6 LINEE GUIDA PER L'INDIVIDUAZIONE E LA RIVISITAZIONE DEI REQUISITI DI SICUREZZA E DI PRIVACY APPLICATIVI

6.1 Linee guida per la modellazione delle minacce

6.1.1 Identificazione degli obiettivi di sicurezza

La sicurezza dei dati è assicurata quando vengono garantite tre caratteristiche di fruizione di questi ultimi, che sono la riservatezza, l'integrità e la disponibilità.
Per raggiungere la sicurezza vengono eseguite azioni, che in caso di:

- riservatezza, proteggono i dati al fine di contrastare la divulgazione non autorizzata;
- integrità, contrastano le modifiche non autorizzate dei dati;
- disponibilità, contrastano la indisponibilità malevola dei dati/servizi.

Gli obiettivi specifici di sicurezza sono un sottoinsieme degli obiettivi di progetto e dovrebbero essere utilizzati per guidare gli sforzi impiegati nella modellazione delle minacce.
Identificare i principali obiettivi di sicurezza permette di concentrarsi con maggiore attenzione sulle aree da proteggere. Ad esempio, se si identificano i dettagli del profilo cliente come dati riservati, che devono essere protetti, è possibile esaminare la modalità di archiviazione sicura di tali dati e il modo in cui l'accesso a tali dati viene controllato e verificato.
Per determinare gli obiettivi di protezione, occorre porsi le seguenti domande:

- Quali dati occorre proteggere?
- Esistono requisiti di conformità? I requisiti di conformità possono includere criteri di protezione, leggi sulla privacy, regolamenti e standard.
- Esistono requisiti di qualità specifici del servizio? I requisiti di qualità del servizio includono tipicamente la disponibilità e i requisiti prestazionali.
- Esistono beni immateriali che devono essere protetti? Tali beni includono ad esempio, la reputazione dell'organizzazione, le informazioni commerciali sensibili e la proprietà intellettuale.

Di seguito sono riportati alcuni esempi di obiettivi di sicurezza comuni:

- Impedire agli aggressori di ottenere dati sensibili, inclusi i codici di accesso e le informazioni sul profilo.
- Soddisfare gli accordi a livello di servizio per la disponibilità delle applicazioni.
- Proteggere la credibilità dell'organizzazione.

6.1.2 Creazione di un disegno ad alto livello dell'applicazione

La modellazione delle minacce è un processo iterativo di analisi, dove ad ogni ciclo si scende sempre più in dettaglio, identificando di livello in livello le funzionalità chiave dell'applicazione, le sue caratteristiche ed i dati da proteggere.
Per avere una panoramica dell'applicazione occorre:

- Disegnare lo scenario di sviluppo dall'inizio alla fine;
- Identificare i ruoli;
- Identificare gli scenari d'uso più significativi;
- Identificare le tecnologie;
- Identificare i meccanismi di sicurezza.

Di seguito sono riportati i dettagli di ciascuna fase.
Disegnare lo scenario di sviluppo dall'inizio alla fine - La prima attività consiste in una modellazione ad alto livello dell'applicazione (composizione e struttura dell'applicazione, relativi sottosistemi e caratteristiche di distribuzione). Dopo il primo disegno, si aggiungono i dettagli sui meccanismi di autenticazione, autorizzazione e comunicazione. Da notare che quando si inizia la

modellazione non sempre si è a conoscenza di tutti i dettagli per cui deve essere sempre possibile ritornare sull'aspetto sotto analisi in un secondo momento per poterlo approfondire ulteriormente. La figura che segue riporta l'esempio di un diagramma iniziale che mostra l'architettura di una applicazione con alcuni dettagli.

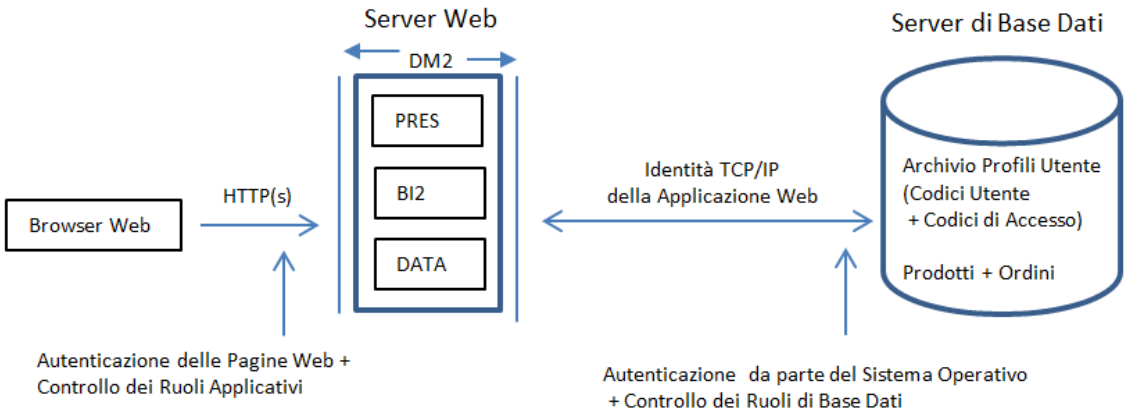

Figura 8 - Esempio di disegno architetturale di una applicazione

In generale il disegno architetturale deve evidenziare i seguenti elementi:

- Topologia fisica e logica dei componenti: il collocamento dei server in rete (Intranet, Extranet e accesso a Internet). Iniziare con le topologie di rete logiche, per poi visualizzare le relative topologie fisiche quando si dispone di tali dettagli. È possibile aggiungere o rimuovere le minacce a seconda della scelta di topologie fisiche.
- Livelli logici: il livello di presentazione (front-end), il livello di business (back-end) e i livelli di accesso ai dati. Successivamente occorre rifinire per includere, una volta noti, i limiti fisici del server;
- Componenti chiave: i componenti importanti all'interno di ogni livello logico. In questa fase è possibile includere i limiti reali del componente e di processo una volta conosciuti;
- Servizi chiave: i servizi importanti. Una volta noti, da mostrare come processi;
- Porte e protocolli di comunicazione: i server, i componenti e i servizi che comunicano tra di loro e come lo fanno. Da mostrare le specifiche dei pacchetti dati in entrata e in uscita, una volta individuati;
- Identità: le identità utente principali usate all'interno dell'applicazione e gli eventuali profili di servizio rilevanti;
- Dipendenze esterne: le dipendenze dell'applicazione da eventuali sistemi esterni. Elencare queste dipendenze è utile per individuare possibili vulnerabilità che potrebbero insorgere in un secondo momento se alcune assunzioni fatte inizialmente sul generico sistema esterno dovessero risultare non più vere o in qualche modo cambiate.

