TRENTINO DIGITALE S.P.A.

Sviluppo sicuro: principali minacce e relative contromisure

PRINCIPALI MODIFICHE RISPETTO ALLA VERSIONE PRECEDENTE

Data	Versione	Modifiche apportate
2011	01.00	Prima emissione
19/09/2019	02.00	Aggiornamento template e contenuto del documento

INDICE

1 Introduzione

1.1 Premessa

Obiettivo della presente linea guida è quello di definire i requisiti di sicurezza che devono essere implementati all'interno di tutti gli applicativi sviluppati da, o per conto di, Trentino Digitale.

In particolare, vengono illustrati i rischi conseguenti ad alcune vulnerabilità a livello applicativo e i criteri di sicurezza, da integrare in fase di sviluppo del software, che consentono di risolverle.

1.2 Perimetro organizzativo

La presente linea guida si applica a tutto il personale dipendente di Trentino Digitale S.p.A. e a tutti i soggetti che collaborano con Trentino Digitale nelle attività di sviluppo e manutenzione applicativa.

1.3 Termini e definizioni

Bug - difetto nel software, derivante da un errore di programmazione, oppure introdotto deliberatamente per causare danni.

Campi nascosti – Campi non visibili all'utente utilizzati per memorizzare informazioni (informazioni che non sono immesse dall'utente).

Contromisura – Strumento di natura tecnologica, organizzativa o fisica atto a contrastare un attacco nei confronti di un sistema.

Disponibilità – Proprietà per la quale le informazioni sono rese accessibili ed utilizzabili su richiesta di un'entità autorizzata (ISO/IEC 13335-1:2004).

ESAPI – Progetto OWASP Enterprise Security API; è un modello per le API di sicurezza necessarie a produrre applicazioni web sicure. ESAPI fornisce implementazioni in Java, .NET, PHP, Classic ASP, Python e Cold Fusion.

Form - Interfaccia di un'applicazione che consente all'utente di inserire e inviare dei dati.

Integrità – Proprietà per la quale l'accuratezza e la completezza degli asset è salvaguardata (ISO/IEC 13335-1:2004).

Man in the Middle (MITM) - Attacco nel quale l'attaccante è in grado di leggere, inserire o modificare a piacere, messaggi tra due parti senza che nessuna delle due sia in grado di sapere se il collegamento che li unisce reciprocamente sia stato effettivamente compromesso da una terza parte.

Minaccia – Potenziale pericolo che può causare dei danni ai beni di un'organizzazione in funzione dell'esistenza di vulnerabilità.

Query – Interrogazione di un database per compiere determinate operazioni (es. inserimento, cancellazione..).

Riservatezza – Proprietà per la quale le informazioni non sono rese disponibili o divulgate a individui, entità o processi non autorizzati (ISO/IEC 13335-1:2004).

Sanitizzazione dei dati – Processo per rendere sicuri dei dati riservati.

SSL (Secure Sockets Layer) – Protocollo crittografico che permette il trasporto di protocolli informatici tradizionalmente insicuri in modalità sicura su reti TCP/IP.

Stored procedure – insieme di istruzioni SQL precompilate e archiviate in un database per ottimizzare l'esecuzione delle query.

VPN (Virtual Private Network) – Una Virtual Private Network o VPN è una rete privata virtuale instaurata tra soggetti che utilizzano un sistema di trasmissione pubblico e condiviso come Internet. Lo scopo delle reti VPN è di garantire un livello di sicurezza pari, o superiore, ad una connessione dedicata utilizzando reti pubbliche condivise.

1.4 Abbreviazioni

ACL – Access Control List.

DB – Data Base.

HTTP – HyperText Transport Protocol.

SQL – Structured Query Language.

URL – Uniform Resource Locator.

XSS - Cross Site Scripting.

1.5 Riferimenti

Norme di legge	*DLGs 196 del 30/06/2003 "Codice in materia di protezione dei dati personali"*
Standard di Riferimento	ISO/IEC 13335-1:2004 - *"Information technology -- Security techniques -- Management of information and communications technology security -- Part 1: Concepts and models for information and communications technology security management"*
A documenti del Sistema Sicurezza	SIC-POL-08 *"Sicurezza nella progettazione e sviluppo di soluzioni informatiche"*
A documenti del Sistema di Gestione della Qualità	-

2 Linea guida

2.1 Premessa

Le applicazioni sviluppate da, o per conto di, Trentino Digitale devono garantire un livello di sicurezza commisurato alla criticità dei servizi cui tali applicativi sono dedicati.

