



L'Ecole Nationale des Sciences Appliquées Kénitra  
Deuxième année cycle ingénieur – S8 – Génie informatique

Module : Cryptographie

---

# Rapport du projet : Attaque CSRF en PHP et JAVA

---

Année scolaire 2023-2024

Présenté par : BADRY ZAKARIA / EL GOURCHI HAMZA

Numéro Apogée : 22014301 / 20000690

Sous la direction de : Mr. Khalid choug dali

# **1- Introduction**

## **A. -Définition de CSRF et son importance dans la sécurité des applications web**

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.

Lorsque l'utilisateur est authentifié sur une application web, le navigateur stocke des cookies de session qui permettent au serveur de reconnaître les requêtes venant de cet utilisateur. Le CSRF prend avantage de ce mécanisme en incitant le navigateur à envoyer des requêtes spécifiques à l'application cible sans l'intervention directe de l'utilisateur. Par exemple, un attaquant pourrait envoyer un email contenant un lien malveillant ou intégrer un script dans une page web tierce. Lorsque l'utilisateur clique sur le lien ou visite la page, le navigateur envoie la requête à l'application vulnérable avec les cookies de session valides, donnant l'apparence que la requête est légitime.

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.

**Divulgaration 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.

## **B. Explorer l'impact de CSRF dans les applications web utilisant des modules de cryptographie.**

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.

**Exemple d'attaque CSRF dans un contexte cryptographique :**

**Scénario de l'application bancaire :** Considérons une application bancaire en ligne qui utilise des modules de cryptographie pour sécuriser les transactions financières. Cette application chiffre les

données de transaction, utilise SSL/TLS pour sécuriser les communications et stocke les informations sensibles de manière chiffrée.

**Vulnérabilité CSRF :** Si cette application ne met pas en œuvre de mesures de protection contre CSRF, un attaquant peut créer une page web malveillante qui, lorsqu'elle est visitée par un utilisateur authentifié, envoie une requête POST pour transférer de l'argent vers un compte contrôlé par l'attaquant. La requête inclut les cookies de session de l'utilisateur, et le serveur de l'application bancaire traite la requête comme légitime, car elle provient d'un utilisateur authentifié.

**Impact sur la sécurité :** Même si les données de la transaction sont chiffrées et sécurisées en transit, l'attaque CSRF contourne ces protections en utilisant les cookies de session valides pour effectuer des actions non désirées. Ainsi, l'intégrité et la confidentialité des données ne sont pas directement compromises, mais la sécurité de l'application est contournée, entraînant des pertes financières et compromettant la confiance des utilisateurs.

**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.

## **2. Explication du mécanisme de CSRF : comment fonctionne-t-il ?**

L'objet de cette attaque est de transmettre à un utilisateur authentifié une requête HTTP falsifiée qui pointe sur une action interne au site, afin qu'il l'exécute sans en avoir conscience et en utilisant ses propres droits. L'utilisateur devient donc complice d'une attaque sans même s'en rendre compte. L'attaque étant actionnée par l'utilisateur, un grand nombre de systèmes d'authentification sont contournés.

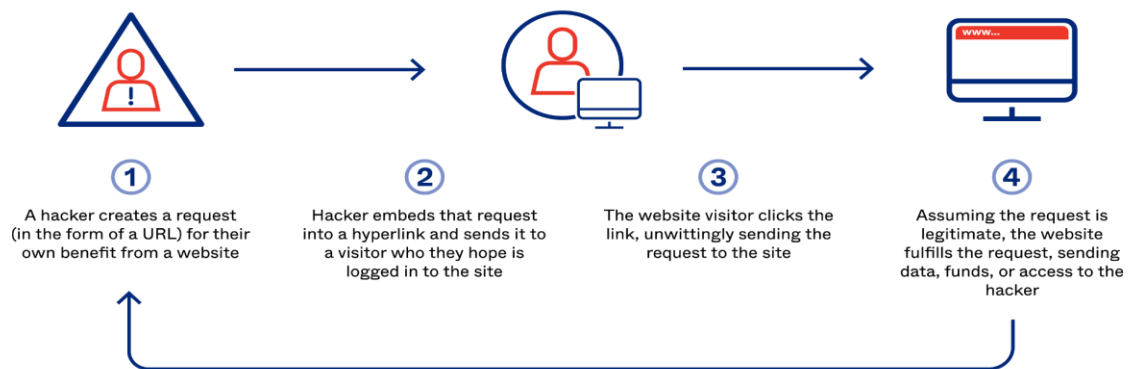
De manière générale, lorsqu'une fonctionnalité est vulnérable aux attaques CSRF, les attaquants cherchent à pousser les utilisateurs à charger une page malveillante.

Par exemple l'attaquants envoyer une page , Cette page contenant du code malveillant enverra des requêtes via le navigateur d'un utilisateur comme s'il s'agissait d'une action légitime de ce dernier, ce qui permet à l'attaquant de réaliser des actions telles que la modification du mot de passe et/ou d'autres informations personnelles ou le changement des configurations et des paramètres personnels de l'utilisateur sur la plateforme vulnérable.

