5 PROGETTAZIONE E SVILUPPO DELL'APPLICAZIONE: DIRETTIVE STANDARD

5.1 Progettazione dell'applicazione

L'architettura dell'applicazione deve essere progettata e sviluppata secondo i paradigmi standard dell'industria del software, quali: Architettura monolitica (mainframe), Client server, Service Oriented Architecture (SOA), ecc.
Nel corso della fase di progettazione è necessario garantire un adeguato livello di sicurezza applicativa e infrastrutturale attraverso l'analisi e la modellazione delle minacce relative agli applicativi coinvolti, delle interfacce e degli agenti che potrebbero minacciare il sistema. Per l'analisi della sicurezza applicativa di un'architettura di sistema si adotta un approccio differente a seconda che si tratti di progettazione di applicazioni ex-novo (approccio Secure by Design) piuttosto che di reingegnerizzazione di applicazioni esistenti (approccio Security Control). Nel dettaglio:

- **PROGETTAZIONE SICURA BY DESIGN** - Durante le fasi di analisi della sicurezza applicativa di una architettura di sistema (da definire o in fase di rivisitazione) è necessaria l'attuazione di pratiche di progettazione sicura attraverso l'individuazione di requisiti di sicurezza e contromisure secondo i Security by Design Principles. Le pratiche di progettazione sicura realizzano la sicurezza delle informazioni attraverso un approccio di "Defense in Depth" del layer applicativo. La "difesa in profondità" ha come scopo limitare al minimo i danni in caso di attacco riuscito. In pratica, nell'ipotesi che un attaccante riesca a oltrepassare il primo livello di difesa (ad esempio aggirando il controllo di autenticazione), ulteriori misure più restrittive devono intervenire per ostacolarne l'avanzata (ad esempio, restringendo al minimo i privilegi d'accesso alle risorse o applicando la compartimentazione dell'applicazione al fine di ostacolare bloccare la propagazione dell'attacco all'intero sistema).
- **SECURITY CONTROL** (su applicazione esistente) – È necessario: 1) Identificare, quantificare e risolvere i rischi di sicurezza associati ad un'interfaccia, un'applicazione e/o un sistema esistenti. 2) Validare dal punto di vista della sicurezza applicativa gli sviluppi realizzati da terze parti (sicurezza della supply chain). 3) Tutelare il proprio patrimonio informativo e i dati.

Le tecniche di modellazione delle minacce e d'identificazione delle relative contromisure, finalizzate a indirizzare i requisiti di sicurezza applicativa di un'architettura di sistema, insieme alle pratiche di progettazione sicura, sono trattate in dettaglio nell'*Allegato 4* - ***Linee Guida per la modellazione delle minacce ed individuazione delle azioni di mitigazione conformi ai principi del Secure/Privacy by Design.***

5.2 Sviluppo dell'applicazione – Criteri Generali

Nel corso della fase di sviluppo di un'applicazione, si raccomanda l'adozione dei criteri generali riportati nei paragrafi successivi.

5.2.1 Performance

Le soluzioni di programmazione impiegate devono ridurre al minimo l'impatto sulle risorse di sistema.
È necessario:

- non ottimizzare mai manualmente ciò che può essere ottimizzato dai compilatori;
- per i linguaggi che accedono direttamente alla memoria del sistema, evitare di avere puntatori multipli ad una determinata risorsa;
- utilizzare i data-types appropriati (es: non utilizzare long quando int è sufficiente);
- utilizzare switch/case al posto di strutture nidificate di if;
- porre le risorse più frequentemente utilizzate le une vicine alle altre;
- allocare la memoria il più tardi possibile (costruzione degli oggetti);

- deallocare la memoria il più presto possibile (distruzione degli oggetti) laddove tale operazione non pregiudichi la sicurezza dell'applicazione;
- compilare il software per la piattaforma di utilizzo (es: non compilare per architettura hardware 64-bit se non è necessario).

5.2.2 Password nel codice sorgente

I dati di accesso (username/password/nome db/ecc..) ai database o a sistemi di altra natura non devono mai essere inseriti all'interno dei sorgenti.
Nei casi in cui non sia possibile, tali dati devono apparire in forma cifrata. Per le chiavi di cifratura e in generale per tutte le informazioni riservate valgono le stesse indicazioni.

5.2.3 Privilegi esecutivi minimi

Quando l'applicazione viene avviata all'interno del sistema operativo, porta con sé i privilegi dell'utenza che effettua l'operazione. L'applicazione non deve essere lanciata con i privilegi amministrativi.

5.2.4 Metodi TRACE e TRACK

Uno dei principi di sicurezza più saggi afferma che ciò che non viene utilizzato dovrebbe essere disabilitato. Nelle applicazioni Web è obbligatoria la disattivazione lato server del metodo HTTP TRACE o del metodo TRACK (utilizzato in ambienti Microsoft IIS). Tali metodi consentono al client di vedere ciò che viene ricevuto dal web server. Tali informazioni possono poi essere utilizzate per organizzare un attacco di Cross Site Scripting. Si parla di "Cross Site Tracing" (XST).

5.2.5 Assenza di codice malevolo

L'applicazione non deve contentere alcun tipo malware (malicious software): virus, trojan, rootkit, worms, ramsonware, ecc.
Sono da considerare potenzialmente pericolose anche le backdoor amministrative, poiché consentono l'accesso alle macchine in rete bypassando il processo di autenticazione. Un attaccante che trovasse il modo di manomettere una backdoor amministrativa, potrebbe penetrare nelle macchine e prenderne il controllo.

5.2.6 Fattore integrità

Il concetto di integrità del software include la resilienza agli attacchi informatici e alle violazioni della privacy, ma essenzialmente sta a indicare che possano essere impedite modifiche non autorizzate.
La fase di progettazione e la successiva fase d'implementazione devono assicurare che tutti gli errori e le eccezioni rilevati durante il processamento e l'elaborazione dei dati acquisiti in ingresso siano correttamente gestiti, in modo che non causino il danneggiamento o la perdita di integrità delle informazioni.

5.2.7 Input character validation

L'applicazione deve assicurare, attraverso opportuni meccanismi di convalida, che tutti i parametri in input, specificati dall'utente, siano congruenti a quanto atteso.
In particolare, sui dati acquisiti in ingresso, l'applicazione deve prevedere l'implementazione di meccanismi di controllo che limitino il set di caratteri o valori, inseribili dall'utente, solo a quelli congruenti ai campi richiesti e/o alle form di pertinenza, al fine di identificare e annullare gli effetti dei seguenti errori:

- Valori out-of-range o non pertinenti (ad esempio l'immissione di caratteri non numerici nel campo "anno di nascita");
- Caratteri invalidi negli stream o nei data field;
- Dati mancanti o incompleti;
- Limite del minimo volume di dati richiesti non soddisfatto o del massimo volume di dati acquisibile in ingresso raggiunto;

Per quanto riguarda i caratteri speciali, se presenti/richiesti in input, sono considerati pericolosi (innescano diverse vulnerabilità, Cfr. [paragrafo 6.1.1]) poichè la loro combinazione non può essere considerata semplice 'testo'.
Di seguito qualche esempio di possibile combinazione:

Caratteri pericolosi	Possibile utilizzazione
< >	identificano tag HTML
! | & ;	esecuzione comandi
' " * %	database queries
? $ @	programmi e script
( ) [ ]	programmi e script
..\./	filesystem paths

Inoltre, caratteri speciali quali:

; | ! ~ ' " - * % ` \ / < > ? $ @ : ( ) [ ] { } .

devono essere identificati e neutralizzati (input sanitizing) con tecniche specifiche quali l'escaping (i caratteri pericolosi devono essere sempre convertiti prima del salvataggio), di seguito alcuni esempi di sostituzione:

Carattere pericoloso	Sostituito con
<	&lt;
>	&gt;
#	&#35;
&	&#38;
(	&#40;
)	&#41;

Il controllo, quindi, deve sempre verificare che non siano inseriti script potenzialmente dannosi. È importante sottolineare che la convalida dell'input utente non deve mai essere svolta lato client, ma sempre dal back-end, sul server, poiché sul client i dati sono sempre visibili e modificabili.

5.2.8 Gestione dell'output

L'applicazione deve fornire in output solamente le informazioni pertinenti e conformi alle richieste avanzate dagli utenti, al fine di evitare qualsiasi raccolta d'informazioni (information gathering) o rivelazione di dati (disclosure) non autorizzate.

5.3 Formattazione del codice

La formattazione del codice e la sintassi devono seguire le seguenti direttive standard:

- Ogni file deve contenere un'intestazione (file header) in cui si specificano l'autore del codice, la data di creazione dello stesso e la storia degli aggiornamenti successivi (se presenti);
- Ogni file header deve contenere la dichiarazione di una ed una sola classe;
- Le dichiarazioni correlate ad una classe riportata all'interno di un file, devono essere poste all'interno dello stesso file;
- Le righe di codice devono avere un numero di caratteri uguale o inferiore a quello previsto dal formato ISO/ANSI per la descrizione delle dimensioni dello schermo (80 caratteri x 24 righe).

5.3.1 Stile e sintassi

Alla dichiarazione di ogni funzione, metodo o classe deve sempre precedere un commento che riporti:

- Scopo della funzione;
- Parametri di input e output a/dalla funzione;
- Valori di ritorno dei parametri di output;
- Tracciamento degli aggiornamenti del codice della funzione (data ultima modifica).
- Le parentesi graffe, nel codice, devono essere apposte sulla riga superiore e inferiore rispetto alla dichiarazione del costrutto linguistico (struttura, classe, funzione, metodo, etc.).
- È raccomandato che ogni funzione assolva un unico compito, in maniera efficiente ed efficace.

5.3.2 Algoritmi

Nell'ottica di rendere l'applicazione conforme agli standard internazionali è richiesto l'utilizzo esclusivo di algoritmi riconosciuti nell'industria del software. Gli standard internazionali devono essere strettamente seguiti per lo sviluppo di algoritmi crittografici e processi di autenticazione.

5.3.3 Utilizzo funzioni di gestione delle stringhe

Tutto l'input utente processato dall'applicazione deve passare per funzioni sicure di gestione delle stringhe che ne prevedono il bound-checking (controllo del range di validità). L'applicazione deve risultare immune da problematiche di tipo stack overflow, off by one/off by few overflow o heap overflow.

5.3.4 Specifica del formato delle stringhe

Nei sorgenti dell'applicazione il formato delle stringhe deve essere sempre specificato nei parametri delle funzioni che lo prevedono e mai dato per assunto. L'applicazione deve risultare immune da problematiche di tipo format string overflow.

5.3.5 Casting e variabili numeriche

L'input utente deve essere filtrato in modo che alle variabili o strutture dati interne dell'applicazione non sia possibile assegnare valori negativi (ad esempio dichiarando array come signed integer) ad eccezione dei casi previsti e per i quali sia stata pianificata la gestione. In fase di comparazione di due variabili numeriche dove il contenuto di almeno una deriva da input utente, il casting o l'assegnazione di un valore da una variabile all'altra deve avvenire in base alla stessa tipologia (ad esempio assegnare un valore intero a una variabile di tipo short è un errore). L'applicazione deve risultare immune da problematiche di tipo integer overflow, cambi di segno, troncamento di valori numerici o altri errori di programmazione logico-computazionali.

5.4 Tracciamento e Raccomandazioni di "Alarm Detection"

Per il tracciamento degli eventi di "Alarm Detection" si raccomanda l'adozione dei criteri generali riportati nei paragrafi (Cfr. [5.4.1- 5.4.4]) che seguono.

5.4.1 Tracciamento eventi

L'applicazione deve essere predisposta sia per il tracciamento di attività "anomale" sia per le "eccezioni" verificatesi sui sistemi.
Il tracciamento degli eventi può essere attivato su:

- Eventi andati a buon fine;
- Eventi non andati a buon fine;
- Errori di sistema o utente;
- La configurazione del sistema di tracciamento e detection degli allarmi sarà predisposta sulla base delle policy stabilite nell'ambito dei requisiti dell'applicazione.

Gli eventi per i quali è richiesto il tracciamento riguardano:

- Autenticazione e processi correlati;
- Start e Stop delle componenti dell'applicazione;
- Violazioni dei criteri o delle policy configurate;

- Modifiche alle configurazioni dell'applicazione;
- Accesso ai dati (inserimento, modifica, lettura, rimozione), ai file e alle risorse dell'applicazione e tipo di accesso;
- Disattivazione del meccanismo di tracciamento;
- La procedura di tracciamento sarà predisposta per l'emissione di "Alert" al verificarsi di uno o più eventi configurabili dall'amministratore del sistema.