È importante revisionare il disegno del sistema nel corso del tempo per verificare se tutti gli elementi individuati siano ancora come descritti, se devono essere cambiati o se hanno bisogno di un ulteriore livello di dettaglio.

6.1.2.1 Identificazione dei Ruoli

È importante identificare i ruoli all'interno dell'applicazione, ovvero, chi può fare cosa.
La fase di identificazione dei ruoli è utilizzata sia per determinare ciò che dovrebbe accadere (accesso alle risorse autorizzate come stabilito per lo specifico ruolo) che per determinare ciò che non dovrebbe accadere (accesso a risorse per le quali non si ha l'autorizzazione).
L'assegnazione dei ruoli deve essere centralizzata e a ciascun ruolo deve essere associato un profilo 'autorizzativo' che regolamenta i comandi, le transazioni e gli accessi ai dati.

Per ridurre la superficie d'attacco è necessario stabilire quali devono essere i privilegi minimi per ciascun componente dell'applicazione. Nella gestione della sicurezza delle risorse (quali Database, Active Directory, file, etc.), i ruoli applicativi devono essere quindi stabiliti a partire dalla logica di business con l'obiettivo di delimitare il perimetro di azione nell'accesso alla risorsa.

6.1.2.2 Identificare gli Scenari d'Uso Chiave

In questa fase occorre identificare le principali funzionalità e modalità d'uso e dettagliare gli aspetti legati alle attività di creazione, lettura, aggiornamento e cancellazione dei dati. Le caratteristiche chiave vengono spesso descritte nel contesto dei casi d'uso e consentendo la comprensione di come l'applicazione è destinata ad essere utilizzata e come può essere utilizzata in modo improprio. I casi d'uso permettono di identificare i flussi di dati e di focalizzarsi sull'analisi di eventuali minacce nelle fasi successive di dettaglio della modellizzazione.

6.1.2.3 Identificare le Tecnologie

Occorre identificare tutte le tecnologie utilizzate e le loro caratteristiche: Sistemi operativi; Server Web; Server di Base Dati; le tecnologie utilizzate per implementare la presentazione dei dati a livello utente, per gestire le regole di business, per l'accesso ai dati sottostanti e il linguaggio di sviluppo utilizzato.
L'identificazione delle tecnologie consente di concentrarsi sulle minacce che possono nascere in un momento successivo alla modellizzazione, legate alle specifiche tecnologie in uso. Inoltre, aiuta a determinare le eventuali tecniche di mitigazione del rischio.

6.1.2.4 Identificare Meccanismi di Sicurezza Applicativa

Un'altra fase importante è l'identificazione dei meccanismi di sicurezza applicativa, in particolare occorre analizzare i seguenti aspetti:

- Validazione input e dati;
- Autenticazione;
- Autorizzazione;
- Gestione della configurazione;
- Dati sensibili;
- Gestione della sessione;
- Crittografia;
- Manipolazione dei parametri;
- Gestione delle eccezioni;
- Audit e gestione dei log.

Lo scopo dell'identificazione di questi elementi è quello di aggiungere il più possibile ulteriori dettagli, anche poco conosciuti. Ad esempio è utile documentare come l'applicazione viene autenticata dalla base dati o come gli utenti vengono autorizzati, oppure quali sono le aree preposte all'autenticazione e all'autorizzazione.

6.1.3 Scomposizione dell'applicazione

La scomposizione dell'applicazione è utile per scoprire le minacce e le vulnerabilità del sistema. Nell'ottica di sicurezza, i componenti più importanti sono:

- confini di fiducia (trust boundaries);
- flussi di dati;
- punti di ingresso (entry points);
- punti di uscita (exit points).

6.1.3.1 Confini di fiducia (Trust boundaries)

Identificare i confini di fiducia dell'applicazione aiuta a concentrare l'analisi sulle aree di maggiore interesse. I confini di fiducia evidenziano dove cambiano i livelli di fiducia. In quest'ambito, la fiducia è intesa in chiave di riservatezza e integrità. Ad esempio, una modifica nei livelli di controllo di accesso all'applicazione, dove è necessario un livello o un privilegio specifico per accedere a una risorsa o un'operazione, comporterebbe una modifica del livello di fiducia. Un altro esempio, potrebbe essere in un punto di entrata nell'applicazione ove è necessario filtrare i dati di accesso.
Per identificare i confini di fiducia occorre:

- Iniziare individuando i confini del sistema esterno. Ad esempio, l'applicazione può scrivere un file sul server X, può effettuare chiamate al database sul server Y e può chiamare il servizio Web Z. Ciò definisce il limite di sistema.
- Identificare i punti di controllo di accesso o i luoghi chiave in cui l'accesso richiede privilegi aggiuntivi o l'appartenenza ad un dato ruolo. Ad esempio, l'accesso ad una pagina particolare, potrebbe essere limitata ai soli dirigenti, nel qual caso richiederebbe un accesso autenticato e inoltre che l'utente ricopra un certo ruolo.
- Identificare i confini di fiducia da una prospettiva di flusso di dati. Per ogni sottosistema, considerare se il flusso di dati a monte o l'input dell'utente sia affidabile e se non lo è, considerare in che modo il flusso di dati e l'input possono essere autenticati e autorizzati. Conoscere quali punti di ingresso esistono tra i confini di fiducia, consente di concentrare l'identificazione delle minacce in tali punti considerati chiave.