### **A- Voici comment cela pourrait fonctionner :**

1. L'utilisateur authentifié se connecte à un site légitime et reçoit un cookie d'authentification qui est stocké dans son navigateur.
2. Pendant que la session sur ce site est active, l'utilisateur visite un site malveillant.
3. Le site malveillant contient des scripts ou des requêtes spécialement conçus qui, lorsqu'ils sont déclenchés, envoient des requêtes HTTP à des actions spécifiques sur le site légitime.
4. Ces requêtes sont conçues pour effectuer des actions non autorisées ou nuisibles, telles que changer le mot de passe de l'utilisateur, effectuer des achats, ou supprimer des données.

## How Cross Site Request Forgeries (CSRFs) Work



## B- 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.** Le déclenchement de l'attaque implique une ou plusieurs requêtes HTTP et l'application s'appuie seulement sur les cookies pour permettre l'authentification de l'utilisateur.
- **Tous les paramètres de requête doivent être prévisibles.** Les requêtes contiennent des paramètres dont l'attaquant peut déterminer ou deviner les valeurs.
- **L'application cible doit contenir une fonctionnalité intéressante et/ou critique pour un attaquant.**

## C- Comment se protéger contre les attaques CSRF ?

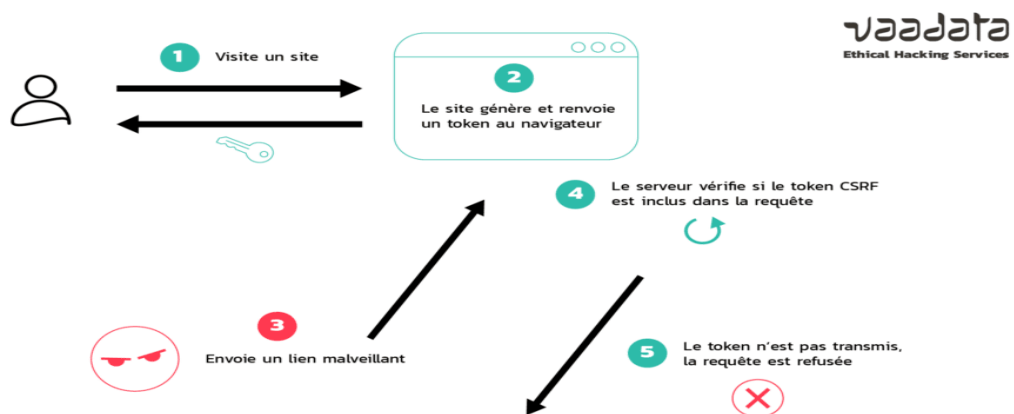
Les attaques CSRF sont relativement simples à contrer. Pour ce faire, deux mesures de sécurité peuvent être mises en œuvre :

- **Implémenter des jetons CSRF pour prévenir les attaques**

Pour se protéger contre les attaques CSRF, une mesure de sécurité courante consiste en l'implémentation de jetons CSRF.

Les jetons CSRF sont des valeurs uniques générées aléatoirement côté serveur et envoyées au client. Ces valeurs étant uniques pour chaque requête, elles permettent de renforcer la sécurité d'une application en rendant difficile (voire impossible) pour un attaquant de les deviner et donc d'exploiter une vulnérabilité CSRF.

Par ailleurs, certains frameworks, tels que Django (Python) et Laravel (PHP), intègrent des middlewares activés par défaut assurant une protection contre les attaques CSRF.



- **Demander des confirmations à l'utilisateur pour les actions critiques, au risque d'alourdir l'enchaînement des formulaires :**

Lorsque l'utilisateur clique sur ce bouton, une boîte de dialogue de confirmation s'affiche, lui demandant de confirmer s'il souhaite vraiment supprimer son compte.

Cela ajoute une couche de sécurité supplémentaire en obligeant l'utilisateur à confirmer son intention de supprimer son compte, ce qui réduit le risque de suppressions accidentelles ou malveillantes.

- **Demander une confirmation de l'ancien mot de passe à l'utilisateur pour changer celui-ci ou utiliser une confirmation par email ou par SMS.**

Ces mesures ajoutent une sécurité supplémentaire en s'assurant que l'utilisateur est bien autorisé à changer son mot de passe, en vérifiant d'abord l'ancien mot de passe ou en demandant une confirmation via un e-mail ou un SMS. Cela réduit considérablement les risques de changement de mot de passe non autorisé.

### **1- Eviter d'utiliser des requêtes HTTP GET pour effectuer des actions critiques de modification des données ,**

En évitant d'utiliser des requêtes HTTP GET pour des actions critiques, on réduit le risque d'attaques CSRF. Cela signifie que les actions qui modifient les données devraient utiliser des requêtes HTTP POST, ce qui rend plus difficile pour un attaquant de manipuler les actions à travers des liens malveillants.

Mais laissera passer les attaques fondées sur JavaScript, lesquelles sont capables très simplement de lancer des requêtes HTTP POST.

## **3 . Implémentation de l'attaque CSRF :**

### **A-Implementation java :**

Dans cette implémentation, nous avons essayé de développer une application web full-stack en utilisant les frameworks suivants :

**Spring Boot** : est un framework Java open-source qui facilite le développement d'applications web et de microservices en fournissant une configuration par défaut. Il permet de créer des applications autonomes avec un minimum de configuration. Spring Boot simplifie l'intégration des composants de l'écosystème Spring.

**Spring Security** : est un framework de sécurité que l'on peut utiliser dans les applications Spring boot , fournissant une authentification et une autorisation complètes. Il intègre des mécanismes pour protéger les applications contre les menaces courantes telles que les attaques CSRF et les failles de sécurité des sessions.

