مدرس: **احسان خامس‌پناه**
طراحان: **امین عارف‌زاده، امیرحسین احمدی، شهریار سلطان‌پور**

مهلت تحویل: جمعه ۱۹ اردیبهشت ۱۳۹۹، ساعت ۲۳:۵۵

لقمه!

مقدمه

هدف از این پروژه، آشنایی با روش‌های پیاده‌سازی احراز هویت[1] و کسب اجازه[2] و اطمینان از وجود برخی از پارامترهای امنیتی در برنامه می‌باشد.

فاز هفتم پروژه

راه‌اندازی فرآیند ثبت‌نام کاربر

ابتدا لازم است که یک API جدید بنویسید که در آن پس از دریافت فیلدهای مربوط ایجاد حساب کاربری جدید، یک کاربر جدید با مشخصات داده شده در پایگاه‌داده ایجاد کنید. فیلدهای مورد نیاز برای این API برای ایجاد یک حساب کاربری جدید عبارت‌اند از:

- **نام**
- **نام خانوادگی**
- **کلمه عبور**
- **آدرس ایمیل**

سپس باید صفحه‌ی ثبت‌نام در بخش سمت‌کاربری خود را به این API جدید وصل کنید. (شما ظاهر این صفحه را در پروژه‌ی شماره‌ی ۴ طراحی کرده‌اید.)

نکات:

۱. قبل از ارسال اطلاعات، ورودی خود را در سمت کاربر اعتبارسنجی کنید.
۲. کلمه‌ی عبور را به هیچ وجه به شکل plain text در پایگاه‌داده ذخیره نکنید و از عبارت hash گرفته‌شده‌ی آن استفاده کنید.
۳. انجام اعتبارسنجی‌هایی نظیر تکراری نبودن ایمیل جدید با کاربران ثبت شده در پایگاه‌داده الزامیست.

احراز هویت به‌کمک JWT[3]

در این بخش می‌خواهیم به کمک JWT، که یک روش stateless است(نیازی به حافظه در سمت سرور ندارد)، احراز هویت را به برنامه اضافه‌کنیم.

---

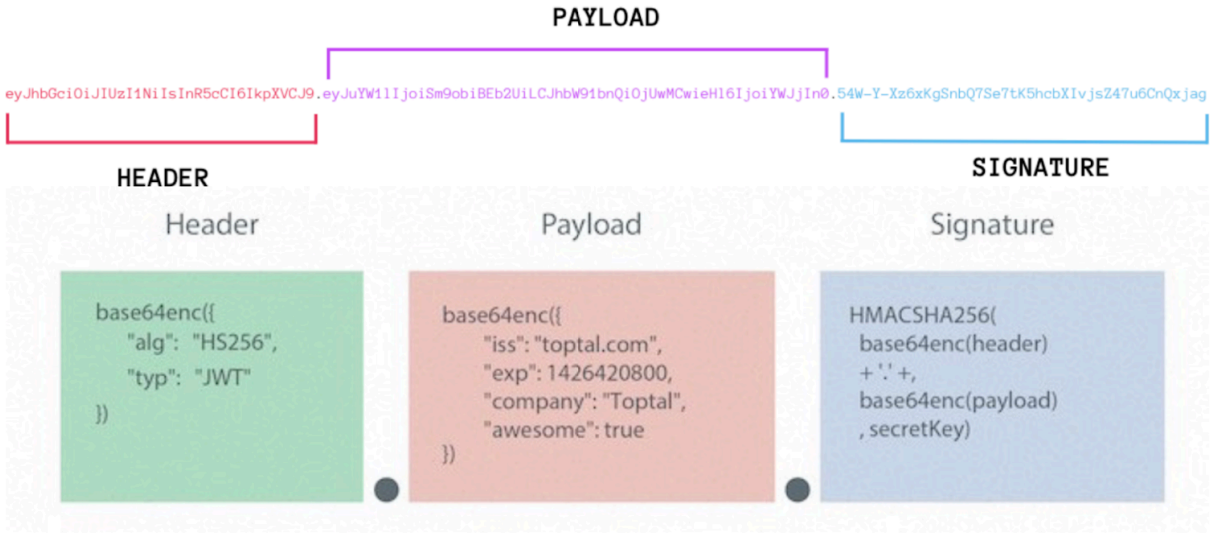
[1] Authentication

[2] Authorization

[3] Json Web Token

استاندارد JWT برای اکثر زبان‌ها پیاده‌سازی شده و برای Java نیز چندین پیاده‌سازی برای آن وجود دارد.
هر JWT شامل سه بخش است:
۱. Header:شامل اطلاعات الگوریتم مورد استفاده برای signature و نوع token است.
۲. Payload:شامل claimهای JWT است.در این پروژه استفاده ازclaimهای ثبت‌شده‌ی **iss**، **iat** و **exp** (با زمان انقضای یک روز) اجباری و استفاده از هر claim استاندارد یا غیراستاندارد دیگر (مثلاً یکclaimبه نام userId برای هویت کاربر) آزاد است.
۳. Signature:برای اطمینان از صحت اعتبار JWT است و به‌کمک الگوریتم‌هایHMAC یا RSA محاسبه می‌شود.در این تمرین برای راحتی از **الگوریتم HMACSHA256** همراه با **کلید loghme** استفاده کنید.در این حالت signature به‌صورت زیر تولید می‌شود:

HMACSHA256(base64UrlEncode(header) + ‘.’ + base64UrlEncode(payload), “loghme”)



از آنجا که signature تنها توسط سرور قابل تولید است (چون فقط سرور key secret را دارد)، پس میتوان اعتبار JWT را بر اساس آن مشخص کرد.برای فهم بیشتر این موضوع می‌توانیدکمی بیشتر در مورد امضای دیجیتال جستجو کنید.
کل JWT به‌صورت زیر تولید می‌شود:

base64UrlEncode(header) + ‘.’ + base64UrlEncode(payload) + ‘.’ + signature

سایت jwt.io نیز منبع بسیار خوبی برای یادگیری JWT و ساختن و درستی‌سنجی آن است.

راه‌انـدازی فـرایند ورود کاربـر

احراز هویت را در این فاز می‌بایست به دو صورت داخلی[4] و به وسیله Google Authentication انجام دهید.

احراز هویت داخلی

یک API برای ورود کاربر پیاده‌سازی کنید.این API نام‌کاربری وکلمه‌ی عبور کاربر را دریافت می‌کند و در صورت درستی، برای او JWT صادر می‌کند.در غیر این صورت، پیغام مناسبی همراه با HTTP code شماره‌ی ۴۰۳ به کاربر می‌فرستد.

[4] Internal

در صورت درستی باید در بخش سمت کاربر، این JWT را نگه‌داری کنید و در درخواستهای بعدی در header authorization بفرستید.

استفاده از Google Authentication

حال می‌بایست از Google API برای فرایند احراز هویت استفاده کنید. مراحل کامل استفاده از Google authentication در این لینک آمده است.

ابتدا دکمه استفاده از Google برای احراز هویت را در صفحه ورود قرار دهید. می‌بایست برنامه خود را به گوگل معرفی کنید تا بتوانید با استفاده از آن به برنامه ورود کنید. گوگل از شما URL می‌خواهد و به دلیل اینکه برنامه شما به صورت لوکال اجرا می‌شود، IP آن 127.0.0.1 است. ولی گوگل آدرس‌های لوکال را قبول نمی‌کند و می‌بایست از روش‌های HTTP Tunneling به وسیله ابزارهایی مانند ngrok استفاده کنید. برای معرفی برنامه به گوگل و همچنین فرایند احراز هویت از همین لینک استفاده کنید.

در صورتی که بعد از دریافت اطلاعات از Google، ایمیل فرد در بین افراد ثبت‌نامی نبود می‌بایست او را به صفحه ثبت‌نام ارجاع دهید. در صورتی که ایمیل فرد در پایگاه‌داده وجود داشت همانند احراز هویت داخلی برای فرد JWT بسازید و از این به بعد فرد به وسیله JWT کار می‌کند.

استفاده از فیلتر برای درستی‌سنجی JWT

به‌جای اینکه در ابتدای هر servlet به بررسی این موضوع بپردازید، باید از فیلتر، که یکی از API های بسیار کاربردی Java EE است، استفاده کنید. یک فیلتر بسازید و در آن درستی JWT دریافت شده را بررسی‌کنید. در این شرایط سه حالت قابل بررسی است:

- در صورت وجود مشکل در ساختار JWT، پاسخی با HTTP code شماره‌ی ۴۰۳ را برای کاربر ارسال کنید.
- در صورت درستی ساختار JWT، کاربر با مشخصات داخل payload را از پایگاه‌داده بگیرید و به عنوان attribute برای request معین کنید.
- در صورتی‌که کاربر JWT را ارسال نکرده بود (بدیهی است که در این حالت کاربر احراز هویت نشده است.) و قصد دسترسی به صفحاتی‌که نیاز به احراز هویت کاربر است را داشت، پاسخی با HTTP code شماره‌ی ۴۰۱ به او ارسال کنید. بهتر است این فیلتر را بر سر همه‌ی API های موجود در سیستم به جز API مربوط به ثبت‌نام و ورود بگذارید.

نیازمندی های در سمت رابط‌کاربری

دقت کنید که در سمت کاربر دو حالت برای احراز هویت وجود دارد:

- یا کاربر به سیستم وارد نشده و JWT ندارد (یا JWT غیر مجاز دارد) که در این حالت، تنها می‌تواند صفحات ورود یا ثبت‌نام را مشاهده کند. (در صورت واردکردن آدرس سایر صفحات در url، کاربر را به صفحه‌ی ورود هدایت کنید.)
- یا کاربر به سیستم وارد شده که در این حالت می‌تواند به تمامی صفحات به جز ورود و ثبت‌نام دسترسی پیدا کند. (به‌جای صفحات ورود و ثبت‌نام، کاربر را به صفحه‌ی اصلی هدایت‌کنید.) در صورتی‌که هویت‌کاربر احراز شده باشد، با بازخوانی[5] صفحه، همچنان کاربر باید در برنامه بماند. برای نگهداری JWT نیز می‌توانید از حافظه‌ی محلی مرورگر استفاده‌کنید و در هر بار بالا آمدن صفحه آن را از حافظه‌ی محلی مرورگر بخوانید تا بفهمید کاربر در برنامه احراز هویت شده یا نه. سپس برای هر درخواست آن را در authorization header قرار دهید.

