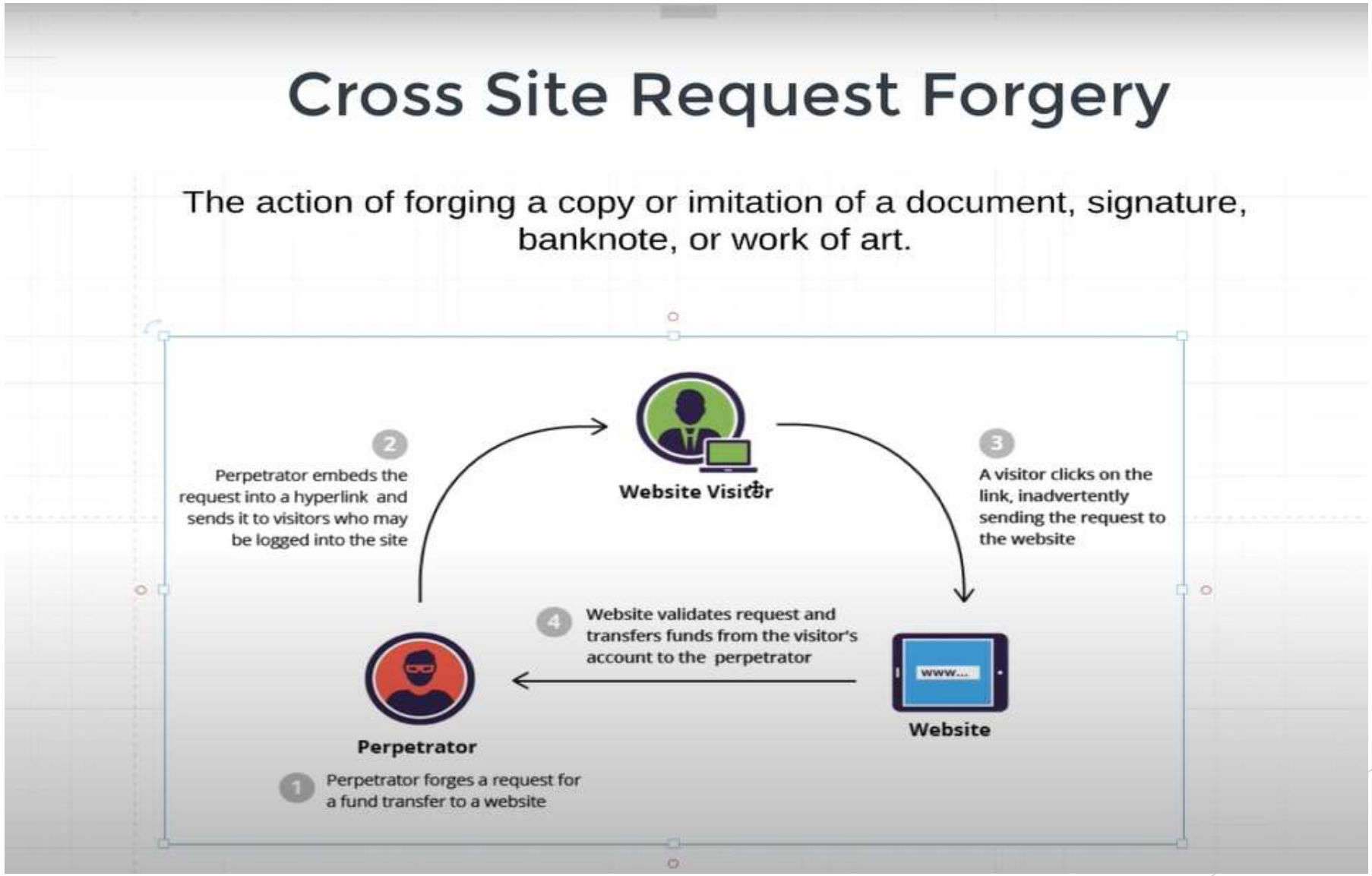
CSRF

CSRF

- Cross-site request forgery - Saytlararo so'rovni qalbakilashtirish.
- **Cross-site request forgery**, also known as **one-click attack** or **session riding** and abbreviated as **CSRF**.
- Saytlararo so‘rovlarni qalbakilashtirish, shuningdek, bir marta bosish hujumi yoki seansni boshqarish deb nomlanadi va CSRF sifatida qisqartiriladi.

