



ATTAQUE CSRF

JAVA / PHP

ÉQUIPE



BADRY ZAKARIA



PLAN DE LA PRÉSENTATION

- 1 - INTRODUCTION
- 2 - DÉFINITION DE L'ATTAQUE CSRF ET SON IMPORTANCE
- 3 - EXPLICATION DU MÉCANISME DE CSRF
- 4 - MESURES DE PROTECTION RECOMMANDÉES
- 5 - IMPLÉMENTATION EN PHP
- 6 - IMPLÉMENTATION EN JAVA

INTRODUCTION

Introduction

Les modules de cryptographie sont essentiels pour garantir la sécurité et l'intégrité des données dans les applications web. Ils permettent de chiffrer les informations sensibles, d'assurer l'authenticité des communications et de protéger les données contre les attaques.

Cependant, même les applications utilisant des modules de cryptographie peuvent être vulnérables aux attaques CSRF, si des mesures de sécurité spécifiques ne sont pas mises en place pour contrer cette menace.

DÉFINITION DE L'ATTAQUE CSRF ET SON IMPORTANCE

Définition de CSRF et son importance

Le Cross-Site Request Forgery (CSRF) est une attaque dans laquelle un utilisateur malveillant trompe le navigateur d'un utilisateur légitime pour envoyer des requêtes non désirées à une application web dans laquelle l'utilisateur est authentifié. Cette attaque exploite la confiance d'un site dans le navigateur d'un utilisateur, utilisant les cookies de session pour envoyer des requêtes malveillantes à l'insu de l'utilisateur.

Définition de CSRF et son importance

CSRF peut entraîner des conséquences graves, telles que :

- **Modification non autorisée de données** : Un attaquant peut changer des informations personnelles de l'utilisateur comme l'adresse email, le mot de passe ou les informations de paiement.
- **Exécution de transactions non désirées** : Dans les applications de commerce électronique ou bancaires, un attaquant peut initier des transferts d'argent ou des achats non autorisés.

