semplice, o scomponendolo in componenti molto semplici e totalmente separati. Una volta limitate le vie d'accesso, è necessario iniziare a pensare a "Quanti attacchi diversi sono possibili contro ciascuna via d'accesso? Anche in questo caso, è opportuno limitare il tutto rendendo le vie di per sé piuttosto semplici. Come una porta con una chiave unica, invece di una porta che può essere aperta con cinque chiavi diverse. Una volta fatto ciò, è necessario contenere i possibili danni che un attacco potrebbe arrecare se portato con successo. Ritornando all'esempio dell'edificio, dovremmo fare in modo che una singola porta consenta l'accesso ad una sola stanza. Il fatto di aumentare la complessità del codice determina con buona probabilità una crescita del livello di entropia e di conseguenza un proporzionale incremento dell'esposizione del software a possibili vulnerabilità. La complessità rende la sicurezza facile da ignorare e quindi aumenta la probabilità che questa possa fallire. All'atto pratico:

- Riutilizzare quei componenti di cui è nota l'affidabilità.
- Evitare architetture complesse e codifiche che presentano l'uso di doppi negativi, es: un costrutto condizionale del tipo "if(!item.isNotFound())" dovrebbe essere ricondotto nella sua forma semplice "if(item.isFound())".

13) Risolvere nel modo corretto le problematiche di sicurezza.
Una volta che un problema di sicurezza viene identificato, è importante attuare un test e comprendere la causa che origina il problema al livello di massima 'profondità'. Quando si utilizzano schemi predefiniti di progettazione, è probabile che eventuali problematiche di sicurezza presenti su un modulo vengano diffuse in tutte le baseline di codice, per cui è essenziale sviluppare la giusta risoluzione senza introdurre regressioni. Per esempio, un utente ha scoperto di poter vedere le informazioni di un altro utente manipolando il proprio cookie. La correzione sembra essere relativamente semplice, ma poiché il codice di gestione dei cookie è condiviso tra tutte le applicazioni, una modifica ad una sola applicazione può essere estesa a tutte le altre. La correzione, ed il contesto su cui opera (una correzione può avere effetti diversi su implementazioni differenti) deve quindi essere verificata su tutte le applicazioni interessate.

5.2.2 Pratiche di secure design

5.2.2.1 Best practice di secure design per le applicazioni web

Il seguente elenco definisce, a titolo esemplificativo e non esaustivo, un insieme di buone pratiche che, se adottate in fase di design, consentono di risolvere a priori vulnerabilità applicative note considerate maggiormente critiche.

- Gestione degli errori e attività di logging
  - Mostrare all'utente finale messaggi di errore di carattere generico.
    - Descrizione: I messaggi di errore non devono mai rivelare dettagli sullo stato interno dell'applicazione. Ad esempio, percorsi del file system e informazioni sullo stack non devono essere mai esposte all'utente attraverso messaggi di errore. (Vedi CWE 209).
  - Gestire tutte le eccezioni nel codice sorgente.
    - Descrizione: Non consentire mai che si verifichi un'eccezione non gestita attraverso i linguaggi e i framework impiegati nello sviluppo di applicazioni web. La gestione degli errori deve essere implementata in modo tale da comprendere anche gli errori imprevisti restituendo l'output all'utente sempre in modo controllato. (Vedi CWE 391).
  - Nascondere, bloccare o gestire gli errori generati da eventuali framework o software di terze parti in uso.
    - Descrizione: L'uso di un eventuale framework o piattaforma di sviluppo può generare messaggi di errore predefiniti. Questi dovrebbero essere bloccati o

sostituiti con messaggi di errore personalizzati, in quanto tali messaggi potrebbero rivelare informazioni sensibili all'utente. (Vedi CWE 209).

- Registrare su log tutte le attività di autenticazione e validazione.
  - Descrizione: Registrare su log tutte le attività di autenticazione e di gestione delle sessioni eseguite lato software insieme a tutti gli errori di convalida dell'input. Tutti gli eventi relativi alla sicurezza devono essere registrati. Questi possono essere utilizzati successivamente per rilevare attacchi passati o in corso. (Vedi CWE 778).
- Registrare su log tutte le attività di cambiamento dei privilegi.
  - Descrizione: Registrare su log tutte le attività o le occorrenze in cui il livello di privilegi di un utente subisce un cambiamento. (Vedi CWE 778).
- Registrare su log tutte le attività amministrative.
  - Descrizione: Registrare su log tutte le attività eseguite a livello amministrativo sull'applicazione o su uno qualsiasi dei suoi componenti. (Vedi CWE 778).
- Registrare su log qualsiasi accesso a informazioni sensibili.
  - Descrizione: Registrare su log qualsiasi accesso ai dati sensibili. Ciò è particolarmente importante per quelle organizzazioni che devono soddisfare i requisiti normativi come HIPAA, PCI o SOX. (Vedi CWE 778).
- Non registrare su log dati inappropriati o non necessari.
  - Descrizione: Sebbene la registrazione su log degli errori e l'accesso per l'auditing siano importanti, i dati sensibili non dovrebbero mai essere registrati in forma non cifrata. Ad esempio, sotto HIPAA e PCI, costituirebbe una violazione del log in termini di dati sensibili in esso presenti, a meno che il log non sia criptato sul disco. Inoltre, potrebbe dar luogo ad un pericoloso punto di esposizione nel caso in cui l'applicazione stessa venga compromessa. (Vedi CWE 532).
- Persistere i dati di log in modo sicuro.
  - Descrizione: I log devono essere conservati e mantenuti in modo appropriato per prevenire eventuali perdite di informazioni o la possibile compromissione da parte di intrusi. La conservazione dei log deve inoltre rispettare le politiche di conservazione stabilite dall'organizzazione al fine di soddisfare i requisiti normativi e fornire le sufficienti informazioni necessarie per le attività forensi e di risposta agli incidenti. (Vedi CWE 533).