5.4.2 Tracciamento eventi di "Alarm Detection"

Oltre ad attenersi alle prescrizioni riportate nei paragrafi precedenti, durante lo sviluppo del codice è essenziale inserire particolari funzioni di tracciamento che, operanti in determinati e specifici punti dell'applicativo, permettano la rilevazione e il logging di eventi anomali o di frode, significativi per la sicurezza dell'organizzazione.
Attraverso l'inserimento di specifiche stringhe di codice all'interno dell'applicativo, si vogliono rilevare alcuni eventi ritenuti sensibili ai fini del mantenimento della riservatezza, integrità e disponibilità del dato applicativo.
In seguito, le segnalazioni prodotte e inserite in appositi file di Log, discriminate per mezzo di TAG (DetCode) opportuni, possono essere elaborate da un sistema di correlazione e utilizzate come fonte per attività di Audit (Ex/Post) degli eventi di sicurezza.

Questa nuova strategia di rilevazione, risulta strettamente necessaria per superare i limiti tecnologici intrinseci delle tecnologie Anti-Intrusione commerciali. In particolare, tali tecnologie non permettono:

- l'analisi di flussi applicativi di applicazioni dell'ente di tipo "Make" (le soluzioni di mercato sono progettate per l'esclusivo utilizzo su applicazioni di tipo commerciale);
- l'analisi di flussi applicativi che fanno uso di meccanismi di cifratura delle informazioni;
- la rilevazione di vulnerabilità software determinate da errori in input commessi dall'utente;
- la rilevazione di vulnerabilità software determinate dall'assenza di controlli applicativi durante le operazioni di allocazione di blocchi di memoria nelle aree di memoria volatile.

5.4.3 Scopo e campo di applicazione per eventi di "Alarm Detection"

Il software sviluppato e personalizzato per l'organizzazione è realizzato seguendo le indicazioni e le necessità espresse dall'organizzazione medesima, nel rispetto dei vincoli di sicurezza imposti nel Piano di Sicurezza (in seguito PdS).
Nella fase di produzione e/o aggiornamento del Piano di Sicurezza di una specifica applicazione, insieme all'esame del funzionamento, all'analisi delle informazioni da esso trattate e all'analisi dei flussi applicativi pertinenti (input, output, accesso a DB, autenticazione, ecc.), si procederà all'individuazione delle raccomandazioni degli eventi di Alarm Detection che permetteranno, alle competenti linee di Sviluppo, di identificare e implementare gli opportuni meccanismi di generazione delle informazioni di tracciamento.

5.4.4 Raccomandazioni generali per eventi di "Alarm Detection"

L'attivazione ed il tracciamento per gli eventi di Alarm Detection, di seguito elencati, sono fortemente raccomandati, poichè riguardano alcune delle principali debolezze applicative che, se utilizzate per scopi malevoli, possono comportare un elevato fattore di rischio:

- **Validazione Input**: si devono tracciare tutti gli input (provenienti da Client o da Server) non conformi con quanto atteso dall'applicativo (Cfr. [paragrafo 5.2.7]);
- **Buffer Overflow**: si devono tracciare tutti gli avvisi e/o le eccezioni generate dall'applicativo a fronte di un evento di Buffer Overflow (Cfr. [paragrafo 6.1.7]);
- **Sessioni applicative anomale**: si devono tracciare le occorrenze di eventi che non rientrano nella corretta gestione delle sessioni applicative, come tentativi massivi di autenticazione, sessioni multiple dell'utente non previste e/o consentite, presenza di cookie con contenuti incomprensibili, referrer errato o inconsistente con la funzione o con la pagina chiamata, etc.(Cfr.[paragrafo 6.1.2]);