La fase di sviluppo di un applicativo è un momento fondamentale per assicurare un adeguato livello di sicurezza alle informazioni trattate: infatti errori in fase di sviluppo possono compromettere la riservatezza, l'integrità e la disponibilità dei dati.

I criteri per lo sviluppo sicuro del software illustrati all'interno del presente documento riguardano le misure di sicurezza da considerare nella fase di progettazione e sviluppo degli applicativi, come definito all'interno del documento SIC-POL-08 *"Sicurezza nella progettazione e sviluppo di soluzioni informatiche"*.

Nei paragrafi seguenti vengono quindi illustrate le principali vulnerabilità dei software, i possibili attacchi che possono sfruttare tali vulnerabilità e gli accorgimenti che, se implementati, consentono di neutralizzare tali minacce di sicurezza.

2.2 Principali rischi di sicurezza applicativa

È fondamentale che lo sviluppo di un applicativo sia effettuato considerando che è necessario prevenire, oltre i possibili bug che ne possono compromettere il corretto funzionamento, eventuali vulnerabilità del codice in grado di pregiudicare la sicurezza dei dati da esso acceduti, elaborati, trasmessi, archiviati.

Il presente paragrafo è dedicato ad illustrare alcuni tra i più comuni rischi che compromettono la sicurezza delle applicazioni, descrivendo per ciascuno di essi le vulnerabilità corrispondenti e gli attacchi che possono sfruttarle, i malfunzionamenti da essi causati agli applicativi esposti ad Internet, i danni possibili e le contromisure atte a contrastarlo.

[redacted]Iniezioni di codice

Le "Injection Flaws", come SQL Injection, OS Injection, e LDAP injection, si verificano quando dati non validati vengono inviati come parte di un comando o di una query al loro interprete. Il dato infetto può quindi ingannare tale interprete, eseguendo comandi non previsti o accedendo a dati per i quali non si ha l'autorizzazione.

Un agente di minaccia per questo tipo di rischio è chiunque possa mandare dati non controllati al sistema, includendo utenti interni, esterni ed amministratori.

L'attaccante invia un semplice testo che usa la sintassi dell'interprete obiettivo dell'attacco. Quasi ogni sorgente di dati può essere un vettore di injection, comprese le sorgenti interne.

Un Injection flaw può essere facilmente individuato esaminando il codice, mentre è più difficile tramite il testing.

Danni/conseguenze

Un Injection può portare ad una perdita o corruzione dei dati, mancanza di tracciabilità o rifiuto di accesso. In alcuni casi può permettere il controllo totale del sistema.

Per quanto concerne gli impatti sul business, si deve considerare il valore dei dati impattati e della piattaforma su cui gira l'interprete. Tutti i dati potrebbero essere rubati, modificati o cancellati.

Riconoscere le vulnerabilità

Il modo migliore per verificare se un'applicazione è vulnerabile ad un "injection" è di verificare che tutti gli interpreti di comandi dell'applicazione separino chiaramente i dati di input dai comandi e dalle query. Per le chiamate SQL, questo comporta l'uso di variabili bind in tutte le istruzioni e stored procedure, evitando query di tipo dinamico.

Il controllo del codice è un modo veloce ed accurato per verificare se un'applicazione utilizza i moduli in maniera sicura. Gli strumenti di analisi del codice possono aiutare gli analisti di sicurezza nel verificare l'uso dei moduli e tracciare il data flow nell'applicazione.

I penetration test possono verificare le carenze di controlli creando exploit che confermino la vulnerabilità.

Gli scanner automatici possono evidenziare l'esistenza di injection conosciuti ma non possono raggiungere sempre tutti i moduli e hanno difficoltà nel verificare se un attacco è andato a buon fine. Una gestione degli errori carente rende un injection flaw più facile da scoprire.

Contromisure di sicurezza

Prevenire un injection richiede che i dati di input siano tenuti separati da comandi e query.