**Angular** : est un framework open-source , Il permet de construire des interfaces utilisateur complexes avec des composants réutilisables. Angular est idéal pour le développement de single-page applications (SPA).

### **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 :

```

20 @Bean
21 @ public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
22     http
23         .exceptionHandling(customizer -> customizer.authenticationEntryPoint(userAuthEntryPoint))
24         .csrf(Customizer.withDefaults())
25         .sessionManagement(customizer -> customizer.sessionCreationPolicy(SessionCreationPolicy.ALWAYS))
26         .authorizeHttpRequests((requests) -> requests
27             .requestMatchers(HttpMethod.POST, ...patterns: "/signIn").permitAll()
28             .requestMatchers(HttpMethod.GET, ...patterns: "/csrf/token").permitAll()
29             .requestMatchers(HttpMethod.OPTIONS, ...patterns: "/*").permitAll()
30             .anyRequest().authenticated());
31     return http.build();
32 }

```

- génération de Csrf token : Lorsqu'un client envoie une requête GET vers l'endpoint /csrf/token, cette méthode est invoquée. Spring Security fournit automatiquement un objet CsrfToken en tant qu'argument de la méthode. Cette méthode retourne ensuite ce token CSRF au client :

```

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

```

### 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();
}

```

- 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);
}

```

- 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 :

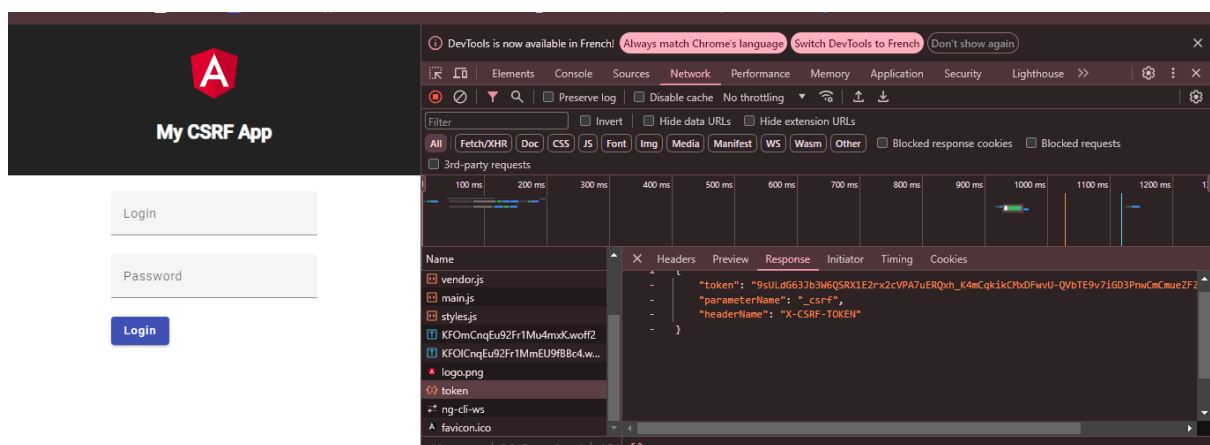
```

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}
  );
}

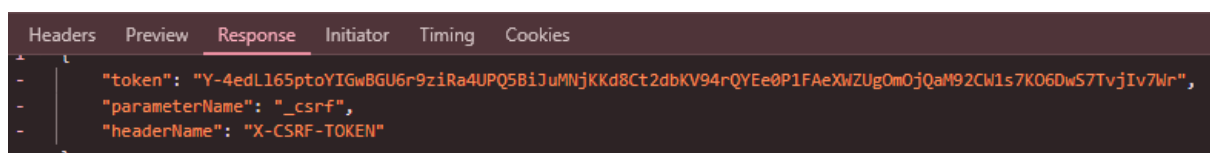
```

## 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.

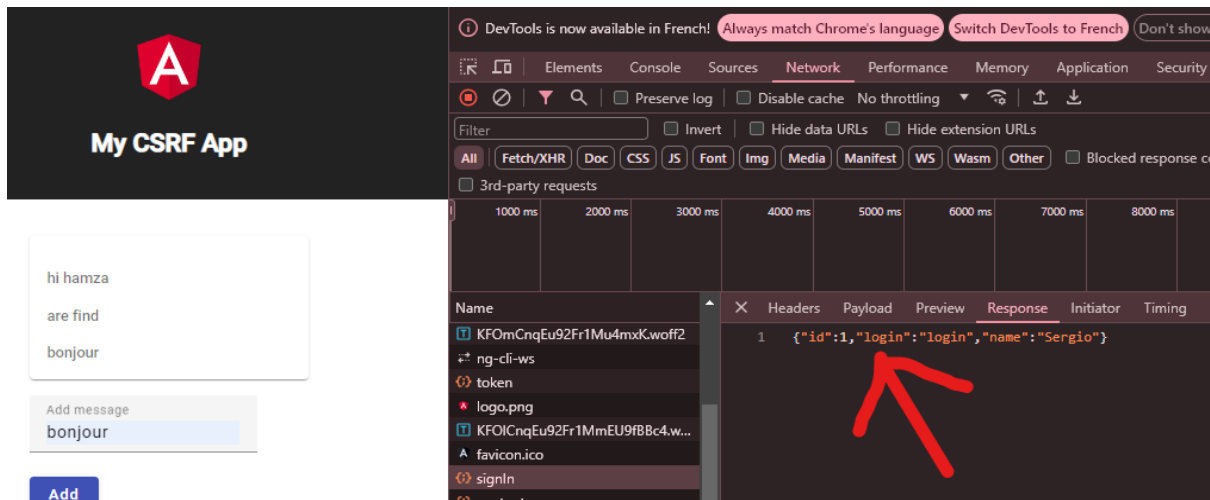


- le token généré :