- **Tentativi di accesso a risorse inibite**: si devono tracciare tutti i tentativi di accesso a risorse inibite ai servizi come, ad esempio, tentativi di accesso alla root di un server web, modifica a configurazioni per mezzo di credenziali non appropriate, etc. (Cfr. [paragrafo 5.6]);
- **Violazioni delle policy configurate**: si devono tracciare le violazioni o i tentativi di bypass delle regole di autorizzazione che definiscono ruolo e permessi assegnati all'utente nonché le operazioni ad esso concesse in base alla tipologia di profilatura (Cfr. [paragrafo 5.6]);
- **Process Issue**: si devono tracciare gli avvisi, generati in ambito Server Applicativo, relativi all'esecuzione di moduli applicativi che risultano diversi in quantità e dimensione rispetto a quanto atteso/definito in fase di progettazione/realizzazione dell'applicativo stesso (ad es. numero eccessivo di istanze duplicate, esecuzione di istruzioni non previste, eccessiva occupazione di memoria, etc.) -(Cfr. [paragrafo 6.1.7]);
- **Funzioni input/output anomale**: si devono tracciare i tentativi inaspettati di dichiarazioni di funzioni e/o comandi in input ed in output (Cfr. [paragrafo 5.2.7 , 5.2.8, cap. 6]);
- **Disattivazione anomala del meccanismo di tracciamento**: devono essere osservati e tracciati tutti i cambiamenti di stato (attivo ↔ disattivo) delle funzioni di tracciamento e generazione allarmi, su tutte le componenti funzionalmente coinvolte. Altresì, è necessario tenere sempre sotto controllo le attività di download/upload dell'utente, al quale è stato consentito l'accesso al sistema, al fine di individuare situazioni anomale (generazione di allarmi laddove la quantità di dati superi una certa soglia che tiene conto del livello/ruolo di accesso dell'utente).

5.5 Compilazione dell'applicazione

Per la compilazione del codice dell'applicazione si raccomanda l'adozione dei criteri riportati nei paragrafi (Cfr. [5.5.1,5.5.2]) che seguono.

5.5.1 Stack Canary

I sorgenti dell'applicazione e delle librerie che la compongono (DLL o altre forme comparabili in ambienti operativi differenti) devono essere compilati con funzionalità di stack canary. A runtime viene impostato un valore (spesso un intero) in memoria e viene verificato che non venga sovrascritto da un eventuale buffer overflow, dopo una chiamata allo stack. Ciò permette di bloccare gli effetti di un buffer overflow in tempo utile. In fase di compilazione, devono essere attivate opzioni di anti-sovversione dei puntatori ai gestori delle eccezioni (ad esempio SafeSEH), relativamente alla piattaforma dell'applicazione.

5.5.2 Correttezza del sorgente

La compilazione dei sorgenti deve terminare senza alcun tipo di warning.

5.6 Ambiente operativo dell'applicazione

In merito agli ambienti operativi di sviluppo e test delle applicazioni, si raccomanda l'adozione dei criteri riportati nei paragrafi (Cfr. [5.6.1 - 5.6.6]) che seguono.

5.6.1 Separazione degli ambienti

I sistemi di sviluppo, test e produzione devono essere separati fisicamente e/o logicamente.

5.6.2 Test dell'Applicazione

- L'applicazione deve essere consegnata e portata in produzione/esercizio solo dopo essere stata verificata la rispondenza ai requisiti dati.
- I casi di test devono includere controlli sull'usabilità dell'applicazione, sulla sicurezza e sulla compatibilità con l'infrastruttura hardware/software in cui andrà installata.
- È raccomandato l'utilizzo di appositi strumenti di stress test prima dell'avvio in esercizio dell'applicazione, al fine di certificare la corretta implementazione delle procedure di input data validation e security menzionate in questo documento.

5.6.3 Strumenti

Compilatori, editor ed altri strumenti di sviluppo non devono essere presenti nei sistemi di produzione in cui l'applicazione risiede.

5.6.4 Profili utente

I profili utente dell'applicazione che risiede nei sistemi di produzione devono essere differenti da quelli configurati e utilizzati nei sistemi di sviluppo e test. L'applicazione deve implementare un meccanismo di avviso della tipologia di profilatura, ruoli e permessi assegnati all'utente a seguito dell'accesso (vedasi per maggior dettaglio "Procedura di accesso dell'applicazione" Cfr. [paragrafo 5.7.3]).

5.6.5 Trattamento dei dati

I dati personali e critici, gestiti dall'applicazione, che risiedono nell'ambiente di esercizio, non devono essere copiati negli ambienti di test e sviluppo. In caso di utilizzo dell'applicazione al solo fine di test questi devono essere rimossi immediatamente dopo il completamento di detta fase.

5.6.6 Protezione dei sorgenti e delle librerie

I sorgenti dell'applicazione e delle librerie correlate, fatta eccezione per i linguaggi interpretati, non devono risiedere in testo chiaro all'interno dei sistemi di esercizio, bensì sotto forma di oggetti compilati. Nel caso di linguaggi interpretati, il sorgente dell'applicazione che risiede nei sistemi di esercizio deve essere offuscato.
Una copia non offuscata deve comunque sempre essere conservata su un supporto diverso (esempio copia su CD o DVD).

