TECHNICAL REPORT

Indice

1. Introduzione

Questo report descrive le vulnerabilità individuate nell'applicazione web "GeekFactory". 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
GeekFactory	Test	http://localhost:8080/GeekFactory2/

L'applicazione "GeekFactory" 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: mariorossi@gmail.com
- Password: 12345678

Invece, per accedere come *admin*, è possibile utilizzare le seguenti credenziali:
- Username/email: geekfactory@gmail.com
- Password: 12345678

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 *src/main/java/model/DriverManagerConnectionPool.java* (riga 26 e riga 27, 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:

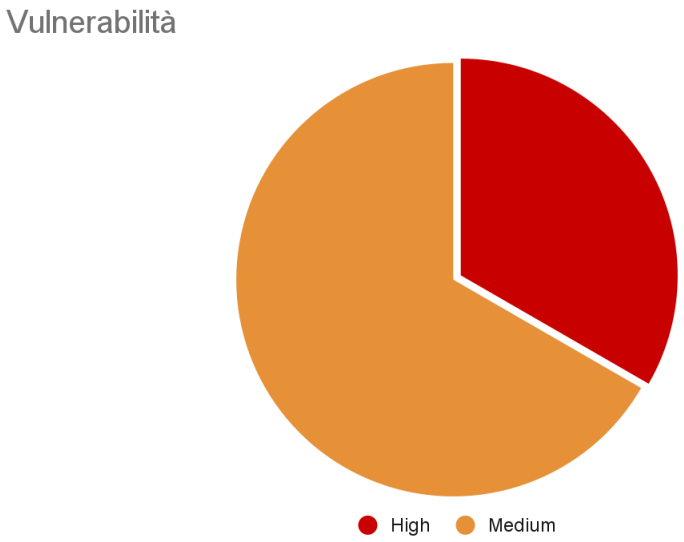
SET PASSWORD FOR 'root'@'localhost' = 'root';

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 *src/main/webapp*, la quale è analoga alla cartella *WebContent*.

4. Riepilogo

Questa sezione riepiloga le vulnerabilità individuate nell'applicazione.



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 documentano 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	SQL Injection	High	Open
ID-2	Assenza controllo di integrità delle risorse (Subresource Integrity)	High	Open
ID-3	Manomissione dei parametri	Medium	Open
ID-4	Cross-site Scripting	Medium	Open
ID-5	Header anti-clickjacking mancante	Medium	Open
ID-6	Primitiva crittografica rischiosa	Medium	Open

5.1 Vulnerabilità dell’applicazione

5.1.1 SQL Injection

High

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 concatenzione 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 Collection<ProductBean> doRetrieveAll(String where)

della classe:

ProductModel

del file:

ProductModel.java

Il rischio è rappresentato dalla concatenazione del parametro *where* all’interno della stringa assegnata alla variabile *selectSQL* (riga 124). 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/main/java/model/ProductModel.java	124, 129

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. Si consiglia di utilizzare query SQL parametriche ed evitare la concatenzione di stringhe nel caso in cui i dati forniti in input dall’utente rappresentino *valori* dei campi di una tabella; assicurarsi di settare correttamente i parametri delle query SQL parametriche con l’input inserito dall’utente.

Riferimenti

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

5.1.2 Assenza controllo di integrità delle risorse (Subresource Integrity)

High

CVSS Score: 7.5

OWASP: A08:2021 - Software and Data Integrity Failures

Status: Open

Descrizione

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità di assenza controllo di integrità delle risorse (Subresource Integrity). Tale controllo permette agli sviluppatori web di garantire che le risorse remote che vengono referenziate nell'applicazione web non siano state manomesse, quindi che siano integre. L'utilizzo di risorse remote senza controlli di integrità può portare all'esecuzione imprevista di codice dannoso nell'applicazione. Un attaccante potrebbe modificare il contenuto di tali risorse remote con del codice malevolo, in modo tale che questo possa essere eseguito incosapevolmente lato client da un utente mentre naviga il sito.

