



UNIVERSITÉ
DE LORRAINE

UFR MATHÉMATIQUES INFORMATIQUE
MÉCANIQUE ET AUTOMATIQUE

Étude des vulnérabilités applicatives : failles XSS et CSRF

Rüdiger MORIN-DOCTER* et Thomas PEDROTTI**

*Université de Lorraine, UFR MIM
Master 1 Informatique
3 Rue Augustin Fresnel, 57070 Metz
rudiger.morin-docter7@etu.univ-lorraine.fr

**Université de Lorraine, UFR MIM
Master 1 Informatique
3 Rue Augustin Fresnel, 57070 Metz
thomas.pedrotti2@etu.univ-lorraine.fr

Travail encadré par : Mme HERRMANN

Résumé

Dans cet article, après avoir présenté une étude bibliographique des attaques de type XSS et CSRF, nous présentons notre maquette de test puis notre méthodologie de détection de failles CSRF. Dans un second temps, nous présentons la mise en place de notre logique de détection dans une extension du logiciel Burp Suite et présentons les résultats obtenus sur notre maquette avec ce nouvel outil.

Mots-clés : CSRF, XSS, Sécurité, CWE, Burp Suite

1 Introduction

La sécurité informatique est un sujet dont l'importance grandit chaque année. Tous les systèmes traitant des données doivent aujourd'hui présenter un degré plus ou moins important de sécurisation et de protection contre des attaquants potentiels. En effet, il n'existe pas de convention de Genève en informatique : un attaquant peut tout se permettre et la victime ne peut pas contre-attaquer. Ainsi, de nombreuses failles sont découvertes et exploitées à des fins malveillantes. Parmi ces failles, il existe les failles de type "Cross-Site Scripting" et "Cross-Site Request Forgeries".

Les attaques "Cross-Site Scripting" et "Cross-Site Request Forgeries" sont respectivement appelées XSS et CSRF. Ce sont des attaques qui ont gagné en popularité au cours des dernières années auprès des personnes peu scrupuleuses. De ce fait, ces attaques, jusqu'alors sous-estimées, gagnent en visibilité, et les communautés de développement commencent à proposer des méthodes pour se protéger de ces attaques, visant les applications web.

Dans ce document, après une étude bibliographique de différents articles sur le sujet, nous mettons d'abord en place une maquette sur laquelle nous testons des attaques CSRF visant les fonctionnalités d'authentification. Puis, nous mettons en œuvre des méthodes de détection de failles CSRF existantes avant de proposer un algorithme de détection automatisé. Enfin, nous développons notre algorithme afin de créer une extension Burp Suite capable de générer des requêtes CSRF lors de la navigation sur l'application web à tester.

2 Etude Bibliographique des failles XSS et CSRF

La littérature scientifique est riche en articles et recherches effectuées sur les XSS et CSRF.

2.1 Classification des CSRF et XSS

La Common Weakness Enumeration (CWE) propose une liste démonstrative des problèmes les plus communs et les plus impactants qui ont été expérimentés au cours des 2 dernières années [CWE, 2020]. Ces faiblesses sont dangereuses car elles sont souvent faciles à trouver, exploiter et peuvent permettre à des adversaires de complètement prendre le contrôle sur un système, de voler des données ou d'empêcher le bon fonctionnement d'une application. Les CWE qui nous intéressent sont les CWE-79 et CWE-352, respectivement classées en première et huitième positions, attestant ainsi de leur popularité auprès des agresseurs potentiels.

La CWE-79 représente la mauvaise neutralisation d'entrée lors du chargement de page web [Eric Dalci, 2021b]. On parle donc ici d'attaque de type XSS.

XSS signifie "Cross-site Scripting". Une vulnérabilité XSS apparaît lorsque des données non fiables entrent dans l'application web et que cette application web ne gère pas comme il faut ces données. Si l'application web n'empêche pas qu'il y ait des données exécutables par un navigateur, alors elle va générer une page web contenant ces données non fiables, et lorsqu'un utilisateur va visiter le site, le script malicieux sera exécuté.