Alcuni esempi di confini di fiducia sono: un firewall perimetrale, il confine tra il web server e il server di base dati, punti di ingresso di componenti di business che espongono dati privilegiati, dunque, protetti da ulteriori controlli di accesso, il limite tra l'applicazione e i servizi di terze parti.

6.1.3.2 Flussi di Dati

È importante tracciare il flusso dei dati all'interno dell'applicazione dal punto di ingresso al punto d'uscita. Questa attività è necessaria per comprendere come interagisce l'applicazione con i sistemi esterni, i sistemi client e come interagiscono i componenti interni. È importante anche, prestare particolare attenzione al flusso di dati che attraversa i confini di fiducia e come tali dati vengono convalidati nel punto di entrata. Inoltre occorre fare molta attenzione ai dati sensibili e come questi attraversano il sistema, se passano attraverso una rete e/o se vengono persistiti. Un buon approccio è quella di analizzare il flusso dei dati tra i singoli sottosistemi a partire dal livello più alto e poi via via a scendere ai livelli più bassi.

6.1.3.3 Punti d'Ingresso (Entry Points)

I punti di ingresso dell'applicazione servono anche come punti di ingresso per gli attacchi. Il front-end di una applicazione web che è in ascolto di richieste http è un esempio di punto di ingresso vulnerabile agli attacchi. Questo punto di ingresso è destinato ad essere esposto agli utilizzatori. Altri punti di ingresso, come i punti di accesso interni esposti dai sotto componenti negli strati dell'applicazione, possono esistere solo per supportare la comunicazione interna con altri componenti. Tuttavia, occorre conoscere dove sono localizzati e quali tipi di input ricevono nel caso in cui un aggressore riesca ad aggirare l'interfaccia dell'applicazione e attaccare direttamente un punto di ingresso interno.
A titolo esplicativo si elencano di seguito ulteriori esempi di Entry Points:

- Front-end applicativo (form di Login, form di Ricerca);
- Funzioni applicative (funzione di login, web service esposti, ..);
- Console di amministrazione del database;

- External reporting;
- Porte TCP/UDP (network socket).

6.1.3.4 Punti di Uscita (Exit Points)

È importante identificare da dove l'applicazione invia i dati all'utente o ai sistemi esterni, dando priorità ai punti di uscita in cui l'applicazione scrive dati che includono l'input proveniente dall'utente o dati provenienti da fonti non attendibili, ad esempio basi dati condivise.
A titolo esplicativo si elencano di seguito ulteriori esempi di Exit Points:

- File di Log;
- Database;
- Interfacce esterne (es. web service).

6.1.4 Identificazione delle minacce

In questa fase, è possibile individuare minacce e attacchi che potrebbero compromettere l'applicazione e gli obiettivi di sicurezza. Il processo di identificazione consiste in sessioni di brainstorming tra i team di sviluppo e test. Idealmente, il team di lavoro è costituito da architetti software, professionisti della sicurezza, sviluppatori, tester e amministratori di sistema.
Esistono due approcci per affrontare questa fase:

- Iniziare, elencando minacce e attacchi comuni. Con questo approccio, si inizia con un elenco di minacce comuni raggruppate per categorie di vulnerabilità. Successivamente, occorre adattare tale elenco alla propria architettura. Ad esempio, utilizzando gli scenari identificati per esaminare i flussi di dati, prestando particolare attenzione ai punti di ingresso e in particolare a quelli che attraversano i confini di fiducia. In questo modo si potranno eliminare immediatamente alcune minacce, in quanto non applicabili ai casi d'uso.
- Utilizzare un approccio basato su domande. Un approccio basato su questionari può aiutare a identificare le minacce e i possibili attacchi. La categorizzazione STRIDE si basa su categorie di minacce molto estese, quali spoofing (assunzione impropria di identità), manomissione di dati, ripudio, divulgazione indesiderata di informazioni e interruzione di servizio. Il modello STRIDE deve essere usato per porre domande relative a qualsiasi aspetto dell'architettura e del design dell'applicazione. Questo è un approccio basato su obiettivi, in cui si prendono in considerazione tutti gli obiettivi di un possibile aggressore (punto di vista di un attaccante).

Per identificare le minacce, si esaminano tutti i livelli dell'applicazione, livello per livello e funzione per funzione. Ponendo l'attenzione sulle categorie di vulnerabilità, ci si concentra sulle aree in cui vengono spesso effettuati errori di sicurezza. Occorre identificare le potenziali minacce e le possibili azioni che un aggressore potrebbe tentare di eseguire per sfruttare le vulnerabilità a cui l'applicazione è esposta. Durante questa attività di identificazione delle minacce si eseguono le seguenti attività:

- Identificazione delle minacce e degli attacchi comuni.
- Identificazione delle minacce annidate nei casi d'uso.
- Identificazione delle minacce annidate nei flussi di dati.

6.1.4.1 Identificazione delle minacce e attacchi comuni

Esistono una serie di minacce e attacchi comuni che si basano su vulnerabilità di carattere comune. Questa sezione identifica una serie di domande chiave da porsi per ciascuna categoria.
Autenticazione:

- Come potrebbe un aggressore rubare una identità?
- Come potrebbe un utente malintenzionato accedere all'archivio delle credenziali?
- Come potrebbe un aggressore portare un attacco? Come vengono memorizzate le credenziali dell'utente e quali criteri di codici di accesso vengono applicati?

- Come può un utente malintenzionato modificare, intercettare o eludere il meccanismo di ripristino delle credenziali dell'utente?

Autorizzazione:

- Come potrebbe un utente malintenzionato influenzare i controlli di autorizzazione per accedere a operazioni privilegiate?
- Come potrebbe un utente malintenzionato elevare i propri privilegi?