L'opzione preferita è quella di usare API sicure che evitino l'accesso diretto all' interprete o che forniscano un'interfaccia parametrizzata (fare attenzione ad API, quali stored procedures, che sono parametrizzate, ma che possono ancora introdurre injection); se non sono disponibili API parametrizzate, è necessario evitare caratteri speciali usando soluzioni sintattiche (escape) specifiche per quell'interprete.

Infine, la validazione di input in "white list" con metodi di normalizzazione è consigliabile, ma non basta, poiché alcune applicazioni richiedono caratteri speciali nei loro input.

Cross-Site Scripting (XSS)

Le vulnerabilità di tipo XSS si verificano quando un'applicazione web riceve dei dati provenienti da fonti non affidabili e li invia ad un browser senza una opportuna validazione e/o "escaping". Il XSS permette agli attaccanti di eseguire degli script malevoli sui browser delle vittime; tali script possono dirottare la sessione dell'utente (prendendone il controllo), fare un deface del sito web o re-direzionare l'utente su un sito malevolo.

Si può considerare "agente di minaccia" chiunque possa inviare dati non controllati al sistema, inclusi utenti interni ed esterni nonché gli amministratori.

Sviluppo Sicuro: Principali Minacce E Relative Contromisure

Quasi tutte le sorgenti di dati possono essere veicoli di attacco, anche i database interni.

XSS è la vulnerabilità di sicurezza più diffusa nelle applicazioni WEB. Si manifesta quando un applicativo genera una pagina con dati forniti dall'utente senza prima averli validati o averne fatto l'escaping.

Individuare gli XSS è abbastanza facile tramite l'analisi del codice sorgente o tramite attività di penetration test applicativo.

Danni/conseguenze

L'attaccante può eseguire del codice dentro il browser della vittima per impadronirsi della sessione, effettuare il defacing del sito, inserire contenuti o stili, reindirizzare l'utente, ecc.

Per valutare gli impatti sul business si deve considerare il valore del sistema attaccato e di tutti i dati che elabora.

Si deve altresì considerare anche l'impatto sul business della divulgazione dell'esistenza della vulnerabilità.

Riconoscere le vulnerabilità

Ci si deve accertare che qualunque input utente re-inviato al browser dall'applicazione sia controllato (attraverso una validazione degli input) e che tali input dell'utente siano assoggettati ad "escaping" prima di essere inviati alle pagine in output. Il corretto trattamento di tali dati assicura che siano sempre recepiti dal browser come stringhe di testo e non come contenuti attivi che possono essere eseguiti.

Sia i tools statici che dinamici possono trovare automaticamente alcuni problemi di XSS. Siccome ogni applicativo costruisce differentemente le proprie pagine di output e si basa su differenti interpreti (JavaScript, ActiveX, Flash, Silverlight), il riconoscimento automatico è spesso difficile. Quindi una copertura completa del tema impone anche, oltre all'uso di tools automatici, l'ispezione del codice e penetration test manuali.

Tecnologie per Web 2.0, come AJAX, rendono inoltre l'XSS molto più difficile da riconoscere automaticamente.

Contromisure di sicurezza

Prevenire il XSS richiede di tenere i dati non controllati separati dal contenuto attivo del browser.

L'approccio preferito, per tutti i dati non affidabili, è quello di fare l'"escaping" appropriato al contesto HTML in cui verranno utilizzati (body, attribute, JavaScript, CSS, or URL). Lo sviluppatore dovrà includere questo escaping nell'applicativo a meno che il proprio framework di User Interface non lo faccia già.

Un'altra tecnica possibile è la validazione dell'input basata su approccio di tipo "whitelist", o positiva, unita ad una appropriata interpretazione delle stringhe. Comunque, in alcuni casi, non è una difesa completa, dato che alcuni applicativi richiedono caratteri speciali nelle stringhe input. Tali validazioni dovrebbero decodificare tutto l'input, validare la lunghezza, i caratteri, i formati e l'aderenza alle business rules prima di accettarli come input per l'applicazione.

[redacted]Compromissione delle procedure di autenticazione e di gestione della sessione

Le procedure applicative relative all'autenticazione e alla gestione della sessione sono spesso implementate in modo non corretto, permettendo agli attaccanti di compromettere password, chiavi, token di sessione o sfruttare debolezze implementative per assumere l'identità di altri utenti.