Localizzazione della vulnerabilità

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità di assenza controllo di integrità all'interno del tag:

<head>

del file:

intro.jsp

e riguarda la risorsa esterna *jQuery*.

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	Linea di codice
src/main/webapp/intro.jsp	13

Remediation

Per verificare l'integrità di un artefatto remoto, si consiglia di utilizzare un *Subresource Integrity Hash Generator*, quindi uno strumento che permetta di generare l'hash della risorse remota; tale hash va specificato come valore dell'attributo *integrity* all'interno del tag che referenzia la risorsa remota. L'hash identifica il contenuto di una risorsa; di conseguenza, se il contenuto della risorsa è stato modificato (ad esempio da un attaccante), il browser se ne accorge perché l'hash del contenuto modificato (calcolato a tempo di esecuzione) è diverso da quello specificato dallo sviluppatore (assegnato come valore dell'attributo *integrity*). L'hash deve essere calcolato con un algoritmo di hash sicuro, come SHA512, SHA384 o SHA256. Oltre all'attributo *integrity*, bisogna anche aggiungere l'attributo *crossorigin* per specificare che la richiesta della risorsa esterna non deve trasmettere informazioni sensibili dell'utente.

Riferimenti

1. CWE-353: Missing Support for Integrity Check

5.1.3 Manomissione dei parametri

Medium

CVSS Score: 6.5

OWASP: A04:2021 - Insecure Design

Status: Open

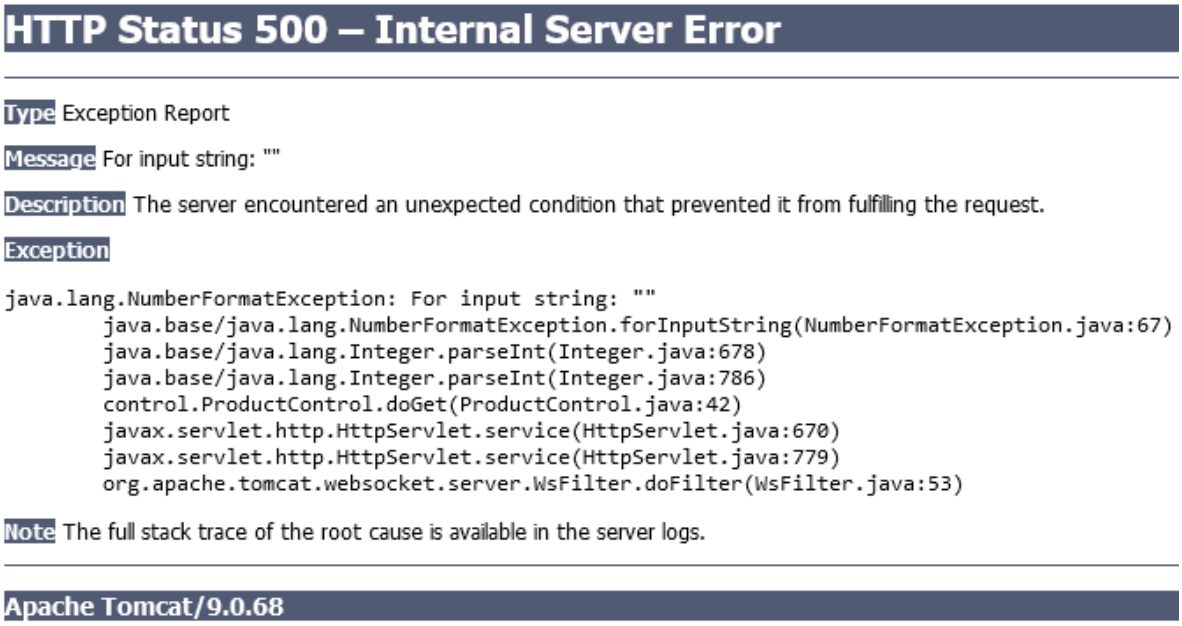
Descrizione

L'analisi dinamica dell'applicazione web ha evidenziato una vulnerabilità di manomissione dei parametri che consente di visualizzazione di pagine di errore contenenti lo stack trace di Java. Quest'ultimo potrebbe contenere informazioni sensibili sull'applicazione e/o sul database (es. struttura del codice sorgente o delle tabelle).

