cdrf

מקודם ראינו שאם לא רשמנו את השורה – ()csrf().disable, כשניסינו לעשות פעולות שמשנות מידע(לא GET) נזרקה לנו שגיאה 403, עכשיו נסביר מה זה csrf

Cross Site Request Forgery - csrf

Forgery – פעולה של זיוף.

שתוקף לא יצליח לגרום לנו לשלוח בקשה לשרת שאנחנו מאומתים אצלו וינצל את זה לטובתו. למשל אם מישהו יצליח לגרום לי ללחוץ על לינק שהלינק הזה גורם לי לשלוח בקשת POST לאתר בנק שלי שאומרת לבנק להעביר כסף לאותו אתר.

בעצם הפעולות שמתבצעות הם:

- הלקוח עושה login
- השרת שולח CSRF Token
- כל פעם שהלקוח שולח בקשה של שינוי מידע (POST, PUT...) השרת מאמת את הטוקן

אז בעצם מקודם לא השתמשנו בטוקן הזה ולכן הבקשות שלנו לא היו מאומתות ולכן ביטלו את CSRF

ההמלצה של מתי להשתמש בCSRF היא בשביל כל בקשה שהיכולה להתבצע בדפדפן ע"י משתמש רגיל. אם משתמשים בשירות שהוא ללא לקוחות דפדפן עדיף לבטל אותו.

אז איך התוקן הזה נוצר, איך משתמשים בו, ואיפה נמצא אותו?

אז כדי שיופיע לי העוגייה הזאת כתבנו את השורה הבאה

(and זה כדי להמשיך את הפקודות אחרי זה)

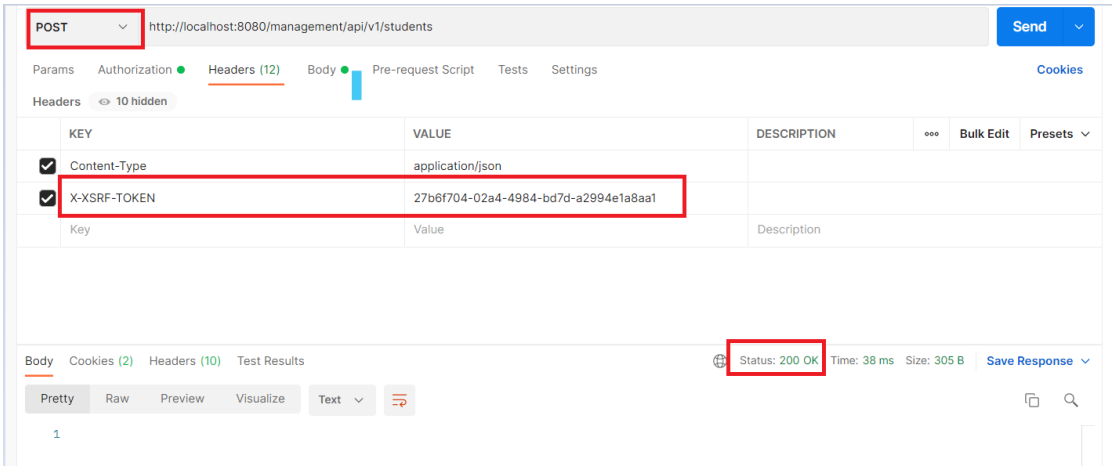
```
.csrf().csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())  CsrfConfigurer<HttpSecurity>
.and()  HttpSecurity
.authorizeRequests()// אנחנו רוצים לאמת בקשות
```

ואז נוכל לראות אותו בפוסטמן, פשוט נלך לcookies ונמצא אותו שם (אחרי שעשינו בקשת get)

Body | Cookies (2) | Headers (11) | Test Results | Status: 200 OK | Time: 6 ms | Size: 475 B | Save Response

Name	Value	Domain	Path	Expires	HttpOnly	Secure
JSESSIONID	D76A187F66f...	localhost	/	Session	true	false
XSRF-TOKEN	2768e6b4-2t...	localhost	/	Session	false	false

ואז כדי להשתמש בו פשוט נשים אותו בheader (כ X-XSRF-TOKEN)



אז איך הוא נוצר?

הוא נוצר בעזרת המתודה שראינו קודם.

מה המתודה withHttpOnlyFalse() עושה?