Il existe différents types d'attaque XSS :

- Un XSS réfléchi est non permanent et est le plus commun. Il demande des données au client qui sont directement utilisées par le script malveillant pour produire une page résultat.
- Un XSS stocké est permanent et permet des attaques plus puissantes. Les données fournies par l'utilisateur sont stockées sur le serveur et sont ré-affichées à postériori.
- Un XSS basé sur le DOM (Document Object Model). Pour rappel, une DOM est une interface de programmation permettant à des scripts de modifier le contenu d'un navigateur web. Il arrive que le client injecte lui-même un XSS sur la page web.

En résumé, une attaque XSS est due à des données non fiables, notamment du Javascript, qui entrent dans une application web et qui ont un effet malveillant sur le client lors du chargement de l'application par un navigateur.

La CWE-352 représente les Cross-Site Request Forgeries [Eric Dalci, 2021a]. Etant donné que l'article [Zeller and Felten, 2008] parle déjà en détail de ce type d'attaque, nous allons être brefs. La description qui est faite ici d'une attaque CSRF est qu'un attaquant fait charger une requête à un client sans que ce dernier n'en ait conscience. Cette requête sera traitée comme une requête provenant du navigateur de l'utilisateur innocent et sera considérée comme voulue. Différents vecteurs peuvent être utilisés, comme une URL ou bien le chargement d'une image. Il est également indiqué que ce type d'attaque a lieu lorsque le site web ne peut pas vérifier si une requête valide a été envoyée intentionnellement par l'utilisateur qui a soumis la requête.

2.2 Exploitation et prévention des CSRF

Dans l'article de William Zeller et Edward W. Felten intitulé "Cross-Site Request Forgeries : Exploitation and Prevention" [Zeller and Felten, 2008] de 2008, on apprend de multiples choses sur les attaques de type CSRF. Au préalable, ces auteurs nous fournissent une autre définition de CSRF : "Un site malicieux attaque le navigateur de l'utilisateur, provoquant une action non voulue sur un site sûr". Ils décrivent ce type d'attaque comme étant un "Géant sommeillant", ceci étant dû au manque de protection efficace contre cette attaque ainsi qu'aux développeurs ignorants celle-ci. Ils indiquent également qu'une protection contre XSS ne protège pas contre CSRF, alors que de nombreux développeurs confondent les deux.

Nous y apprenons aussi qu'une attaque CSRF a la même "puissance" que celle de l'utilisateur innocent sur le site attaqué. Le but d'une attaque CSRF est de faire en sorte qu'un utilisateur innocent, connecté sur le site ciblé, charge avec son navigateur une requête non voulue par l'utilisateur qui aura un effet sur le site cible. Ceci peut être obtenu, par exemple, en visitant un site malveillant qui charge des images dans lesquelles sont camouflées les requêtes visant le site cible. En général, une attaque CSRF a lieu sans que l'utilisateur innocent ne se rende compte qu'il devient le catalyseur de l'attaque.

CSRF vise en particulier le mécanisme d'authentification. Le problème principal permettant la plupart des attaques est dû au fait que certains sites authentifient le navigateur et pas l'utilisateur. Ainsi, un attaquant peut utiliser le navigateur authentifié de l'utilisateur pour arriver à ses fins.

Pour qu'une attaque soit couronnée de succès, il faut que l'utilisateur servant de catalyseur soit connecté sur le site cible, puis visite le site "attaquant". Si le serveur du site cible accepte des requêtes GET, alors il suffira simplement d'utiliser des URL pour attaquer le site cible. S'il

utilise des requêtes POST, il faudra forger des formulaires qui sont automatiquement soumis lors de la visite.

Les auteurs indiquent également qu'ils sont réussis à trouver des failles CSRF plus ou moins graves sur plusieurs sites, et notamment sur NYTimes.com, le site d'ING, sur Meta-Filter et sur Youtube. Evidemment, l'article datant de 2008, les failles ont été corrigées, mais ceci indique que ce type d'attaque touche potentiellement n'importe quel site, même ceux des grandes compagnies.

Pour en finir avec cet article, nous avons également étudié avec intérêt la partie où les auteurs décrivent des méthodes de prévention contre les attaques CSRF.

La première méthode proposée est la plus efficace : intégrer le mécanisme de défense du côté du serveur. Voici les règles qu'elle suit :

- Il faut utiliser les requêtes GET uniquement pour récupérer des données, et jamais pour les modifier.
- Il faut que toutes les requêtes POST utilisent un système de vérification sur cookies qui possèdent des valeurs pseudo-aléatoires générées lorsque l'utilisateur se connecte au site.
- Cette valeur pseudo-aléatoire doit être indépendante du compte de l'utilisateur.