5.7 Autenticazione, Autorizzazione e Gestione degli accessi

Per le politiche degli accessi si raccomanda l'adozione dei criteri riportati di seguito.

5.7.1 Policy standard "Everything is generally forbidden unless expressly permitted"

L'applicazione deve implementare un meccanismo di access control adeguato. Tutte le operazioni svolte dagli utenti e le fasi di autorizzazione e assegnazione dei permessi devono essere subordinate alla policy standard : "Ogni azione è negata se non espressamente consentita".

5.7.2 Assegnazione dei privilegi utente

L'applicazione non deve assegnare alcun privilegio/permesso all'utente fin quando il processo di autenticazione e autorizzazione non è stato completato.

5.7.3 Procedura di accesso dell'applicazione

La procedura di accesso e log-on dell'applicazione deve ridurre al minimo le informazioni fornite agli utenti non ancora autenticati e prevedere determinati comportamenti. In particolare:

- Non deve con messaggi specifici fornire alcun tipo di aiuto, né rendere comprensibile se il processo di autenticazione è fallito a causa del nome utente o della password errata;
- Non deve fornire alcuna chiara indicazione sui ruoli e sui permessi assegnati a un utente fin quando il processo di autenticazione non viene completato;
- Deve visualizzare un messaggio di avviso sulle sanzioni derivate dall'accesso fraudolento all'applicazione;
- Deve prevedere il mascheramento della password digitata dall'utente non rendendola visibile o nascondendola attraverso simboli (ad esempio con asterischi);
- Non deve trasmettere in rete la password in chiaro;
- Deve "processare" le informazioni fornite dall'utente per l'accesso solo quando sono complete;
- Deve prevedere procedure configurabili di blocco momentaneo dell'account dopo una serie di tentativi d'accesso infruttuosi;

- Deve visualizzare, al completamento della procedura di autenticazione, la data, l'ora e le informazioni sull'ultimo sistema (indirizzo IP/FQDN) che ha completato con successo la fase di log-on per una specifica utenza;
- Deve visualizzare nella console dell'amministratore o nei file di log, i dettagli di tutti i precedenti tentativi infruttuosi di accesso per una specifica utenza;
- L'autenticazione non deve mai essere un processo convalidato lato client.

5.7.4 Account standard

L'applicazione non deve essere rilasciata da chi la sviluppa, con account utente standard di tipo amministrativo/operativo o con account protetti tramite password di default.

5.7.5 Autorizzazione

L'applicazione deve sempre operare un controllo sui reali privilegi d'accesso dell'utente prima di autorizzare qualsiasi operazione in lettura, scrittura, esecuzione o cancellazione.
L'autorizzazione non deve mai essere un processo convalidato lato client.

5.7.6 Generazione dei token

I token dell'applicazione devono essere generati utilizzando algoritmi true random ed analizzati ogniqualvolta l'utente richiede autorizzazione a svolgere una qualsiasi azione, al fine di determinarne permessi e privilegi.

5.7.7 Generazione dei cookie

Nelle applicazioni web i cookie di sessione applicativa devono essere cifrati, non persistent, avere il flag secure attivato e l'attributo HttpOnly impostato.

5.7.8 Contenuto del cookie

Un cookie non deve contenere informazioni critiche quali password o essere composto da parti predicibili come username o valori elaborati basandosi su algoritmi sequenziali. L'identificatore della sessione nel cookie deve avere un'entropia pari almeno a 128 bit.

5.7.9 Scadenza del cookie

Nelle applicazioni web, ciascun cookie generato deve essere soggetto a un tempo di scadenza oltre il quale non deve più essere considerato valido.

5.7.10 Logout utente

Quando un utente ha effettuato il log-out, la sessione relativa deve essere invalidata sia sul server (sganciandola nella Entry Table delle sessioni attive) che sul client (ad esempio rimuovendo il cookie o svuotando il suo contenuto).

5.7.11 Timeout di sessione

L'applicazione deve prevedere il rilascio della sessione utente dopo un certo periodo configurabile di inattività della sessione stessa.

5.7.12 Isolamento delle funzioni dall'applicazione

È vietata l'implementazione della sicurezza attraverso l'oscuramento delle funzioni a livello di presentazione. È obbligatorio invece isolare e rendere inutilizzabili le funzioni che non devono essere rese accessibili agli utenti, direttamente a livello logico (es: imponendo la consultazione del token della sessione per determinarne i reali privilegi di esecuzione).