Input e dati di convalida:

- Come potrebbe un utente malintenzionato iniettare comandi SQL?
- Come potrebbe un utente malintenzionato eseguire un attacco di cross-site scripting?
- Come potrebbe un utente malintenzionato eludere la validazione degli input?
- Come potrebbe un utente malintenzionato inviare un input non valido per influenzare la logica di protezione adottata sul server?
- Come potrebbe un utente malintenzionato sollevare un errore di input per bloccare l'applicazione?

Gestione della configurazione:

- Come potrebbe un utente malintenzionato accedere alle funzioni di amministratore della configurazione?
- Come potrebbe un utente malintenzionato accedere ai dati di configurazione dell'applicazione?

Dati sensibili:

- Dove e come l'applicazione memorizza i dati sensibili?
- Quando e in quale punto i dati sensibili vengono passati attraverso la rete?
- Come potrebbe un utente malintenzionato visualizzare i dati sensibili?
- Come potrebbe un utente malintenzionato manipolare i dati sensibili?

Gestione delle sessioni:

- Si utilizza un algoritmo di crittografia personalizzato e ci si fida di tale algoritmo?
- Come potrebbe un aggressore prendere il controllo della sessione di un utente?
- Come potrebbe un utente malintenzionato visualizzare o modificare lo stato della sessione di un altro utente?

Crittografia:

- Di cosa ha bisogno un aggressore per sovvertire il meccanismo di crittografia adottato?
- Come potrebbe un utente malintenzionato ottenere l'accesso alle chiavi crittografiche?
- Quali standard crittografici si stanno utilizzando? Quali sono gli attacchi noti su tali standard?
- Si vuole adottare un proprio meccanismo di crittografia?
- In che modo la tipologia di distribuzione potenzialmente influenzerà la scelta dei metodi crittografici?

Manipolazione dei parametri:

- In che modo un aggressore potrebbe manipolare i parametri per influenzare la logica di protezione implementata sul server?
- Come potrebbe un utente malintenzionato manipolare i dati sensibili presenti nei parametri?

Gestione delle eccezioni:

- Come potrebbe un utente malintenzionato bloccare l'applicazione?
- Come potrebbe un utente malintenzionato ottenere dettagli utili ai propri fini?

Revisione e registrazione (audit):

- Come potrebbe un aggressore coprire le sue tracce?
- Come si può dimostrare che un utente malintenzionato (o un utente legittimo) ha eseguito azioni specifiche?

6.1.4.2 Identificazione delle potenziali minacce annidate nei casi d'uso

Occorre esaminare i casi d'uso chiave, che sono stati individuati nella fasi precedenti, per comprendere il modo in cui un utente potrebbe influenzare premeditatamente o involontariamente l'applicazione ad eseguire un'operazione non autorizzata o a divulgare dati riservati o privati. In questa fase ci si pongono domande immedesimandosi nella figura dell'aggressore. Alcuni esempi di domande da porsi:

- Come può un utente iniettare un input dannoso in un caso d'uso specifico?
- I dati vengono pubblicati in base all'input fornito dall'utente o in base all'input non validato fornito dall'utente?
- In che modo un aggressore potrebbe manipolare i dati della sessione?
- Come potrebbe un utente malintenzionato ottenere dati sensibili quando questi vengono trasmessi attraverso la rete?
- In che modo un aggressore potrebbe eludere i controlli di autorizzazione?

6.1.4.3 Identificazione delle potenziali minacce annidate nei flussi di dati

Occorre rivedere i casi d'uso e gli scenari chiave e analizzare i flussi di dati, in particolare, i flussi di dati tra i singoli componenti dell'architettura. Il flusso di dati che attraversa un confine di fiducia richiede particolare attenzione. Nella stesura del codice, si deve assumere che, tutti i dati esterni al confine di fiducia dell'applicativo siano dannosi. Nel codice si dovrebbe eseguire una validazione dei dati adeguatamente robusta. Per identificare le minacce associate ai flussi di dati, ci si deve porre le seguenti domande:

- Quale è il percorso del flusso di dati dal front-end al back-end dell'applicazione?
- Quali componenti chiamano altri componenti?
- Quale aspetto hanno i dati validi?
- Dove viene eseguita la validazione dei dati?
- Quali sono i vincoli imposti ai dati?
- Come vengono validati i dati in base ai parametri previsti in termini di lunghezza, intervallo di valori, formato e tipo?
- Quali dati sensibili vengono trasmessi tra i componenti dell'applicazione e attraverso le reti, e come vengono protetti durante il transito?

L'uso della documentazione quali ad esempio, i diagrammi DFD e i diagrammi di sequenza UML, aiuta nell'analisi dell'applicazione e nell'identificazione dei flussi di dati.

6.1.4.4 Esplorare ulteriori minacce utilizzando gli alberi delle minacce/attacchi

Vedi Paragrafo 5.5.4.2.

6.1.5 Identificazione delle vulnerabilità

Così come è stato fatto nel processo di identificazione delle minacce, si fornisce di seguito una rassegna delle diverse categorie di vulnerabilità. In questa fase, l'obiettivo è quello di analizzare le

vulnerabilità (le minacce sono state già trattate nel paragrafo precedente) partendo dal principio che per poter analizzare correttamente un'applicazione è necessario valutare le sue vulnerabilità ad ogni livello.
Di seguito alcuni esempi di domande da porsi a secondo della fase.
Autenticazione:

- I nomi utente e le password sono inviati in chiaro su un canale non protetto? Si utilizza una crittografia ad hoc per le informazioni sensibili?
- Le credenziali vengono memorizzate? Se vengono memorizzate, come vengono memorizzate e protette?
- Viene imposto l'uso di un meccanismo di strong authentication? Quali altre politiche sull'uso dei codici di accesso vengono applicate?
- Come vengono verificate le credenziali?
- Come viene identificato l'utente autenticato dopo l'accesso iniziale?