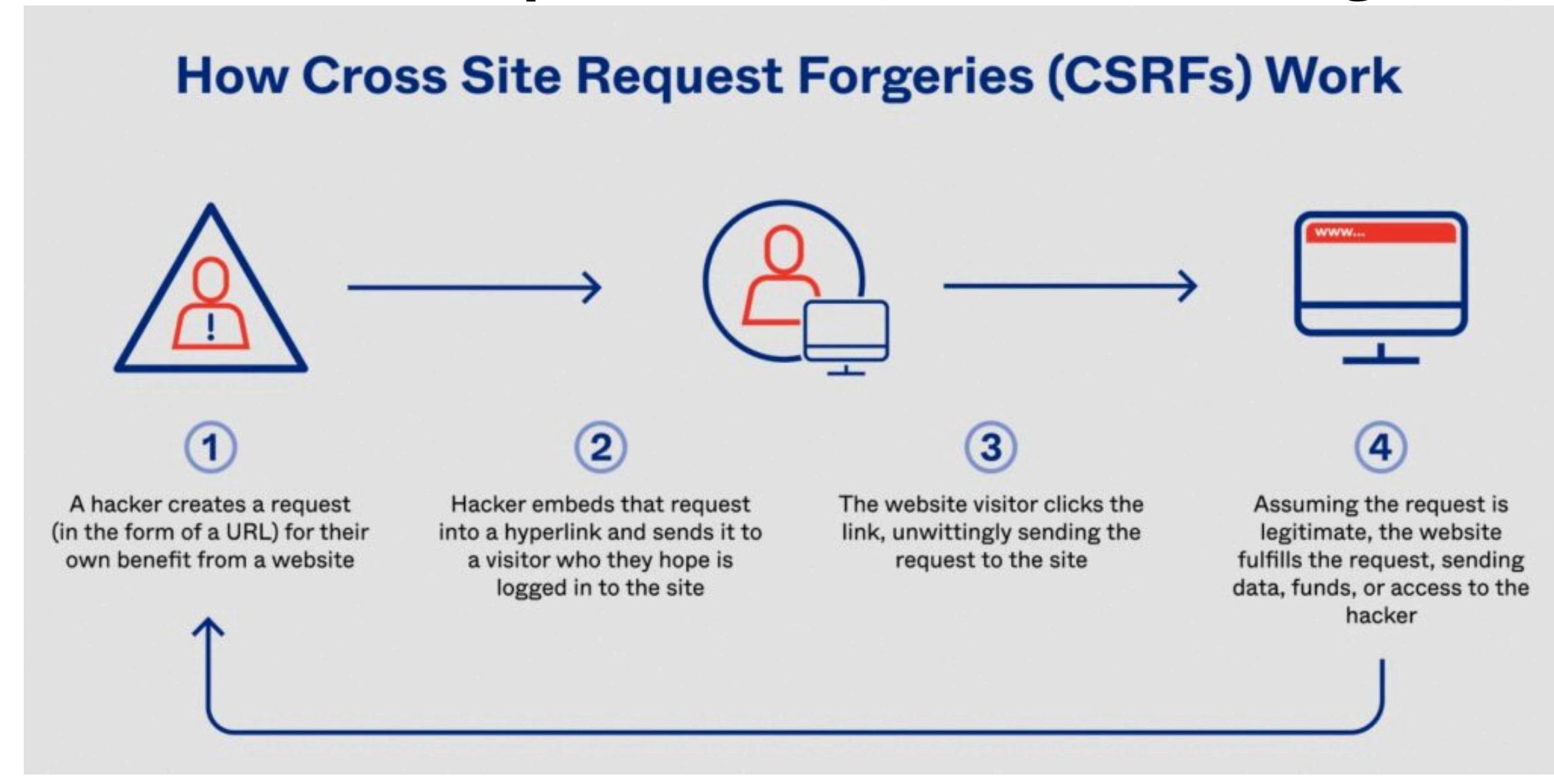
Définition de CSRF et son importance

- **Divulgation d'informations sensibles** : Un attaquant peut accéder et exposer des données sensibles en incitant le navigateur de l'utilisateur à envoyer des requêtes qui retournent des informations confidentielles.

EXPLICATION DU MÉCANISME DE CSRF

comment fonctionne attaque Csrf ?

L'utilisateur authentifié se connecte à un site légitime et reçoit un cookie d'authentification qui est stocké dans son navigateur.



Quelles sont les conditions nécessaires à une attaque CSRF ?

Pour mener à bien une attaque CSRF, certaines conditions doivent être remplies :

- L'authentification doit être basée uniquement sur les cookies.
- Tous les paramètres de requête doivent être prévisibles.
- L'application cible doit contenir une fonctionnalité intéressante et/ou critique pour un attaquant.

MESURES DE PROTECTION RECOMMANDÉES

Mesures de protection recommandées

Utilisation de tokens CSRF : Intégrer des tokens CSRF dans chaque formulaire et requête sensible pour vérifier que les requêtes proviennent de sources légitimes.

Authentification à deux facteurs (2FA) : Ajouter une couche supplémentaire de sécurité en exigeant une vérification supplémentaire pour les actions sensibles, réduisant ainsi le risque de réussite des attaques CSRF.

Examens de sécurité réguliers : Effectuer des audits de sécurité réguliers pour identifier et corriger les vulnérabilités potentielles, y compris celles liées aux attaques CSRF.

IMPLEMENTATION EN PHP

Création d'une application vulnérable à CSRF

Pour illustrer une application à une attaque CSRF, nous avons développé une simulation d'une banque en ligne appelée "Crypto-K". Cette application est construite en utilisant des technologies web de base telles que PHP, HTML et CSS. L'application permet aux utilisateurs de se connecter, de consulter leur solde et de transférer de l'argent à d'autres comptes.



