مهندسی اینترنت

تمرین شماره‌ی ۶

مدرس: **دکتر خامس‌پناه**

طراح: علیرضا اسمعیل زاده

مهلت تحویل: ۱ خرداد ساعت ۲۳:۵۹

MizDooni

مقدمه

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

[1] Authentication

[2] Authorization

فرآیند ثبت‌نام کاربران

در این قسمت قرار است فرآیند ثبت‌نام کاربران را طراحی کنید که در آن پس از وارد کردن فیلدهای لازم، یک کاربر جدید با مشخصات داده شده به پایگاه‌داده اضافه شود. در این صفحه فیلدهای نام، نام کاربری، ایمیل و کلمه‌ی عبور از کاربر دریافت می‌شود.

نکات:

- قبل از ارسال اطلاعات به سرور، فرمت ورودی‌ها را در سمت کاربر اعتبارسنجی کنید.
- کلمه‌ی عبور را به هیچ وجه به صورت plain text در پایگاه‌داده ذخیره نکنید، بلکه از hash آن استفاده کنید.
- انجام اعتبارسنجی‌هایی نظیر تکراری نبودن نام کاربری و ایمیل در سمت سرور الزامیست. اگر نام کاربری یا ایمیل تکراری باشد، باید خطای مناسب به کاربر نمایش داده شود.

مانند فازهای قبلی، همچنان دریافت لیست کاربران در هنگام اجرا شدن پروژه و اضافه کردن آن‌ها به پایگاه‌داده را خواهید داشت و فرآیند ثبت‌نام، صرفا برای کاربران جدید است. در این تمرین، کلمه‌ی عبور تمام کاربران را به صورت hash شده ذخیره کنید.

http://91.107.137.117:55/users

احراز هویت به کمک JWT

در این بخش به کمک JWT که یک روش بدون حالت است (بدون نیاز به حافظه در سمت سرور)، احراز هویت را به برنامه‌ی خود اضافه می‌کنید. استاندارد JWT در اکثر زبان‌های برنامه‌سازی پیاده‌سازی شده و برای جاوا 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 claim های JWT است. در این پروژه استفاده از claim های iss، iat و exp (با زمان انقضای یک روز) اجباری است. در کنار آن‌ها می‌توانید از claim های استاندارد یا غیراستاندارد دیگر نیز استفاده کنید (مثلاً یک claim به نام userEmail برای هویت کاربر).

۳. Signature: این قسمت شامل امضای دیجیتال سرور است که برای اطمینان از صحت JWT اضافه می‌شود. این امضا معمولا به کمک الگوریتم‌های HMAC و RSA محاسبه می‌شود. در این تمرین برای راحتی از الگوریتم HMACSHA256 همراه با کلید mizdooni2024 استفاده کنید. در این حالت امضا به‌صورت زیر تولید می‌شود:

HMACSHA256(base64UrlEncode(header) + ‘.’ + base64UrlEncode(payload), “mizdooni2024”)

به دلیل اینکه signature تنها توسط سرور قابل تولید است (چون فقط سرور کلید را دارد)، پس میتوان صحت JWT را بر اساس آن سنجید. در صورت علاقه می‌توانید کمی در مورد امضای دیجیتال تحقیق کنید.

احراز هویت با مکانیزم OAuth

در این بخش، شما باید امکان ورود کاربر با استفاده از سرویس Google را پیاده سازی نمایید.

برای این کار لازم است ابتدا یک OAuth Application در Google ایجاد کنید. نحوه ایجاد اپلیکیشن در این لینک موجود است.

سپس لینکی را که کاربر باید بر روی آن کلیک کند تا به Google ریدایرکت شود و به اپلیکیشن شما دسترسی بدهد، در صفحه لاگین و ثبت‌نام قرار دهید. این لینک به شکل زیر خواهد بود:

https://accounts.google.com/o/oauth2/auth?client_id=CLIENT_ID&response_type=code&scop=SCOP

که باید به جای CLIENT_ID از مقداری که هنگام ایجاد اپلیکیشن در Google تولید شده است، استفاده کنید.

سپس لازم است یک صفحه Callback در بخش فرانت‌اند و یک اندپوینت Callback در بخش بک‌اند پروژه خود پیاده‌سازی نمایید.

صفحه Callback فرانت‌اند، یک صفحه خالی است که کاربر پس از دادن دسترسی به اپلیکیشن شما، با پارامتر "code" به آن ریدایرکت می‌شود. این کد توسط Google ایجاد شده و با ریدایرکت کردن کاربر به صفحه Callback، در واقع آن را در اختیار اپلیکیشن شما قرار می‌دهد.

