML ed è stato tradotto in diverse lingue. Nella stessa pagina sono evidenziati i tool a supporto: http://www.opensamm.org/download/

8.1.2 Systems Security Engineering Capability Maturity Model (SEE-CMM)

Il modello SSE-CMM si indirizza sui requisiti per l'implementazione della sicurezza in un sistema. Le attività di ingegneria di sicurezza coprono l'intero ciclo di vita del sistema (definizione dei concetti, analisi dei requisiti, progettazione, sviluppo, integrazione, installazione, manutenzione e disattivazione). L'SSE-CMM si applica a tutti i tipi di organizzazioni, a prescindere dalle loro dimensioni, da quelle commerciali a quelle di carattere governativo o accademico.

URL	**http://www.sse-cmm.org**
Country of HQ location	**US**
Geographic Scope	**International**
Type	**Industry (not for profit)**

Questo modello ha undici aree di processo di sicurezza ciascuna delle quali comprende un insieme di pratiche di base. Queste aree si concentrano sui controlli, sulle minacce, sulla scoperta e sull'eliminazione delle vulnerabilità:

- Administer Security Controls
- Assess Impact
- Assess Security Risk
- Assess Threat

- Assess Vulnerability
- Build Assurance Argument
- Coordinate Security
- Monitor Security Posture
- Provide Security Input
- Specify Security Needs
- Verify and Validate Security

Risultati più significativi:

Maturity Model	Capability Maturity Model - Model Description Document - http://all.net/books/standards/ssecmmv3final.pdf
Standard	ISO/IEC 21827 - http://www.iso.org/iso/catalogue_detail.htm?csnumber=44716

8.1.3 Building Security In Maturity Model (BSIMM)

BSIMM non è una guida completa 'how to' di sicurezza software, ma piuttosto una raccolta di idee e attività che sono oggi in uso all'interno delle aziende di sviluppo software. Il modello Building Security In Maturity (BSIMM) è uno studio delle iniziative di sicurezza del software in uso all'interno delle aziende che si occupano di sviluppo software. Mettendo insieme le pratiche di molte organizzazioni diverse, è possibile descrivere le misure comuni, quelle condivise da molti, e le peculiarità che rendono unico ogni singolo sistema. Il BSIMM è stato creato attraverso un processo di comprensione e analisi dei dati del mondo reale provenienti dalle esperienze di numerose aziende. Quelle che partecipano allo studio BSIMM provengono da differenti settori verticali, inclusi i servizi finanziari, il software indipendente, la tecnologia, la sanità, l'elettronica di consumo, ecc. Ogni mese al campione si aggiungono nuove aziende. Nove imprese nell'ambito sicurezza software, che sono stati a seguire validati e regolamentati con i dati provenienti da 21 aziende aggiuntive. Il BSIMM mette quindi insieme le esperienze di trenta imprese di sviluppo software - la maggior parte di essi si trovano negli Stati Uniti - che hanno implementato iniziative di sicurezza del software.

URL	https://www.bsimm.com/
Country of HQ location	US
Geographic Scope	International (mainly the US)
Type	Industry

BSIMM ha sviluppato il https://www.bsimm.com/ (SSF), che fornisce un vocabolario comune per descrivere gli elementi più importanti di un quadro di sicurezza software all'interno di una società.

Sono stati identificati quattro domini e pratiche comuni alla maggior parte delle esperienze. Il BSIMM descrive 109 attività che ogni organizzazione può mettere in pratica. Le attività sono descritte in termini di SSF, che identifica dodici pratiche raggruppati in 4 domini, 3 pratiche di dominio, come mostrato nella figura presa dal documento BSIMM2: di funzionalità, all'interno dei quali sono previste delle attività da svolgere.