SMART Portefeuilles Nouvelles Techniques de cryptage

La meilleure plateforme d'achat des
cryptomonnaies

REJOINDRE-NOUS



Création d'une application vulnérable à CSRF

Le principe de cette application est simple : les utilisateurs peuvent initier des transferts d'argent en remplissant un formulaire et en soumettant une requête POST au serveur. Le serveur vérifie ensuite si le solde de l'utilisateur est suffisant et, si c'est le cas, déduit le montant du compte de l'utilisateur et crédite le compte du destinataire. Cette logique est implémentée de manière basique sans aucune protection contre les attaques CSRF, ce qui en fait une cible idéale pour démontrer la vulnérabilité.

CryptoK

Change Profil

Déconnexion

Bonjour, zakaria

Votre solde est : 19000.00

Transférer

Charger

Débiter

Transférer des fonds

Adresse email du destinataire:

Montant à transférer:

[Transférer](#)[Transférer](#)[Charger](#)[Débiter](#)

Création d'une application vulnérable à CSRF

L'attaque CSRF exploitée ici consiste à envoyer un email à l'utilisateur, contenant un lien ou un formulaire caché qui, lorsqu'il est cliqué ou visité, envoie une requête POST malveillante au serveur de la banque. Cette requête semble provenir de l'utilisateur légitime car elle utilise les cookies de session de l'utilisateur authentifié. Dans notre scénario, l'email promet à l'utilisateur qu'il a gagné 1 BTC, et en cliquant sur le lien, une requête est envoyée à l'application pour transférer de l'argent au compte de l'attaquant sans que l'utilisateur ne s'en rende compte.

Félicitations !

Vous avez gagné 1 BTC gratuit ! Cliquez sur le bouton ci-dessous pour réclamer votre prix.

[Réclamer Prix](#)

Sans csrf token

Lorsque l'utilisateur clique sur le lien dans l'email, une requête POST est automatiquement envoyée à l'application, utilisant les cookies de session de l'utilisateur pour transférer de l'argent au compte de l'attaquant.

CryptoK

Change Profil Déconnexion

Bonjour, zakaria

Votre solde est : 17000.00

Transférer Charger Débiter

DevTools is now available in French!

Always match Chrome's language Switch DevTools to French Don't show again

Elements Console Sources Network Performance > x 5 1 ⋮ ⋮

Preserve log Disable cache No throttling ⋮ ⋮ ⋮

Filter Invert Hide data URLs Hide extension URLs

All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other

Blocked response cookies Blocked requests 3rd-party requests

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms 8000 ms 9000 ms

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
process_transfer.php	▼ General						
home.php?success=1							
style.css							
css2?family=Poppins:wght@200							
pxiEyp8kv8JHgFVrJfecg.woff2							
pxiByp8kv8JHgFVrlCz7Z1xFQ...							
content.js							
executor.js							
js.js							
dom.js							
dom.js							
js.js							
dom.js							
js.js							
14 requests 1.1 MB transferred							
Console What's new							

