**ANALISI STATICA**

**Lifetime dei dati sensibili**

il lifetime degli oggetti contenenti informazioni sensibili nella memoria heap è stato ridotto al minimo essenziale affinché compiessero il loro ruolo.

[LoginDao.java] [RegistrationDao.java]

**Immagine che contiene testo, schermata, Carattere, schermo

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente**

…

**Immagine che contiene testo, Carattere, schermata, calligrafia

Descrizione generata automaticamente**

…

****

**Lettura file**

l metodo “BufferedReader.readLine()” fornisce una stringa, una possibile soluzione è utilizzare ByteBuffer.

[Token.java]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Cookie e Sessione http**

* I cookie vengono creati da un server Web e vengono memorizzati per un periodo di tempo sul client
* Quando il client si riconnette al server, fornisce il cookie, che identifica il client sul server
* il server successivamente fornisce le informazioni sensibili.

Gestione richiesta di salvataggio del cookie tramite funzionalità “rememberme”.

[LoginServlet.java]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

Generazione del token per il cookie, esso sarà costituito da:

* username
* password
* token ottenuto dal file “token.txt”

inoltre, viene applicato l’AES sul token, con rispettiva generazione della chiave nel file “secret.key”.

[Aes.java] Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Creazione del cookie ed assegnazione del token.

[CookieManagment.java]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

Verifica presenza del cookie in fase di login, e se esso contiene: username, password, e token validi.

[Login.jsp]

**Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente**

Inserimento automatico credenziali se il cookie è presente

[Login.jsp]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Cross-site scripting (XSS)**

I cookie non proteggono le informazioni sensibili dagli attacchi di cross-site scripting (XSS).

Per ovviare al problema è stato applicata la seguente verifica ai file .txt, per rigettarli in caso contenessero uno script al loro interno.

[File.java]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**protocollo HTTPS**

Comunicazione https garantita da un certificato ssl/tls «self-signed»

* Utili per motivi di test, consentono di utilizzare il protocollo https ma nessuno ha garantito per il sito. Il sito potrebbe esser stato creato da chiunque e quindi non certificato.

Inoltre, è stato applicato un limite di tempo di inattività durante la sessione.

[web.xml]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Logout**

Per questioni di sicurezza è stato fatto in modo di poter invalidare una sessione https attiva tramite il tasto di logout presente nella home del sito.

Inoltre, la sua funzione è quella di cancellare il cookie dal browser

[LofoutServlet.java]

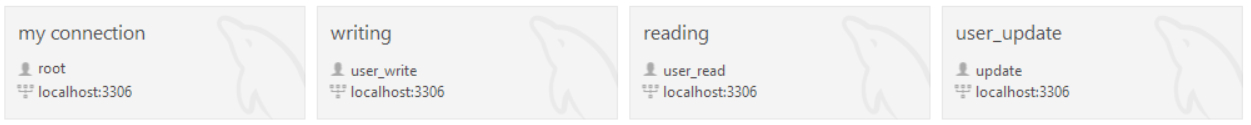
Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Java Database Connectivity**

Per l’accesso al DB sono stati creati diversi profili per limitare al minimo necessario i permessi per effettuare quary al database e dunque possibili attacchi SQL Injection:

* user\_read, per la lettura dei dati;
* user\_write, per la scrittura dei dati nel databse;
* user\_update, per la sovrascrittura dei dati nel database.



Questi profili sono accessibili dal metodo “Connection connect(String username, String password)” , che permette di effettuare una connessione ai diversi profili tramite i due parametri “username” e “password”.

Inoltre, questo metodo assimila i restanti parametri di configurazione da un file chiamato “config.ini”.

[Config.ini]

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

[Database.java]

Immagine che contiene testo, schermata

Descrizione generata automaticamente

**Caricamento file**

Le applicazioni che accettano i caricamenti di file devono garantire che un utente malintenzionato non possa caricare o trasferire file dannosi; Se un file contenente codice viene eseguito dal sistema di destinazione, può compromettere le difese dell'applicazione in esecuzione. Un utente malintenzionato può inviare un file HTML valido con XSS che verrà eseguito in assenza di una routine di escape, oppure potrebbe caricare file con estensioni pericolose come .exe e .sh causando l'esecuzione di codice su applicazioni lato server.

