- 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?