Bezpieczeństwo Systemów Informatycznych

Adam Baranowski, Kamil Tomaszewski, Andrzej Włodarczyk, Michał Worsowicz

Styczeń 2023

Spis treści

1 Wstęp

Celem projektu jest stworzenie aplikacji reprezentującej podatności jakie mogą pojawić się w przypadku wszelkiego rodzaju aplikacji webowych. Inspiracje dla projektu stanowiły takie przedsięwzięcia jak OWASP WebGoat oraz Damn Vulnerable Web Application (DVWA). Podatności przedstawione zostaną na prostej aplikacji typu TODO List.

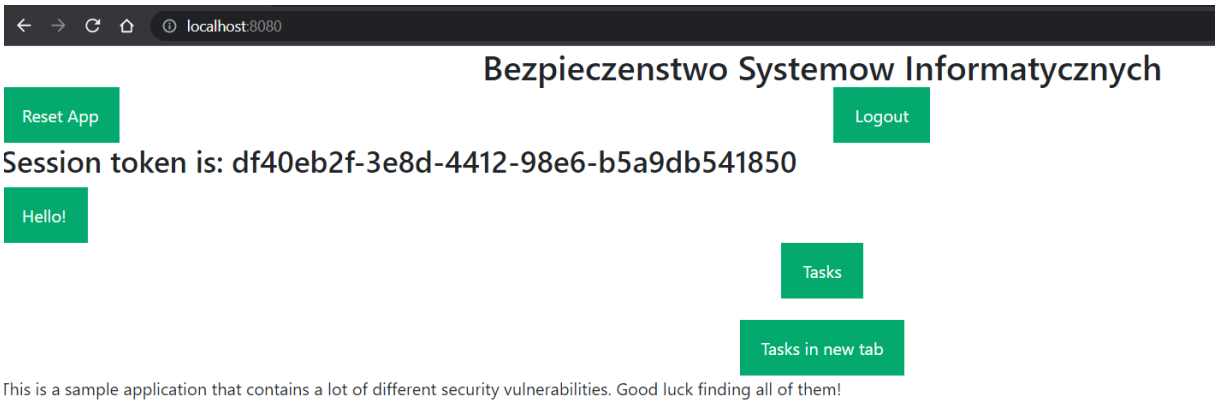
2 Aplikacja

Stworzony program to prosta aplikacja webowa przypominająca systemy klasycznych list zadań (tzw. TODO List) dla zalogowanych użytkowników. Aplikacja powstała przy użyciu Java 11 za pomocą frameworka Spring Boot, z wykorzystaniem bazy danych in memory H2. System uwierzytelniania użytkowników został napisany własnoręcznie i opiera sie na identyfikatorze sieci.

Podstawowe widoki w aplikacji:

1. Strona główna

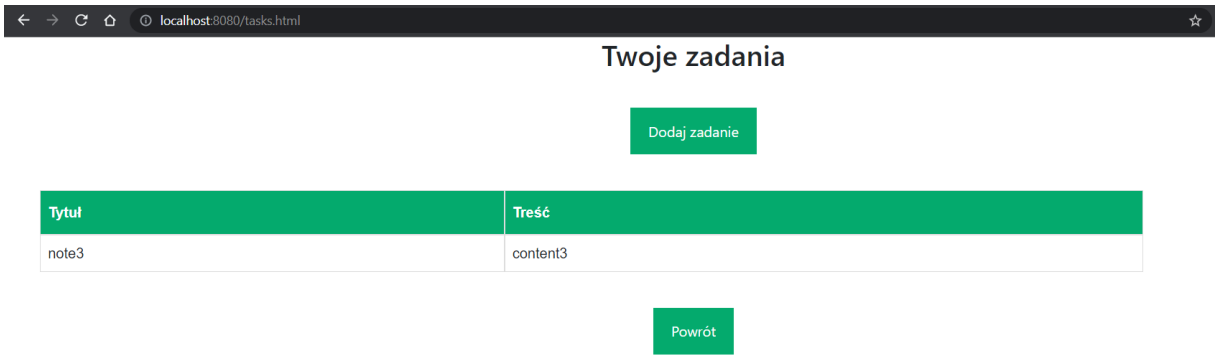
   Strona pozwalająca na zalogowanie się użytkownikowi, wyświetla również identyfikator obecnej sesji, a także pozwala wypisać przywitanie dla użytkownika oraz przejść do listy zadań



Rysunek 2.1: Główna strona aplikacji

2. Strona z zadaniami

   Strona ta pozwala zarówno przeglądać jak i dodawać nowe zadania do listy użytkownika



Rysunek 2.2: Podstrona aplikacji z zadaniami

3 Podatności

W tej sekcji omówione zostaną zaimplementowane podatności oraz sposoby w jakie można je wywołać w programie.