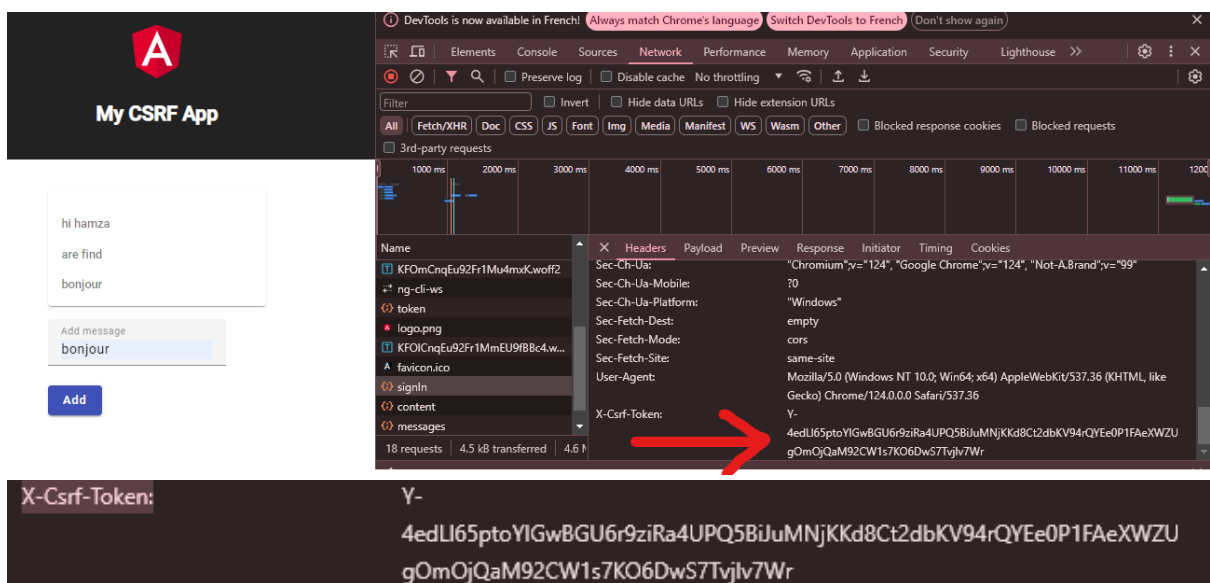


- Si l'utilisateur est bien authentifié, le serveur envoie la réponse suivante au format JSON, contenant votre identifiant, votre login et votre nom.