Cette méthode est considérée comme étant légère et très efficace, le cookie étant "server-wide" pour l'utilisateur. Ainsi, un attaquant ne pourra pas forger de requête sans connaître la valeur pseudo-aléatoire générée par le site cible, étant donné qu'elle sert de "token" pour toutes les requêtes qu'effectuera l'utilisateur.

La deuxième méthode proposée est moins efficace : utiliser un plugin côté client pour intercepter et traiter les requêtes HTML. Ce plugin suit les règles suivantes :

- Si la requête n'est pas un POST, on l'autorise.
- Si le site requérant et le site cible sont de même origine, on autorise.
- Si le site requérant est autorisé par l'Adobe cross-domain policy¹ à faire une requête sur le site cible, on autorise.
- Sinon, un pop-up indique la requête et demande si on veut l'effectuer.

Cette méthode a pour but de protéger l'utilisateur uniquement, et ne fonctionne que pour les requêtes POST. Il faut donc noter que les attaques CSRF utilisant GET fonctionneront toujours.

Cet article, riche en informations, nous a permis de découvrir et de mieux comprendre le fonctionnement des attaques CSRF. Pour compléter cet article, nous avons lu un autre article parlant de la détection d'attaques CSRF.

2.3 Détection de types d'attaques CSRF

Après plusieurs recherches et la lecture de différents articles dans le domaine des CSRF et XSS, nous avons choisi un article datant de 2017 intitulé "Large-scale Analysis and Detection of Authentication Cross-Site Request Forgeries" et proposé par Sudhodanan et al [Sudhodanan et al., 2017].

Dans cet article, les chercheurs se focalisent sur les attaques CSRF visant les fonctionnalités d'authentification et de gestion d'identité. Ils les appellent les "Auth-CSRF".

1. Il s'agit d'un fichier xml qui permet d'indiquer si on peut utiliser des données venant d'un autre domaine. Lien vers un pdf explicatif : adobe.com/devnet-docs/acrobatetk/tools/AppSec/CrossDomain_PolicyFile_Specification.pdf

Ils donnent au préalable plusieurs méthodes de défense contre les attaques CSRF :

- Validation par Token secret. Cette méthode reprend la première méthode de [Zeller and Felten, 2008] et il est clair que cette méthode est devenue un standard de protection.
- Validation du HTTP Referer-Origin Header. Quand une action de modification est enclenchée, on vérifie si le domaine dont elle provient est de confiance.
- Validation avec HTTP Header personnalisé. En créant nos propres headers, on peut vérifier côté serveur que les headers des requêtes reçues sont bien correspondants à ceux que l'on a inventés.
- Mécanismes (semi-)automatiques de protection. Ceci représente tous les plugins et programmes conçus pour limiter ou empêcher les attaques CSRF. Ils sont, pour la plupart, peu pratiques d'usage.

Nous pouvons donc voir que les fondations posées par [Zeller and Felten, 2008] ont mené à des pratiques de protection plus évoluées. Mais cet article parle, par la suite, des méthodes d'analyse et de détection de failles CSRF.

L'équipe a d'abord défini sept processus communs qui, lorsqu'ils sont mal implémentés, présentent des vulnérabilités.