صفحه Callback لازم است بلافاصله پس از load شدن، اندپوینت Callback بک‌اند را با همان کد داده شده در پارامتر، فراخوانی کند و در پاسخ، توکن JWT مربوط به کاربر را از بک‌اند دریافت کرده و کاربر را به صفحه خانه هدایت کند.

در بدنه Callback پیاده سازی شده در بک‌اند، لازم است ابتدا یک درخواست به Google برای دریافت Access Token کاربر ارسال شود. این درخواست باید شامل code داده شده از بخش فرانت‌اند، client_id و client_secret باشد.

این درخواست باید با متد POST و به آدرس زیر ارسال شود (توجه کنید برای آن که پاسخ به صورت json برگردانده شود، هدر Accept را برابر application/json قرار دهید):

https://oauth2.googleapis.com/token?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&code=CODE

client_id و client_secret هنگام ایجاد اپلیکیشن در Google به شما داده می‌شود. با وجود این 2 پارامتر در درخواست ارسالی، Google اپلیکیشن شما را احراز هویت کرده و با توجه به code داده شده، اجازه دریافت Access Token کاربر را به شما می‌دهد و آن را در پاسخ با کلید access_token، برمی‌گرداند. از این پس با قرار دادن این توکن در درخواست‌های خود به Google API، می‌توانید اطلاعاتی را که کاربر اجازه دسترسی آن‌ها را به شما داده است، دریافت کنید.

برای این پروژه فقط لازم است اطلاعات کاربر را از اندپوینت زیر دریافت کنید:

https://www.googleapis.com/oauth2/v2/userinfo

از این لینک می‌توانید سند مربوط به این API را مشاهده کنید.

در درخواست به API باید Access Token دریافتی را در هدر Authorization به صورت زیر قرار دهید:

Authorization: “token ACCESS_TOKEN”

در میان اطلاعات کاربر، برای ما تنها ایمیل (email) و نام کاربری (username) اهمیت دارد.

پس از دریافت اطلاعات کاربر، یک کاربر جدید در دیتابیس خود ایجاد کنید یا اگر ایمیل مورد نظر موجود بود، اطلاعات کاربر را **به‌روز** کنید. سپس برای کاربر مورد نظر یک توکن JWT ایجاد کرده و به عنوان خروجی برگردانید تا فرایند احراز هویت کاربر تمام شود.

برای راهنمایی بیشتر درمورد Google OAuth Authentication می‌توانید این لینک را مطالعه نمایید.

برای کاربران **جدید** که با این روش احراز هویت می‌شوند، رمز عبور را در دیتابیس null در نظر بگیرید.

اعتبارسنجی کاربر

پس از موفقیت آمیز بودن فرآیند ورود، یک JWT برای کاربر صادر می‌شود که کاربر برای درخواست‌های بعدی خود این JWT را به سرور می‌فرستد. در نتیجه باید در درخواست‌های بعدی بر اساس JWT فرستاده شده، اعتبارسنجی کاربر را انجام دهید.

برای این کار به‌جای اینکه در ابتدای هر servlet به بررسی این موضوع بپردازید، از فیلتر که یکی از پرکاربردترین امکانات JavaEE است، استفاده کنید. یک فیلتر بسازید و در آن درستی JWT دریافت شده را بررسی‌کنید. در پیاده‌سازی این فیلتر باید سه حالت زیر را در نظر بگیرید:

- در صورت درست بودن JWT، اطلاعات کاربر را از پایگاه‌داده بگیرید و به عنوان یک attribute برای request تعیین کنید تا در ادامه به راحتی به این اطلاعات دسترسی داشته باشید.
- در صورت وجود مشکل در JWT، پاسخی با status code شماره‌ی 401 را برای کاربر ارسال کنید.
- در صورتی‌که کاربر JWT را ارسال نکرده بود و قصد دسترسی به صفحاتی را که نیازمند احراز هویت کاربر هستند داشت، پاسخی با status code شماره‌ی 401 به او ارسال کنید.

پس از پیاده سازی این فیلتر، آن را بر روی همه‌ی API های موجود در سیستم که نیاز به احراز هویت دارند، بگذارید. دقت کنید که API های ثبت‌نام و ورود و Callback نیازی به احراز هویت ندارند.

نیازمندی های سمت رابط کاربری

در سمت کاربر دو حالت برای احراز هویت وجود دارد:

- کاربر وارد سیستم نشده و JWT ندارد که در این حالت **تنها** می‌تواند صفحات ورود، ثبت‌نام و Callback را مشاهده‌کند (در صورت واردکردن آدرس سایر صفحات، کاربر را به صفحه‌ی ورود هدایت کنید).
- کاربر به سیستم وارد شده که در این حالت می‌تواند به تمامی صفحات به جز ورود، ثبت‌نام و Callback دسترسی پیدا کند (در صورت واردکردن آدرس این صفحات، کاربر را به صفحه‌ی اصلی هدایت کنید).