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/GeekFactory2/ProductControl?action=dettaglio&codice=
2. Viene visualizzata la seguente pagina web:



3. La pagina visualizzata evidenzia un errore lato server e mostra lo stack trace di Java con le funzioni invocate e i nomi delle classi coinvolte.

Remediation

Gestire l'errore 500 interno al server in modo appropriato, ad esempio predisponendo delle pagine di errore. In ogni caso, non permettere la visualizzazione di informazioni sensibili, come ad esempio lo stack trace di Java.

Riferimenti

1. CWE-472: External Control of Assumed-Immutable Web Parameter

5.1.4 Cross-site Scripting

Medium

CVSS Score: 4.8

OWASP: A03:2021 - Injection

Status: Open

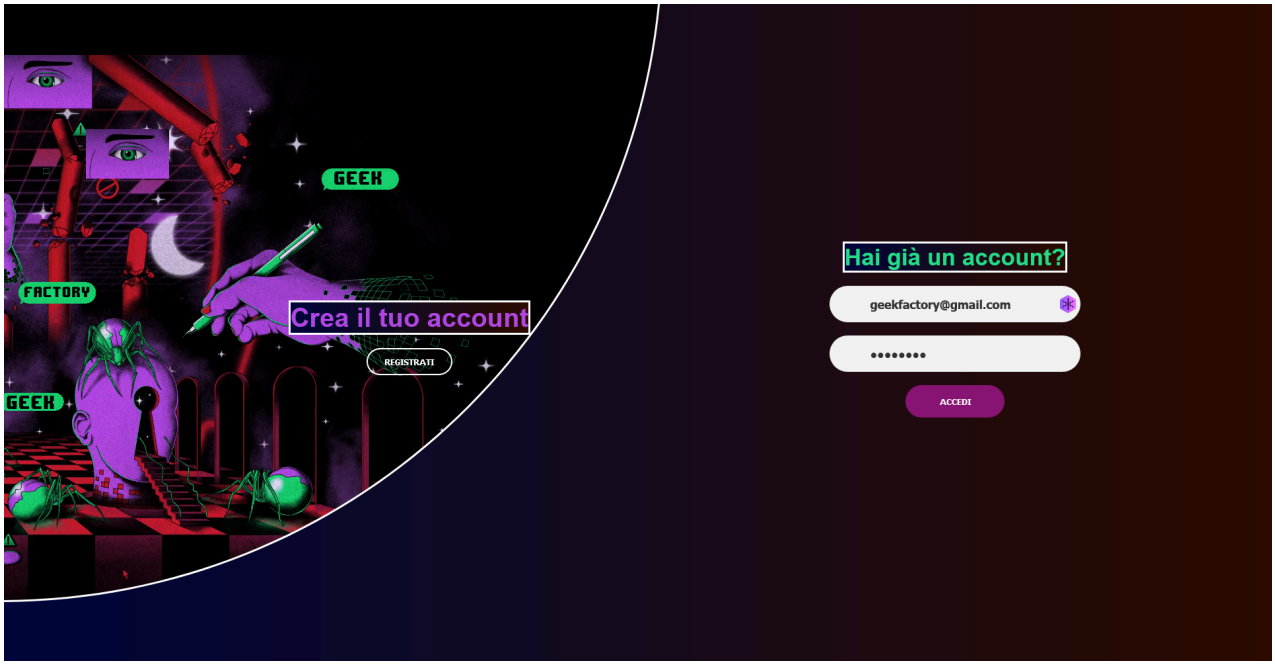
Descrizione

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.