- User authorization dan o’tganidan keyin boshqa bitta vqlatkada hakerni linkini bosadi (sovrunni qo’lga kiritish deb nomlangan).
- Shu hakerni linkida userni login parolini o’zgartirishga boshqa bir request bo’ladi. Link bosilganda login va parolni o’zgartiradigan link ishlab ketadi va uni paroli va logini haker hohlagan login va parolga o’zgaradi. Buni user bilmaydi. Haker boyagi login parol bilan saytga kirib nima qilsa qilaveradi.

Problem


Cross Site Request Forgery
The action of forging a copy or imitation of a document, signature, banknote, or work of art.
Website Visitor
2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website

CSRF

- Default holatda CSRF enabled (yoniq) bo’ladi
- Front Get Request jo’natganda SpringSecuruty unga csrf token generate qilib cookie da berib yuboradi. Front cookie dan XSRF-TOKEN key ni olishi mumkin.
- Har gal Frontend dan Post,Put,Delete request kelganida SpringSecurity Header da csrf token bor yo’qligiga tekshiradi. Agar token bo’lmasa uni CrosSiteRequestForgery attack deb tushunadi va 403 return qiladi.
- Front da hargal Headerda X-XSRF-TOKEN qilib berib yuborishimiz kerak.
- Bo’lmasa 403-401 kelaveradi.
- Nima uchun cookie ?
- Sababi bitta vklatkadan turib boshqa bitta vklatkadagi cookie larni olib bo’lmaydi.

- Agar CSRF ni disabled (o’chirib) qo’ysak SpringSecurity csrf token create qilmaydi va post,put,delete request larda uni userdan kutmaydi.

When to use CSRF protection

- When you use CSRF protection?
- Our recommendation is to use CSRF protection for any request that could be processed by a browser by normal users. If you are only creating a service that is used only by non browser clients, you will likely want to disable CSRF protection.
- Qachon CSRF himoyasidan foydalanish kerak?
- Browser tanomidan keladigan requestlar uchun CSRF ni ishlatish kerak. Agar siz faqat brouser bolmagan clientlar tomonidan ishlatiladigan service cha qilayotgan bo’lsangiz CSRF ni ishlatishingiz kerak emas.

- Browser tomonidan ishlatiladi deganda Java based web bo’lsa CSRF ni ishlating. Bo’lmasa vashe kerak emas.

Custom CsrfTokenRepository 1

- By default Spring Security stores the expected CSRF token in the HttpSession using HttpSessionCsrfTokenRepository. There can be cases where users will want to configure a custom CsrfTokenRepository. For example, it might be desirable to persist the CsrfToken in a cookie to support a JavaScript based application.
- By default the CookieCsrfTokenRepository will write to a cookie named XSRF-TOKEN and read it from a header named X-XSRF-TOKEN or the HTTP parameter _csrf. These defaults come from AngularJS

Custom CsrfTokenRepository 2

- You can configure CookieCsrfTokenRepository in Java Configuration using:
- Code explanation in next slide

```
@EnableWebSecurity
public class WebSecurityConfig extends
                    WebSecurityConfigurerAdapter {

        @Override
        protected void configure(HttpSecurity http) {
                http .csrf(csrf -> csrf
                                .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
                        );
        }
}
```