The Software Security Framework (SSF)			
Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

Figura 15 - BSIMM SSF

Per ogni livello di pratica e di maturità vi è un'associazione “one activity - one objective”. I domini sono:

1. Governance - Pratiche che aiutano a organizzare, gestire e calibrare un framework di sicurezza del software. Anche l’addestramento del personale è una pratica da far rientrare nella governance centrale.
2. Intelligence - Un insieme di conoscenze aziendali utili per svolgere attività di sicurezza del software all'interno di un'organizzazione; comprende sia indicazioni proattive sulla sicurezza che modelli di organizzazione delle minacce.
3. SSDL Touchpoints - Pratiche associate all'analisi e alla sicurezza di particolari sviluppi software, artefatti e processi. Tutte le metodologie di sicurezza del software includono queste pratiche.
4. Deployment - Pratiche che si rifanno alla sicurezza della rete tradizionale e alla manutenzione del software. La configurazione del software, la manutenzione e altri problemi ambientali hanno un impatto diretto sulla sicurezza.

Il modello di maturità si presenta come una serie di attività connesse con le pratiche. Gli obiettivi per ogni livello di pratica sono identificati. Gli obiettivi possono essere ulteriormente suddivisi in obiettivi per la pratica/livello e sono associati alle attività. A titolo di esempio, la figura seguente, tratta dal documento BSIMM2, mostra il modello di maturità per la pratica di addestramento del dominio Governance.

GOVERNANCE: TRAINING		
Objective	**Activity**	**Level**
promote culture of security throughout the organization	provide awareness training	1
ensure new hires enhance culture	include security resources in onboarding	
act as informal resource to leverage teachable moments	establish SSG office hours	
create social network tied into dev	identify satellite during training	
build capabilities beyond awareness	offer role-specific advanced curriculum (tools, technology stacks, bug parade)	2
see yourself in the problem	create/use material specific to company history	
reduce impact on training targets and delivery staff	offer on-demand individual training	
educate/strengthen social network	hold satellite training/events	
align security culture with career path	reward progression through curriculum (certification or HR)	3
spread security culture to providers	provide training for vendors or outsource workers	
market security culture as differentiator	host external software security events	
keep staff up-to-date and address turnover	require annual refresher	

Figura 16 - Training practice BSIMM

Risultati più rilevanti:

Maturity Model	BSIMM2 - https://www.bsimm.com/download/

8.2 Analisi dei Processi SSDLC

8.2.1 McGraw's Secure Software Development Life Cycle Process

McGraw[30] [1] si propone di accrescere il processo SDLC (cascata o iterativo) attraverso l'integrazione di alcune attività SSD. In sostanza, il processo di McGraw si focalizza su:

- incorporazione dei requisiti di sicurezza,
- esecuzione dell'analisi dei rischi durante le diverse fasi di sviluppo,
- applicazione di metodi di security assurance quali test di sicurezza risk-based,
- analisi statica e test di penetrazione.

Il processo suggerisce anche di utilizzare l'analisi dei rischi durante la fase di progettazione. Per la fase di security assurance, McGraw suggerisce di utilizzare gli abuse cases e i requisiti di sicurezza per guidare i test di penetrazione.

[30] G. McGraw, Software Security: Building Security In, Addison Wesley, 2006

8.2.2 Microsoft Software Development Life Cycle (MS SDL)

MS SDL è un modello che pone molta attenzione alla fase di specifica dei requisiti durante la quale prevede di interagire con il cliente (end-user), al fine di identificare gli obiettivi e le caratteristiche di sicurezza necessarie.

L'incorporazione di queste caratteristiche/funzionalità di sicurezza sono guidate da standard di settore e criteri di certificazione. Durante la fase di progettazione MS SDL suggerisce di svolgere le seguenti attività: l'identificazione dei componenti critici per la sicurezza, l'identificazione di tecniche di progettazione e linee guida, l'identificazione dei punti di accesso degli attacchi, la modellazione delle minacce e analisi del rischio componente per componente, l'identificazione dei requisiti di sicurezza per mitigare le minacce, l'identificazione dei componenti che necessitano di particolare attenzione durante le fasi di test e review del codice, e i criteri per il completamento del software.

