

TECHNICAL REPORT

Indice

1. Introduzione	3
2. Metodologia di analisi	3
3. Il perimetro di analisi	4
4. Riepilogo	5
4.1 Istruzioni per affrontare le vulnerabilità	5
5. Dettagli tecnici	7
5.1 Vulnerabilità dell'applicazione	8
5.1.1 <i>Path Traversal</i>	8
5.1.2 <i>SQL Injection</i>	10
5.1.3 <i>Cross-site Scripting</i>	11
5.1.4 <i>Regular expression Denial of Service</i>	14
5.1.5 <i>Esposizione di informazioni di debug</i>	15
5.1.6 <i>Mancata crittografia delle password</i>	17

1. Introduzione

Questo report descrive le vulnerabilità individuate nell'applicazione web "GamesEmpire". I risultati presentati in questo report tecnico non rappresentano necessariamente una dichiarazione esaustiva di tutte le vulnerabilità e criticità esistenti. È quindi possibile che esistano o sorgano vulnerabilità non identificate durante la nostra revisione. Quello che viene richiesto è risolvere soltanto le vulnerabilità descritte in questo report.

2. Metodologia di analisi

Le vulnerabilità sono state individuate attraverso tecniche di analisi dinamica e statica.

- L'**analisi dinamica** ha identificato comportamenti insicuri dell'applicazione in esecuzione. Per tali vulnerabilità, viene indicata la proof of concept, cioè come replicare il comportamento insicuro durante l'esecuzione dell'applicazione. **Nota bene:** non essendo evidenziati i file e le linee di codice vulnerabili, la risoluzione di tali vulnerabilità richiede allo sviluppatore di identificare sia i file che le righe di codice da modificare.
- L'**analisi statica** del codice sorgente ha esaminato l'intero codice sorgente dell'applicazione alla ricerca di vulnerabilità. Per tali vulnerabilità, vengono indicate i file e le linee di codice in cui sono state individuate. **Nota bene:** la risoluzione di tali vulnerabilità *potrebbe* richiedere la modifica di *altre* righe di codice.

Ciascuna vulnerabilità individuata è stata classificata secondo i rischi di sicurezza della OWASP Top 10 (versione 2021) e con il CVSS Score. In accordo al CVSS Score, alle vulnerabilità presentate in questo documento sarà assegnata una delle seguenti livelli di severity:

Livelli di severity	Descrizione	Range CVSS
Critical	La vulnerabilità permette all'attaccante di compromettere del tutto l'asset.	9.0 - 10.0
High	La vulnerabilità consente l'esecuzione di codice malevolo, ma non permette l'accesso amministrativo alle risorse o ai dati presenti nel database.	7.0 - 8.9
Medium	La vulnerabilità consente di acquisire informazioni per successivi attacchi. Potrebbe inoltre permettere all'attaccante di modificare le normali operazioni dell'asset.	4.0 - 6.9
Low	La vulnerabilità consente di recuperare informazioni sulle attività del cliente, ma con un impatto molto basso sulle attività.	0.1 - 3.9
Informational	La vulnerabilità è presente ma non è stato possibile sfruttarla durante l'attività di security assessment.	0

3. Il perimetro di analisi

Il perimetro di analisi è descritto dalla seguente tabella:

Applicazione	Ambiente	URL
GamesEmpire	Test	http://localhost:8080/GamesEmpireApp/

L'applicazione "GamesEmpire" è stata analizzata alla ricerca di vulnerabilità in ambiente di test. L'applicazione web è stata hostata in *localhost* sulla *porta 8080*.

Nota bene: rieseguire l'applicazione sul proprio computer (e con le proprie configurazioni per il web server) potrebbe prevedere un URL diverso (es. un diverso numero di porta).

Per avviare l'applicazione web, si suggerisce di utilizzare:

- Eclipse IDE for Enterprise Java and Web Developers - 2021-12
- Apache Tomcat v9.0 Server at localhost
- JavaSE-17

Inoltre, si consiglia di utilizzare MySQL come Database Management System. È possibile utilizzare lo script "creazione_e_popolamento_database.sql" per creare e popolare il database che verrà utilizzato dall'applicazione web. Tale script si trova nella cartella "database".