Per questa ragione durante la fase di registrazione, nel quale si inserisce un’immagine di profilo, e nella home, quando l’utente inserisce una proposta progettuale, viene effettuata una verifica del tipo dal metodo String fileType(String file).

[RegistrationServlet.java]

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

[HomeServlet.java]

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Questo metodo utilizza la libreria Tika per analizzare direttamente i metadati del file, in questo modo si può essere certi del tipo del file, ed individuare, ad esempio, possibili .exe la cui reale estensione è stata nascosta.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

Successivamente alla verifica dell’estensione del file sono state adottate misure di sicurezza per quanto riguarda gli attacchi XSS, resi possibili tramite l’inserimento file validi contenenti HTML malevolo. Nella homeServlet è stato applicato un ulteriore controllo sul contenuto del file .txt richiamando il metodo boolean containsScript(Part filePart); In caso di positività il file viene scartato.

[HomeServlet.java]

Immagine che contiene testo, Carattere, schermata, Elementi grafici

Descrizione generata automaticamente

[File.java]

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Gestione delle password**

I programmi che memorizzano le password come testo non crittografato rischiano l'esposizione di tali password in vari modi. I programmi dovrebbero garantire che le password non vengano archiviate come testo in chiaro, una tecnica accettabile per limitare l'esposizione delle password è l'uso di funzioni hash che consentono ai programmi di confrontare indirettamente una password di input con la stringa della password originale senza memorizzare un testo in chiaro o una versione decrittata della password.

Come prima cosa sono stati applicati dei controlli basilari sulla password inserita dall’utente, come il “conferma password”, e la formattazione della password (lettera minuscola, maiuscola, numero e carattere speciale) in modo che sia sufficientemente sicura; stesso controllo è stato attuato anche per il formato dell’username ([text@text@.domain](mailto:text@text@.domain)).

[RegistrationServlet.java]

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Una buona pratica è quella di aggiungere sempre un sale (salt) alla password che viene sottoposta a hash; Il suo uso, insieme all’hash e all’utilizzo di una password robusta prevengono attacchi di tipo brute force.

[RegistrationDao.java]

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

La generazione della stringa casuale, da utilizzare come sale è stata generata in modo sicuro dalla libreria “security.SecureRandom”.

[Password.java]

Immagine che contiene testo, schermata, schermo, Carattere

Descrizione generata automaticamente

Invece per hash della password ed il sale è stato utilizzato l’algoritmo SHA-256 in quanto pubblicamente ritenuto sicuro.

[Password.java]

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Infine, la password hashata è stata caricata all’interno del database.

Password in chiaro, sale e password hashata, essendo comunque informazioni sensibili sono state rimosse dalla memoria non appena completato il loro utilizzo tramite il metodo “Array.fill”. ciò è stato applicato anche per tutte le altre variabili sensibili nel progetto.

[RegistrationDao.java]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Programmazione difensiva**

I meccanismi del linguaggio Java dovrebbero essere utilizzati per limitare l'ambito, la durata e l'accessibilità delle risorse del programma. Un buon principio generale per la programmazione difensiva è la semplicità, per cui il programma dovrebbe aiutare il sistema runtime Java limitando le risorse che utilizza e rilasciando le risorse acquisite quando non sono più necessarie.

L’elevata accessibilità va in contrasto con il principio dell'incapsulamento indebolendo la sicurezza delle applicazioni Java, per cui per ciascun metodo e variabile ne è stato minimizzato lo scope.

Immagine che contiene testo, schermata, Carattere, ricevuta

Descrizione generata automaticamente

Inoltre, dove non necessario che le classi fossero pubbliche, ne è stato limitato l’accesso unicamente da parte delle altre classi nello stesso package.