3.1 SQL Injection

Jedna z najczęstszych i najniebezpieczniejszych podatności w aplikacjach webowych, wynikająca z braku odpowiedniej walidacji parameterów przekazywanych przez użytkownika. Pozwala między innymi na ominięcie mechanizmu uwierzytelniania oraz wykonanie kodu w systemie bazy. Wykorzystywany głównie do odczytu lub nadpisywania danych, do których dostęp jest ograniczony.

W naszej aplikacji przykład ten zaobserwować można w kilku polach input, jeden z nich znajduje się chociażby w oknie logowania użytkownika. Przykładowo, podczas wprowadzenia nazwy użytkownika o treści:

```
'; UPDATE users SET password='haslo' WHERE username='admin';--
```

dokonamy zmiany hasła dla określonego użytkownika (w tym przypadku admina) na wybrane przez nas, co pozwoli następnie zalogować się na jego konto.

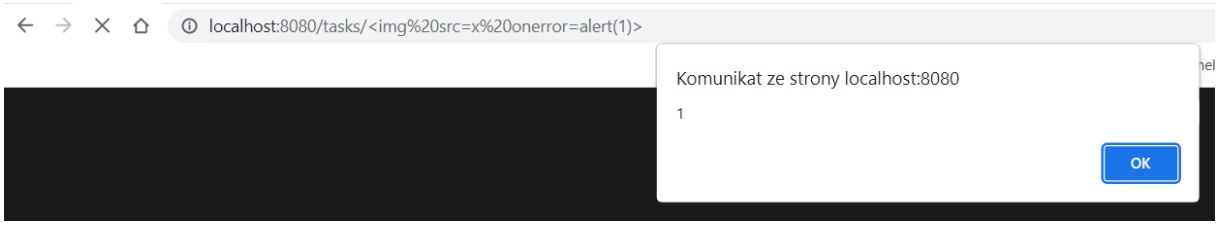
3.2 Cross-Site-Scripting

Cross-Site-Scripting (XSS) to atak na klienta korzystający z podatności webaplikacji, polegający na wstrzyknięciu do przeglądarki ofiary fragmentu kodu Javascript (lub innego języka skryptowego), który może być uruchomiony w przeglądarce. Dzięki temu, atakujący może dokonać dowolnej akcji w przeglądarce, co skutkować może wykradnięciem cookies, podmienieniem zawartości strony czy hostowaniem malware-u.

W naszej aplikacji wstrzyknięcie przykładowo pod adres *http://localhost:8080/tasks/* następującej treści:

```
http://localhost:8080/tasks/<img src=x onerror=alert(1)>
```

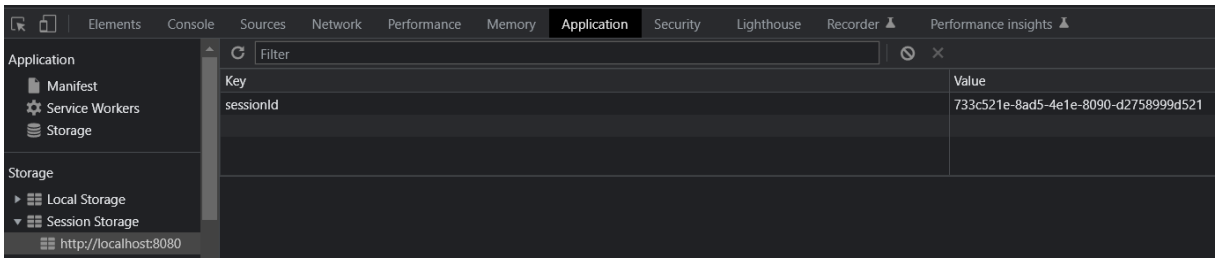
skutkuje w wyświetleniu alertu dla użytkownika. Jest to bezpieczny przykład tego jak wpłynąć można na przeglądarkę z której korzysta użytkownik naszej aplikacji webowej.



