Cross Site Request Forgery (CSRF) Saldırılarını Önleme

CSRF Nedir?

- Kullanıcı, tarayıcısı üzerinden **authenticated bir servis**e erişim halinde iken, saldırganların kullanıcının authenticated durumundan yararlanarak bu site üzerinde **bir işlem gerçekleştirebildiği** saldırı şeklidir
- Açığın temeli authentication bilgilerinin istemci tarafında **cookie şeklinde tutulması** ve **her istekte sunucuya geri gönderilmesi**dir

Örnek Bir CSRF Saldırısı

- Kullanıcı **WEBMAIL** hesabına **login** olur
- Bir süre hesabında vakit geçirdikten sonra **logout olmadan** tarayıcıda **yeni bir tab** açar
- Burada da **farklı bir sistenin sayfası**na erişir
- Bu sayfada **WEBMAIL sunucusuna kötü amaçlı istek gönderen bir HTML tag** yer alır

Örnek Bir CSRF Saldırısı

- Örneğin, bu **tag'in tetiklediği istek**te bir mesajı silme, kullanıcı ayarlarını değiştirme veya başka birilerine spam mail gönderme **komut**u olabilir
- Kullanıcının **WEBMAIL oturumu hala açık** olduğu için bu **istek kullanıcının yetkileri dahilinde işletilir**

CSRF Nasıl Önlenir?

- Sunucu her web requesti için **benzersiz ve gizli bir token** üretir (synchronizer token)
- Bu token'ı kullanıcıya gönderir
- Kullanıcı **bir sonraki request'i bu token ile birlikte** gerçekleştirir
- Token sunucu tarafında **her seferinde kontrol** edilir
- Eğer geçerli ise işleme izin verilir

CSRF ve HTTP GET

- Saldırıları önlemede birinci nokta state değişikliğine neden olan, hassas veri içeren istekler kesinlikle **GET ile yapılmamalı**dır
- Bu tür istekler sadece **PUT, POST, DELETE** gibi HTTP metotları ile **yapılmalı**dır
- HTTP GET ile yapılan işlemler verinin **Request-URI**'da da yer almasına neden olmaktadır
- Request URI, genellikle **üçüncü şahısların görebileceği yerler**de loglanabildiği için bu da ayrıca bir açık teşkil etmektedir

Spring Security CSRF Konfigürasyonu

- Spring Security çözüm olarak **Synchronizer Token Pattern** yöntemini kullanmaktadır
- **<http>** elemanı altında **<csrf/>** elemanı ile konfigüre edilir

```
<security:http>
    <security:csrf/>
</security:http>
```

token-repository-ref attribute'u default durumda **HttpSessionCsrfTokenRepository** sınıfından bir bean kullanmaktadır

request-matcher-ref attribute'u request'de CSRF kontrolünün uygulanıp uygulanmayacağına karar verir. Default durumda GET, TRACE, HEAD, OPTIONS metotları haricinde uygulanır

Spring Security CSRF Konfigürasyonu

- GET metodu ile yapılan herhangi bir istekte **yeni bir CSRF token** üretilir ve istemciye dönülür
- Bu CSRF tokenın sonraki **web request'lerinde yer alması** sağlanmalıdır
- Bunun için bütün POST, PUT, DELETE, PATCH metotlarında bir şekilde (örneğin bir hidden input alan ile) **CSRF token'ı web request'ine eklenmeli**dir

Spring Security CSRF Konfigürasyonu

```
<form action="${actionUrl}"
    method="post">
  <input type="submit"
    value="Click It!" />
  <input type="hidden"
    name="${_csrf.parameterName}"
    value="${_csrf.token}"/>
</form>
```

- **GET isteği tekrarlandığı** vakit CSRF token'ı HTTP Session'da saklandığı için **yeni bir token yerine aynı token** dönülecektir

- Spring Security, geçersiz bir CSRF token ile karşılaşıldığında **InvalidCsrfTokenException** fırlatır
- Bu exception **AccessDeniedHandler** tarafından ele alınır ve default olarak **403 access denied hatası** üretilir

Dikkat Edilmesi Gereken Noktalar:Login

- Login formlarında da CSRF protection kullanılması önerilir
- Bu durumda login sırasında da **HttpSession'a ihtiyaç** duyulacaktır
- Çünkü **CsrfToken**'a erişilir erişilmez bir **HttpSession** yaratılmaktadır
- RESTful/stateless uygulamalar için bu bir **overhead** yaratabilir
- **RESTful** servislerde **CSRF devre dışı** bırakılabilir

Dikkat Edilmesi Gereken Noktalar: Logout

- Spring Security 4, CSRF aktif ise logout işlemini **GET ile yapmaya izin vermemekte**dir
- Bu durumda **logout** işlemi için de ayrı bir **form** kullanılması ve formun **POST metodu ile submit** edilmesi gerekir

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com