Occorre rivedere l'autenticazione cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Le credenziali di autenticazione o cookie di autenticazione vengano passate su collegamenti di rete non crittografati, che possono portare al furto delle credenziali o della sessione;
- Si fa uso di codici d'accesso e procedure di accesso deboli, che possono portare ad un accesso non autorizzato;
- Si fa uso di dati personali come forma di autenticazione.

Autorizzazione:

- Quali controlli di accesso vengono utilizzati nei punti di ingresso dell'applicazione?
- L'applicazione prevede l'uso di ruoli? Se utilizza ruoli, questi sono sufficientemente granulari ai fini del controllo degli accessi?
- L'inserimento erroneo di un codice di accesso fallisce in maniera sicura?
- È possibile limitare l'accesso alle risorse di sistema?
- Vengono attuate politiche restrittive nell'accesso alla base dati?
- Quale processo autorizzativo viene adottato a livello di base dati?

Occorre esaminare l'autorizzazione cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Si fa uso di ruoli e profili utente sovra privilegiati;
- Granularità di ruoli insufficiente;
- Non viene attuata alcuna restrizione di accesso alle risorse di sistema o viene attuata limitatamente ad alcuni profili.

Inserimento e Validazione Dati:

- Occorre identificare le vulnerabilità che riguardano la convalida dei dati di ingresso e dei dati in generale, domandandosi quanto segue:
    - I dati di ingresso vengono tutti validati?
    - I dati di ingresso, vengono validati in termini di lunghezza, intervallo, formato e tipo?
    - Si fa affidamento sulla sola validazione lato client?
    - Un aggressore potrebbe iniettare comandi o dati dannosi nell'applicazione?
    - I dati presenti all'interno delle pagine Web vengono scritti senza ricorrere a particolari accorgimenti o si preferisce codificarli in HTML per impedire possibili attacchi di cross-site scripting?
    - L'input, viene validato prima di essere utilizzato nella costruzione delle istruzioni SQL al fine di impedirne l'iniezione di malware?
    - I dati vengono validati al punto di ingresso del componente destinatario in virtù del fatto che oltrepassano sia il confine di fiducia del mittente che del destinatario?

  - Ci si può fidare dei dati presenti nel database?
  - Si accettano come input, nomi di file, URL e nomi utente? In tal caso, sono state considerate le problematiche di canonicalizzazione?
- Occorre rivedere la convalida degli input cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:
  - Ci si affida soltanto alla validazione lato client;
  - Viene eseguito un controllo a posteriori dei dati già immessi anziché applicare un filtro al momento dell'inserimento;
  - Si fa uso di dati non validati nella composizione di pagine Web;
  - Si fa uso di input non validato per generare query SQL;
  - All'interno del codice sorgente, si fa uso di tecniche di accesso ai dati non sicure, che possono aumentare il rischio di minaccia da iniezione SQL;
  - Le decisioni intraprese nelle procedure di sicurezza si basano su nomi di file, URL o nomi utente forniti in input.

Gestione della Configurazione:

- Come vengono protette le interfacce di amministrazione remota?
- Gli archivi della configurazione vengono protetti?
- I dati di configurazione sensibili vengono crittografati?
- I privilegi di amministratore vengono separati rispetto a quelli assegnati alle altre tipologie di utenti?
- Vengono usati profili utente di processo e servizio con privilegi limitati?

Revisionare la gestione della configurazione cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- I dati confidenziali di configurazione, quali le stringhe di connessione e le credenziali del profilo utente di servizio, vengono memorizzati in chiaro;
- Non esiste protezione negli aspetti di gestione della configurazione dell'applicazione, incluse le interfacce di amministrazione;
- Si fa uso di profili utente di processo e profili utente di servizio con privilegi non strettamente necessari.

Dati Sensibili:

- I dati confidenziali vengono persistiti?
- I dati sensibili, come vengono storicizzati?
- I dati confidenziali presenti in memoria, vengono storicizzati?
- Si trasferiscono dati sensibili attraverso la rete?
- I dati sensibili vengono registrati nei log dell'applicazione?

Revisionare le modalità di trattamento dei dati sensibili cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- I dati confidenziali vengono memorizzati quando ciò non è necessario;
- I dati confidenziali vengono codificati direttamente nel codice dell'applicazione;
- I dati confidenziali vengono memorizzati in chiaro;
- I dati sensibili vengono passati in chiaro sulla rete.

Gestione della Sessione:

- Come vengono generati i cookie di sessione?
- Come vengono scambiati gli identificativi di sessione?
- Come viene protetto lo stato della sessione mentre attraversa la rete?
- Come viene protetto lo stato della sessione per impedirne il furto?
- Come viene protetto in generale lo stato della sessione?
- La durata della sessione viene limitata?

- In che modo l'applicazione esegue l'autenticazione utilizzando l'archivio delle sessioni?
- Le credenziali sono trasmesse attraverso la rete e vengono mantenute all'interno dell'applicazione? Se sì, come vengono protette?

Controllare gli aspetti di gestione della sessione con il fine di individuare una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Gli identificativi di sessione vengono passati su canali non crittografati;
- La durata della sessione è prolungata;
- Gli stati di sessione non vengono protetti;
- Gli identificativi di sessione vengono passati nelle stringhe di query.

Crittografia:

- Quali algoritmi e tecniche di crittografia vengono usate?
- Vengono impiegati algoritmi di crittografia personalizzati?
- Perché vengono utilizzati determinati algoritmi piuttosto che altri?
- Quale è la durata di validità prevista per le chiavi crittografiche e come queste vengono protette?
- Quante volte vengono riciclate le chiavi crittografiche?
- Come e con quali utenti vengono distribuite le chiavi crittografiche?

Occorre esaminare l'utilizzo della crittografia cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Vengono impiegati algoritmi crittografici personalizzati;
- Si fa uso di un algoritmo errato o di una chiave di dimensione troppo piccola;
- Le chiavi crittografiche non vengono protette;
- Si fa uso della stessa chiave per un periodo di tempo prolungato.

