Secure Coding TP 2

Arsène Lapostolet

Introduction

In this report I will explain and demonstrate the Cross Site Request Forgery (CSRF) attack. A CSRF attack has three components :

- An authenticated action to perform on a website. For instance : changing account info like password and/or email to take control of the user's account
- Cookie-based browser authentication
- Predictable request parameters : the attacker must be able to guess any required parameter (ex : you don't need to provide old password to change your password)

The attacker will build a fake page to the user, using social engineering (i.e : phishing) and if the user is connected to the targeted website on his browser, the attacker site will be able to perform a malicious request to the target site using the cookie authentication token stored on the user's browser.

Demonstration

I will demonstrate this attack using a very simple website that features simple user login, using C#, .NET 6 and ASP .NET Core MVC.

I built a simple example of an application which is vulnerable to CSRF attack. This application features login via username and password that works with a cookie-based HTTP session. Once logged in, the user can change his password.

Quick breakthrough of the app :

- `GET /login` : login form
- `POST /login` : authenticate the FormData credentials and grants a session cookie
- `GET /home` : Welcome message and change password form, protected by auth
- `POST /home/password` : change the password of the user, protected by auth

When the login form is validated with the valid credentials, the server gives a cookie, containing the user's database ID (a GUID) and the user's username. The presence of this cookie will allow him to be authenticate when he request the `GET /home` page and validate the form at `POST /home/password`.

The app is hosted at : https://csrf-demo-vulnerable-site.arsenelapostolet.fr/login for demonstration.

I have also set up an attacker's website, which is just a simple HTML form, which does a `POST /home/password` with the value "arbitrary password" as password, using a hidden form field :

```
<form action="@ViewBag.Target/home/password" method="POST">
    <input type="hidden" name="NewPassword" value="arbitrary password" />
    <button type="submit">Gagner 1 000 000 €</button>
</form>
```

Submitting the form on the attackers website, while being on a web session connected to the vulnerable application will send the following HTTP request :

""http request POST /home/password HTTP/1.1 Host: csrf-demo-vulnerable-site.arsenelapostolet.fr Content-Type: application/x-www-form-urlencoded Content-Length: 30 Cookie: loggedUserId=46f85bd0-2948-4ce8-8ef7-ef118e2c0ece;loggedUserName=test

NewPassword=arbitrary+password ""

The hijacked cookie will authenticate the fraudulent request, and the password will be changed, allowing the attacked to take the control of the user's account, just because the user visited the attacker's website while being authenticated on the vulnerable site.

The attacker's site is hosted at : https://csrf-demo-attacker-site.arsenelapostolet.fr/

If you want to test the attack, I have set the credentials of the user as :

- Login : `test`
- Password : `test`

After the button is clicked on the attacker's site, the password will be "arbitrary password".

How to protect an application against CSRF attacks ?

The most efficient way to protect against CSRF attacks is the implementation of a CSRF Token that is tied to the user session on the sever side.

It should be transmitted as :

- Hidden field of the forms
- Query string
- Custom Header

Some thing to know to use secure CSRF tokens :

- The CSRF token must be validated for all HTTP methods
- The CSRF token must be validated even if not present
- The CSRF token must be tied to the user's session

In the case of the vulnerable application that I developed, the best way to secure it would be the use the server-side session implementation which is provided by the framework. This implementation is secure, it uses various mechanisms including, but not limited to, CSRF tokens, to prevent CSRF attacks.

The code of both site are available at : https://github.com/Ombrelin/efrei-m2-secure-coding-tp2