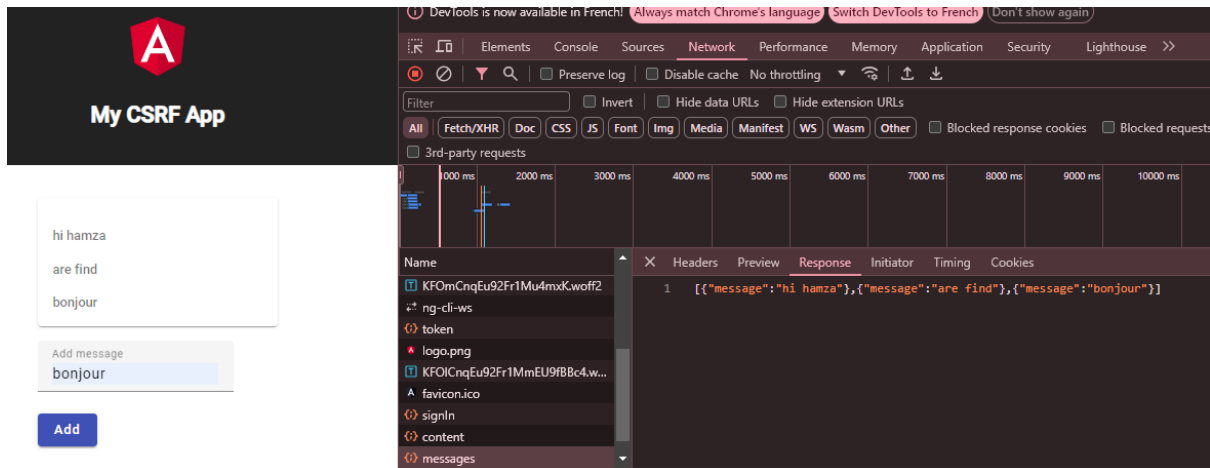




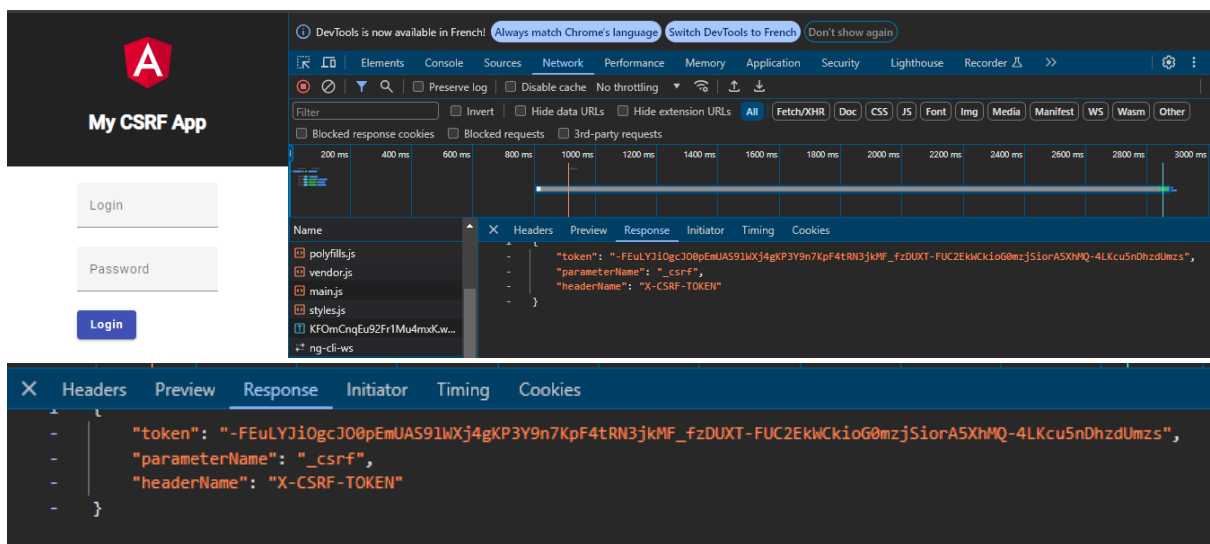
- Si l'utilisateur veut envoyer un message (requête POST HTTP), le token CSRF sera envoyé dans l'en-tête de la requête HTTP. Ce token CSRF sera vérifié par le serveur pour vérifier son existence et s'il est associé à la session de l'utilisateur connecté.
- Par exemple, l'utilisateur va envoyer un message "bonjour", qui sera le contenu de la requête HTTP avec un en-tête contenant le token CSRF (X-CSRF-Token).



- La réponse envoyée par le serveur au format JSON contient l'ensemble des messages de cet utilisateur.



- 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 de celui généré pour les autres sessions, car le token CSRF généré dépend de la date système.



## B- Implementation en PHP

### Étapes pour créer une application PHP Full-stack vulnérable à CSRF :

Les technologies de base utilisées dans le développement web : PHP, HTML et CSS :

**PHP (Hypertext Preprocessor)** est un langage de script côté serveur largement utilisé pour le développement web. Il est intégré au HTML et est utilisé pour gérer la logique côté serveur, comme l'interaction avec des bases de données, la gestion des sessions utilisateur et le traitement des formulaires.

**HTML (HyperText Markup Language)** est le langage standard utilisé pour créer et structurer le contenu des pages web. Il permet de définir la structure des documents web avec des éléments tels que les paragraphes, les en-têtes, les formulaires, et les liens hypertexte.

**CSS (Cascading Style Sheets)** est utilisé pour styliser les documents HTML. Il permet de définir l'apparence visuelle des éléments HTML, comme les couleurs, les polices, les marges et les alignements.

## **Création d'une application vulnérable à CSRF**

Pour illustrer une application vulnérable 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.

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.


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.

## **Démonstration de l'attaque sans et avec CSRF**

-page d'accueil de l'application :

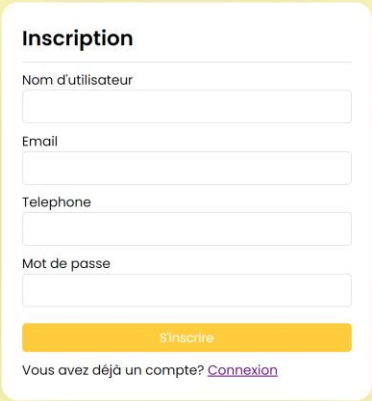


-page de connexion :



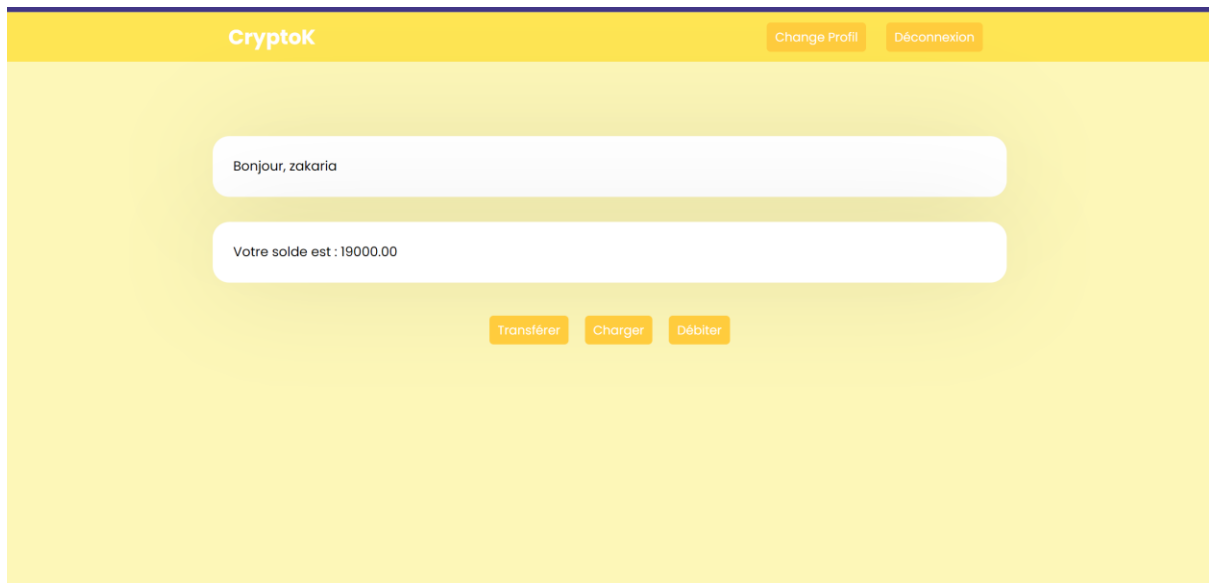
A login form titled "Connexion" is centered on a yellow background. The form is a white rounded rectangle containing the following elements: a title "Connexion", an "Email" label above a text input field, a "Mot de passe" label above a text input field, an orange "Connexion" button, and a link "Pas de compte? [Créer un compte](#)".