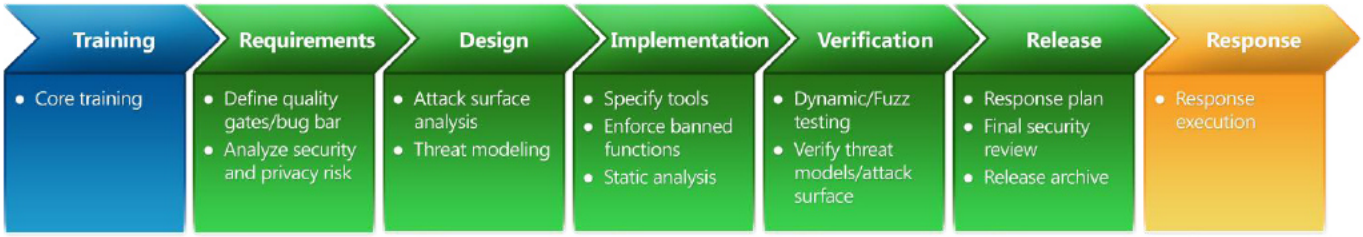

Figura 17 - Microsoft SDL

MS SDL consiglia di seguire gli standard di secure coding nella fase di implementazione. L'accento è posto su:

- test specifici di sicurezza,
- analisi statica del codice utilizzando i tool SDL utili a tale scopo,
- revisione del codice (code review) nell'ultimo step della fase di implementazione. Terminata la fase di implementazione, il software completo viene nuovamente verificato attraverso un ulteriore test di sicurezza che si concentra principalmente sui componenti critici (si esaminano ad esempio, i punti di ingresso alle possibili aree di attacco).

URL	https://www.microsoft.com/en-us/securityengineering/sdl
Contact	https://support.microsoft.com/it-it/contactus/?ws=mscom#tab0 Email, chat, phone and address
Country of HQ location	US
Geographic Scope	International
Type	Industry (Microsoft)

Risultati più rilevanti:

Guidance	Microsoft SDL Process Guidance Questa guida illustra il modo in cui Microsoft applica il SDL ai suoi prodotti e tecnologie. Include requisiti di sicurezza e privacy e raccomandazioni per lo sviluppo di software sicuro. Si rivolge ai modelli di sviluppo Cascade, Spiral, Agile. I responsabili delle politiche IT e le organizzazioni di sviluppo software possono sfruttare questi

	contenuti per migliorare l'aspetto di sicurezza e di privacy.
	Microsoft SDL for Agile Development
	Microsoft SDL for Line-of-Business Applications
	The Security Development Lifecycle Fornisce una guida attraverso ogni fase della SDL, dall'istruzione e progettazione alla sperimentazione e post-rilascio. Gli autori sono esperti di sicurezza del team Microsoft Security Engineering.
	Simplified Implementation of the Microsoft SDL Questo documento illustra i concetti chiave e le singole attività di sicurezza che devono essere eseguite per la conformità con il processo Microsoft SDL. Gli aspetti da tenere in considerazione includono ruoli e responsabilità, attività di sicurezza obbligatorie, attività di sicurezza opzionali e processo di verifica della sicurezza dell'applicazione.
	SDL Quick Security Reference (QSR) Con SDL QSR, il team SDL introduce una serie di documenti di orientamento di base, progettati per affrontare le vulnerabilità comuni, dal punto di vista di molteplici ruoli aziendali: decisori aziendali, architetti, sviluppatori e tester / QA.
	Securing Applications Questa documentazione è rivolta agli sviluppatori di .NET Framework per la scrittura di codice sicuro. Comprende: concetti chiave sulla sicurezza, sicurezza dell'accesso al codice, sicurezza basata sui ruoli, servizi crittografici, gestione delle politiche di sicurezza, best practice sulle politiche di sicurezza, linee guida per la codifica sicura e strumenti di sicurezza.
Tools & Templates	Microsoft SDL Threat Modeling Tool La modellazione delle minacce consente agli architetti di software di identificare e mitigare tempestivamente potenziali problemi di sicurezza, quando risolverli è relativamente facile ed economico. È uno strumento gratuito che richiede Visio. Lo strumento è focalizzato sulle tecniche di analisi del progetto.
	Microsoft SDL Process Template Un modello scaricabile che incorpora automaticamente la policy, il processo e gli strumenti associati a SDL nell'ambiente di sviluppo software di Visual Studio.
	MSF-Agile+SDL Process Template Un modello scaricabile che incorpora automaticamente la policy, il processo e gli strumenti associati alla guida allo sviluppo di SDL per Agile, nel Microsoft Solutions Framework per lo sviluppo di software Agile (MSF-Agile) e nell'ambiente Visual Studio.
	The Microsoft SDL Tools Una mappa degli strumenti e dei modelli gratuiti disponibili per ogni fase SDL.