Gli sviluppatori spesso realizzano approcci personalizzati di gestione della sessione e dell'autenticazione, ma implementarli in maniera corretta è difficile. Il risultato è che spesso questi approcci personalizzati contengono difetti nelle aree quali il log-out, la gestione delle password, il time-out, il "ricordami su questo computer", la domanda segreta, l'aggiornamento dell'account, ecc. Scoprire tali difetti può essere difficile, poiché ciascuna realizzazione è unica.

Come "agenti di minaccia" si devono considerare attaccanti privi di credenziali o utenti con credenziali legittime che tentano di rubare gli account altrui per ottenere maggiori privilegi e/o per celare le proprie azioni. L'attaccante sfrutta le vulnerabilità presenti nel sistema di gestione della sessione o dell'autenticazione (es.: esposizione delle password o degli account, identificativi della sessione, ecc.) per impersonare l'utente.

[redacted]*Danni/conseguenze*

Questa categoria di vulnerabilità può consentire un accesso diretto verso uno o più account. In caso di successo l'attaccante ottiene gli stessi privilegi della vittima; per questa ragione obiettivi frequenti sono gli account dotati di privilegi elevati.

Per valutare gli impatti sul business si deve considerare il valore dei dati o delle funzioni applicative coinvolte. Considerare anche l'impatto sul business della divulgazione dell'esistenza della vulnerabilità.

[redacted]*Riconoscere le vulnerabilità*

Cruciale è la protezione di credenziali ed ID di sessione. Chiedersi se:

1. Le credenziali archiviate sono protette con tecniche di offuscamento o di crittografia? Vedere par. 2.2.7.
2. Le credenziali possono essere indovinate o sovrascritte (es.: creazione account, modifica o recupero delle password, identificativi della sessione deboli, ecc.)?
3. L'ID di sessione è in chiaro nelle URL (es.: URL rewriting)?
4. L'ID di sessione è vulnerabile ad attacchi di Session Fixation?
5. Gli ID hanno una scadenza? L'utente può fare log-out?
6. Dopo la fase di Login, gli ID di sessione vengono rinnovati?
7. Le password, ID di sessione e le altre credenziali sono trasmessi solo su connessioni crittografate utilizzando, ad esempio, TLS? Vedi par. 2.2.9.

[redacted]*Contromisure di sicurezza*

E' opportuno dotarsi di unico set di controlli per la gestione della Strong Authentication e delle sessioni e porre la massima cura nell'evitare difetti di tipo XSS che potrebbero consentire di sottrarre gli ID di sessione (vedi par. 2.2.2).

Sviluppo Sicuro: Principali Minacce E Relative Contromisure

Riferimento diretto ad un oggetto non sicuro

Un "riferimento diretto ad un oggetto" si verifica quando uno sviluppatore espone un riferimento all'implementazione interna di un oggetto, come un file, una directory, o una chiave in un database. Senza un opportuno controllo degli accessi o altre protezioni analoghe, gli attaccanti possono manipolare questi riferimenti in modo da accedere a dati non autorizzati.

Le applicazioni utilizzano spesso il nome effettivo o la chiave di un oggetto durante la generazione di pagine web. Le applicazioni non sempre verificano se l'utente è autorizzato all'accesso di un determinato oggetto. Questo può causare la vulnerabilità chiamata Insecure Direct Object Reference.

Come "agenti di minaccia" si devono considerare le differenti categorie di utenti del sistema e i loro privilegi di accesso agli oggetti. Il vettore di attacco più comune è costituito da un utente autorizzato che, modificando semplicemente il valore di un parametro (es. il percorso di una url), che fa riferimento diretto ad un oggetto, accede ad altri oggetti senza averne l'autorizzazione.

Gli addetti al testing possono facilmente evidenziare la presenza della vulnerabilità sia agendo sui valori dei parametri che eseguendo l'analisi del codice.

Danni/conseguenze

Tali difetti possono compromettere tutti i dati che sono referenziati da quel parametro.

A meno che il name space sia offuscato (es: nomi e chiavi arbitrari o randomici), è semplice per un attaccante indovinare i riferimenti e accedere a tutti i dati di quel tipo.

Per gli impatti sul business si deve considerare il valore dei dati esposti.

Considerare anche gli impatti sul business in relazione alla divulgazione della vulnerabilità del sistema.