Rysunek 3.1: Przykład XSS

3.3 Session Hijacking

Session Hijacking to wszelkiego rodzaju ataki mające na celu uzyskanie dostępu do sesji użytkownika, przez przydzielony do niej identyfikator. Atak ten pozwala na korzystanie z sesji w ten sam sposób co zalogowany do niej użytkownik, dając możliwość na odczyt oraz edycję danych.



Rysunek 3.2: Przykładowa wartość dla Session Storage

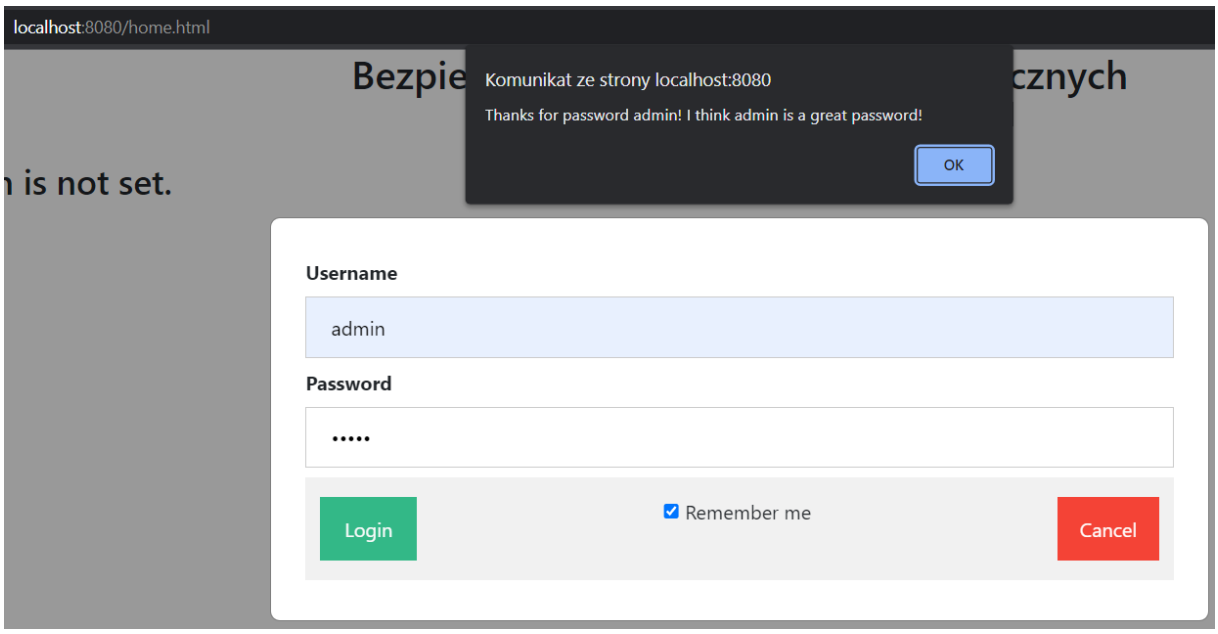
W przypadku naszej aplikacji wystąpić może **Session sniffing** poprzez odczyt identyfikatora sesji z głównej strony aplikacji webowej u zalogowanego użytkownika.

Następnie, ID może zostać dodane do Session Storage dla strony *http://localhost:8080/* za pomocą Browser Developer Tools (uruchamiane za pomocą F12 w przeglądarce). Session Storage znajduje sie w zakładce Application.

3.4 Reverse Tabnabbing

Reverse tabnabbing to atak polegający na nadpisaniu strony przez link do zewnętrznej strony znajdujący się na stronie, która zostanie nadpisana. Z punktu widzenia użytkownika, wciśnięty link otworzy nową kartę w przeglądarce na którą użytkownik zostanie automatycznie przełączony. Czego użytkownik nie widzi, to fakt, że w czasie uruchamiania nowej karty, zawartość poprzedniej strony zostaje nadpisana przez phishingową stronę, najczęściej wyglądającą tak jak ta poprzednio wyświetlana.

Nasza aplikacja pozwala na przetestowanie tej podatności za pomocą przycisku *Tasks in new tab*, do którego dostęp ma jedynie zalogowany użytkownik. Po jego wciśnięciu uruchamiana zostaje nowa karta z zadaniami, a więc przycisk spełnia swoje założenie, jednak dochodzi równocześnie do podmienienia strony, którą użytkownik właśnie opuścił. Zostaje ona zastąpiona identyczną stroną startową z wylogowanym użytkownikiem. W momencie, gdy użytkownik, przekonany że został wylogowany ze strony, spóbuje ponownie się zalogować, strona wyświetli mu odpowiedni komunikat sugerujący że jego login oraz hasło zostały wykradnięte.



Rysunek 3.3: Komunikat występujący dla tabnabbingu

3.5 Clickjacking

Clickjacking to typ ataku polegający na wciśnięciu przez użytkownika niewidocznego elementu strony, w celu wykonania określonej akcji (dokonania transakcji, pobrania maleware'u itp.). Jest on najczęściej dokonywany przez wyświetlenie niewidocznego elementu HTML w *iframe* na górnej warstwie użytkowanej strony. Widząc treść pod spodem, użytkownik wciska zawartość *iframe*.

Nasza aplikacja wykorzystuje clickjacking na spreparowanej stronie *home.html*, wykorzystywanej również do tabnabbingu. Tak jak w przypadku normalnej strony startowej znajduje się na niej przycisk *Hello!* pozwalający na wyświetlenie przywitania dla zalogowanego użytkownika. W przypadku tej podstrony, nad przyciskiem znajduje się element *iframe* z video z platformy YouTube, który to w momencie wciśnięcia odtworzy wideo (z punktu widzenia użytkownika "zauważalny"będzie jedynie dźwięk).

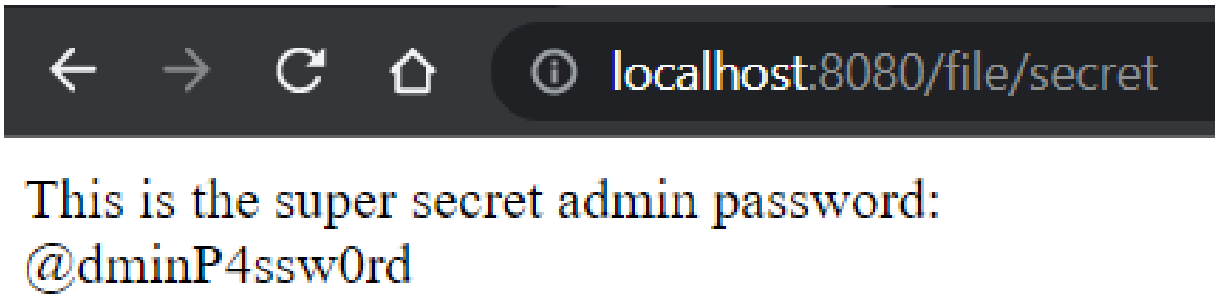
Rysunek 3.4: Nad przyciskiem znajduje się niewidoczne wideo Ricka Astley'a - Never Gonna Give You Up

3.6 Local File Inclusion

Local File Inclusion (LFI) to technika ataku polegająca na wymuszeniu przez atakującego wyświetlenia albo uruchomienia plików na serwerze webowym. Taki atak może posłużyć do zdobycia wrażliwy danych, a także przeprowadzenia dalszych ataków XSS.

W przypadku naszej aplikacji, plik *about.txt* zostaje zaczytywany na stronie głównej, w celu wyświetlenia informacji o stronie, za pomocą endpointu *\file\about*. Enpoint ten służy do odczytu pliku z zadanej ścieżki, pozwalając równocześnie na dostęp do wszystkich pozostałych plików znajdujących się w tym folderze. W celu zaprezentowania problemu, nasza aplikacja

posiada w tym folderze również plik *secret.txt* zawierający hasło admina, którego treść może zostać w prosty sposób wyświetlona, za pomocą endpointu *\file\secret*



Rysunek 3.5: Zdobyte za pomocą LFI hasło