זה אומר שהעוגייה לא תהיה זמניה בעזרת JS

```
public static CookieCsrfTokenRepository withHttpOnlyFalse() {
    CookieCsrfTokenRepository result = new CookieCsrfTokenRepository();
    result.setCookieHttpOnly(false);
    return result;
}
```

ואם ניכנס למחלקה CookieCsrfTokenRepository נוכל לראות דברים מעניים כמו השם הדיפולטיבי של העוגייה ושל הheader.

```
public final class CookieCsrfTokenRepository implements CsrfTokenRepository {
    static final String DEFAULT_CSRF_COOKIE_NAME = "XSRF-TOKEN";
    static final String DEFAULT_CSRF_PARAMETER_NAME = "_csrf";
    static final String DEFAULT_CSRF_HEADER_NAME = "X-XSRF-TOKEN";
    private String parameterName = "_csrf";
    private String headerName = "X-XSRF-TOKEN";
    private String cookieName = "XSRF-TOKEN";
    private boolean cookieHttpOnly = true;
    private String cookiePath;
    private String cookieDomain;
    private Boolean secure;
    private int cookieMaxAge = -1;
```

בוא נסתכל על המתודה saveToken

```java
public void saveToken(CsrfToken token, HttpServletRequest request, HttpServletResponse response) {
    String tokenValue = token != null ? token.getToken() : "";
    Cookie cookie = new Cookie(this.cookieName, tokenValue);
    cookie.setSecure(this.secure != null ? this.secure : request.isSecure());
    cookie.setPath(StringUtils.hasLength(this.cookiePath) ? this.cookiePath : this.getRequestContext(request));
    cookie.setMaxAge(token != null ? this.cookieMaxAge : 0);
    cookie.setHttpOnly(this.cookieHttpOnly);
    if (StringUtils.hasLength(this.cookieDomain)) {
        cookie.setDomain(this.cookieDomain);
    }

    response.addCookie(cookie);
}
```

הוא מייצר טוקן חדש אם צריך, ואז יותר עוגייה, שם את הsecurity, שם את הpath, את הmaxAge, ואז מוסיף את העוגייה ל response.

עכשיו בוא נסתכל על המחלקה CsrfFilter

היא מרחיבה את המחלקה OncePerRequestFilter – כלומר הוא בשביל בקשה אחת

```java
package org.springframework.security.web.csrf;

import ...

public final class CsrfFilter extends OncePerRequestFilter {
    public static final RequestMatcher DEFAULT_CSRF_MATCHER = new CsrfFi
```

והמתודה שמעניינת אותנו היא doFilterInternal

```java
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {
    request.setAttribute(HttpServletResponse.class.getName(), response);
    CsrfToken csrfToken = this.tokenRepository.loadToken(request);
    boolean missingToken = csrfToken == null;
    if (missingToken) {
        csrfToken = this.tokenRepository.generateToken(request);
        this.tokenRepository.saveToken(csrfToken, request, response);
    }

    request.setAttribute(CsrfToken.class.getName(), csrfToken);
    request.setAttribute(csrfToken.getParameterName(), csrfToken);
    if (!this.requireCsrfProtectionMatcher.matches(request)) {
        if (this.logger.isTraceEnabled()) {
            this.logger.trace("Did not protect against CSRF since request did not match " + this.requireCsrfProtectionMatcher);
        }

        filterChain.doFilter(request, response);
    } else {
        String actualToken = request.getHeader(csrfToken.getHeaderName());
        if (actualToken == null) {
            actualToken = request.getParameter(csrfToken.getParameterName());
        }

        if (!equalsConstantTime(csrfToken.getToken(), actualToken)) {
            this.logger.debug(LogMessage.of(() -> {
                return "Invalid CSRF token found for " + UrlUtils.buildFullRequestUrl(request);
            }));
            AccessDeniedException exception = !missingToken ? new InvalidCsrfTokenException(csrfToken, actualToken) : new MissingCsrfTokenException(actualToken);
            this.accessDeniedHandler.handle(request, response, (AccessDeniedException)exception);
        } else {
            filterChain.doFilter(request, response);
        }
    }
```

אז הוא שם כל מיני תכונות שהוא צריך

ואז הוא לוקח את הטוקן, אם הוא לא קיים הוא יוצר חדש והוא שומר אותו לtokenRepository

ואז אם זו בקשה שמשנה דברים הוא בודק את הטוקן.