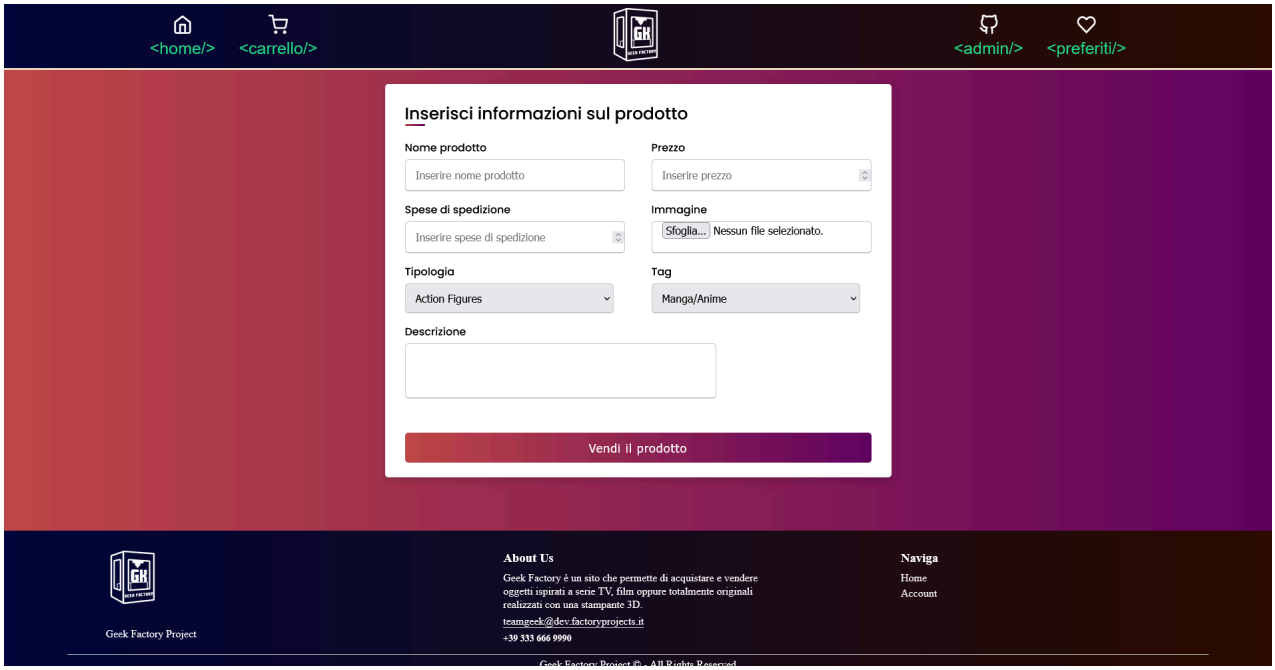
Proof of Concept

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

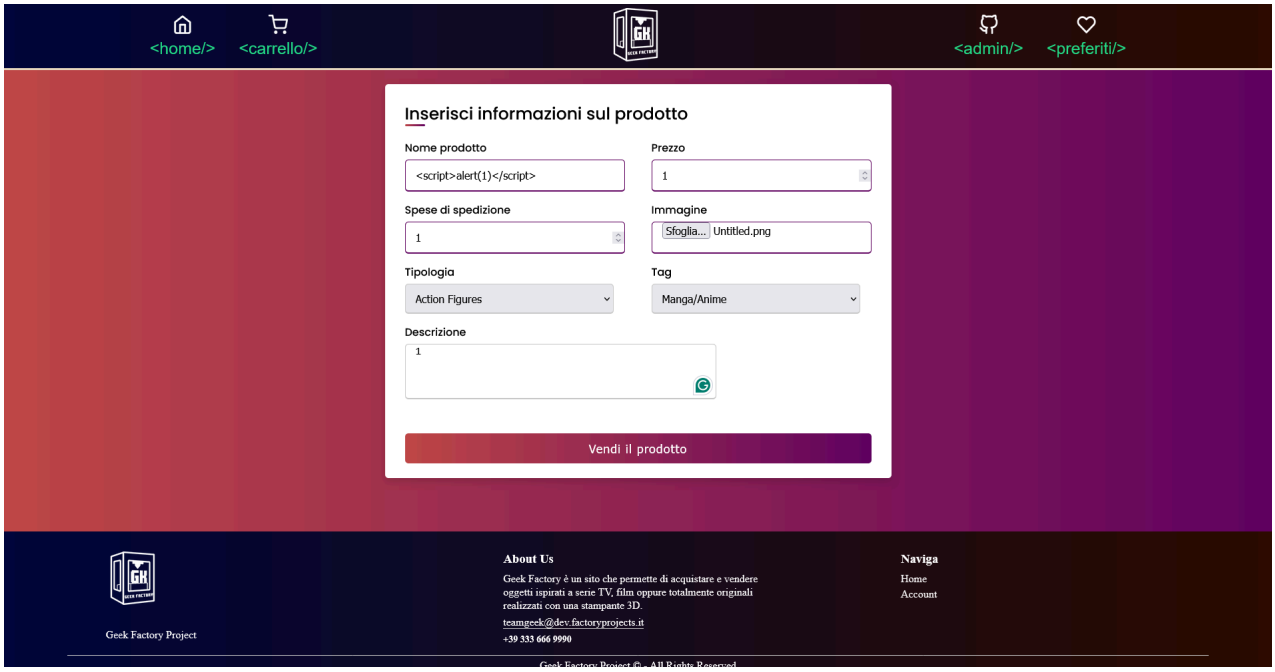
1. È stata visitata la pagina web http://localhost:8080/GeekFactory2/loginPage.jsp e sono state inserite le credenziali per effettuare l'accesso come *admin* (username=geekfactory@gmail.com, password=12345678):