عملیاتی را نیز برای خروج کاربر[6] از حسابش مشخص کنید که با توجه به stateless بودن JWT، تنها کافی است توکن (و سایر داده‌های ذخیره شده در Local Storage) را در سمت کاربر پاک کنید.

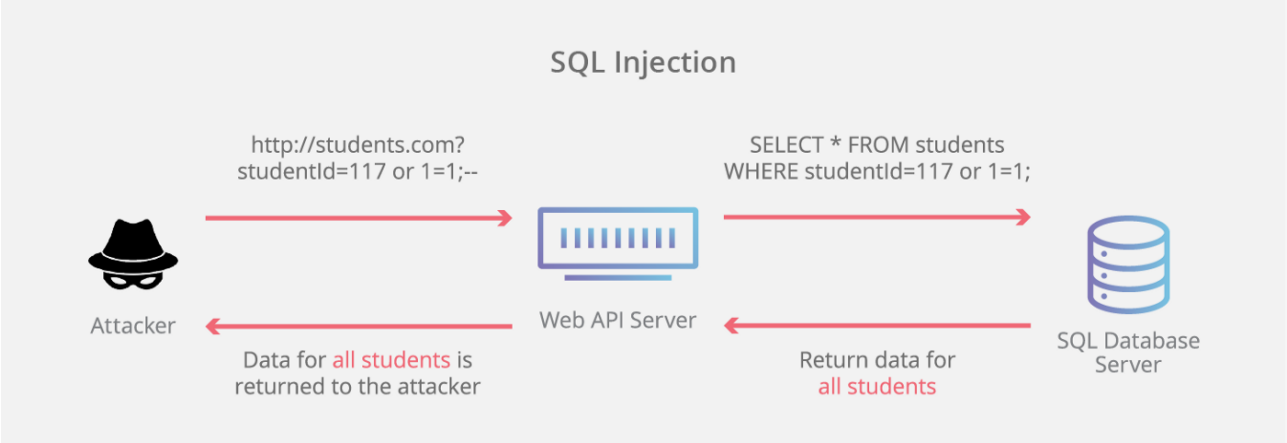
---

[5] Refresh

[6] Logout

SQL Injection مسئله‌ی امنیتی؛ مقاومت در برابر حملات

حمله‌ی Injection به حمله‌ای گفته می شود که در آن متخاصم در داده‌های ارسالی خود به سیستم، از دستورات یا کوئری‌هایی استفاده می‌کند که در صورت اجرا شدن در سرور، می‌توانند مشکل‌زا باشند. حمله‌ی SQL Injection نیز نوعی از حمله‌ی Injection است که از زبان SQL در داده‌های کاربر استفاده می‌شود. به عنوان مثال می‌توانید به سناریوی زیر دقت کنید که در آن متخاصم به اطلاعات تمامی دانش‌آموزان یک سیستم دسترسی پیدا می‌کند.



برای این بخش، شما باید APIهای مربوط به احراز هویت کاربر را طوری طراحی کنید که در برابر حملات SQL Injection مقاوم باشند. همچنین باید با بررسی مجدد تمام APIهای خود اطمینان حاصل کنید که آنها نیز دچار چنین مخاطراتی نمی‌شوند.

نکات پایانی

- کافی است که یکی از اعضای گروه در دو خط آدرس مخزن‌های پروژه (back-end, front-end) را در گیت‌لب و در خط‌های بعدی Hash مربوط به آخرین کامیت‌های این مخزن‌ها را در سایت درس آپلود کند. در هنگام تحویل، پروژه روی این کامیت‌ها مورد ارزیابی قرار می‌گیرد.
- ساختار صحیح و تمیزی کد برنامه بخشی از نمره‌ی این فاز پروژه‌ی شما خواهد بود. بنابراین در طراحی ساختار برنامه دقت به خرج دهید.
- متاسفانه استفاده از API توسعه‌دهندگان گوگل برای ایرانیان امکان‌پذیر نیست و برای دسترسی به آن می‌توانید از شکن یا FoD استفاده کنید.
- این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، ۵۰٪ نمره‌ی کل پروژه‌ها از گروه متقلب و تقلب‌دهنده کسر خواهد شد و در صورت تکرار، این روند ادامه خواهد داشت.
- سوالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن‌ها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این فاز پروژه ارتباط برقرار کنید. توجه داشته باشید که دیگر شبکه‌های اجتماعی مانند تلگرام راه ارتباطی رسمی با دستیاران آموزشی نیست و دستیاران آموزشی موظف به پاسخگویی در محیط‌های غیررسمی نیستند.
- ایمیل طراحان پروژه:
  - amirhmi1377@gmail.com
  - aminarefzadeh1376@gmail.com
  - Shahryar.Soltanpour@gmail.com