8.2.3 Appropriate and Effective Guidance for Information Security (AEGIS)

AEGIS [31] [32] [2] [3] è un processo SSDLC basato sul modello a spirale e si concentra sulla specifica dei requisiti di sicurezza, identificando gli elementi principali ed eseguendo l'analisi dei rischi. Le fasi di analisi dei

[31] I. Flechais, M.A. Sasse, and S.M.V. Hales, "Bringing Security Home: A Process for Developing Secure and Usable Systems," In Proc. of the New Security Paradigms Workshop (NSPW'07), Ascona, Switzerland, ACM Press, 2003, pp. 49-57.

requisiti e di disegno sono strettamente collegati. Il modello propone quattro sessioni di progettazione tra gli sviluppatori e gli stakeholders del software. La prima e la seconda sessione modellano le caratteristiche principali del software e le loro relazioni, identificando i requisiti di alto livello di riservatezza, integrità e disponibilità. Nella terza sessione vengono identificati rischi, vulnerabilità e minacce per il software. La quarta sessione, orientata alla progettazione, indica i requisiti di sicurezza per rimuovere le vulnerabilità identificate.

AEGIS suggerisce anche una metodologia di analisi dei rischi da utilizzare durante le sessioni 3 e 4 finalizzate alla progettazione. Questo metodo di analisi dei rischi ha le seguenti fasi principali:

- Determinazione delle vulnerabilità.
- Determinazione del costo e della probabilità di un attacco in ambiente distribuito (inclusi i ruoli delle persone coinvolte e i task che verranno eseguiti sul software).
- Selezione dei requisiti di sicurezza basate sulle indicazioni dell'esperto di sicurezza.
- Valutazione costi-benefici dei requisiti di sicurezza selezionati.
- Il confronto tra il costo di ogni attacco, commisurato con la probabilità che possa verificarsi, e il costo dei requisiti di sicurezza.
- Selezione dei requisiti di sicurezza sulla base dell'efficacia e dei costi.

8.2.4 Secure Software Development Model (SSDM)

SSDM[33] è un processo che incorpora diverse attività di sicurezza in un modello SDLC a cascata (cascade). Secondo SSDM, la modellazione delle minacce dovrebbe essere eseguita in fase di specifica dei requisiti. Il risultato di questa modellazione dovrebbe essere una check-list contenente tutte le potenziali vulnerabilità e attacchi. Tali elenchi di fatto dovrebbero essere dati in input alla fase di sviluppo.

Dopo la modellazione delle minacce, è necessario definire una policy che indichi chiaramente come saranno raggiunti gli obiettivi di sicurezza prefissati.

Tale policy, come sottolineato dal SSDM, è un insieme di decisioni di gestione di alto livello come ad esempio minimizza l'impatto degli errori in tutto il processo di sviluppo, correggendoli non appena vengono rilevati. I test di penetrazione rappresentano, nel modello SSDM, l'unica attività SSD per la fase security assurance.

8.2.5 Aprville and Pourzandi's Secure Software Development Life Cycle Process

[4]Aprville e Pourzandi[34] propongono un processo SSDLC sulla base della loro esperienza, maturata durante lo sviluppo di un software di instant messaging. Secondo il loro processo [5], il primo passo nella fase di specifica dei requisiti è quello di individuare gli obiettivi di alto livello, per quanto riguarda la sicurezza (riservatezza, integrità e disponibilità) del software in fase di sviluppo, considerando il suo ambiente di distribuzione. Per gli obiettivi di sicurezza a basso livello, la modellazione delle minacce dovrebbe essere di supporto nella costruzione di un insieme di requisiti di sicurezza. La priorità di tali requisiti può essere

---

[32] I. Flechais, C. Mascolo, and M.A. Sasse, "Integrating Security and Usability into the Requirements and Design Process," International Journal of Electronic Security and Digital Forensics, Inderscience Publishers, Geneva, Switzerland, 2007, vol. 1, no. 1, pp. 12-26.