-page d'enregistrer :

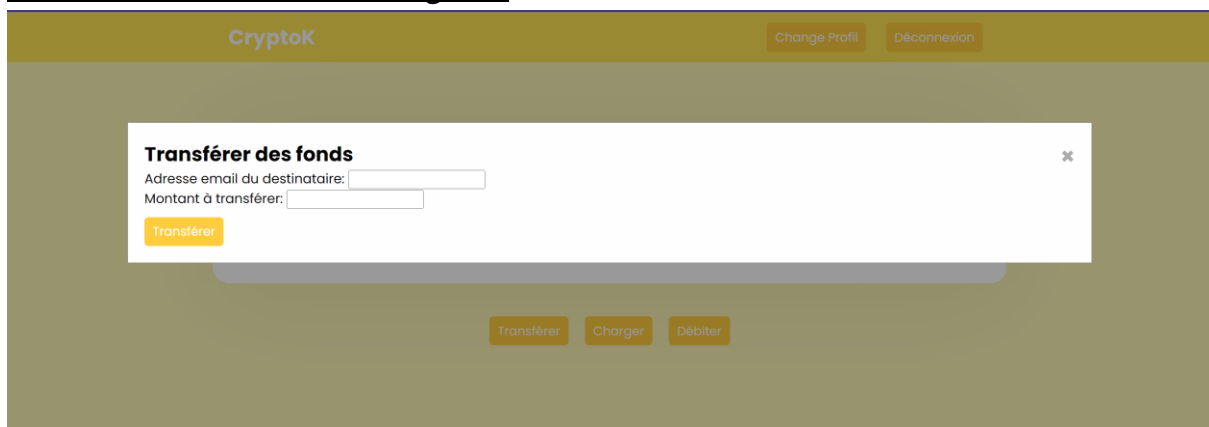


A registration form titled "Inscription" is centered on a yellow background. The form is a white rounded rectangle containing the following elements: a title "Inscription", a "Nom d'utilisateur" label above a text input field, an "Email" label above a text input field, a "Telephone" label above a text input field, a "Mot de passe" label above a text input field, an orange "S'inscrire" button, and a link "Vous avez déjà un compte? [Connexion](#)".

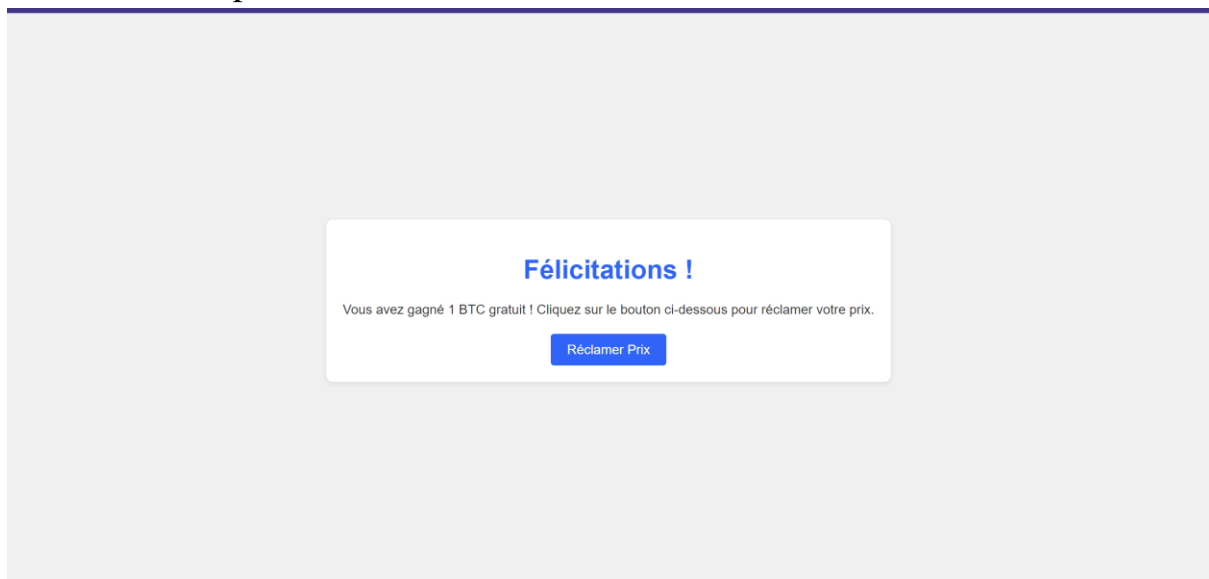
-page du compte utilisateur :



### -formulaire de transfert d'argent :



### -Scénario sans protection CSRF :



localhost/sans-csrf/attaqueb.php

Félicitations !