Infine, sono stati utilizzati al posto dei metodi void, metodi boolean che fornissero un feedback sull’loro output. Un esempio è il seguente:

[Aes.java]



[Aes.java]

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Identifica i file utilizzando più informazioni**

Molte vulnerabilità di sicurezza relative ai file derivano da un programma che accede a un oggetto di tipo File in maniera non intenzionale. L'associazione di un nome file a un oggetto File viene rivalutata ogni volta che viene utilizzato il nome file in un'operazione, questa rivalutazione può introdurre un bug del tipo time-of-check, time-of-use (TOCTOU) in un'applicazione. Oggetti di tipo java.io.File e di tipo java.nio.file.Path sono associati a oggetti file sottostanti solo a tempo di accesso al file.

Per ovviare a questo problema sia per l’immagine dell’profilo utente durante la registrazione, sia durante l’inserimento del file proposta dell’utente nella home, è stato applicato un controllo sulla dimensione del file. Non appena viene preso in input viene memorizzato in una variabile la dimensione del file.

[RegistrationServlet.java]

****

Prima che esso venga inserito all’interno del db si effettua un ultimo controllo per verificare se la dimensione del file sia rimasta invariata, e dunque se non è stato compromesso. Ciò riduce al minimo la finestra di Time-Of-Check Time-Of-Use in cui il file può essere sostituito.

[RegistrationDao.java]

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Thread-Safety**

L'annotazione del codice sorgente è un meccanismo per associare i metadati a un elemento del programma e renderlo disponibile al compilatore, agli analizzatori, ai debugger o alla JVM (Java Virtual Machine).

Nel progetto sono state utilizzate diverse annotazioni per documentare la sicurezza del thread utilizzando la libreria JCIP: @ThreadSafe, @NotThreadSafe, @Immutable e @GuardedBy.

Di seguito alcuni esempi:

****[HomeDao.java] [Proposal.java]

**Immagine che contiene testo, Carattere, schermata, Elementi grafici

Descrizione generata automaticamente**

**Immagine che contiene testo, Carattere, Elementi grafici, tipografia

Descrizione generata automaticamente**  [Aes.java]

**ANALISI DINAMICA: Test uso**

**Registrazione**

Registrazione OK

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, Carattere, logo

Descrizione generata automaticamente**

Registrazione con file non valido

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente**

Registazione con formato username non valido

**Immagine che contiene testo, schermata, linea, bianco

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente**

Registrazione con password diversa da conferma password

Immagine che contiene testo, Carattere, schermata, design

Descrizione generata automaticamente

Registrazione con formato password non valido

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Login**

Login OK

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata

Descrizione generata automaticamente**

Login KO

Immagine che contiene testo, schermata, Carattere, logo

Descrizione generata automaticamente

**Home**

Caricamento proposta progettuale OK



Caricamento proposta progettuale KO

****

Inserimento file con formato non valido

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

Visualizzazione lista delle proposte progettuali

**Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente**

Visualizzazione contenuto di una specifica proposta progettuale

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

**Session timeout**

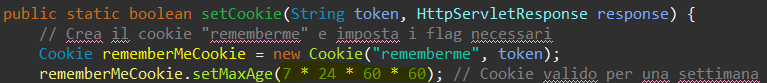
Il timeout della sessione è stato configurato a 15 minuti

**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente**

**Cookie scaduto**

La lifetime del cookie è stata settata ad una settimana, dopo di che il cookie semplicemente verrà eliminato e non sarà più presente in fase di login

****

**Logout**

**Immagine che contiene testo, schermata, Carattere, logo

Descrizione generata automaticamente**

**ANALISI DINAMICA: Test abuso**

**Registrazione**

registrazione con file non valido (.exe)

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente**

registrazione con file non valido (fake .jpeg)

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente **Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente**

**Login**

Login bypass via SQL Injection su email

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente Immagine che contiene testo, schermata, Carattere, logo

Descrizione generata automaticamente

**Home**

Caricamento di un file .txt di proposta progettuale e contenente script/codice per attacco XSS



Immagine che contiene Carattere, testo, Elementi grafici, tipografia

Descrizione generata automaticamente

****