Highlights from the Chrome 124 update

```
<form action="process_transfer.php" method="post">

    <div class="form-group">
        <label for="recipientEmail">Adresse email du destinataire:</label>
        <input type="email" id="recipientEmail" name="recipientEmail" required>
    </div>
    <div class="form-group">
        <label for="amount">Montant à transférer:</label>
        <input type="number" id="amount" name="amount" min="0" step="0.01" required>
    </div>
    <button type="submit" class="btn">Transférer</button>
</form>
```

```
process_transfer.php > ...
3 session_start();
4 include("config.php");
5 // Vérifiez si l'utilisateur est connecté
6 if (!isset($_SESSION['email'])) {
7     header("Location: login.php");
8     exit();
9 }
10 // Vérifiez si les données du formulaire sont soumises
11 if ($_SERVER["REQUEST_METHOD"] == "POST") {
12     // Récupérez les données du formulaire
13     $recipientEmail = $_POST['recipientEmail'];
14     $amount = $_POST['amount'];
15     // Récupérez l'email de l'utilisateur connecté
16     $email = $_SESSION['email'];
17     // Récupérez le solde actuel de l'utilisateur
18     $query = mysqli_prepare($connection, "SELECT balance FROM utilisateurs WHERE email = ?");
19     mysqli_stmt_bind_param($query, "s", $email);
20     mysqli_stmt_execute($query);
21     $result = mysqli_stmt_get_result($query);
22     if ($result && $row = mysqli_fetch_assoc($result)) {
23         $currentBalance = $row['balance'];
24         // Soustrayez le montant transféré du solde de l'utilisateur
25         $newBalance = $currentBalance - $amount;
26         // Mettez à jour le solde de l'utilisateur dans la base de données
27         $updateQuery = mysqli_prepare($connection, "UPDATE utilisateurs SET balance = ? WHERE email = ?");
28         mysqli_stmt_bind_param($updateQuery, "ds", $newBalance, $email);
29         mysqli_stmt_execute($updateQuery);
30         // Redirigez l'utilisateur vers la page d'accueil avec un message de succès
31         header("Location: home.php?success=1");
32         exit();
33     } else {
34         // Gestion des erreurs de requête
35         header("Location: home.php?error=database_error");
36         exit();
37 }
```

Avec csrf token

Exécution de l'attaque : Lorsque l'utilisateur clique sur le lien dans l'email, une requête POST est automatiquement envoyée à l'application. Cependant, comme cette requête ne contient pas le token CSRF valide, elle est rejetée par le serveur.

CSRF token validation failed

DevTools is now available in French!

Always match Chrome's language Switch DevTools to French Don't show again

Elements Console Sources Network Performance > x 5 1 ⋮ ×

Filter Invert Hide data URLs Hide extension URLs

All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other

Blocked response cookies Blocked requests 3rd-party requests

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
process_transfer.php	General						
content.js	Request URL:	http://localhost/csrf-php/process_transfer.php					
executor.js	Request Method:	POST					
js.js	Status Code:	200 OK					
dom.js	Remote Address:	[::1]:80					
js.js	Referrer Policy:	strict-origin-when-cross-origin					
dom.js	Response Headers	<input type="checkbox"/> Raw					
js.js	Cache-Control:	no-store, no-cache, must-revalidate					
	Connection:	Keep-Alive					
	Content-Length:	29					
	Content-Type:	text/html; charset=UTF-8					
	Date:	Wed, 22 May 2024 21:56:02 GMT					
	Expires:	Thu, 19 Nov 1981 08:52:00 GMT					
	Keep-Alive:	timeout=5, max=98					
	Pragma:	no-cache					