[33] A.S. Sodiya, S.A. Onashoga, and O.B. Ajayi, "Towards Building Secure Software Systems," Issues in Informing Science and Information Technology, Informing Science Institute, California, USA, 2006, vol. 3, pp. 635-646.

[34] A. Apvrille and M. Pourzandi, "Secure Software Development by Example," IEEE Security and Privacy, IEEE CS Press, 2005, vol. 3, no. 4, pp. 10-17.

modificata in base ai risultati dell'analisi del rischio. In fase di progettazione, si raccomanda l'uso di [6]UMLsec[35]. Per la fase di implementazione, si suggerisce di scegliere un linguaggio di programmazione che meglio soddisfa gli obiettivi di sicurezza. Inoltre, particolare attenzione deve essere posta su come evitare: (i) buffer overflow, (ii) format string vulnerabilities. Essi sottolineano di utilizzare per la crittografia algoritmi già verificati. Per la fase di security assurance vengono indicate le seguenti attività: static vulnerability code scanning, code reviews, ad-hoc unit e system security testing, fuzz testing.

8.2.6 Secure Software Development Model (SecSDM)

SecSDM[36] utilizza l'analisi dei rischi nella fase di specifica dei requisiti al fine di dare priorità alla modellazione delle minacce. Gli obiettivi di sicurezza di alto livello quali la riservatezza, l'integrità e la disponibilità sono poi identificati sulla base delle minacce rilevate [7].

In fase di progettazione, vengono identificate e selezionate le funzionalità di sicurezza per mitigare le minacce e raggiungere gli obiettivi di sicurezza. SecSDM propone di seguire standard di secure coding durante la fase di implementazione.

8.2.7 Software Security Assessment Instrument (SSAI)

SSAI[3738] raggruppa un insieme di attività che utilizzano determinate risorse e strumenti per lo sviluppo di software sicuro. [8] [9] La prima risorsa che SSAI fornisce è un database online[39] che contiene informazioni sulle varie vulnerabilità e le indicazioni per la loro mitigazione. [10] La seconda risorsa SSAI è una security checklist che può essere sviluppata e utilizzata come guida per lo sviluppo sicuro. Sono forniti i dettagli di come redigere una checklist e quali sono gli elementi potenziali che possono essere inclusi[40]. La terza risorsa è un elenco di strumenti accessibili pubblicamente, per la scansione statica del codice. SSAI fornisce anche Flexible Modeling Framework (FMF), uno strumento di modellazione e il property-based testing tool (PBT), che utilizza le proprietà di sicurezza specificate nella security checklist o nel FMF come base dei test per il software.

8.2.8 Hadawi's Set of Secure Development Activities

Hadawi[41] identifica 25 vulnerabilità (common vulnerabilities) da evitare durante lo sviluppo [11]. Egli propone anche una serie di requisiti di sicurezza per le fasi di progettazione e implementazione che, se adottati, aiuterebbero ad evitare queste vulnerabilità [12].

---

[35] J. Juerjens, Secure Systems Development with UML, Springer, 2005.

[36] L. Futcher and R.v. Solms, "SecSDM: A Model for Integrating Security into the Software Development Life Cycle," In IFIP International Federation for Information Processing, Volume 237, Proc. of the 5th World Conference on Information Security Education, Springer, 2007, pp. 41-48.

[37] D.P. Gilliam, T.L. Wolfe, J.S. Sherif, and M. Bishop, "Software Security Checklist for the Software Life Cycle," In Proc. of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria, IEEE CS Press, 2003, pp. 243-248.

[38] D. Gilliam, J. Powell, E. Haugh, and M. Bishop, "Addressing Software Security Risk and Mitigations in the Life Cycle," In Proc. of the 28th Annual NASA Goddard Software Engineering Workshop (SEW'03), Greenbelt, Maryland, USA, 2003, pp. 201-206.