- "Form-based Registration" : l'attaquant envoie un formulaire au nom de l'utilisateur. Cette attaque est particulièrement efficace sur les sites qui nous authentifient directement après inscription.
- "URL-based Account Activation" : l'attaquant se crée un compte sur le site cible. Il implémente l'URL d'activation qu'il a reçue dans un site malveillant. Quand un utilisateur innocent se connecte sur le site attaquant, il charge l'url d'activation et est connecté sur le compte appartenant à l'attaquant, qui aura accès à l'historique ou autres fonctionnalités. Cette attaque ne marche que sur les sites qui nous authentifient directement après activation d'un compte.
- "Form-based Login" : L'attaquant se crée un compte ressemblant à celui d'un utilisateur existant. L'utilisateur va se connecter sur le site ciblé puis va visiter le site malveillant. Un formulaire de connexion au compte de l'attaquant est alors envoyé depuis le navigateur de l'utilisateur innocent. Il va alors se connecter avec le compte de l'attaquant sans s'en rendre compte.
- "SSO Login" : Le mécanisme de Single Sign-on (SSO), ou d'Authentification unique, représente la possibilité de se connecter avec un compte d'une autre plateforme existante (par exemple, se connecter avec son compte Google). L'attaquant peut forger une requête pour utiliser ce moyen de connexion dans le dos de l'utilisateur, qui sera donc connecté sur le compte SSO de l'attaquant.
- "SSO-Based Account association" : C'est lorsque le site propose de lier son compte avec un compte Google ou autre. Si un utilisateur est connecté avec son compte sur le site cible, et visite le site de l'agresseur, dans lequel il a forgé une requête pour lier son compte SSO, alors l'attaquant peut lier son compte SSO à un compte utilisateur innocent et avoir plus ou moins de pouvoir.
- "Primary Email Change" : Très répandue, la possibilité de changer son adresse mail principale est une des vulnérabilités les plus dangereuses. Si un attaquant est capable de forger une requête pour modifier l'email actuel par un email de son choix, alors un utilisateur innocent qui visitera le site malveillant verra l'email principal de son

compte modifié, et ainsi l'attaquant aura le contrôle total du compte de cet utilisateur (en utilisant la fonction "mot de passe oublié").

- "Password Change" : De manière similaire à l'attaque ci-dessus, si un attaquant peut forger une requête permettant de changer le mot de passe d'un compte, alors n'importe quel utilisateur innocent visitant le site attaquant verra son mot de passe modifié et ainsi l'attaquant aura le contrôle du compte.

Après l'identification de ces processus, l'équipe prépare des méthodologies de tests pour vérifier la résistance d'un site web à chacune de ces vulnérabilités Auth-CSRF potentielles. Nous verrons ces méthodologies plus en détail dans la partie 4.1.

Enfin, l'équipe teste ses méthodologies sur le top 300 des sites d'après Alexa.com². Sur ces 300 sites, leur expérience a fonctionné sur 133 d'entre eux, et sur ces 133 sites, 90 possèdent au moins une vulnérabilité parmi les sept décrites ci-dessus.

Ceci indique qu'entre 2008 et 2017, même s'il y a eu des progrès en matière de protection contre les attaques CSRF, il est relativement commun de tomber sur des sites présentant encore des failles. De plus, d'après [CWE, 2020], les failles CSRF et XSS font partie des attaques les plus fréquentes de 2021. Ainsi, le travail que nous pouvons fournir dans ce domaine est d'actualité.

Avec les informations et les découvertes que nous avons faites en lisant tous ces articles, nous allons d'abord mettre en place une maquette et tester les différentes attaques CSRF.

3 Maquette et mise en oeuvre

En nous inspirant des sept failles potentielles décrites dans la partie 2.2, nous avons créé une maquette de site "gentil", avec toutes les fonctionnalités nécessaires, ainsi qu'un site "méchant", sur lequel nous mettons des images qui, lorsque l'on clique dessus, chargent des requêtes CSRF visant les différents systèmes implémentés dans notre site "gentil".

Le site "gentil" fait un simple affichage d'images de fleurs. Lorsqu'un utilisateur se connecte au site, on affiche alors plus d'images de fleurs. Il y a également un petit système pour indiquer quel utilisateur est actuellement connecté.

Le site "méchant" est un site qui affiche des images de squelettes de dinosaures. Lorsque l'utilisateur clique sur une image, une requête CSRF visant le site "gentil" est chargée par le navigateur. Il faut noter qu'ici, la requête CSRF n'est lancée que lorsque l'utilisateur clique dessus. Nous avons construit de telle manière le site "méchant" afin de pouvoir plus facilement travailler, car en réalité le site attaquant lancerait directement la requête lorsque l'image est chargée par le navigateur du client.

Nous avons également lié le site "gentil" à une base de données MySQL en utilisant Easy-Php³. Ainsi, nous pouvons voir les différents résultats de nos attaques dans la base de données.

Il y a sept types d'attaques Auth-CSRF à tester, nous allons en détailler deux dans la suite de cette section. Les autres attaques seront décrites dans les annexes⁴ de ce document, afin de

2. Système de classification des sites web par leur trafic mis à jour tous les mois, géré par Amazon. Lien : alexa.com/topsites

3. Environnement de développement PHP. Lien : <https://www.easypHP.org/>

4. Annexe E - Démonstration des attaques Auth-CSRF.

fournir toutes les informations possibles à toute personne souhaitant reprendre ou améliorer notre travail⁵.