8 requests | 1.1 MB transferred

⋮ Console What's new ×

Highlights from the Chrome 124 update

```
<!-- Formulaire de transfert (initiallement masqué) -->
<div id="transferModal" class="modal">
  <div class="modal-content">
    <span class="close" onclick="closeModal()">&times;</span>
    <h2>Transférer des fonds</h2>
    <?php
      if (!isset($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
      }
    ?>
    <form action="process_transfer.php" method="post">
      <input type="hidden" name="csrf_token" value="<?php echo hash('sha256', htmlspecialchars($_SESSION['csrf_token'])); ?>">

      <div class="form-group">
        <label for="recipientEmail">Adresse email du destinataire:</label>
        <input type="email" id="recipientEmail" name="recipientEmail" required>
      </div>
      <div class="form-group">
        <label for="amount">Montant à transférer:</label>
        <input type="number" id="amount" name="amount" min="0" step="0.01" required>
      </div>
      <button type="submit" class="btn">Transférer</button>
    </form>
  </div>
</div>
```

```
process_transfer.php > ...
2  <?php
3  session_start();
4  include("config.php");
5  if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== hash('sha256', $_SESSION['csrf_token'])) {
6      die('CSRF token validation failed');
7  }
8  // Vérifiez si l'utilisateur est connecté
9  if (!isset($_SESSION['email'])) {
10     header("Location: login.php");
11     exit();
12 }
13 // Vérifiez si les données du formulaire sont soumises
14 if ($_SERVER["REQUEST_METHOD"] == "POST") {
15     // Récupérez les données du formulaire
16     $recipientEmail = $_POST['recipientEmail'];
17     $amount = $_POST['amount'];
18
19     // Récupérez l'email de l'utilisateur connecté
20     $email = $_SESSION['email'];
21
22     // Récupérez le solde actuel de l'utilisateur
23     $query = mysqli_prepare($connection, "SELECT balance FROM utilisateurs WHERE email = ?");
24     mysqli_stmt_bind_param($query, "s", $email);
25     mysqli_stmt_execute($query);
26     $result = mysqli_stmt_get_result($query);
27     if ($result && $row = mysqli_fetch_assoc($result)) {
28         $currentBalance = $row['balance'];
29
30         // Soustrayez le montant transféré du solde de l'utilisateur
31         $newBalance = $currentBalance - $amount;
32
33         // Mettez à jour le solde de l'utilisateur dans la base de données
34         $updateQuery = mysqli_prepare($connection, "UPDATE utilisateurs SET balance = ? WHERE email = ?");
35         mysqli_stmt_bind_param($updateQuery, "ds", $newBalance, $email);
36         mysqli_stmt_execute($updateQuery);
```

IMPLEMENTATION EN JAVA

Implémentation de code :

Les outils utiliser dans cette implmentation :

Spring Boot : Framework Java open-source facilitant le développement d'applications web .

Spring Security : Framework de sécurité pour applications Spring Boot, offrant une authentification et autorisation complètes.

Angular : Framework open-source pour construire des interfaces utilisateur

Implémentation de code :

BackEnd : Spring boot , Spring security

- configuration la sécurité de l'application Spring Boot en définissant la protection CSRF, gestion de session et l'autorisation des requêtes HTTP :

```
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .exceptionHandling(customizer -> customizer.authenticationEntryPoint(userAuthEntryPoint))
            .csrf(Customizer.withDefaults())
            .sessionManagement(customizer -> customizer.sessionCreationPolicy(SessionCreationPolicy.ALWAYS))
            .authorizeHttpRequests((requests) -> requests
                .requestMatchers(HttpMethod.POST, ...patterns: "/signIn").permitAll()
                .requestMatchers(HttpMethod.GET, ...patterns: "/csrf/token").permitAll()
                .requestMatchers(HttpMethod.OPTIONS, ...patterns: "/**").permitAll()
                .anyRequest().authenticated());
        return http.build();
    }
```

Implémentation de code :

BackEnd : Spring boot , Spring security

génération de Csrf token : Lorsqu'un client envoie une requête GET vers l'endpoint /csrf/token, cette méthode est invoquée:

```
@GetMapping("/csrf/token")
public CsrfToken csrf(CsrfToken token) { return token; }
```

Implémentation de code :

FrontEnd : Angular .

- Lorsque l'utilisateur accède à l'application, la première chose que l'application fait est de générer un jeton CSRF (faire appelle a getCsrf()) :

```
ngOnInit(): void {  
    this.http.getCsrf();  
}
```

Implémentation de code :

FrontEnd : Angular .

getCsrf() : Cette méthode envoie une requête GET spéciale pour récupérer le token CSRF du serveur et le stocke dans la propriété csrfToken :

```
getCsrf() {  
    return this.http.get("http://localhost:8080/csrf/token", {withCredentials: true})  
        .subscribe((data: any) => this.csrfToken = data.token);  
}  
}
```

Implémentation de code :