Così come riportato all'interno dello script "creazione_e_popolamento_database.sql", è possibile utilizzare le seguenti credenziali per accedere come *cliente*:

- Username/email: user
- Password: user

Invece, per accedere come *admin*, è possibile utilizzare le seguenti credenziali:

- Username/email: admin
- Password: admin

Nota bene: le credenziali per la connessione al database sono:

- Username: root
- Password: root

In caso fosse necessario cambiarle (per accedere al database), è possibile modificarle nel file *WebContent/META-INF/context.xml* (riga 8 e riga 9, rispettivamente).

Nota bene: per modificare la password di un utente di MySQL, è possibile utilizzare una query SQL; tale query SQL, per l'utente avente come username "*root*", è la seguente:

```
SET PASSWORD FOR 'root'@'localhost' = 'root';
```

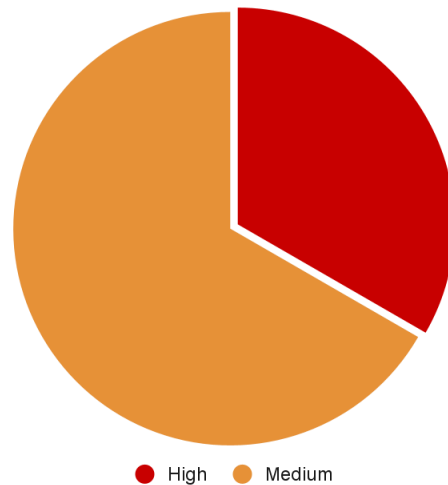
Nota bene: rieseguire l'applicazione sul proprio computer (e con le proprie configurazioni per il web server) potrebbe prevedere un URL diverso (es. un diverso numero di porta).

Nota bene: i file web.xml, JSP, JavaScript, CSS, etc. si trovano sotto la cartella *WebContent*, la quale è analoga alla cartella *src/main/webapp*.

4. Riepilogo

Questa sezione riassume le vulnerabilità individuate nell'applicazione.

Vulnerabilità



L'attività svolta ha evidenziato la presenza di 6 vulnerabilità, distribuite in questo modo:

- **2** vulnerabilità ad impatto **high**;
- **4** vulnerabilità ad impatto **medium**.

Il livello viene determinato in accordo agli standard de facto internazionali (CVSS) per la sicurezza delle informazioni, tenendo conto delle vulnerabilità che potrebbero compromettere l'integrità e la riservatezza dell'applicazione e dei dati in essa contenuti.

4.1 Istruzioni per affrontare le vulnerabilità

- Le seguenti sezioni di questo documento illustrano le vulnerabilità che è necessario individuare e risolvere all'interno dell'applicazione web.
- Per ciascuna vulnerabilità, vengono riportate le seguenti informazioni:
 - Nome
 - CVSS Score
 - Classificazione in accordo alla OWASP Top 10
 - Status
 - Descrizione, la quale serve a fornire informazioni sul problema riscontrato
 - *Proof of concept* o la *localizzazione*, a seconda se la vulnerabilità è stata individuata attraverso analisi dinamica o statica dell'applicazione web
 - La remediation, la quale suggerisce strategie per la risoluzione della vulnerabilità
 - Riferimenti alla classificazione CWE
- Per mettere in sicurezza l'applicazione web, si richiede di leggere la documentazione fornita in questo technical report e di individuare e risolvere le vulnerabilità.

- **È OBBLIGATORIO individuare e risolvere ciascuna vulnerabilità nell'ordine in cui vengono presentate nel technical report.** Questo significa che dovrete affrontare prima della vulnerabilità avente ID-1, poi ID-2, etc.
- **Quando avete finito di affrontare una vulnerabilità, DOVETE effettuare un commit per comunicarlo!**
- **Questo deve essere fatto sia in caso avete risolto la vulnerabilità che altrimenti.** Ad esempio, potreste decidere di non continuare ad affrontare una certa vulnerabilità e di voler passare alla successiva.
- **Prima di passare ad una nuova vulnerabilità, è NECESSARIO effettuare un commit per dire che avete finito di affrontare la vulnerabilità che stavate affrontando.** Questo significa che iniziate ad affrontare prima la vulnerabilità avente ID-1; poi, quando avete finito, EFFETTUATE UN COMMIT e DOPO passate alla vulnerabilità ID-2 (e così via per tutte le altre vulnerabilità).
- **Il messaggio di commit DEVE avere la seguente struttura:**
 - Nel caso in cui pensate di aver risolto la vulnerabilità: ***id-1 resolved***
 - Sostituite *id-1* con l'id della vulnerabilità che avete affrontato (id-1 è per la prima vulnerabilità, id-2 è per la seconda vulnerabilità, etc.)
 - Nel caso in cui non avete risolto la vulnerabilità: ***id-1 unresolved***
 - Sostituite *id-1* con l'id della vulnerabilità che avete affrontato (id-1 è per la prima vulnerabilità, id-2 è per la seconda vulnerabilità, etc.)
 - Se non avete modificato alcun file, potreste ritrovarvi nella condizione in cui non riuscite ad effettuare il commit per dire che non avete risolto la vulnerabilità (non ci riuscite perché per effettuare un commit è necessario aver modificato almeno un file). In tal caso, aprite un file qualsiasi ed effettuate una modifica non significativa, come l'aggiunta di uno spazio; successivamente, effettuate il commit.

5. Dettagli tecnici

Questa sezione offre un'analisi dettagliata di ciascuna vulnerabilità riscontrata durante le attività presentate.

La tabella immediatamente successiva mostra una panoramica dei risultati e, per ciascuna vulnerabilità, viene dettagliata un'apposita tabella contenente il livello di criticità ed un ID per future referenze.

ID	Nome	Severity	Stato
ID-1	Path Traversal	High	Open
ID-2	SQL Injection	High	Open
ID-3	Cross-site Scripting	Medium	Open
ID-4	Regular expression Denial of Service	Medium	Open
ID-5	Esposizione di informazioni di debug	Medium	Open
ID-6	Mancata crittografia delle password	Medium	Open

5.1 Vulnerabilità dell'applicazione

5.1.1 Path Traversal



CVSS Score: 8.1

OWASP: A01:2021 - Broken Access Control

Status: Open

Descrizione

L'analisi dinamica dell'applicazione web ha evidenziato una vulnerabilità relativa a Path Traversal. La vulnerabilità Path Traversal consente a un attaccante di accedere a file, directory e comandi che risiedono lato server ma al di fuori dell'applicazione web (ad esempio, i file context.xml e web.xml). Infatti, la maggior parte delle applicazioni web limita l'accesso degli utenti soltanto a una porzione specifica del file system (lato server). Per accedere a queste risorse in modo illecito, l'attaccante può modificare strategicamente l'URL specificando il percorso che gli interessa.

Proof of Concept

Al fine di sfruttare la vulnerabilità, sono stati seguiti i seguenti step:

1. È stata visitata la seguente pagina web:
<http://localhost:8080/GamesEmpireApp/home?page=META-INF/context.xml>
2. Viene visualizzata la seguente pagina web:

Sembra che il file XML specificato non abbia un foglio di stile associato. L'albero del documento è visualizzato di seguito.

```
<Context>
  <Resource name="jdbc/storage" auth="Container" driverClassName="com.mysql.cj.jdbc.Driver" type="javax.sql.DataSource"
    username="root" password="root" url="jdbc:mysql://localhost:3306/storage?
    useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC"/>
</Context>
```

3. La pagina visualizzata riporta il contenuto del file context.xml; tra queste informazioni, ci sono anche quelle per connettersi al database!
4. È stata visitata anche la seguente pagina web:
<http://localhost:8080/GamesEmpireApp/home?page=WEB-INF/web.xml>
5. Viene visualizzata la seguente pagina web:

Sembra che il file XML specificato non abbia un foglio di stile associato. L'albero del documento è visualizzato di seguito.

```
<web-app xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <welcome-file-list>
    <welcome-file>Home.jsp</welcome-file>
  </welcome-file-list>
  <resource-ref>
    <description>JNDI reference to a data source</description>
    <res-ref-name>jdbc/storage</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

6. La pagina visualizzata riporta il contenuto del file web.xml!

Nota bene: potrebbe capitare di visualizzare una pagina web completamente bianca; questo non significa che non è stato effettuato l'accesso a `context.xml` o `web.xml`, ma che il browser ha avuto problemi nella visualizzare di un file senza fogli di stile (infatti, l'immagine precedente riporta la scritta "Sembra che il file XML specificato non abbia un foglio di stile associato. L'albero del documento è visualizzato di seguito.").

Remediation

L'applicazione web non deve consentire la navigazione verso i file `web.xml` e `context.xml`. Per questo motivo, controllare il valore che può assumere il parametro `page` e assicurarsi che non sia possibile navigare verso `META-INF/context.xml` e `WEB-INF/web.xml`; nel caso in cui il parametro `page` indichi un reindirizzamento verso `META-INF/context.xml` oppure `WEB-INF/web.xml`, forzare un reindirizzamento verso un'altra pagina del sito.

Riferimenti

1. [CWE-22: Improper Limitation of a Pathname to a Restricted Directory \('Path Traversal'\)](#)

5.1.2 SQL Injection



CVSS Score: 8.0

OWASP: A03:2021 - Injection

Status: Open

Descrizione

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità di SQL injection dovuta alla concatenazione di stringhe rappresentative di query SQL. Le query SQL ottenute dalla concatenazione di più stringhe possono costituire un rischio di SQL injection perché potrebbero venire concatenati dei valori malevoli (proveniente, ad esempio, dai parametri di form) all'interno della query SQL; al momento dell'esecuzione della query SQL, tali valori malevoli porterebbero portare ad eseguire delle query SQL inaspettate perché essi stessi potrebbero contenere delle altre query SQL (che sono decise dall'attaccante e non dallo sviluppatore e che possono riguardare, ad esempio, la modifica o distruzione del database).

Localizzazione della vulnerabilità

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità SQL injection nella funzione:

```
public synchronized ArrayList<ProdottoBean> doRetrieveAll(String order)
```

della classe:

ProdottoDao

Il rischio è rappresentato dalla concatenazione del parametro *order* all'interno della stringa assegnata alla variabile *selectSQL* (riga 157). Tale stringa rappresenta una query SQL.

La seguente tabella riporta le linee di codice, identificate dall'analisi statica, in cui la vulnerabilità può essere sfruttata da un attaccante. **Nota bene:** la risoluzione di tale vulnerabilità *potrebbe* richiedere la modifica di *altre* righe di codice, *anche* in altri file.

File	Linee di codice
src/it/unisa/model/ProdottoDao.java	157, 162

Remediation

Un'applicazione è vulnerabile alle injection quando i dati forniti in input dall'utente (ad esempio attraverso un form) non sono validati, filtrati o sanificati dall'applicazione.

Nel caso della SQL injection, la query SQL è ottenuta concatenando stringhe fornite in input dall'utente come parametri di richieste HTTP.

Quando ad essere concatenati sono valori rappresentativi di *colonne* di una tabella (come ad esempio nella clausola *ORDER BY*), si consiglia di verificare che tali valori rappresentino davvero le colonne della tabella che si sta interrogando. Per fare ciò, è possibile effettuare dei controlli rispetto ad una lista contenente valori corrispondenti alle colonne della tabella che si sta interrogando.

Riferimenti

1. [CWE-89: Improper Neutralization of Special Elements used in an SQL Command \('SQL Injection'\)](#)

5.1.3 Cross-site Scripting

Medium

CVSS Score: 4.8

OWASP: A03:2021 - Injection

Status: Open

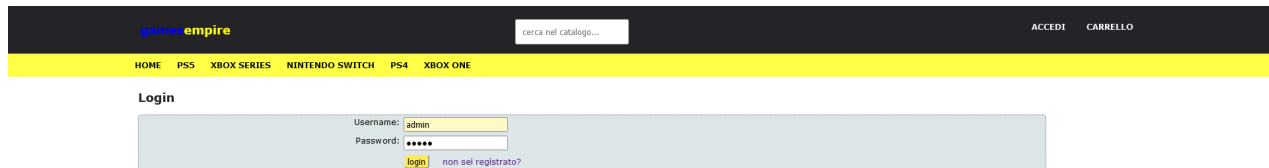
Descrizione

L'analisi dinamica dell'applicazione web ha evidenziato una vulnerabilità relativa a Cross-site Scripting. Tale vulnerabilità permette ad un attaccante di inserire o eseguire codice lato client (ad esempio codice JavaScript) al fine di attuare un insieme variegato di attacchi quali; ad esempio, raccolta, manipolazione e reindirizzamento di informazioni riservate, visualizzazione e modifica di dati presenti lato server, alterazione del comportamento dinamico delle pagine web, ecc.

Proof of Concept

Al fine di sfruttare la vulnerabilità, sono stati seguiti i seguenti step:

1. È stata visitata la pagina web <http://localhost:8080/GamesEmpireApp/Login.jsp> e sono state inserite le credenziali per effettuare l'accesso come *admin* (username=admin, password=admin):



Games Empire © 2021.

2. È stata visitata la pagina web per l'inserimento di un nuovo prodotto nel catalogo: <http://localhost:8080/GamesEmpireApp/admin/AddProdotto.jsp>

games:empire

cerca nel catalogo...

ACCOUNT CARRELLO

HOME PS5 XBOX SERIES NINTENDO SWITCH PS4 XBOX ONE

AGGIUNGI PRODOTTO

Nome:

Descrizione:

Iva:

Prezzo:

Data:

Quantità:

Immagine:

Piattaforma:

Genere:

Descrizione dettagliata:

aggiungi

Games Empire © 2021.

3. Sono stati inseriti i seguenti dati per il nuovo prodotto:

games:empire

cerca nel catalogo...

ACCOUNT CARRELLO

HOME PS5 XBOX SERIES NINTENDO SWITCH PS4 XBOX ONE

AGGIUNGI PRODOTTO

Nome:

Descrizione:

Iva:

Prezzo:

Data:

Quantità:

Immagine:

Piattaforma:

Genere:

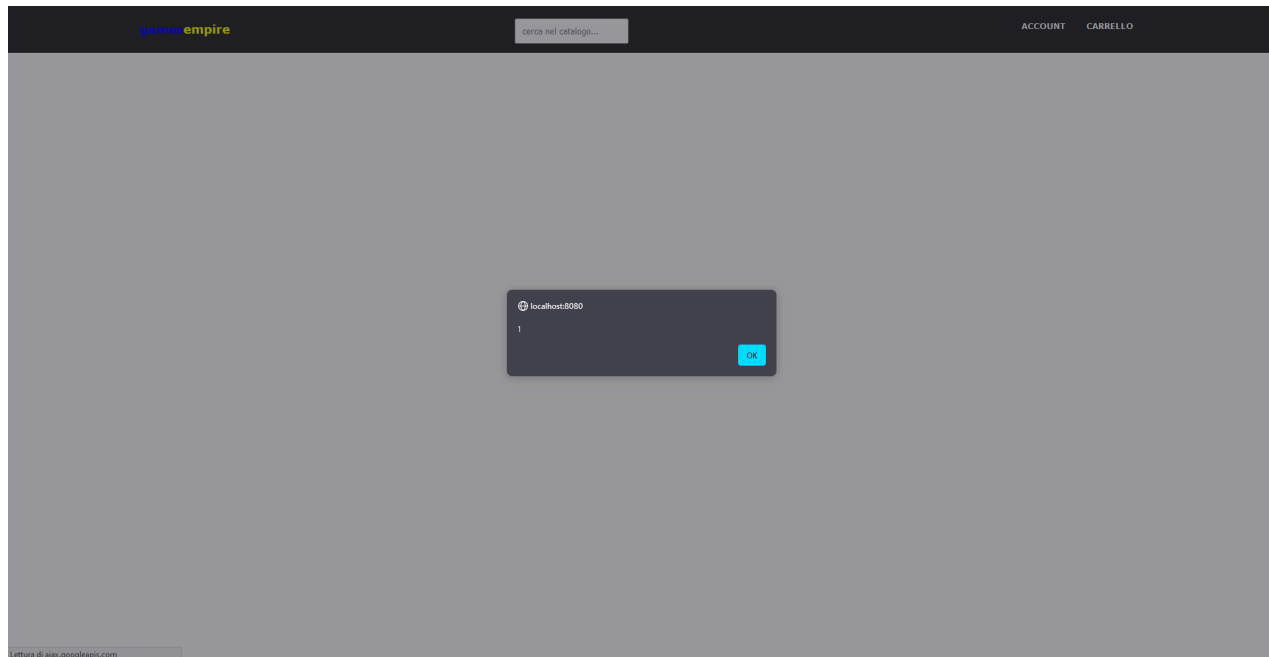
Descrizione dettagliata:

aggiungi

Games Empire © 2021.

Come è possibile osservare, per il campo “Nome” è stato inserito il seguente codice JavaScript: **<script>alert(1)</script>**.

4. È stata visitata la pagina web per vedere il catalogo con il nuovo prodotto inserito:
<http://localhost:8080/GamesEmpireApp/admin/GestioneCatalogo.jsp>