3.1 Attaque type avec requête GET

Etant donné que la mise en place d'attaques CSRF sur requêtes GET est facile et rapide, nous avons testé toutes les attaques de type Auth-CSRF décrites pour mieux les comprendre. Mais ici nous n'en décrirons qu'une seule : l'attaque type "Form-Based Registration".

Nous divisons cette attaque en deux parties : la préparation puis l'exécution.

3.1.1 Préparation

Dans cette section, nous représentons l'attaquant qui possède son site "méchant". Il veut y camoufler une requête forgée. Nous devons d'abord construire cette requête. Dans cette attaque, on utilise la faille (supposée découverte par l'attaquant) du formulaire d'inscription. Comme l'inscription se fait en utilisant une requête GET, il suffira d'utiliser un outil simple⁶ pour capturer la requête de création d'un compte. Cette requête peut être utilisée pour forger une requête CSRF.

Dans notre maquette, ce formulaire d'inscription possède les champs 'nom', 'prenom', 'pseudo', 'motdepasse', 'confirmation', 'email', 'tel' et 'inscription'. Ainsi, nous pouvons forger une requête avec une simple URL, où l'on place les variables correctement. Voici l'URL représentant notre attaque CSRF :

```
http://127.0.0.1/edsa-IR_projet/formulaire_entree_process.php?  
nom=Mechant&prenom=Xavier&pseudo=xavier&motdepasse=facile&conf  
irmation=facile&email=xavier@gmail.fr&tel=0658476952&inscripti  
on=Submit
```

Nous camouflons donc cette requête dans une image sur notre site malveillant. Et nous attendons qu'un utilisateur tombe dans le piège...

3.1.2 Exécution

Dans cette section, nous représentons l'utilisateur innocent qui va activer l'attaque avec son navigateur sans le vouloir. Tout d'abord, nous nous connectons sur le site "gentil". Pour cette attaque, le fait d'être connecté en tant qu'anonyme ou sur son compte ne fait pas de différence, car le site "gentil" nous connecte automatiquement sur un compte qui vient d'être créé. Nous pouvons voir sur la figure 1 que, pour cette démonstration, nous sommes initialement connecté en tant qu'anonyme.

5. Si vous souhaitez regarder le code source de notre maquette, voici un lien Gitlab : gitlab.com/RudigerMorinDocter/ir_site_test_csrf

6. L'attaquant peut faire usage d'un proxy ou d'un plugin chrome par exemple, ou bien simplement analyser le code source de la page d'inscription pour récupérer les éléments du formulaire d'inscription.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : Anonymous

[Login](#)
[Inscription](#)
[Login VIP](#)
[Inscription VIP](#)
[Login SSO](#)
[Login SSO avec POST](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)

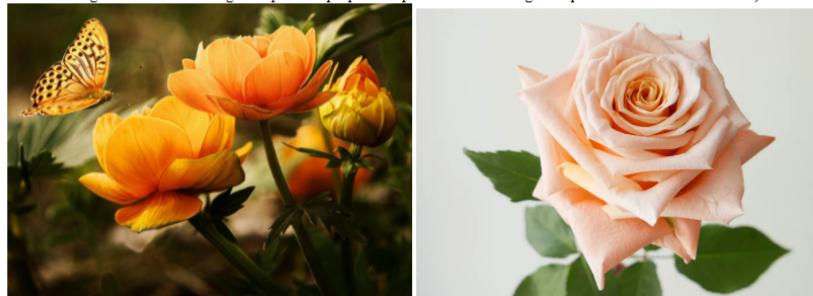


FIGURE 1 : Connecté en tant qu'anonyme avant l'attaque "Form-Based Login"

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : xavier

[Logout](#)
[Changement d'email](#)
[Lier son compte SSO](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