Riconoscere le vulnerabilità

Il modo migliore per verificare se un'applicazione sia effettivamente vulnerabile all'Insecure Direct Object Reference è verificare che tutti i riferimenti agli oggetti siano opportunamente protetti. Considerare che:

- Per i riferimenti diretti ad una risorsa soggetta a restrizioni di accesso, l'applicazione deve verificare che l'utente sia autorizzato ad accedere alla risorsa richiesta.
- Se il riferimento è di tipo indiretto, il mapping al riferimento diretto deve essere limitato ai valori autorizzati per l'utente corrente.

La revisione del codice può rapidamente verificare se il metodo di protezione è stato correttamente implementato. Anche il security testing è efficace per individuare eventuali vulnerabilità nei riferimenti diretti ad oggetti. Gli strumenti automatici in genere non cercano tali difetti perché non possono conoscere a priori il livello di protezione ed autorizzazione adeguato di una risorsa.

Contromisure di sicurezza

Prevenire le vulnerabilità di Insecure Direct Object Reference richiede di gestire ciascuna risorsa accessibile dall'utente (es: object ID, nomefile) utilizzando i riferimenti ad oggetti indiretti, assegnati per utente o sessione, oppure garantendo un granulare controllo dell'accesso per garantire che l'utente sia autorizzato a tale accesso.

[redacted]Cross-Site Request Forgery (CSRF)

Un attacco di CSRF forza il browser della vittima ad inviare una richiesta http opportunamente "forgiata" – includendo i cookie di sessione della vittima ed ogni altra informazione di autenticazione – ad una applicazione web vulnerabile che la interpreta come legittimamente inviata dall'utente.

CSRF è favorito dalle applicazioni che permettono di prevedere/indovinare agevolmente tutti i dettagli di una particolare funzione applicativa. Dato che i browser inviano automaticamente le credenziali come cookie di sessione, l'attaccante può creare pagine web malevoli che generano richieste contraffatte, indistinguibili da quelle legittime.

Per individuare gli "agenti di minaccia" si deve considerare che chiunque può forzare gli utenti generando una richiesta su un sito web. L'attaccante crea richieste HTTP malevoli ingannando la vittima attraverso immagini, tag, XSS o numerose altre tecniche. Se l'utente è autenticato, l'attacco riesce.

L'individuazione di vulnerabilità di CSRF è abbastanza semplice attraverso penetration test o code analysis.

[redacted]*Danni/conseguenze*

L'attaccante può forzare le vittime a modificare le informazioni sull'applicazione bersaglio o ad utilizzare tutte le funzioni alle quali la vittima ha accesso.

Per individuare gli impatti sul business si deve considerare il valore delle informazioni e delle funzioni applicative, non essendo sicuri che siano gli utenti effettivi ad usarle.

E' opportuno anche considerare l'impatto sulla reputazione.

[redacted]*Riconoscere le vulnerabilità*

Il modo più facile per controllare se un'applicazione web è vulnerabile, è verificare se tutti i link e i form, focalizzando l'attenzione su link e form che invocano funzioni particolarmente importanti, contengono token casuali e non prevedibili per ogni utente. In caso contrario l'attaccante può generare richieste malevoli.

E' opportuno anche controllare le operazioni a più step, in quanto non sono naturalmente immuni. L'attaccante, utilizzando tag multipli o codice Javascript, può facilmente generare non una, ma una serie di richieste malevoli.

[redacted]*Contromisure di sicurezza*

Per prevenire il CSRF è necessario includere token non prevedibili nel corpo della pagina o negli URL di ogni richiesta HTTP. Come minimo, questi token dovrebbero essere unici per ogni sessione utente, anche se sarebbe meglio l'utilizzo di token unici per ogni richiesta. Preferibilmente inserendo il token in un campo nascosto. Ciò permette che il token venga inviato nel corpo della richiesta, evitando di includerlo nell'URL, che è più soggetto a intercettazioni.

[redacted]Configurazioni di sicurezza non corrette

Per garantire un adeguato livello di sicurezza è necessario configurare opportunamente le applicazioni, i framework, i server applicativi, i server web, i database e le piattaforme. Tutte le configurazioni devono

essere definite, implementate e manutenute, poiché raramente le configurazioni di default sono sicure. Questo implica mantenere tutto il software aggiornato, includendo in esso anche tutte le librerie usate dall'applicazione.