2. È stata visitata la pagina web per l'inserimento di un nuovo prodotto nel catalogo: http://localhost:8080/GeekFactory2/vendita.jsp

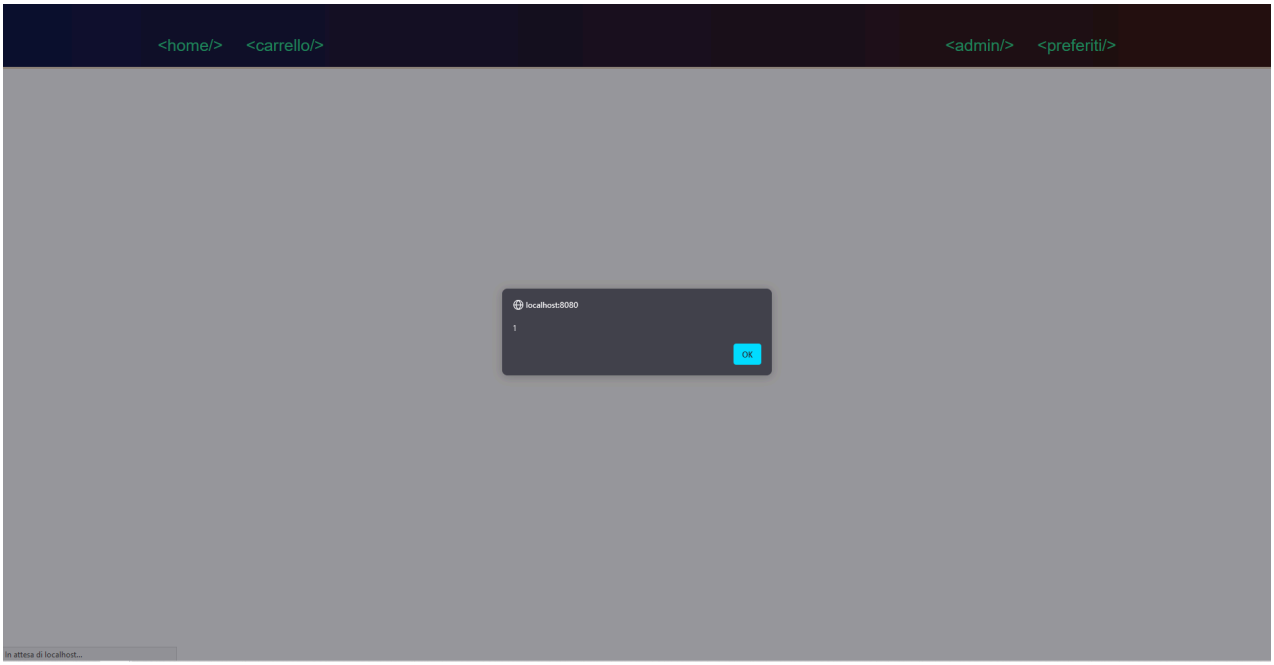


3. Sono stati inseriti i seguenti dati per il nuovo prodotto:



Come è possibile osservare, per il campo “Nome prodotto” 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/GeekFactory2/Tipologia?tipologia=Action-Figures
   Nota bene: dato che il prodotto inserito è della tipologia “Action Figures”, il parametro tipologia nell’URL ha come valore “Action-Figures”.
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. Si consiglia di effettuare dei controlli sui caratteri dei parametri inseriti in input dagli utenti, evitando l'inserimento di caratteri speciali come <, /, >, “, etc. anche in formato encoded.

Riferimenti