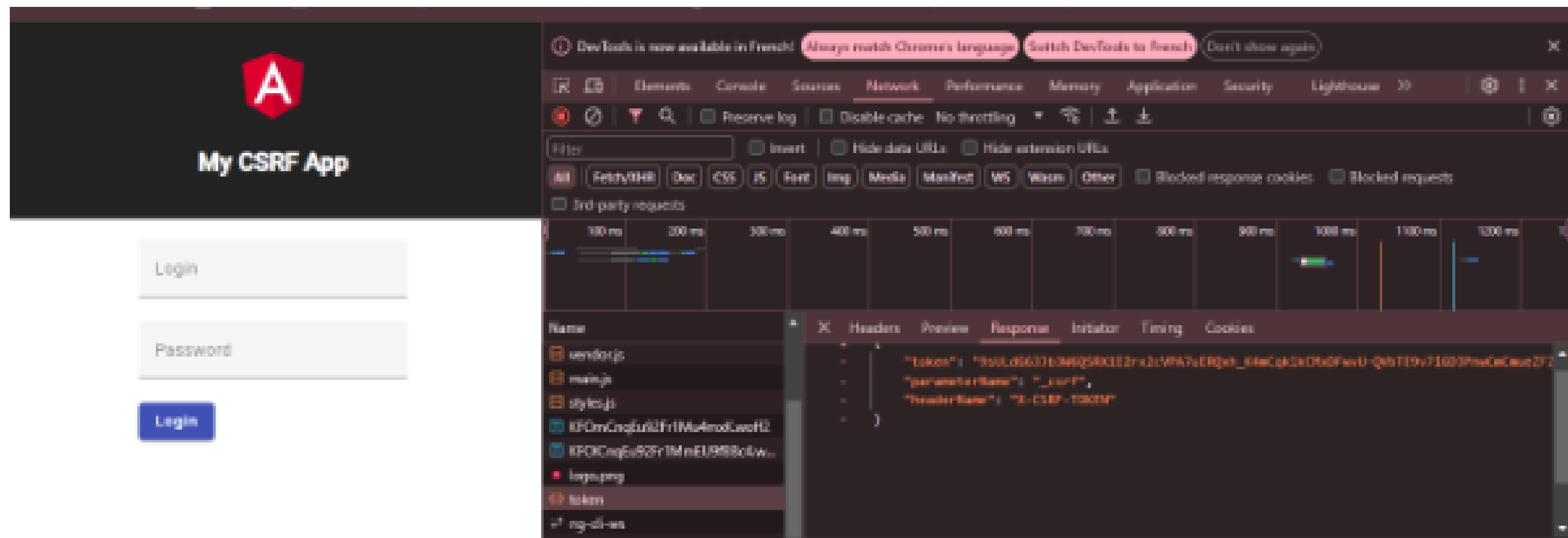
FrontEnd : Angular .

Si l'utilisateur est authentifié avec succès, le jeton CSRF sera envoyé dans l'en-tête de chaque requête POST envoyée par l'utilisateur, en incluant dans l'en-tête "X-Csrf-Token", utilisant cette fonction. Ce jeton sera ensuite traité par le serveur pour identifier qu'il est lié à cette session de l'utilisateur connecter :

```
post(url: string, data: any): any {
  return this.http.post(
    "http://localhost:8080" + url,
    data,
    {headers: new HttpHeaders({"X-CSRF-TOKEN": this.csrfToken}), withCredentials: true}
  );
}
```