Manipolazione dei Parametri:

- Tutti i valori dei parametri di ingresso vengono validati?
- Tutti i valori dei parametri inseriti nei campi delle pagine che comprendono la GUI, nei dati dei cookie e nelle intestazioni http, subiscono un processo di validazione?
- I dati sensibili vengono trasferiti attraverso parametri?
- L'applicazione è in grado di rilevare la manomissione dei parametri?

Esaminare gli aspetti di gestione dei parametri cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Non è possibile validare tutti i parametri di ingresso. Ciò rende l'applicazione vulnerabile ad attacchi di tipo "interruzione di servizio" e ad attacchi indirizzati alla modifica del codice sottostante, quali SQL injection e XSS.
- I dati sensibili vengono inclusi nei cookie non crittografati. I dati del cookie possono essere modificati lato client o possono essere acquisiti e modificati in quanto vengono passati attraverso la rete.
- I dati sensibili vengono inclusi nelle stringhe di query e nei campi delle pagine di front-end. Le stringhe di query e i campi all'interno di pagine di front-end sono facilmente modificabili lato client.
- Si fa affidamento sulle informazioni presenti nell'intestazione HTTP. Queste informazioni sono facilmente modificabili lato client.

Gestione delle Eccezioni:

- In che modo l'applicazione gestisce le condizioni di errore?
- Viene mai permesso alle eccezioni di propagarsi fino al client?
- Che tipo di informazioni mostrano i messaggi di errore?
- Vengono rivelate troppe informazioni al client?
- Dove vengono registrati i dettagli delle eccezioni? I file di log vengono protetti?

Occorre esaminare i meccanismi di gestione degli errori cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Non è possibile convalidare tutti i parametri di ingresso;
- Vengono rivelate troppe informazioni al client.

Audit and Gestione dei Log:

- Sono state individuate delle attività chiave per l'audit?
- L'attività di audit copre tutti i livelli e i server dell'applicazione?
- Come vengono protetti i file di log?

Occorre esaminare i meccanismi di logging cercando una corrispondenza tra le seguenti vulnerabilità di carattere comune:

- Mancata revisione e registrazione (audit) dei tentativi d'accesso falliti;
- Mancata protezione dei file di log;
- Mancata revisione e registrazione (audit) nei vari livelli del server dell'applicazione.

Indicazioni aggiuntive. Dopo aver completato l'attività di modellazione delle minacce, si procede come segue:

- Se si vuole descrivere il modello delle minacce in un documento, mantenere il documento di facile lettura in modo da potere essere consultato frequentemente. Tale documentazione dovrebbe includere gli obiettivi di sicurezza, gli scenari chiave, le risorse protette, un elenco di minacce e un elenco di vulnerabilità.
- Analizzare le vulnerabilità per contribuire a predisporre la progettazione e l'implementazione della sicurezza.
- Analizzare le vulnerabilità per pianificare e implementare il test di sicurezza del sistema.
- Tracciare e aggiornare l'elenco delle vulnerabilità riscontrate utilizzando un sistema di tracciamento.
- Se sono state identificate delle minacce a cui è stata attribuita una priorità molto alta, ma per le quali non sono state individuate le corrispettive vulnerabilità, è necessario decidere se indagare ulteriormente o meno con il rischio di essere esposti a possibili attacchi, oppure di continuare l'analisi alla ricerca di una possibile vulnerabilità.
- Comunicare le informazioni acquisite ai membri del team di lavoro.

6.2 Identificazione del Processo di Sviluppo del Software Sicuro

L'applicazione di un processo di gestione del rischio[58] nello sviluppo di un sistema abilita le organizzazioni a bilanciare i requisiti per la protezione delle informazioni e degli asset proprietari con il solo costo di implementare le strategie di controllo della sicurezza e di mitigazione attraverso l'SDLC.
Il processo di gestione del rischio, identifica le attività e gli asset critici, nonché le vulnerabilità sistemiche a cui è esposta l'organizzazione. I rischi sono spesso condivisi in tutta l'organizzazione e non sono specifici per sole determinate architetture di sistema.
In questo contesto la metodologia ed il tool sviluppato da AGID a tale scopo (https://www.sicurezzait.gov.it) sono strategici per la gestione strutturata del rischio. Per ulteriori dettagli si rinvia all' Allegato 1- Linee Guida per l'adozione di un ciclo di sviluppo di software sicuro.
Alcuni dei vantaggi apportati con l'integrazione degli aspetti di sicurezza nel ciclo di vita di sviluppo del software, sono:

[58] Fare riferimento alla metodologia e al tool sviluppato da AGID a tale scopo (**Cyber Risk Management** - **https://www.sicurezzait.gov.it/Home**). Per ulteriori dettagli si rinvia all' *Allegato 1- Linee Guida per l'adozione di un ciclo di sviluppo di software sicuro.*

- L’individuazione preventiva e la mitigazione delle vulnerabilità e dei problemi di sicurezza presenti nella configurazione dei sistemi, con conseguente riduzione dei costi per l'implementazione dei controlli di sicurezza e delle tecniche di mitigazione delle vulnerabilità;
- La consapevolezza delle potenziali sfide ingegneristiche dovute ai controlli di sicurezza obbligatori;
- L’identificazione dei servizi di sicurezza condivisi e riutilizzo delle strategie e degli strumenti di sicurezza che riducono i costi di sviluppo e migliorano la condizione di sicurezza complessiva del sistema, attraverso l'applicazione di metodi e tecniche collaudate;
- La facilitazione nell'attuazione delle decisioni prese da parte dei dirigenti, attraverso l'applicazione tempestiva di un processo completo di gestione del rischio;
- La documentazione di importanti decisioni di sicurezza prese durante il processo di sviluppo, per informare la direzione sulle considerazioni di sicurezza intraprese durante tutte le fasi dello sviluppo;
- Il miglioramento dell'organizzazione e della fiducia degli utenti nel promuovere l'adozione e l'uso dei propri sistemi;
- Una migliore interoperabilità e integrazione dei sistemi che sarebbe difficile raggiungere se la sicurezza fosse considerata separatamente ai vari livelli.