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

5.1.5 Header anti-clickjacking mancante

Medium

CVSS Score: 6.2

OWASP: A05:2021 - Security Misconfiguration

Status: Open

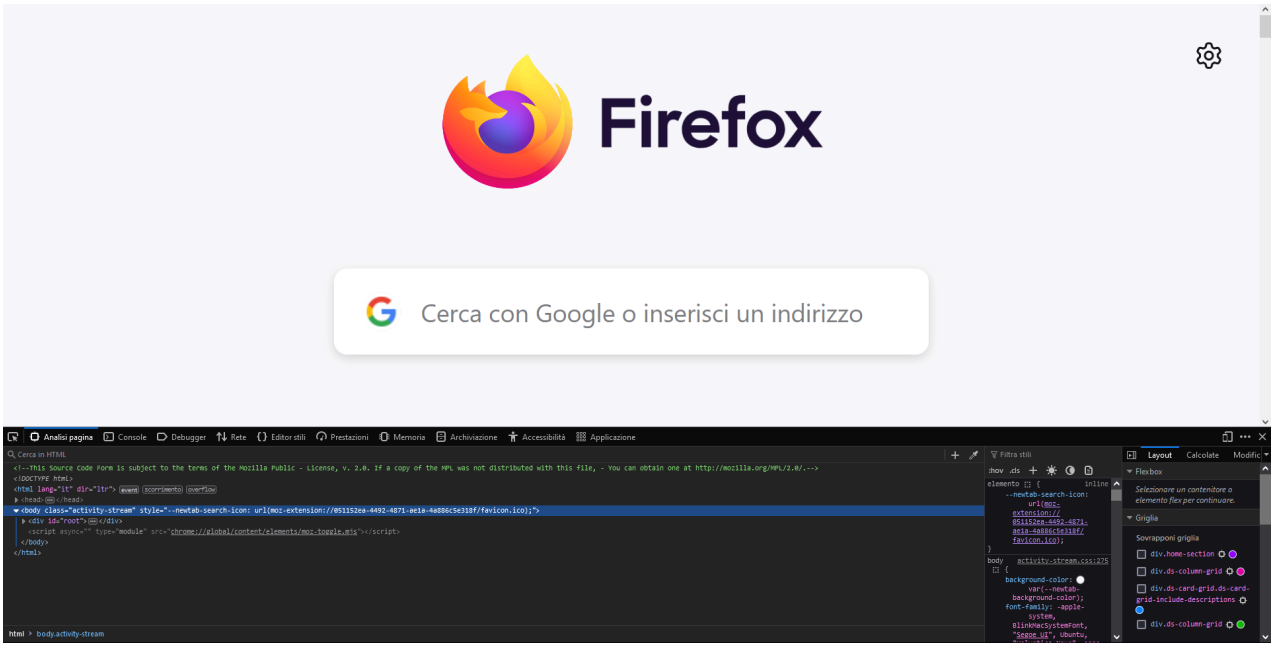
Descrizione

L'analisi dinamica dell'applicazione web ha evidenziato una vulnerabilità relativa all'assenza di header anti-clickjacking; nello specifico, la risposta non include né Content-Security-Policy con la direttiva "frame-ancestors" né X-Frame-Options per proteggere dagli attacchi ClickJacking. Durante una normale navigazione web, ciò potrebbe portare l'utente a cliccare con il puntatore del mouse su di un oggetto (ad esempio un link), ma in realtà il suo clic viene reindirizzato, a sua insaputa, su di un altro oggetto. Ciò può portare all'invio di spam, al download di file, etc.