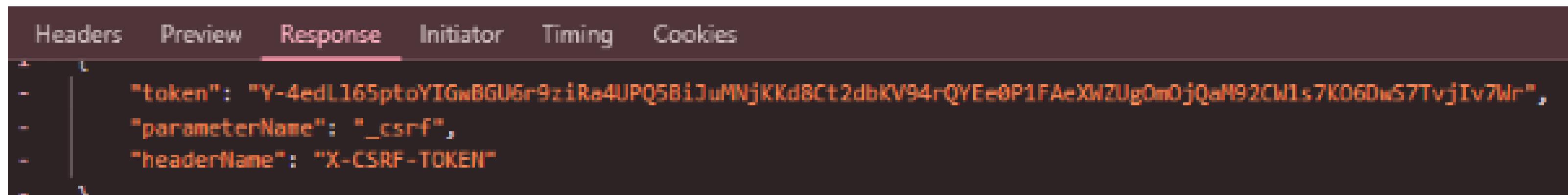
Démonstration

- Lorsqu'un utilisateur demande la page root, le serveur, utilisant une authentification stateful, génère un token CSRF qui est envoyé en réponse à la requête HTTP GET. Pour chaque session créée sur le serveur, un nouveau token CSRF est généré.



Démonstration

le token généré :



A screenshot of a browser's developer tools Network tab. The Response section is selected and shows the following JSON data:

```
{
  "token": "Y-4edL165ptoYIGwBGU6r9ziRa4UPQ5BiJuMNjKKd8Ct2dbKV94rQYEe0P1FAeXWZUg0m0jQaM92CW1s7K06DwS7TvjIv7Mr",
  "parameterName": "_csrf",
  "headerName": "X-CSRF-TOKEN"
}
```

Démonstration

Si l'utilisateur est bien authentifié, le serveur envoie la réponse suivante au format JSON, contenant votre identifiant, votre login et votre nom.

The screenshot illustrates a CSRF attack demonstration. On the left, a web application titled "My CSRF App" is shown. It features a message list with items like "hi hamza", "are find", and "bonjour". Below the list is an "Add message" form with a text input containing "bonjour" and a blue "Add" button. On the right, the Chrome DevTools Network tab is open, displaying a list of network requests. A specific JSON response is highlighted with a red arrow pointing to it. The response payload is visible in the "Response" tab:

```
{"id":1,"login":"login","name":"Sergio"}
```

Démonstration

- Par exemple, l'utilisateur va envoyer un message “bonjour”, qui sera le contenu de la requête HTTP avec un en-tête contenant le token CSRF (X-CSRF-Token).

The screenshot shows a web application titled "My CSRF App" with a red logo. On the left, there's a sidebar with "Hi home", "new friend", "import", and a "Add message" button with "bonjour" typed into it. Below that is an "Add" button. On the right, the Network tab of the developer tools is open, showing a list of requests. A red arrow points to the "X-CSRF-Token" header in the request details for the "Add message" POST request. The value of the token is displayed below the screenshot.

X-CSRF-Token:

Y-

4edU65ptoYIGwBGU6r9ziRa4UPQ5BiJuMNjKKd8Ct2dbKV94rQEe0P1FAeXWZU
gOmOjQaM92CW1s7KO6DwS7Tvjv7Wr

Démonstration

Si nous essayons de nous connecter à partir d'un autre navigateur dans lequel nos cookies ne sont pas enregistrés, le serveur va générer un autre token CSRF, totalement différent

The screenshot shows a browser developer tools interface with the Network tab selected. On the left, there is a preview of a login page titled "My CSRF App" with fields for "Login" and "Password" and a "Login" button. The Network tab displays a list of requests. One request is highlighted, showing its Headers, Preview, Response, Initiator, Timing, and Cookies sections. The Headers section shows the following data:

Name	Value
Content-Type	application/x-www-form-urlencoded
Cookie	polyfill.js; session=; main.js; styles.js; EPMCnCqfQh0B9rHMu4mzv...; ng-id=ns
HeaderName	X-CSRF-TOKEN

The Preview section shows the JSON response body:

```
{"token": "-FEuLYJiOgcJ00pEmUAS91wXj4gKP3Y9n7KpF4tRN3jkMF_fzDUXT-FUC2EkwOkloG8mzjsIorASXhMQ-4LKcu5nDhzdUmzs", "parameterName": "_csrf", "headerName": "X-CSRF-TOKEN"}
```

MERCI POUR VOTRE
ATTENTION