L'attaccante può usare credenziali di default, pagine non più usate, vulnerabilità non risolte, file e cartelle non protetti, ecc. per accedere al sistema o raccogliere informazioni riservate.

Gli errori di configurazione possono presentarsi a qualsiasi livello dello stack applicativo come: piattaforma, web server, application server, framework e codice personalizzato. Sviluppatori e amministratori di rete devono lavorare insieme per assicurare che l'intero stack sia correttamente configurato.

Come "agenti di minaccia" si devono considerare attaccanti esterni o utenti interni che tentino di compromettere il sistema o che intendono svolgere attività illecite utilizzando identità altrui o fittizie.

Gli scanner automatici sono utili per scovare patch mancanti, errori di configurazione, credenziali di default, servizi non necessari, ecc..

Danni/conseguenze

Alcune vulnerabilità permettono all'attaccante di avere accesso non autorizzato a sistemi, dati o funzionalità. A volte, alcune di queste compromettono l'intero sistema.
Per gli impatti sul business si deve considerare che il sistema può essere compromesso senza lasciare evidenze. I dati possono essere rubati o modificati nel tempo.
Gli eventuali costi di recupero possono essere elevati.

Riconoscere le vulnerabilità

È necessario un processo ben strutturato e ripetibile per implementare e mantenere le configurazioni in sicurezza in quanto occorre verificare se tutto lo stack applicativo è stato configurato correttamente. Nello specifico, ad esempio, verificare se:

- è presente un processo per la gestione degli aggiornamenti software (OS, Web/Application Server, Database, Applicazioni e Librerie);
- tutti gli elementi non strettamente necessari sono stati disabilitati, rimossi o non installati (es. porte, servizi, pagine, account, privilegi);
- le credenziali di default sono state cambiate/disabilitate;
- la gestione degli errori è stata configurata per prevenire la visualizzazione di messaggi che potrebbero rilasciare informazioni utili;
- le impostazioni di sicurezza dei framework (es. Struts, Spring, ASP.NET) sono opportunamente configurate.

Contromisure di sicurezza

Le principali raccomandazioni consistono:

- nell'implementare un processo ripetibile per la messa in sicurezza dei vari ambienti coinvolti nello sviluppo del software. Gli ambienti di sviluppo, QA e produzione devono essere tutti configurati allo stesso modo;

- nell'istituire un processo per la gestione degli aggiornamenti software per ogni ambiente coinvolto. In questo processo vanno inclusi anche gli aggiornamenti di tutte le librerie di codice, che sono frequentemente tralasciate;
- nel prevedere una solida architettura applicativa che permette di tenere separati i vari componenti;
- nell'effettuare scansioni e audit periodici per identificare eventuali errori di configurazione o aggiornamenti mancanti.

[redacted]Archiviazione cifrata non sicura

Molte applicazioni web non proteggono opportunamente i dati sensibili. Degli attaccanti potrebbero quindi rubare o modificare tali dati (debolmente protetti).

La vulnerabilità più comune consiste nel non cifrare i dati che dovrebbero esserlo. Se invece i dati sono cifrati, è una pratica diffusa usare chiavi poco robuste o algoritmi deboli. Se sono presenti hash, è frequente l'utilizzo di algoritmi deboli. Come "agenti di minaccia" si devono considerare gli utenti del proprio sistema, che potrebbero accedere a dati protetti senza opportuna autorizzazione, compresi gli amministratori di sistema.

Tipicamente gli attacchi vengono effettuati individuando le copie in chiaro dei dati o le chiavi crittografiche (in caso di crittografia debole).

Gli attaccanti esterni hanno generalmente difficoltà ad individuare queste vulnerabilità. Avendo un accesso limitato devono prima sfruttare altre vulnerabilità per accedere ai dati.

[redacted]*Danni/conseguenze*

Sfruttando questo attacco è possibile compromettere i dati che dovrebbero essere protetti come dati sanitari, credenziali d'accesso, ecc.

Per l'impatto sul business si deve considerare il valore economico e l'impatto sulla reputazione in caso di furto o perdita dei dati.