Uno studio della Forrester Consulting sullo stato della sicurezza applicativa ha riportato che le organizzazioni che implementano un processo MS-SDL hanno mostrato risultati di ROI migliori rispetto agli altri approcci metodologici. Anche, la Aberdeen Group ha dimostrato come l'adozione di un processo MS-SDL aumenti la sicurezza e riduca la gravità e il costo degli incidenti dovuti alla presenza di vulnerabilità nel software, generando al contempo un ritorno sugli investimenti (quattro volte maggiore) rispetto ad altri approcci di sicurezza adottati nello sviluppo di software.
MS-SDL è supportato da una rilevante quantità di risorse, tra cui documentazione, tutorial e strumenti software. Tale ricchezza di informazioni e strumenti rende sicuramente MS-SDL un'opzione interessante per le organizzazioni che intendono adottare nuove iniziative di sicurezza del software.
La modellazione delle minacce è un approccio per analizzare la sicurezza di un'applicazione. Si tratta di un approccio strutturato che consente di identificare, quantificare e affrontare i rischi di sicurezza associati ad una applicazione. La modellazione delle minacce non è un approccio orientato alla revisione del codice, ma integra il processo di revisione del codice da un punto di vista della sicurezza. L'inclusione della modellazione delle minacce nell'SDLC può contribuire a garantire che le applicazioni vengano sviluppate con la sicurezza integrata fin dall'inizio (Secure by Design/ Secure by Default).
Questo, in combinazione con la documentazione prodotta nell'ambito del processo di modellazione delle minacce, può fornire al revisore una maggiore comprensione del sistema. Consente inoltre al revisore di vedere dove si trovano i punti di accesso all'applicazione e quali sono le potenziali minacce associabili a ciascun punto di accesso.
Il concetto di modellazione delle minacce non è nuovo, ma negli ultimi anni si è verificato un chiaro cambiamento di mentalità. La modellazione delle minacce, oggi guarda ad un sistema dal punto di vista di un potenziale attaccante, piuttosto che dal punto di vista della difesa. Microsoft è stata forte sostenitrice di tale processo negli ultimi anni, facendo della modellazione delle minacce una componente fondamentale del proprio SDLC, che sostiene essere una delle principali ragioni di maggiore sicurezza riscontrabile nei suoi prodotti.
Quando l'analisi del codice sorgente, ad esempio di applicazioni esistenti, viene eseguita al di fuori dell'SDLC (in quanto già realizzate), i risultati della modellazione delle minacce, aiutano a ridurre la complessità dell'analisi del codice sorgente, promuovendo un approccio maggiormente circoscritto. Invece di rivedere tutto il codice sorgente con uguale attenzione, è possibile assegnare una priorità alla revisione di sicurezza del codice, basandosi sul risultato ottenuto dal processo di modellazione che individua le minacce a più alto rischio facendo sì che la revisione del codice possa essere indirizzata in modo più puntuale.

Questi i vantaggi introdotti dal processo di modellazione delle minacce:

- la conferma dell'idoneità degli elementi di sicurezza individuati da attuare;
- l'individuazione di eventuali lacune nelle caratteristiche di sicurezza da attuare;
- l'identificazione di eventuali ulteriori elementi di sicurezza;
- l'identificazione dei requisiti di policy e di processo;
- l'identificazione dei requisiti da inserire nelle operazioni di sicurezza;
- l'identificazione dei requisiti in materia di tracciamento e monitoraggio;
- arrivare ai casi di abuso, se utilizzati, secondo la metodologia Agile;
- la comprensione dei requisiti di business continuity;
- la comprensione dei requisiti in materia di capacità e disponibilità.

L' esecuzione del processo di modellazione delle minacce, in fase di progettazione, aiuta nella:

- identificazione delle vulnerabilità che devono essere risolte a livello di progettazione e di implementazione;
- identificazione dei beni informativi che necessitano di controlli di sicurezza;
- mappatura dei controlli di sicurezza, identificati in controlli tecnico/amministrativi/fisici a seconda dei casi (questa attività può essere svolta anche a livello di architettura, ma farlo a livello di progettazione aiuta ad essere più precisi);
- identificazione dei casi di "test di sicurezza"/"scenari di test di sicurezza" nella verifica dei requisiti di sicurezza implementati.

La modellazione delle minacce è il processo di valutazione e documentazione dei rischi associati alla sicurezza di un particolare sistema e/o applicazione software. Mediante l'adozione di opportune tecniche già discusse in precedenza nel documento, è possibile identificare strategie di mitigazione efficaci per contrastare potenziali minacce a cui potrebbe essere soggetta l'applicazione. La modellazione delle minacce consente inoltre, di giustificare l'introduzione o l'eliminazione di eventuali feature all'interno dell'applicazione oltre che governare, al fine di proteggere gli asset dell'applicazione, l'introduzione di nuove policy o pratiche di sicurezza all'interno del sistema. La categorizzazione delle minacce di sicurezza può essere ottenuta mediante l'adozione di un modello denominato STRIDE che è l'acronimo che riunisce la gamma dei rischi a cui può essere soggetta l'applicazione e per i quali deve essere protetta.
Nell'ambito della fase progettuale dell'SDL, nel processo di definizione dei requisiti di sicurezza, un modello come STRIDE può essere di aiuto nel definire i pattern di attacco, tra cui estrarre il modello di attacco (ovvero il sottoinsieme dei possibili attacchi) per il sistema applicativo oggetto di analisi. Lo STRIDE ha l'indubbio vantaggio di non essere eccessivamente astratto ma è piuttosto facilmente riconducibile a situazioni reali. In gran parte della letteratura il modello STRIDE viene definito come un sistema per modellare le minacce. Ma in realtà (in accordo con Gary McGraw) si ritiene più opportuno pensare che STRIDE sia un modello relativo ai possibili attacchi, legando la "minaccia" agli attori (umani e non) che sono invece gli artefici degli attacchi stessi. Le sei categorie di rischi a cui la STRIDE afferisce (Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, and Elevation of privilege) consentono di identificare le vulnerabilità ed i possibili vettori di attacco nelle applicazioni software.