5. Viene visualizzato un *alert* dovuto all’esecuzione del codice JavaScript:
<script>alert(1)</script>



Remediation

Un'applicazione è vulnerabile a Cross-site Scripting quando i dati forniti in input dall'utente (ad esempio attraverso un form) non sono validati, filtrati o sanificati dall'applicazione. Lato *server*, si consiglia di effettuare dei controlli sui caratteri dei parametri inviati dagli utenti per il form dell'aggiunta di un nuovo prodotto, evitando che vengano memorizzati caratteri speciali come “<” e “>” anche in formato encoded. È possibile effettuare tale controllo anche senza utilizzare librerie esterne.

Riferimenti

1. [CWE-79: Improper Neutralization of Input During Web Page Generation \('Cross-site Scripting'\)](#)

5.1.4 Regular expression Denial of Service

Medium

CVSS Score: 6.8

OWASP: A04:2021 - Insecure Design

Status: Open

Descrizione

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità di Regular expression Denial of Service. Il rischio è rappresentato dall'utilizzo di un'espressione regolare molto complessa per verificare che, durante la registrazione di un utente al sito, l'input inserito per il campo email rappresenti davvero un indirizzo email. Ciò può portare un attaccante a sfruttare l'alto tempo di computazione necessario per valutare un'espressione regolare complessa, al fine bloccare l'applicazione web per un tempo molto lungo.

Localizzazione della vulnerabilità

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità di Regular expression Denial of Service all'interno della funzione:

function checkEmail(inputtxt)

del file:

Registrazione.js

La seguente tabella riporta la linea di codice, identificata dall'analisi statica, in cui la vulnerabilità può essere sfruttata da un attaccante. **Nota bene:** la risoluzione di tale vulnerabilità *potrebbe* richiedere la modifica di *altre* righe di codice, *anche* in altri file.

File	Linea di codice
WebContent/script/Registrazione.js	14

Remediation

Sostituire l'espressione regolare avente un'elevata complessità con una espressione regolare avente una complessità minore, ad esempio lineare.

Riferimenti

1. [CWE-400: Uncontrolled Resource Consumption](#)
2. [CWE-1333: Inefficient Regular Expression Complexity](#)

5.1.5 Esposizione di informazioni di debug

Medium

CVSS Score: 5.8

OWASP: A04:2021 - Insecure Design

Status: Open

Descrizione

L'analisi dinamica dell'applicazione web ha evidenziato una vulnerabilità relativa all'esposizione di informazioni di debug. Il rischio è rappresentato da istruzioni di debug, ad esempio relative alla gestione scorretta (o, addirittura, assente) delle eccezioni, i cui messaggi possono far trapelare informazioni sul sistema, come il percorso dell'applicazione o i nomi dei file.

Proof of Concept

Al fine di sfruttare la vulnerabilità, sono stati seguiti i seguenti step:

1. È stata visitata la seguente pagina web: <http://localhost:8080/GamesEmpireApp/account>
2. Viene visualizzata la seguente pagina web:

HTTP Status 500 – Internal Server Error

Type Exception Report

Message Cannot invoke "it.unisa.model.UserBean.getIndirizzo()" because "user" is null

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
java.lang.NullPointerException: Cannot invoke "it.unisa.model.UserBean.getIndirizzo()" because "user" is null
    it.unisa.control.AccountServlet.doPost(AccountServlet.java:106)