[redacted]*Riconoscere le vulnerabilità*

E' fondamentale identificare i dati che necessitano della cifratura per assicurarsi che:

- siano cifrati in qualsiasi punto vengano salvati, in particolare nelle copie di backup;
- solo gli utenti autorizzati possano accedere a copie dei dati in chiaro (per esempio attraverso il controllo degli accessi (vedi par. 2.2.4 - 2.2.8);
- sia utilizzato un algoritmo standard di cifratura sufficientemente robusto;
- siano utilizzate chiavi robuste, debitamente protette da accessi non autorizzati e cambiate periodicamente.

[redacted]*Contromisure di sicurezza*

Per tutti i dati che devono essere protetti è opportuno, come minimo:

Sviluppo Sicuro: Principali Minacce E Relative Contromisure

- considerare le minacce da cui è necessario proteggerli (per esempio utenti interni, attaccanti esterni) e assicurarsi di adottare sistemi di protezione adeguati a seconda delle minacce;
- assicurarsi che i backup siano cifrati e che le chiavi siano gestite e salvate separatamente;
- assicurarsi che siano usati algoritmi standard considerati robusti e che le chiavi siano forti e gestite correttamente;
- assicurarsi che le password siano protette da algoritmi di hash robusti;
- assicurarsi che tutte le chiavi e le password siano protette dagli accessi non autorizzati.

[redacted]Protezione insufficiente per accessi alle url

Le applicazioni non sempre proteggono le pagine ad accesso controllato in modo appropriato. Le applicazioni dovrebbero eseguire dei controlli di accesso ogni qualvolta le pagine vengono accedute, altrimenti gli attaccanti potrebbero alterare le URL in modo da ottenere l'accesso ad aree per cui non sono autorizzati (es. conoscendo il link diretto ad una pagina riservata è sufficiente incollarlo nel browser).

Come "agente di minaccia" si deve considerare chiunque abbia un accesso alla rete e può inviare richieste all'applicazione.

Malgrado la vulnerabilità sia facilmente identificabile non è altrettanto facile individuare gli oggetti non adeguatamente protetti.

[redacted]*Danni/conseguenze*

Questa vulnerabilità permette ad un attaccante di accedere a funzionalità particolari senza opportuna autorizzazione. Principali bersagli sono le pagine delle interfacce di amministrazione.

Per l'impatto sul business si deve considerare il valore economico delle funzioni esposte e dei dati che esse gestiscono.

Inoltre, bisogna considerare l'impatto sulla reputazione se la vulnerabilità viene resa pubblica.

[redacted]*Riconoscere le vulnerabilità*

Il metodo migliore per verificare se un'applicazione non limita correttamente l'accesso ad una URL è quello di verificare ogni pagina, stabilendo se deve essere considerata pubblica o privata. Se la pagina è stata considerata privata è necessario che sia presente un processo di autenticazione e di autorizzazione che gestisca l'accesso alla stessa. E' mandatorio che il processo venga applicato a tutte le pagine, anche a quelle pubbliche.

Tramite attività di Penetration Testing è possibile controllare se la protezione è stata propriamente applicata.

[redacted]*Contromisure di sicurezza*

A prescindere dall'utilizzo di un sistema esterno o interno all'applicazione per la gestione di autorizzazione e permessi, è necessario che:

- le policy di autenticazione ed autorizzazione siano basate sui ruoli per minimizzare l'effort nell'implementazione;

- le policy siano flessibili per minimizzare l'utilizzo di codice applicativo per bloccare l'accesso alle pagine;
- le regole neghino l'accesso per default e che per l'accesso ad ogni singola pagina siano necessari permessi specifici ad utenti specifici;
- l'accesso alle pagine che fanno parte di un flusso applicativo composto da più passi avvenga solo tramite quelli previsti.

[redacted]Protezione insufficiente del Transport Layer

Le applicazioni spesso falliscono nell'autenticare, cifrare, e proteggere la confidenzialità e l'integrità del traffico di rete sensibile. Questo accade a causa dell'uso di algoritmi poco robusti o a causa dell'uso di certificati scaduti, invalidi o che non vengono utilizzati correttamente.

Monitorare il traffico di rete degli utenti non sempre risulta un'operazione complessa. La difficoltà primaria sta nel monitorare il traffico nel momento in cui l'utente sta utilizzando il sito vulnerabile.

Spesso le applicazioni non proteggono correttamente il traffico di rete, utilizzando SSL/TLS solo durante la fase di autenticazione.