Le linee guida per la modellazione delle minacce ispirata a MS-SDL si suddivide nelle seguenti fasi:

- Identificazione degli asset. Cosa il sistema dovrebbe proteggere?
- Creazione di una panoramica dell'architettura. Concentrandosi sui confini di fiducia, ovvero sui flussi di dati scambiati tra componenti appartenenti ad un'entità e componenti appartenenti ad un'altra entità.
- Scomposizione del sistema in sotto-componenti fino al livello più basso possibile.

- Identificazione delle minacce. Utilizzando la STRIDE o un albero delle minacce per facilitare l'enumerazione delle stesse.
    - STRIDE è un acronimo di spoofing, tampering, repudiation, information disclosure, denial of service e elevation of privilege. Queste descrizioni di vulnerabilità non sono intese come categorie che si escludono a vicenda, ma piuttosto come una tecnica euristica per l'enumerazione. Esaminare ciascun componente, ponendo particolare attenzione sui confini di fiducia, valutando se questo presenta vulnerabilità precedentemente indicate.
    - Gli Attack tree, possono eventualmente sostituire la STRIDE.
- Documentazione delle minacce.
- Valutazione delle minacce secondo il modello DREAD, un acronimo che indica il danno potenziale, la riproducibilità, l'utilizzabilità, gli utenti interessati, l'esposizione. Le minacce che si collocano ai primi posti in ciascuna di queste categorie dovrebbero avere una priorità più elevata nella risoluzione.

6.3 Modellazione e Individuazione delle minacce con STRIDE

Un evidente vantaggio dell'approccio STRIDE è l'indipendenza dal codice. Ciò è vantaggioso in quanto aiuta a identificare i problemi di sicurezza nella fase di analisi e progettazione del sistema software. Inoltre consente a tutto il team (oltre agli sviluppatori) di partecipare alla definizione del modello delle minacce del sistema stesso.
Il processo può portare benefici laddove non si dispone di un esperto di sicurezza all'interno del proprio team. L'impiego della STRIDE consente a questi team di individuare le vulnerabilità e le tecniche di mitigazione da attuare per difendersi dalle eventuali minacce identificate.
Le organizzazioni, possono inoltre beneficiare del processo sia per i nuovi progetti che per i progetti in essere, in cui potrebbero esistere delle vulnerabilità non identificate. L'implementazione di una metodologia che identifica e classifica le minacce è un processo ripetibile strutturato che può portare benefici a qualsiasi tipo di progetto.
Il risultato finale è, un elenco di vulnerabilità che i team di sviluppo e gli stakeholder dei prodotti possono quindi valutare, al fine di effettuare una accurata valutazione dei rischi, che insistono sulle vulnerabilità individuate.
La STRIDE è stata identificata come metodologia leader di Threat Modeling nell'industria del software.
Il successo di questa metodologia è dovuto ad un approccio ben strutturato alla modellazione delle minacce, ed è un eccellente supporto ed una risorsa per il team.
A volte la STRIDE viene indicata come "categorie STRIDE" o " tassonomia STRIDE". Tuttavia si evidenzia che la STRIDE non nasce come strumento di categorizzazione, ma con l'obiettivo di aiutare a trovare i possibili attacchi alla sicurezza.

6.4 Valutazione del rischio derivante dalle minacce individuate con DREAD

La metodologia DREAD è stata sviluppata da Microsoft nell'ambito della definizione del Security Development Lifecycle e del Threat Modeling. L'adozione della metodologia DREAD prodiga i seguenti benefici:

1. è utile per focalizzarsi sui reali rischi di una minaccia specifica;
2. obbliga a considerare fattori aziendali come la criticità del sistema e l'impatto sul business;
3. le cinque categorie sono tra loro scarsamente correlate (una di esse non implica le altre): considerare fattori indipendenti è un'ottima garanzia per formulare una corretta valutazione del rischio.

4. è largamente impiegata (ad es. OWASP[59] e OpenStack[60]) .

6.5 Modellazione e Individuazione delle minacce di privacy con LINDUN

La privacy è un aspetto molto importante, soprattutto nella società odierna, dove i dati personali sono onnipresenti e purtroppo questa, viene spesso trascurata durante lo sviluppo del software. Nonostante emergono diverse metodologie orientate alla produzione di requisiti di tutela della privacy (vedi paragrafo 5.8.6), queste non soddisfano appieno quelle che sono le aspettative, o comunque, non sono in grado di fornire una guida metodologica sostanziale da adottare nel corso dell'analisi o mancano del necessario supporto alla tutela della privacy.
Per colmare questa lacuna, si propone la metodologia LINDDUN, descritta nei capitoli precedenti. LINDDUN si ispira infatti a STRIDE, ovvero, un approccio consolidato per la modellazione e l'identificazione delle minacce alla sicurezza. Inoltre, la metodologia LINDDUN è stata costruita sulla base delle classificazioni esistenti in materia di tutela della privacy.
Anche se LINDDUN non è una tecnica di conformità, attua diversi principi imposti dalla legislazione sulla protezione dei dati (ad esempio consenso, minimizzazione, sensibilizzazione, ecc.) e richiama esplicitamente l'attenzione sulla necessità di conformità normativa. Inoltre, le proprietà sulla privacy, riportate nel paragrafo 5.8.1.1, costituiscono la base delle categorie di minacce gestite da LINDDUN. Infine, LINDDUN aderisce anche ai principi della Privacy by Design in quanto mira a introdurre la privacy nelle prime fasi del ciclo di vita di sviluppo del software.

[59] https://www.owasp.org/index.php/Threat_Risk_Modeling

[60] https://wiki.openstack.org/wiki/Security/OSSA-Metrics#DREAD