در صورتی که کاربر وارد برنامه شده باشد، با بازنشانی[3] صفحه همچنان باید اطلاعات مربوط به احراز هویت او ثابت باقی بماند. برای این کار می‌توانید JWT را در حافظه‌ی محلی مرورگر نگه‌داری کنید و در هر بار بازخوانی صفحه آن را از حافظه‌ی محلی مرورگر بخوانید.

امکان خروج[4] کاربر از حسابش را نیز اضافه کنید که با توجه به بدون حالت بودن JWT، تنها کافی است این توکن را در سمت‌کاربر پاک کنید.

هدف اصلی این پروژه یادگیری فرآیندهای مربوط به احراز هویت کاربران است. به همین دلیل مسائل مربوط به طراحی و زیبایی صفحات اهمیتی ندارد و می‌توانید طراحی دلخواه خود را داشته باشید.

---

[3] Refresh

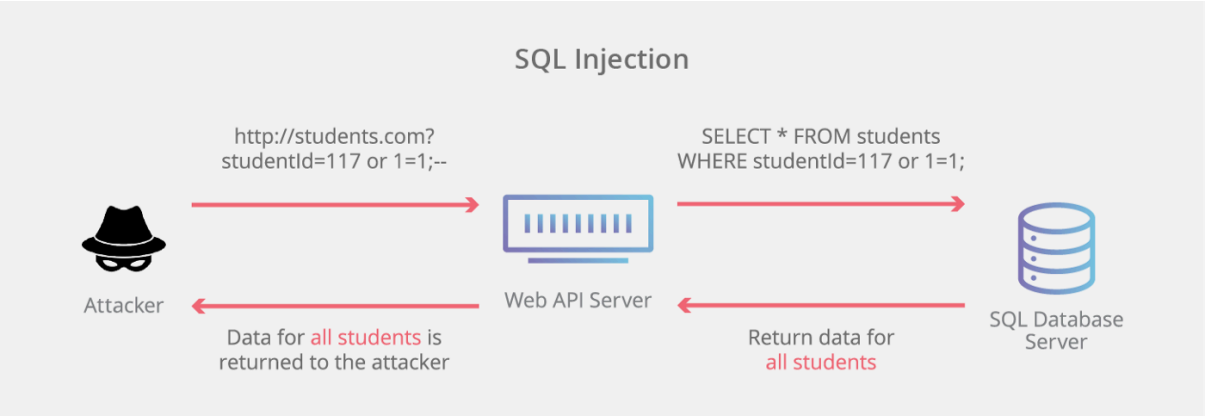
[4] Logout

ایمن سازی در برابر حملات SQL Injection

در حملات Injection فرد حمله کننده در داده‌های ارسالی خود از دستورات یا کوئری‌هایی استفاده می‌کند که در صورت اجرا شدن در سرور، می‌توانند مشکل‌زا باشد. حمله‌ی SQL Injection نیز نوعی از حمله‌ی Injection است که در آن فرد حمله‌کننده کوئری‌های SQL را در داده‌های ارسالی خود به سرور می‌فرستد. به عنوان مثال می‌توانید به سناریوی عکس (۱) دقت کنید که در آن فرد حمله کننده با استفاده از حمله‌ی SQL Injection توانسته به اطلاعات تمامی دانش‌آموزان دسترسی پیدا کند.

در این بخش شما باید API هایی را که در آن از کاربران ورودی دریافت می‌کنید، طوری تغییر دهید که در برابر حملات SQL Injection مقاوم باشند. به دلیل استفاده از کتابخانه Hibernate در فاز قبلی پروژه تا حد زیادی نسبت به این نوع حملات مقاوم هستیم اما استفاده از Hibernate جلوگیری کامل از SQL Injection را تضمین نمی‌کند و کد در صورت رعایت نکردن اصول امنیتی می‌تواند در برابر این حملات آسیب پذیر باشد.

برای راهنمایی بیشتر در این مورد می‌توانید این لینک را مطالعه نمایید.



عکس (۱)

نمونه‌ای از کد آسیب پذیر:

```java
public User getUserByUsername(String username) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction tx = null;
    User user = null;
    try {
        tx = session.beginTransaction();
        // SQL injection vulnerability here!
        Query query = session.createQuery("FROM User WHERE username = '" + username + "'");
        user = (User) query.uniqueResult();
        tx.commit();
    } catch (HibernateException e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
    return user;
}
```

اگر username مخرب مانند "admin' OR '1'='1" وارد شود، کوئری حاصل به صورت زیر خواهد بود:

```
FROM User WHERE username = 'admin' OR '1'='1'
```

در نتیجه اطلاعات تمامی کاربران برای مهاجم قابل دسترسی است.