Come "agente di minaccia" si deve considerare chiunque possa monitorare il traffico di rete degli utenti dell'applicazione, sia front che back-end. Se l'applicazione è esposta su Internet, si deve considerare tale chiunque conosca la modalità di accesso degli utenti.

Per individuare le vulnerabilità più semplici è sufficiente monitorare il traffico. Vulnerabilità più complesse sono invece individuabili esaminando l'architettura dell'applicazione e la configurazione del server.

[redacted]*Danni/conseguenze*

La vulnerabilità espone i dati in transito, comprese le credenziali. La loro compromissione può coinvolgere un intero sito oppure favorire attacchi di phishing e MITM.

Il danno reale è dato dal valore dei dati esposti sui canali di trasmissione in base alla confidenzialità ed integrità richiesta.

[redacted]*Riconoscere le vulnerabilità*

È possibile stabilire se un'applicazione utilizza una protezione insufficiente del canale di trasporto verificando se:

- è utilizzato SSL per tutto il traffico legato al processo di autenticazione e per tutte le risorse all'interno delle pagine e token di sessione, evitando così l'utilizzo misto di SSL e risorse in chiaro;
- vengono utilizzati algoritmi di cifratura considerati robusti;
- tutti i cookie hanno impostato il flag "secure";
- viene utilizzato un certificato server valido, configurato correttamente, rilasciato da un ente autorizzato, non revocato e che corrisponda ai nomi di dominio utilizzati.

Sviluppo Sicuro: Principali Minacce E Relative Contromisure

Contromisure di sicurezza

Considerando che una corretta protezione del canale di trasporto può influire sull'architettura, la soluzione più semplice è utilizzare SSL per l'intera applicazione se non si presentano problemi di performance. In caso contrario limitare la sua applicazione alle pagine riservate o critiche anche se questa scelta può esporre gli ID di sessione e altri dati sensibili.

Redirect e forward non validati

Le applicazioni Web spesso eseguono dei redirect o dei forward degli utenti verso altre pagine o siti, e usano dei dati non validati per determinare le pagine di destinazione. In questo scenario, gli attaccanti possono fare reindirizzare le vittime a siti di phishing o di malware, o utilizzare il forward per accedere a pagine non autorizzate. A volte la pagina bersaglio è specificata in un parametro non validato, che permette di scegliere arbitrariamente la pagina di destinazione.

L'attaccante forza la vittima a cliccare su un link con un redirect non validato. Le vittime non rilevano un potenziale pericolo poiché il link appartiene a un sito valido.

Come "agente di minaccia" si deve considerare chiunque può forzare gli utenti a creare richieste verso il sito.

Rilevare redirect non controllati è semplice. I forward non controllati sono più difficili da trovare perché l'obiettivo sono le pagine interne.

Danni/conseguenze

Tali redirect possono installare dei malware o chiedere agli utenti di inserire credenziali o altre informazioni sensibili. I forward non sicuri possono permettere di evadere i controlli d'accesso.

Per valutare gli impatti sul business si deve considerare l'impatto reputazionale nei confronti degli utenti che scoprono di aver subito questa vulnerabilità.

Riconoscere le vulnerabilità

Il modo migliore per capire se un'applicazione ha redirect o forward non validati consiste nel:

- revisionare il codice di tutti i redirect o forward (chiamato transfer in .NET) per rilevare se nell'URL di destinazione siano inclusi valori parametrici e nel caso accertarsi che i parametri possano contenere solo il valore della destinazione autorizzata;
- indicizzare il sito per verificare se genera dei redirect (codice di risposta HTTP 300-307, in genere 302);
- controllare i parametri immessi prima del redirect per verificare se vengono utilizzati come URL destinazione o parte di esso. In tal caso modificare il parametro ed osservare se il sito effettua il redirect;
- se il codice non è disponibile, controllare tutti i parametri in cerca di URL di destinazione utilizzati in redirect o forward.

Contromisure di sicurezza

Un'implementazione sicura di redirect e forward può essere ottenuta in diversi modi:

- evitare l'uso di redirect e forward se non strettamente necessario;
- se necessario, valutare la possibilità di non adoperare i parametri utente per la destinazione;
- se i parametri di destinazione non possono essere evitati, assicurarsi che il valore inserito sia valido ed autorizzato per quell'utente effettuando una mappatura dei parametri di destinazione e usare del codice lato server per convertire la mappatura in URL di destinazione.