5.8 Password, chiavi e certificati

Per la gestione di dati quali password, chiavi e certificati, si raccomanda l'adozione dei criteri riportati di seguito.

5.8.1 Gestione di password, chiavi e certificati

Le password mantenute dall'applicazione o le chiavi private dei certificati devono essere conservate in forma cifrata. Le informazioni sulle password e le chiavi devono risiedere in container (aree del filesystem, tabelle del database, ecc.) differenti rispetto ai dati dell'applicazione.

5.8.2 Trasmissione delle password in rete

Utilizzare protocolli crittografici, come TLS (Transport Layer Security) o SSH (Secure Socket Shell), che impiegano algoritmi standard di derivazione delle chiavi basata su password (Password-based Key Derivation/key stretching) detti anche algoritmi di slow hashing, come PBKDF2, scrypt o bcrypt, i quali, rallentando di molto la funzione di hashing, rendono inefficaci eventuali attacchi di forza bruta per il password cracking.
Prevedere, inoltre, l'aggiunta di una chiave segreta alla hash, in modo tale da consentire la convalida della password solo a coloro che la conoscono. Ciò si può fare cifrando l'hash con algoritmo AES oppure includendo la chiave segreta nell'hash utilizzando poi un algoritmo di hashing come HMAC.
È sconsigliato l'utilizzo di funzioni di hash crittografico veloce come MD5, SHA-1, SHA-256, SHA-512, RipeMD, WHIRLPOOL, SHA-3.

5.8.3 Generazione/conservazione delle password nel filesystem/DB

Le password memorizzate nel filesystem o nel DB sotto forma di hash (esempio MD5/SHA-1 etc.), devono prevedere l'introduzione di un ulteriore fattore randomico (salt) durante la loro generazione.

5.8.4 Batch Job dell'applicazione

Le informazioni, i dati o gli allegati trasmessi tramite i batch job dell'applicazione (ad esempio sessioni ftp o altri protocolli di rete non cifrati o proprietari), devono utilizzare canali di comunicazione sicuri come SSL o TLS, in cui le chiavi di cifratura simmetriche vengono scambiate all'interno di una comunicazione protetta attraverso algoritmo crittografico asimmetrico (Ad esempio RSA con dimensione delle chiavi uguale o superiore a 1024 bit).

5.8.5 Storage dei dati applicativi

I dati dell'applicazione memorizzati nel database o nel filesystem devono essere cifrati tramite algoritmi simmetrici con chiave pari almeno a 192 bit (inclusi i bit di parità).

5.8.6 Integrità delle informazioni

Tutti i dati di natura critica conservati e mantenuti dall'applicazione, oltre che cifrati, devono prevedere l'utilizzo di algoritmi di hashing o firma digitale per poterne vagliare l'integrità/autenticità.

5.8.7 Meccanismi di autenticazione

L'applicazione sviluppata non deve impiegare meccanismi di autenticazione con chiave condivisa (altrimenti detti pre-shared secret).

5.8.8 Non ripudio delle sessioni

Tutte le sessioni riconducibili all'applicazione, svolte dalle utenze operative/amministrative, devono essere, oltre che supportate da meccanismi di tracciamento idonei, anche cifrate con algoritmi crittografici. In questo modo si garantisce il non ripudio delle singole sessioni. Deve cioè essere possibile determinare con esattezza se un evento si è verificato o meno.

5.8.9 Schemi di sicurezza e crittografici

Gli schemi di sicurezza devono essere semplici e ben documentati. È vietata la predisposizione di schemi di autenticazione , crittografia e/o gestione delle chiavi non-standard, oppure fatta in proprio ("hand-made")..

5.8.10 Weak Keys e Collision

Il processo di creazione/assegnazione delle chiavi di cifratura ai dati dell'applicazione, in base al cipher utilizzato, non deve generare weak keys (chiavi deboli) o, nel caso di algoritmi di hashing, alcuna collision (valori ripetuti).

5.8.11 URL cifrati

Le directory contenenti file o dati di natura personale, critici e sensibili, residenti nella document root del web server devono apparire cifrate nell'URL del client browser.

5.8.12 Normalizzazione dei dati cifrati

Nelle applicazioni web l'utilizzo della codifica *base64* è autorizzato solo per la normalizzazione dei dati, delle stringhe o degli URL cifrati.