Cross-Site Request Forgery

Description

Le {Cross-Site Request Forgery} (CSRF ou XSRF) est une technique d'attaque Web qui force un utilisateur authentifié sur une application vulnérable à réaliser une action en abusant de la confiance qu'a l'application envers les requêtes de ses clients.

L'action forcée dépend du contexte, mais peut être le changement de certains paramètres du compte, le virement bancaire vers un compte arbitraire ou encore l'attribution de nouveaux privilèges à un utilisateur de l'application. Étant donné que la cible est un utilisateur et non l'application en elle-même, cette attaque appartient à la famille des vulnérabilités dites “côté client”.

Ces attaques existent car les navigateurs envoient automatiquement les cookies associés aux sessions authentifiées, sans vérifier l'origine de la requête. Pour la réaliser, un utilisateur malveillant va héberger, au travers d'une page Web, un formulaire HTML permettant d'effectuer la requête réalisant l'action désirée. Ce formulaire est accompagné d'un script JavaScript qui va soumettre le formulaire dès que la page est chargée.

```
<h1>Page hébergée sur hackerman.hack pour CSRF sur example.ex</h1>
<form action="https://example.ex/edit-account" method="post">
  <input type="password" name="password" value="easypass123">
  <input type="password" name="confirm-password" value="easypass123">
  <button type="submit">Edit</button>
</form>
<script>
    document.forms[0].submit();
</script>
```

Exemple de charge utilisée pour forcer le changement de mot de passe sur **exemple.ex**.

Si l'utilisateur qui consulte la page malveillante est authentifié sur l'application ciblée, la requête contiendra son cookie de session. La requête est ensuite reçue par le serveur qui ne fait pas de distinction entre une requête légitime et malicieuse. Enfin, l'action désirée par l'attaquant est effectuée.

L'attaque nécessite que la cible se dirige sur la page hébergée par l'attaquant. En général, c'est au travers des mails de phishing que le CSRF est réalisé. Cependant, d'autres moyens peuvent être utilisés pour diffuser l'attaque, notamment via un service de tickets interne ou encore via une messagerie compromise d'un des employés de l'entreprise ciblée.

Aujourd'hui, plusieurs protections existent pour se protéger des attaques par CSRF. L'attaque étant peu connue des développeurs, celle-ci reste donc très présente dans les applications actuelles.

Pré-requis d’exploitation

Pour mettre en œuvre cette attaque, il est nécessaire d'avoir accès au minimum à un compte de l'application à tester. Ce compte doit avoir les privilèges nécessaires pour effectuer l'action vulnérable aux attaques CSRF.

Compétences nécessaires

- Connaissances de base du protocole HTTP ;
- Notions sur les langages JavaScript et HTML.

Outils nécessaires

- Accès à une console de navigateur ou à un outil de modification et/ou d’interception de requêtes comme **Burp Suite**.

Flux d’exécution

Explorer

La détection d'un CSRF est assez simple, il suffit de parcourir les pages de l'application web avec un compte utilisateur légitime (idéalement deux pour effectuer des tests) à la recherche de fonctionnalités intéressantes (transferts d’argent, changement de mot de passe, etc.) en accord avec les points suivants :

- **Une action exploitable :** l'action réalisée doit être sensible. Un CSRF sur un changement de filtre de recherche n'aura à priori aucune répercussion utile pour l'attaquant, tandis qu'un changement d'adresse email est bien plus intéressant d'un point de vue offensif ;
- **L'authentification est basée uniquement sur un cookie :** l'attaque par CSRF repose sur le fait de réaliser une requête HTTP sous le nom de la victime. Si l'application utilise d'autres facteurs de vérification pour effectuer l'action, il est probable que celle-ci ne passe pas. Par exemple pour un virement bancaire vulnérable qui demanderait à l'utilisateur de confirmer celui-ci sur l'application de son téléphone, rendant ainsi l'attaque impossible ;
- **Le formulaire ne contient que des paramètres prédictibles :** le formulaire ne doit pas envoyer de jeton particulier ou d'informations imprédictibles. Une valeur imprédictible pour l'attaquant dans le formulaire rendrait l'attaque caduque, car l'application n'accepterait pas d'effectuer l'action, un des champs étant erroné. Par exemple, un changement de mot de passe qui demanderait l'ancien mot de passe pour réaliser le changement.

Expérimenter

Si l'on trouve une fonctionnalité qui respecte les trois points de l'étape précédente, il est relativement probable qu'un CSRF soit possible. Pour s'en assurer totalement, il est nécessaire de le tester. Pour cela, le mieux est d'utiliser deux comptes obtenus légitimement. L'un servira à mettre en place l'attaque, l'autre à tester :

- Créer le code HTML / JavaScript qui permet d'effectuer automatiquement l'action ciblée avec des valeurs prédéfinies dans les champs.
- Mettre en place le code sur un serveur accessible par la victime. En CTF, on peut utiliser des services Web publics tels que [Beeceptor→https://beeceptor.com/], [Pipedream→https://pipedream.com/requestbin] ou encore [ngrok→https://ngrok.com/].
- Avec le compte de la victime authentifiée sur l'application, visiter la page qui contient la charge d'attaque.
- Si après avoir navigué sur la page, l'action est réalisée sans avoir eu d'interaction supplémentaire, l'attaque est possible.

<quote> {Pro-tips} : pour utiliser deux comptes authentifiés sur la même application, vous pouvez utiliser la navigation privée, ou les [containers tabs→https://support.mozilla.org/fr/kb/utiliser-conteneurs-firefox]. Sinon, il est possible d'utiliser l'extension [Pwnfox→https://addons.mozilla.org/en-US/firefox/addon/pwnfox/] disponible seulement pour le navigateur Firefox. </quote>

<h4>Exploiter</h4> Consulter les solutions de chaque challenge.

consequences_potentielles} Les conséquences potentielles de ce type d’attaque dépendent grandement de la nature de l’application ciblée. Par exemple, dans le cas d’une application bancaire vulnérable, il peut être possible : -* D’usurper un compte utilisateur (par exemple en exploitant un formulaire de changement de mot de passe ou d'email) ; -* De modifier les informations de compte (telles que les adresses de contact ou encore les informations personnelles) ; -* De déclencher des transactions financières (comme des achats ou des paiements) sans le consentement de l’utilisateur ; -* D’ajouter des bénéficiaires ou des comptes tiers pour les transactions financières.

contres-mesures} Les contre-mesures suivantes peuvent être mises en œuvre : -* utiliser_des_jetons_cryptographiques_pour_associer_une_demande_a_une_action_specifique Le jeton peut être régénéré à chaque demande, de sorte que si une demande avec un jeton invalide est reçue par le serveur, elle peut être écartée de manière fiable. Le jeton est considéré comme invalide s’il est arrivé avec une demande autre que l’action à laquelle il était censé être associé. -* utiliser_un_second_facteur_d_authentification L’utilisateur peut être invité à confirmer une action chaque fois qu’une action concernant des données potentiellement sensibles est invoquée. De cette façon, même si l’attaquant parvient à faire en sorte que l’utilisateur clique sur un lien malveillant et demande l’action souhaitée, l’utilisateur peut encore refuser la confirmation et ainsi être averti que son compte a potentiellement été compromis. -* mettre_en_place_l_option_same-site_sur_les_cookies_de_session L'option {Same-Site} peut être précisée lors de l'attribution d'un cookie par le serveur. Cet attribut permet de réduire le domaine de diffusion du cookie associé selon trois niveaux de politiques : {None}, {Lax} et {Strict}.

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités