# Lern- und Arbeitsauftrag *LA\_183\_07\_BrokenAccessControl.docx*

|  |  |
| --- | --- |
| Titel | Broken Access Control |
| Modul | 183 Informatiker/in EFZ |
| Autor / Version | Adrian Schmid / V1.1 |
| Hilfsmittel: | PR\_183\_Verschlüsselung  PR\_183\_JWT  InsecureApp  <https://jwt.io/> |
| Nachweis |  |
| Sozialform | Einzelarbeit / Partnerarbeit |
| Leistungsziele | 3.5, 3.6, 3.8 |

## Ausgangslage

Die News in der Insecure-App sollten nur im geschützten Bereich aufrufbar sein. Das heisst, ohne Login sollten die News nicht lesbar sein und es sollte nicht möglich sein, News zu erstellen.

Ein Bild, das Luftbild, Design enthält.

Automatisch generierte Beschreibung mit geringer Zuverlässigkeit

Quelle: <https://pixabay.com/vectors/attack-unsecured-laptop-hacker-7647136/>, 03.10.2023

## Aufgabenstellung

Ziel dieser Aufgabe ist es zu analysieren, wie das Login der Insecure-App aktuell implementiert ist und es so zu verbessern, dass es nachher sicher ist.

### Teilaufgabe 1: Analyse der Login Funktionalität

Starten Sie die Entwicklertools ihres Browsers und öffnen Sie den «Netzwerk» Tab, um HTTP-Requests nachverfolgen zu können. Welcher Request wird beim Login Vorgang abgesetzt? Was wird an den Server übermittelt, welche Antwort sendet der Server bei einem Erfolgreichen Login zurück?

|  |  |
| --- | --- |
| **Anfrage an Server** | **Antwort vom Server** |
| Beim Login-Vorgang wird eine POST-Anfrage an den Server gesendet, die Benutzername und Passwort enthält. | Bei erfolgreichem Login sendet der Server eine Antwort zurück, die eine Session-ID oder ein Token für die Authentifizierung des Benutzers beinhaltet. |

Analysieren Sie den Code des JavaScript Clients. Wie merkt sich der Client, ob jemand eingeloggt ist / wer eingeloggt ist? Können Sie dem Client ein Login vortäuschen, ohne die Login Funktion zu verwenden? Falls ja, beschreiben Sie wie das geht:

### Teilaufgabe 2: Analyse der API

Starten Sie das Backend der Insecure-App und öffnen Sie wahlweise Swagger oder ein REST-Client Ihrer Wahl (z.B. Postman). Probieren Sie aus was Sie bei den folgenden Endpoints ohne Login Informationen machen können. Dokumentieren Sie wie die folgenden Endpoints ansprechen können und was das Resultat ist.

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Methode** | **Was ist ohne Login möglich?** |
| /api/News | GET | Ohne Login können keine News abgerufen werden. Der Zugriff ist beschränkt. |
| /api/News | POST | Ohne Login können keine neuen News erstellt werden. |
| /api/News/{id} | GET | Einzelne Newsbeiträge können ausgelesen werden. |
| /api/News/{id} | PATCH | Alle Newsbeiträge können bearbeitet werden. |
| /api/News/{id} | DELETE | Newsbeiträge können gelöscht werden. |
| /api/User/password-update | PATCH | Passwort-Update ist ohne authentifizierten Login nicht durchführbar. |

### Teilaufgabe 3: Verschlüsselungsverfahren

Infomieren Sie sich mithilfe der Präsentation «PR\_183\_Verschlüsselung» zum Thema Verschlüsselungsverfahren.

Suchen Sie im Internet und finden Sie heraus, welche Verschlüsselungsverfahren (Shared Secret, PKI oder Hashing) bei den folgenden Verfahren eingesetzt werden.

|  |  |
| --- | --- |
| **Beispiel** | **Eingesetzte Verfahren** |
| JSON Web Tokens (verwendet HMAC) | Verwendet HMAC (Hash-based Message Authentication Code) für die Verschlüsselung. |
| TLS / SSH | Einsatz von Public Key Infrastruktur (PKI) für die Verschlüsselung |
| Sicheres Speichern von Passwörtern | Typischerweise wird Hashing verwendet, um Passwörter sicher zu speichern. |

### Teilaufgabe 4: JWT-Authentifizierung serverseitig einbauen

Um diese Sicherheitslücken zu schliessen, braucht die Insecure-App eine sichere Authentifizierungslösung. In den Modulen 294 und 295 haben Sie zu diesem Zweck bereits das JSON Web Token (JWT) kennengelernt.

Lesen Sie die Präsentation «PR\_183\_JWT» um das Thema wieder aufzufrischen.

Damit die Insecure App ein bisschen sicherer wird, soll die JWT-Authentifizierung mit folgenden Parametern implementiert werden:

* Im Rahmen dieses Arbeitsauftrags arbeiten wir ohne Refresh Tokens. Das heisst sie müssen eine lange «Lifetime» für die Tokens wählen, damit Ihr Token nicht ständig ungültig wird.
* Die JWT sollen mit HMAC SHA 512 signiert werden
* Signature, Lifetime, Issuer und Audience sollen verifiziert werden. Es sind bereits Werte für Issuer, Audience und Key in der Konfiguration hinterlegt, welche verwendet werden können.
* Das JWT-Token sollte folgende Claims beinhalten:
  + JwtRegisteredClaimNames.Jti: Eine zufällig generierte UUID
  + JwtRegisteredClaimNames.NameId: Die ID (Primary Key) des Benutzers
  + JwtRegisteredClaimNames.UniqueName: Der Benutzername
  + ClaimTypes.Role: «admin» oder «user» Abhängig vom boolean «IsAdmin» des eingeloggten Benutzers
* Alle Endpoints, welche nur von eingeloggten Benutzern verwendet werden sollen (Siehe Teilaufgabe 2), müssen geschützt werden.

Um den Schlüssel aus der Konfiguration auszulesen können Sie den Code aus den zusätzlichen Angaben zum Auftrag verwenden.

Prüfen Sie mithilfe eines REST API Tools (z.B. Swagger oder Postman) ob Ihr Login Endpoint ein JWT-Token zurückgibt, und schauen Sie mithilfe von <https://jwt.io/> ob die Claims korrekt vorhanden sind.

Stellen Sie sicher, dass alle geschützten Endpoints (Siehe Teilaufgabe 2) nicht mehr ohne JWT-Token aufrufbar sind. Prüfen Sie ob die Endpoints mit dem JWT-Token funktionieren. Verwenden Sie dazu folgenden HTTP-Header:

Authorization: Bearer {JWT-Token}

### Teilaufgabe 5: JWT-Authentifizierung im Client einbauen

Die Insecure-App funktioniert durch den Umbau aus Teilaufgabe 4 nicht mehr richtig. Damit Sie wieder funktioniert, muss das JWT-Token, welches Sie in der Antwort des Login-Requests erhalten, abgespeichert werden (zum Beispiel im LocalStorage). Dieses Token muss dann bei jedem HTTP-Request, welcher nur mit Login funktioniert (alle ausser der Request fürs Login selbst) als Authorization-Header mitgegeben werden.

headers: {

'Authorization': 'Bearer {JWT-TOKEN}'

}

Die Logout Funktion und die Funktion zum Prüfen, ob ein Benutzer eingeloggt ist, müssen gegebenenfalls angepasst werden. Die Logout Funktion soll das gespeicherte Token löschen. Die Funktionen, welche auslesen welcher Benutzer eingeloggt ist, ob der Benutzer ein Admin ist, können diese Informationen aus dem JWT-Token auslesen. Sie können dabei die Funktion zum Parsen des JWT-Tokens welche in den zusätzlichen Angaben zum Auftrag ist verwenden.

Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Software, Multimedia-Software, Screenshot enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

## Gütekriterien

Der Lern- und Arbeitsauftrag ist erfüllt, wenn …

* Sie die Schwachstellen im Authentifizierungssystem der InsecureApp erkannt und ausgenutzt haben
* Sie sich über Verschlüsselungsverfahren informiert haben
* Sie JWT als neues Authentifizierungsverfahren in die InsecureApp eingebaut haben

## Zusätzliche Angaben zum Auftrag

**Auslesen des JWT-Keys aus der Konfiguration:**

**Im Program.cs**

String base64Key = builder.Configuration["Jwt:Key"]!;

new SymmetricSecurityKey(Convert.FromBase64String(base64Key));

**Im Controller:**

**Mit Dependency Injection die Konfiguration Injecten:**

private readonly IConfiguration \_configuration;

public LoginController(IConfiguration configuration)

{

    \_configuration = configuration;

}

**Dann können Sie den Key wie folgt auslesen:**

string base64Key = \_configuration.GetSection("Jwt:Key").Value!;

new SymmetricSecurityKey(Convert.FromBase64String(base64Key));

**Parsen des JWT-Tokens in JavaScript:**

function parseJwt(token) {

var base64Url = token.split('.')[1];

var base64 = base64Url.replace(/-/g, '+').replace(/\_/g, '/');

var jsonPayload = decodeURIComponent(window.atob(base64).split('').map(function (c) {

return '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2);

}).join(''));

return JSON.parse(jsonPayload);

}

// Beispielverwendung

var token = localStorage.getItem('jwtToken');

var tokenData = parseJwt(token);

console.log(tokenData); // prüfen Sie welche Daten im Token sind

console.log(tokenData.role); // sollte "admin" oder "user" sein

## Mögliche Erweiterungsaufträge

* Bauen Sie die JWT-Authentifizierung bei Swagger ein.