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

Réclamer Prix

DevTools is now available in French!

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

Elements Console Sources Network Performance >>

Filter

☐ Preserve log ☐ Disable cache No throttling ☐ 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

Name	Status	Type	Initiator	Size	Time	Waterfall
content.js	200	script	extension-load	1.1 MB	39 ms	
executor.js	200	script	content.js:85	1.4 kB	25 ms	
favicon.ico	200	x-icon	Other	31.2 kB	21 ms	
js.js	200	script	content.js:32	1.3 kB	4 ms	
js.js	200	script	content.js:32	2.0 kB	3 ms	
dom.js	200	script	content.js:32	1.3 kB	12 ms	
dom.js	200	script	content.js:32	2.0 kB	16 ms	
js.js	200	script	content.js:32	1.3 kB	2 ms	

9 requests | 1.1 MB transferred | 1.1 MB resources | Finish: 6.90 s | DOMContentLoaded: 197 ms | Load: 533 ms

Console What's new

Highlights from the Chrome 124 update

Scroll-driven animations support

The Animations panel now lets you inspect scroll-driven animations.

New Autofill panel

Le montant est réduit

CryptoK

Change Profil Déconnexion

Bonjour, zakaria

Votre solde est : 18000.00

Transférer Charger Débitier

CryptoK

Change Profil Déconnexion

Bonjour, zakaria

Votre solde est : 17000.00

Transférer Charger Débitier

DevTools is now available in French!

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

Elements Console Sources Network Performance >>

Filter

☐ Preserve log ☐ Disable cache No throttling ☐ 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

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
process_transfer.php	General						
home.php?success=1	Request URL:			http://localhost/sans-csrf/process_transfer.php			
style.css	Request Method:			POST			
css2?family=Poppins:wght@20...	Status Code:			302 Found			
pxiEyp8kv8IHgFvrlfegwof2	Remote Address:			[::1]:80			
pxiByp8kv8IHgFvrlCz7Z1xIFQ...	Referrer Policy:			strict-origin-when-cross-origin			
content.js	Response Headers						
executor.js	Cache-Control:			no-store, no-cache, must-revalidate			
js.js	Connection:			Keep-Alive			
dom.js	Content-Length:			1			
dom.js	Content-Type:			text/html; charset=UTF-8			
js.js	Date:			Wed, 22 May 2024 21:55:13 GMT			
dom.js	Expires:			Thu, 19 Nov 1981 08:52:00 GMT			
js.js	Keep-Alive:			timeout=5, max=100			
js.js	Location:			home.php?success=1			

14 requests | 1.1 MB transferred

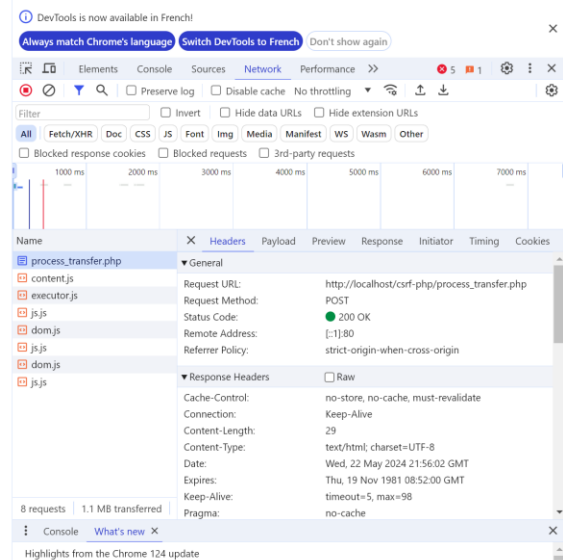
Console What's new

Highlights from the Chrome 124 update

## -Scénario avec protection CSRF :

CSRF token validation failed

CSRF token validation failed



## Partie code

### Sans csrf token

L'application web permet aux utilisateurs de transférer de l'argent en remplissant un formulaire et en soumettant une requête POST au serveur. Cette application ne contient aucune mesure de protection contre CSRF.

Un attaquant crée une page web qui envoie une requête POST au serveur de la banque avec des paramètres spécifiques pour transférer de l'argent au compte de l'attaquant.

L'attaquant envoie un email à l'utilisateur légitime, l'invitant à cliquer sur un lien pour réclamer une fausse récompense (1 BTC).

*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.*

### Formulaire sans protection CSRF :

```

<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>

```

## Traitement de la requête sans validation CSRF :

```

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     }
38 }

```

## Avec csrf token

L'application web permet aux utilisateurs de transférer de l'argent en remplissant un formulaire et en soumettant une requête POST au serveur. Cette application inclut des mesures de protection contre CSRF, telles que des tokens CSRF.

Un attaquant crée une page web qui envoie une requête POST au serveur de la banque avec des paramètres spécifiques pour transférer de l'argent au compte de l'attaquant, mais sans inclure le token CSRF valide.

L'attaquant envoie un email à l'utilisateur légitime, l'invitant à cliquer sur un lien pour réclamer une fausse récompense (1 BTC).