    it.unisa.control.AccountServlet.doGet(AccountServlet.java:29)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:670)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:779)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/9.0.68

3. La pagina visualizzata riporta il contenuto dello stack trace di Java.
4. È stata visitata anche la seguente pagina web:
<http://localhost:8080/GamesEmpireApp/account?action=addS>
5. Viene visualizzata la seguente pagina web:

HTTP Status 500 – Internal Server Error

Type Exception Report

Message Cannot invoke "it.unisa.model.UserBean.getEmail()" because "user" is null

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
java.lang.NullPointerException: Cannot invoke "it.unisa.model.UserBean.getEmail()" because "user" is null
    it.unisa.control.AccountServlet.doPost(AccountServlet.java:75)
    it.unisa.control.AccountServlet.doGet(AccountServlet.java:29)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:670)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:779)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/9.0.68

6. La pagina visualizzata ancora una volta il contenuto dello stack trace di Java!

Remediation

Creare delle pagine di errore per gestire l'errore 500 interno al server ed evitare di far trapelare informazioni sensibili (come i messaggi di errore e lo stack trace). Si consiglia di mappare tali pagine di errore nel *web.xml*.

Riferimenti

1. [CWE-489 - Active Debug Code](#)
2. [CWE-215 - Information Exposure Through Debug Information](#)

5.1.6 Mancata crittografia delle password

Medium

CVSS Score: 6.9

OWASP: A02:2021 - Cryptographic Failures

Status: Open

Descrizione

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità relativa al mancato utilizzo di algoritmi di crittografia per le password degli utenti dall'applicazione web.

Sia durante la registrazione che durante il login, l'applicazione web dovrebbe crittografare con algoritmi sicuri le password inviate dagli utenti attraverso i form. Nello specifico, nel caso della registrazione, l'applicazione web dovrebbe crittografare la password inviata dall'utente e memorizzarla all'interno del database; invece, nel caso del login, l'applicazione web dovrebbe crittografare la password inviata dall'utente e confrontarla con la password crittografata memorizzata all'interno del database.

Localizzazione della vulnerabilità

La seguente tabella riporta le linee di codice, identificate dall'analisi statica, in cui la vulnerabilità può essere sfruttata da un attaccante. **Nota bene:** la risoluzione di tale vulnerabilità *potrebbe* richiedere la modifica di *altre* righe di codice, *anche* in altri file.

File	Linee di codice
src/it/unisa/control/LoginServlet.java	37
src/it/unisa/control/RegistrazioneServlet.java	55
database/creazione_e_popolamento_database.sql	52

Remediation

Utilizzare un algoritmo di crittografia sicuro, come i ben conosciuti SHA-256, SHA-512, SHA-3, per le password inviate dagli utenti.

Quindi, per quanto riguarda la registrazione, gli utenti dovranno essere memorizzati nel database con le password crittografate.

Al momento del login, sarà necessario crittografare la password inviata dall'utente (che tenta l'accesso) e confrontarla con quella presente nel database (in quanto il database contiene le password crittografate).

Di conseguenza, sarebbe necessario modificare anche le password definite per gli utenti nello script di popolamento del database con la versione crittografata, secondo l'algoritmo di crittografia scelto.

Riferimenti

1. [CWE-311: Missing Encryption of Sensitive Data](#)