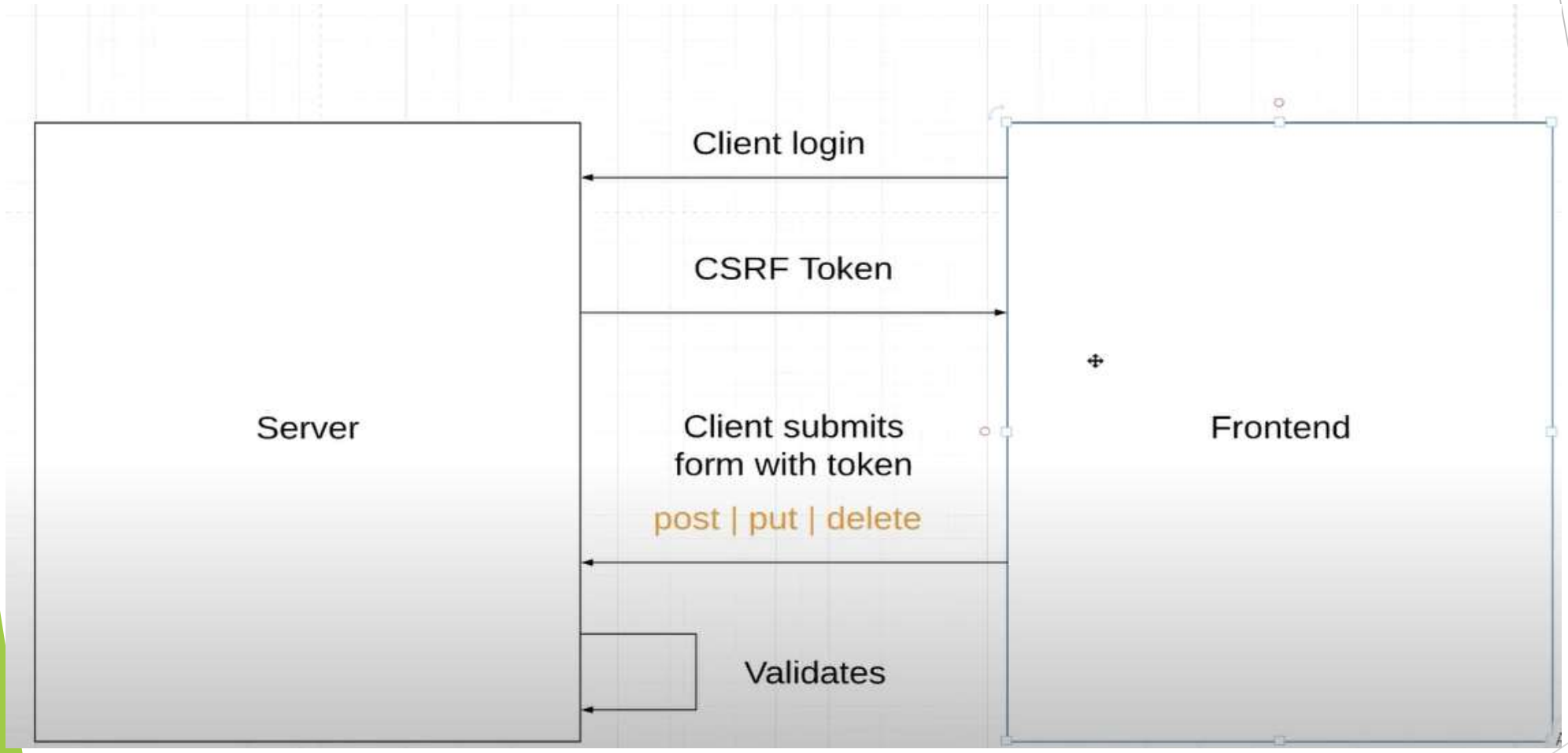
Custom CsrfTokenRepository 3

- The sample explicitly sets cookieHttpOnly=false. This is necessary to allow JavaScript (i.e. AngularJS) to read it. If you do not need the ability to read the cookie with JavaScript directly, it is recommended to omit cookieHttpOnly=false (by using new CookieCsrfTokenRepository() instead) to improve security.

links

- https://www.youtube.com/watch?v=Ub5TLow9GL4
- https://docs.spring.io/spring-security/reference/servlet/exploits/csrf.html
- https://docs.spring.io/spring-security/site/docs/5.0.x/reference/html/csrf.html

Solution


Server
Client login
CSRF Token
Client submits
form with token
post | put | delete
Validates
Frontend

Ajax GET request example

```
<!DOCTYPE html>
<html lang="en">
<body>

<h1>Result: </h1>
<div id="result">dasda</div>
</body>

</html>
<script>
    var xhttps = new XMLHttpRequest();

    xhttps.onreadystatechange = function () {
        document.getElementById("result").innerText = this.responseText;
    };
    xhttps.open("GET", "http://localhost:8080/category", false);
    xhttps.send();
</script>
```

Ajax POST request example

```
<!DOCTYPE html>
<html lang="en">
<body>
<h1>Result: </h1>
<div id="result">dasda</div>
</body>

</html>
<script>
    var xhttps = new XMLHttpRequest();
    var userObj = {name: "Alijon", surname: "Aliyev"};
    var jsonBody = JSON.stringify(userObj);

    xhttps.onreadystatechange = function () {
        document.getElementById("result").innerText = this.responseText;
    };

    xhttps.open("POST", "http://localhost:8080/category/test", true);
    xhttps.setRequestHeader('Content-Type', 'application/json');
    xhttps.send(jsonBody);
</script>
```