Proof of Concept

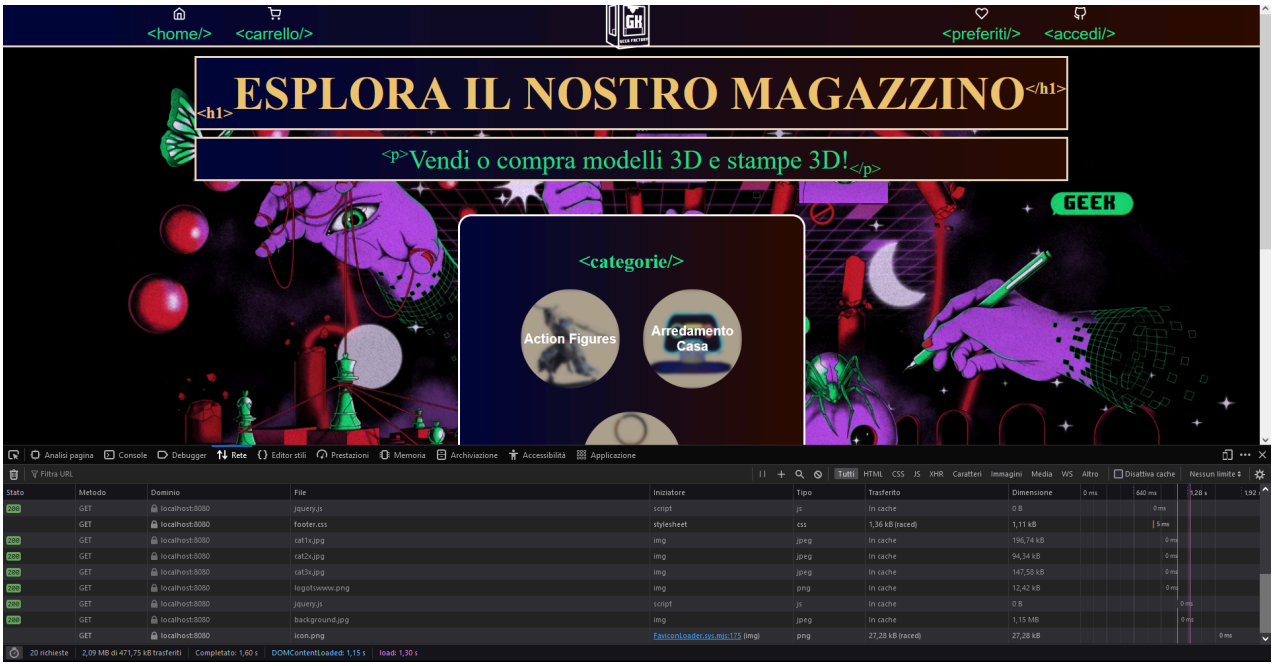
È possibile osservare l'assenza dell'header anti-clickjacking ispezionando gli header delle risposta HTTP. Per verificare tale assenza, utilizzando Mozilla Firefox (analogo per altri browser), sono stati seguiti i seguenti step:

1. È stata aperta la nuova scheda del browser
2. In una finestra del browser, è stato premuto **CTRL+MAIUSC+I** (o **F12**).
3. È stata visualizzata la finestra o il riquadro degli strumenti di sviluppo.



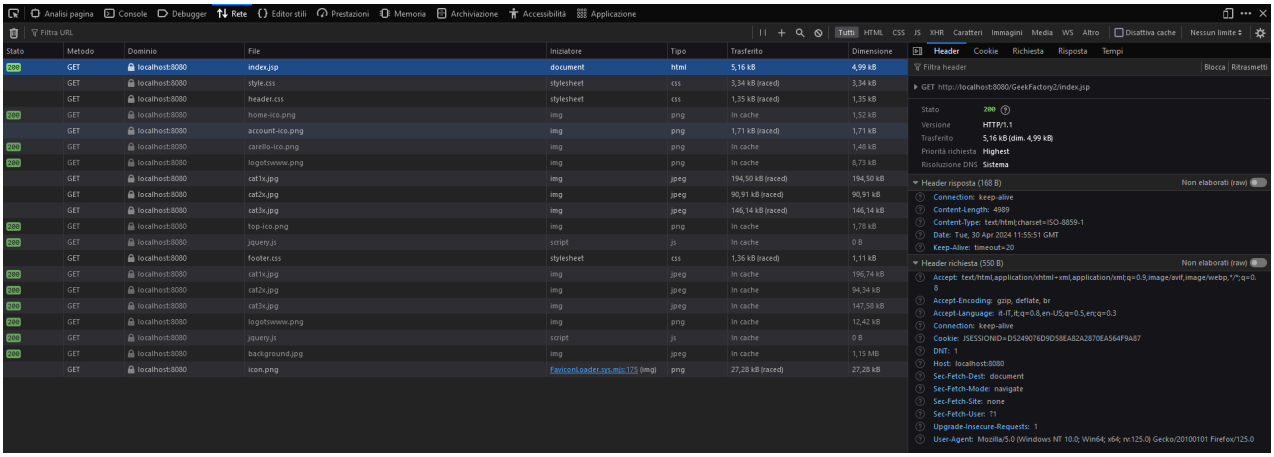
4. È stata selezionata la scheda *Rete*.

5. È stata visitata la seguente pagina web: http://localhost:8080/GeekFactory2/index.jsp



6. È stato cliccato su *index.jsp* in corrispondenza della colonna *File* nella sched *Rete*.

7. Tra gli header della rispota, non appaiono header anti-clickjacking per proteggere dagli attacchi ClickJacking. Nello specifico, la risposta non include né Content-Security-Policy con la direttiva “frame-ancestors” né X-Frame-Options.



Remediation

I web server moderni permettono di integrare security header nelle richieste HTTP attraverso filtri di sicurezza dichiarati nel web.xml.

Riferimenti

[CWE-1021: Improper Restriction of Rendered UI Layers or Frames]

5.1.6 Primitiva crittografica rischiosa

Medium

CVSS Score: 6.8

OWASP: A02:2021 - Cryptographic Failures

Status: Open

Descrizione

L'analisi statica dell'applicazione web ha evidenziato una vulnerabilità relativa all'utilizzo di una primitiva crittografica rischiosa all'interno della funzione:

protected void doPost(HttpServletRequest request, HttpServletResponse response)

della classe:

Login

del file:

Login.java

Il rischio è rappresentato dall'utilizzo dell'algoritmo di hash crittografico MD5, il quale è considerato insicuro per la crittografia delle password degli utenti. Oltre al file *Login.java*, anche i file *Register.java* e *creazione_e_popolamento_database.sql* sono coinvolti perché in essi viene usata una funzione per crittografare le password le password in MD5.

Sia durante la registrazione che durante il login, l'applicazione web dovrebbe crittografare con algoritmi sicuri le password inviate dagli utenti come parametro della richiesta HTTP. Nello specifico, nel caso della registrazione, la password crittografata viene memorizzata all'interno del database; invece, nel caso del login, la password crittograta viene confrontata 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/main/java/control/Login.java	104
src/main/java/control/Register.java	55
database/creazione_e_popolamento_database.sql	da 104 a 134

Remediation

Per le password inviate dagli utenti, sostituire l'utlizzo dell'algoritmo MD5 con alternative più sicure come i ben conosciuti SHA-256, SHA-512, SHA-3.
Quindi, per quanto riguarda la registrazione, gli utenti dovranno essere memorizzati nel database con le password crittograte. Al momento del login, sarà necessario crittograre la password inviata dall'utente (che tenta l'accesso) e confrontarla con quella presente nel database (in quanto il database contiene le password crittograte). Di conseguenza, sarebbe necessario modificare anche le password definite per gli utenti nello script di popolamento del database.

Riferimenti

1. CWE-1240: Use of a Cryptographic Primitive with a Risky Implementation