- Protezione dei dati
  - Prevedere ovunque l'HTTPS per applicazioni web.
    - Descrizione: L'HTTPS dovrebbe essere impiegato preferibilmente in tutti i punti di accesso all'applicazione web. Se è necessario limitarne l'uso, quantomeno lo si deve prevedere per le pagine di autenticazione e tutte le pagine a valle dell'autenticazione. Se è previsto l'invio di informazioni sensibili (ad esempio, informazioni personali) prima dell'autenticazione, anche queste devono essere necessariamente trasmesse tramite HTTPS. Utilizzare sempre se disponibile, la versione HTTPS prevista dal server indicata attraverso l'URL. Consentire il re indirizzamento da HTTP a HTTPS aumenta la possibilità per un possibile attaccante di perseguire un attacco man-in-the-middle senza destare il minimo sospetto dell'utente. (Vedi CWE 311, 319, 523).
  - Disattivare l'accesso tramite HTTP a tutte le risorse protette.
    - Descrizione: Per tutte le pagine che richiedono l'accesso protetto tramite HTTPS, lo stesso URL non deve essere accessibile tramite canale HTTP. (Vedi CWE 319).
  - Impiegare una configurazione forte del TLS.
    - Descrizione: Disattivare su tutti i server eventuali schemi di cifratura considerati deboli. Ad esempio, i protocolli SSL v2, SSL v3 e TLS precedenti alla versione 1.2 presentano debolezze note e non sono considerabili sicuri. Disattivare inoltre, le suite di cifratura NULL, RC4, DES e MD5. Assicurarsi che la lunghezza di tutte le chiavi

sia superiore a 128 bit, utilizzare una rinegoziazione sicura e disabilitare la compressione se presente.

- Utilizzare l'intestazione "Strict-Transport-Security".
  - Descrizione: L'intestazione Strict-Transport-Security garantisce che il browser non parli con il server tramite HTTP. Ciò contribuisce a ridurre il rischio di attacchi di downgrade dell'HTTP.
- Memorizzare le password utente utilizzando un algoritmo forte di salted hashing iterativo.
  - Descrizione: Le password utente devono essere memorizzate utilizzando tecniche di hashing sicuro con algoritmi forti come PBKDF2, bcrypt o SHA-512. Il semplice hashing della password eseguito una sola volta non protegge a sufficienza la password stessa. Per rendere l'hash forte, utilizzare un algoritmo di hashing basato su funzione adattiva, combinato con un salt generato randomicamente per ciascun utente. (Vedi CWE 257).
- Scambiare in modo sicuro le chiavi crittografiche.
  - Descrizione: Se le chiavi di crittografia vengono scambiate o preimpostate nell'applicazione, allora qualsiasi impostazione o scambio di chiavi deve essere effettuato su un canale sicuro.
- Definire e configurare un processo sicuro di gestione delle chiavi crittografiche.
  - Descrizione: Quando le chiavi crittografiche sono memorizzate nel sistema, queste devono essere opportunamente protette e accessibili solo al personale autorizzato rispettando il criterio del "need-to-know". (Vedi CWE 320).
- Utilizzare certificati HTTPS validi rilasciati da una autorità certificante con buona reputazione.
  - Descrizione: I certificati HTTPS devono essere firmati da un'autorità di certificazione di buona reputazione. Il nome sul certificato deve corrispondere all'FQDN del sito web. Il certificato stesso deve essere valido e non scaduto.
- Disabilitare il data caching attraverso l'uso delle intestazione di controllo della cache e disabilitare anche l'auto completamento.
  - Descrizione: Disabilitare la cache dei dati del browser usando le intestazioni HTTP di controllo della cache o i meta tag presenti all'interno della pagina HTML. Inoltre, i campi di input che contengono dati sensibili, come quelli presenti nel form di login, dovrebbero avere nel relativo form HTML, l'attributo "autocomplete "impostato su "off" al fine di forzare il browser a non mettere in cache le credenziali. (Vedi CWE 524).
- Cifrare i dati sensibili persistiti.
  - Descrizione: Crittografare i dati sensibili o critici prima che questi vengano archiviati. (Vedi CWE 311, 312).
- Limitare l'utilizzo e la memorizzazione dei dati sensibili.
  - Descrizione: Condurre una appropriata valutazione al fine di accertarsi che i dati sensibili non vengano inutilmente trasportati o conservati. Ove possibile, utilizzare la tokenizzazione per ridurre i rischi di esposizione dei dati.

- Configurazione e operatività
  - Automatizzare il processo di distribuzione.
    - Descrizione: Con l'avvento dei processi di Continuous Integration e di Continuous Delivery Il ciclo di vita del software si integra sempre più con l'ambiente di produzione. Grazie a ciò, l'automazione del processo di distribuzione del software, garantisce che le modifiche vengano apportate in modo coerente e ripetibile in tutti gli ambienti.
  - Definire e attuare un processo rigoroso di "Change Management".
    - Descrizione: Nell'ambito dell'operatività, è opportuno mantenere un processo rigoroso di gestione dei cambiamenti. Ad esempio, le nuove release di software