*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.*

### Formulaire sans protection CSRF :

```
<!-- Formulaire de transfert (initialement 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>
```

### Traitement de la requête sans validation CSRF :

```
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   // Récupérez le solde actuel de l'utilisateur
22   $query = mysqli_prepare($connection, "SELECT balance FROM utilisateurs WHERE email = ?");
23   mysqli_stmt_bind_param($query, "s", $email);
24   mysqli_stmt_execute($query);
25   $result = mysqli_stmt_get_result($query);
26   if ($result && $row = mysqli_fetch_assoc($result)) {
27     $currentBalance = $row['balance'];
28
29     // Soustrayez le montant transféré du solde de l'utilisateur
30     $newBalance = $currentBalance - $amount;
31
32     // Mettez à jour le solde de l'utilisateur dans la base de données
33     $updateQuery = mysqli_prepare($connection, "UPDATE utilisateurs SET balance = ? WHERE email = ?");
34     mysqli_stmt_bind_param($updateQuery, "ds", $newBalance, $email);
35     mysqli_stmt_execute($updateQuery);
36
37     // Redirigez l'utilisateur vers la page d'accueil avec un message de succès
38     header("Location: home.php?success=1");
39     exit();
40   } else {
41     // Gestion des erreurs de requête
42     header("Location: home.php?error=database_error");
43     exit();
44   }
45 } else {
46   // Redirection si les données du formulaire ne sont pas soumises
47   header("Location: home.php");
48   exit();
49 }
50 ?>
51
```

## **4 . Analyse de résultat :**

### **4.1- En java**

L'implémentation de la protection CSRF dans cette application s'est révélée efficace pour prévenir les attaques de type CSRF. En injectant des tokens CSRF dans les en-têtes de chaque requête, nous avons pu empêcher les attaquants d'exécuter des actions non autorisées, telles que des modifications de données ou des transferts de fonds non autorisés.

Avant l'implémentation de la protection CSRF, cette implémentation présentait des vulnérabilités potentielles aux attaques CSRF, exposant ainsi les utilisateurs à des risques de sécurité. Cependant, grâce à l'utilisation de tokens CSRF dans chaque requête, nous avons considérablement réduit ces risques, renforçant ainsi la sécurité de l'application.

### **4-2 En php**

#### **Scénario sans protection CSRF :**

Dans le scénario sans protection CSRF, l'application web de la permet aux utilisateurs de transférer de l'argent en remplissant un formulaire et en soumettant une requête POST au serveur. Sans aucune protection CSRF en place, un attaquant peut facilement exploiter cette vulnérabilité en envoyant un email de phishing à l'utilisateur. Cet email contient un lien malveillant qui, lorsqu'il est cliqué, envoie une requête POST au serveur de la banque pour transférer de l'argent au compte de l'attaquant. Cette requête utilise les cookies de session de l'utilisateur pour apparaître comme légitime aux yeux du serveur. En conséquence, le transfert d'argent non autorisé est effectué sans la connaissance ou le consentement de l'utilisateur. Ce scénario illustre clairement la gravité des conséquences potentielles des attaques CSRF, incluant la perte financière et la compromission de la sécurité des utilisateurs.

#### **Scénario avec protection CSRF :**

Dans le scénario avec protection CSRF, l'application web inclut des mesures de sécurité telles que des tokens CSRF pour chaque session utilisateur. Le token CSRF est inclus dans les formulaires de l'application et validé par le serveur lors de la soumission de la requête. Lorsque l'attaquant envoie le même email de phishing, la requête malveillante qui en résulte ne contient pas le token CSRF valide. Le serveur, en validant chaque requête POST, détecte l'absence ou l'invalidité du token CSRF et rejette la requête. Ainsi, le transfert d'argent non autorisé échoue, et l'utilisateur reste protégé contre l'attaque. Ce scénario démontre l'efficacité des protections CSRF dans la prévention des attaques, en garantissant que seules les requêtes authentiques sont traitées par le serveur.

## **5- Conclusion**

En conclusion, ce rapport a exploré en profondeur les attaques Cross-Site Request Forgery (CSRF) et leur impact sur les applications web, en mettant l'accent sur une simulation réalisée avec PHP et JAVA . Nous avons démontré comment une application vulnérable peut être facilement exploitée pour effectuer des actions non désirées à l'insu de l'utilisateur. Cette vulnérabilité peut entraîner des

conséquences graves, telles que la modification non autorisée de données et des pertes financières significatives.

La mise en place de mesures de protection, telles que les tokens CSRF, a été présentée comme une solution efficace pour contrer ces attaques. En incluant des tokens CSRF dans les formulaires et en validant leur présence et leur validité côté serveur, les développeurs peuvent empêcher les requêtes malveillantes d'être traitées. Cette approche assure que seules les requêtes authentiques et intentionnelles sont acceptées, renforçant ainsi la sécurité des applications web.

L'implémentation des protections CSRF a été démontrée non seulement avec PHP, mais également avec des technologies modernes comme Java, Spring Boot, Spring Security et Angular. Cette diversité technologique souligne l'importance et la flexibilité des solutions de sécurité CSRF, applicables à divers environnements de développement.

la protection contre les attaques CSRF est essentielle pour garantir la sécurité et la confiance des utilisateurs dans les applications web. Les développeurs doivent intégrer de telles mesures de sécurité dans leurs projets pour prévenir les attaques potentielles et protéger les données sensibles des utilisateurs. En adoptant une approche proactive en matière de sécurité, les entreprises peuvent assurer la robustesse de leurs applications et maintenir la confiance de leurs clients.