[39] DOVES: Database of Vulnerabilities, Exploits, and Signatures, http://seclab.cs.ucdavis.edu/projects/DOVES/. Last Accessed March 2009.

[40] D.P. Gilliam, T.L. Wolfe, J.S. Sherif, and M. Bishop, "Software Security Checklist for the Software Life Cycle," In Proc. of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria, IEEE CS Press, 2003, pp. 243-248.

[41] M.A. Hadawi, "Vulnerability Prevention in Software Development Process," In Proc. of the 10th International Conference on Computer & Information Technology (ICCIT'07), Dhaka, Bangladesh, 2007,

Durante la fase di implementazione, l'unica attività SSD è la scelta di un appropriato linguaggio di programmazione (sicuro). Per la fase di security assurance, Hadawi consiglia di utilizzare: (i) security code reviews, (ii) static code analysis tools.

8.2.9 Comprehensive, Lightweight Application Security Process (CLASP)

Comprehensive, Lightweight Application Security Process (CLASP)[42] identifica un insieme di attività SSD classificate in base ai ruoli svolti durante lo sviluppo. CLASP suggerisce l'impiego di un esperto di sicurezza fin dall'inizio dello sviluppo. Per la fase di specifica dei requisiti, sottolinea la necessità di un'analisi dei rischi e della modellazione delle minacce. L'analisi dei rischi e la modellazione delle minacce devono essere eseguite anche nella fase di progettazione.

CLASP propone di annotare i diagrammi di classe con le informazioni di sicurezza. Nella fase di security assurance, consiglia di effettuare le seguenti operazioni: security code reviews, security code scanning, security testing.

CLASP fornisce anche un elenco di vulnerabilità (common vulnerabilities) con informazioni complete su come e quando possono essere introdotti durante lo sviluppo e come evitarli.

URL	https://www.owasp.org/index.php/CLASP_Concepts

Risultati più rilevanti:

Security Process	CLASP version 1.2

8.2.10 Secure Software Development Process Model (S2D-ProM)

S2D-PROM[43] specifica molteplici strategie possibili per avanzare da ogni fase di sviluppo all'altra [13]. Alla base di questo processo, c'è l'idea di fornire agli sviluppatori opzioni flessibili. Il processo si propone di condurre l'analisi dei rischi durante le fasi di specifica dei requisiti, progettazione, e implementazione. L'analisi del rischio, secondo S2D-PROM, può essere eseguita in modi diversi per ogni fase di sviluppo. I rischi identificati possono essere mitigati utilizzando varie strategie (ad esempio, definendo le norme di sicurezza o utilizzando meccanismi di difesa).

8.2.11 Team Software Process for Secure Software Development (TSP Secure)

[14]TSP-Secure[44] garantisce la sicurezza attraverso:

- la pianificazione per la sicurezza,
- la qualità e la gestione della sicurezza in tutto il ciclo di vita dello sviluppo,
- la formazione degli sviluppatori circa gli aspetti relativi alla sicurezza.

[42] https://www.owasp.org/index.php/CLASP_Concepts
[43] M. Essafi, L. Labed, and H.B. Ghezala, "S2D-ProM: A Strategy Oriented Process Model for Secure Software Development," In Proc. of the 2nd International Conference on Software Engineering Advances (ICSEA'07), Cap Esterel, French Riviera, France, 2007, p. 24.
[44] N. Davis, "Secure Software Development Life Cycle Processes: A Technology Scouting Report", technical note CMU/SEI-2005-TN-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsalyania, USA, 2005.

Durante la fase di progettazione, il team identifica obiettivi di sicurezza e produce un piano dettagliato come guida per lo sviluppo. Le attività di sviluppo possono includere l'identificazione dei rischi, l'identificazione dei requisiti di sicurezza, la progettazione sicura, le revisioni del codice, gli unit test, i fuzz test e l'analisi statica del codice. Il team può scegliere qualsiasi attività SSD che ritiene necessaria.

Secondo TSP-Secure, un membro del team svolge il ruolo di responsabile della sicurezza, facendosi carico di tutte le problematiche relative alla sicurezza.