Software Engineering Best Practices

نکاتی درباره امنیت وب اپلیکیشن

Parameterized Queries

برای جلوگیری از حملات تزریق SQL از پرس و جوهای پارامتری (یا عبارات آماده شده (prepared statement)) استفاده کنید. این تضمین می‌کند که ورودی کاربر به جای کد اجرایی به عنوان داده در نظر گرفته می‌شود.

Input Validation

تمام ورودی‌های کاربر در سمت سرور را تأیید و sanitize کنید تا از تزریق داده‌های مخرب جلوگیری شود. اعتبارسنجی ورودی باید در هر دو طرف مشتری و سرور انجام شود.

تست امنیتی

تست‌های امنیتی و بازبینی کدها به صورت منظم برای شناسایی و کاهش آسیب‌پذیری‌های امنیتی در اوایل چرخه توسعه اهمیت زیادی دارد. تست نفوذ[5] (peneteration test)، اسکن آسیب‌پذیری(vulnerability scanning) و ممیزی‌های امنیتی[6] (security audit) را به طور منظم انجام دهید.

---

[5] A **penetration test (pen test)** is an authorized simulated attack performed on a computer system to evaluate its security.

[6] A **security audit** is a systematic evaluation of the security of a company's information system by measuring how well it conforms to an established set of criteria.

Security Patching and Updates

همه وابستگی‌ها، چارچوب‌ها و کتابخانه‌های نرم‌افزار را با آخرین وصله‌های امنیتی (security patches)[7] و به‌روزرسانی‌ها، به‌روز نگه دارید. توصیه‌های امنیتی را به طور منظم نظارت کنید و وصله‌ها را به سرعت اعمال کنید.

Security Headers

برای محافظت در برابر آسیب پذیری‌های مختلف وب و اعمال سیاست‌های امنیتی در مرورگر، از headerهای امنیتی (به عنوان مثال، X-Frame-Options، X-Content-Type-Options) استفاده کنید.

Cross-Site Scripting (XSS) Prevention

برای پیشگیری از اسکریپت بین سایتی (XSS) محتوای تولید شده توسط کاربر را برای جلوگیری از حملات XSS رمزگذاری و پاکسازی کنید. از چارچوب‌ها یا کتابخانه‌هایی استفاده کنید که ویژگی‌های حفاظتی داخلی XSS را ارائه می‌کنند.

Session Management

تکنیک‌های مدیریت session امن را برای محافظت از داده‌های session و جلوگیری از session hijacking[8] اجرا کنید. از random session IDs، ست کردن زمان انقضای مناسب برای session و ذخیره ایمن داده‌های session استفاده کنید

HTTPS

از HTTPS برای رمزگذاری داده‌های منتقل شده بین کلاینت و سرور استفاده کنید. این امر از استراق سمع و حملات man-in-the-middle جلوگیری می‌کند و محرمانه بودن و یکپارچگی داده‌ها را تضمین می‌کند.

---

[7] **Security patches** are software and operating system updates that aim to fix security vulnerabilities in a program or product.

[8] **Session hijacking** refers to the malicious act of taking control of a user's web session.

هش کردن رمز عبور

همیشه رمزهای عبور را با استفاده از الگوریتم‌های هش (مانند Argon2 ،bcrypt)، قبل از ذخیره آن‌ها در پایگاه‌داده هش کنید. هرگز رمزهای عبور را به صورت متن ساده ذخیره نکنید.

Git Commit

همان‌طور که در پروژه اول توضیح داده شد، کامیت‌ها اهمیت زیادی در توسعه پروژه‌های نرم‌افزاری دارند. در این پروژه نیز باید مواردی که در پروژه اول گفته شدند، رعایت شوند. رعایت این قسمت، بخشی از نمره شما را در این پروژه تعیین می‌کند.

نکات پایانی

- کافی است که یکی از اعضای گروه Hash مربوط به آخرین کامیت پروژه سمت کاربر و سمت سرور را در سایت درس آپلود کند. در هنگام تحویل، پروژه روی این کامیت 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.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاست درس، نمره کسر خواهد شد.
- حتما کاربر IE-S03 را به مخزن خود اضافه کرده باشید.
- در صورت داشتن سوال درباره پروژه به آدرس alirezaesmaeilzadeh113@gmail.com ایمیل بزنید.

موفق و پیروز باشید.