FIGURE 2 : Connecté en tant que 'xavier' après l'attaque "Form-Based Login"

```

<h2>Login SSO</h2>
<div style = "width:300px; border: solid 1px #333333; " align = "left">
    <div style = "background-color:#333333; color:#FFFFFF; padding:3px;"><b>Login</b></div>
    <div style = "margin:30px">
        <form action = "" method = "get">
            <label>Pseudo :</label><br /><input type = "text" name = "pseudo" class = "box"/><br /><br />
            <label>Mot de Passe :<br /></label><input type = "password" name = "mdp" class = "box" /><br /><br />
            <input type = "submit" name="log" value = "Submit"/><br />
        </form>
        <div style = "font-size:11px; color:#cc0000; margin-top:10px"></div>
    </div>
</div>

```

FIGURE 3 : Code source de la page de connexion SSO du site "gentil".

```

<h3>Attaque 4 bis : SSO Based Login mais POST</h3>
<form name="attaqueForm" method="post" action="http://127.0.0.1/edsa-IR_projet/SSO_login_Post.php" >
    <input type="hidden" name="pseudo" value="V1lScorpion">
    <input type="hidden" name="mdp" value="159">
    <button type="submit" name="log" value = "Submit">
        
    </button>
</form>

```

FIGURE 4 : Formulaire forgé après exploration du code source du site "gentil".

Nous visitons ensuite le site "méchant" et nous cliquons⁷ sur l'image de squelette de T-Rex, dans laquelle est cachée la requête forgée préparée par l'attaquant. La requête CSRF est chargée par le navigateur de l'utilisateur innocent, et sans s'en rendre compte, il est connecté sur le compte de l'attaquant, ici nommé 'xavier' (figure 2).

Dans notre maquette, ceci n'est pas très grave, mais en réalité, une attaque de ce type permet à l'attaquant d'avoir accès à l'historique de navigation ou d'achat de l'utilisateur si celui-ci ne fait pas attention au compte actuellement actif.

3.2 Attaque type avec requête POST

Certaines sources, notamment les forums de développement, stipulent que les attaques CSRF sont impossibles si le site web utilise uniquement des requêtes POST. Afin de contrer ces allégations, nous avons mis en place une attaque CSRF sur une requête POST. Contrairement aux requêtes GET, les attaques CSRF avec POST sont plus longues à mettre en place. C'est pourquoi nous n'en avons mis en place qu'une seule. Cependant, la logique de création reste très similaire si l'on souhaite l'adapter pour les autres Auth-CSRF.

L'attaque que nous allons ici décrire est le "SSO-Based Login". De la même manière qu'en 3.1, nous allons diviser l'attaque en deux parties : la préparation et l'exécution.

7. Comme indiqué auparavant, en réalité, l'url d'attaque est lancée lorsque l'image est chargée par le navigateur. Nous avons transformé les images en boutons afin d'éviter de charger toutes nos attaques à chaque visite du site "méchant".

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : PetitRhino
[Logout](#)



FIGURE 5 : Connecté en tant que 'PetitRhino' avant l'attaque "SSO-Based Login"

3.2.1 Préparation

Nous prenons la place de l'attaquant. Nous souhaitons utiliser une faille que nous avons découverte dans le formulaire de connexion SSO. Avec les requêtes POST, nous ne pouvons pas utiliser une simple URL. Il va falloir forger le formulaire en lui-même. Pour ce faire, nous pouvons utiliser l'outil fourni par les navigateurs⁸ pour visionner le code source de la page du formulaire (figure 3).

Nous voyons ici que le formulaire demande un 'pseudo', un 'password' et un 'log'. Nous créons donc un petit script HTML pour camoufler un formulaire forgé dans une image. Un formulaire forgé est un formulaire reprenant la nomenclature du formulaire ciblé dans le site "gentil", mais avec des valeurs pré-remplies par l'attaquant. Vous pouvez observer le formulaire forgé pour cette démonstration sur la figure 4.

Il faut noter qu'ici nous en faisons un bouton, pour que ce soit plus pratique, mais en réalité on chargerait directement le formulaire pré-rempli lors du chargement de l'image. Ceci est faisable très facilement en utilisant la commande Javascript :

```
document . GetElementById( ' yourFormId ' ) . submit ()
```

Maintenant, nous attendons qu'un utilisateur visite notre site...

3.2.2 Exécution

Revenons dans la peau de l'utilisateur. Pour que cette attaque ait un effet, il faut être connecté sur notre compte. Nous observons, sur la figure 5, que nous sommes initialement connecté sur le compte 'PetitRhino'.

8. L'outil d'exploration du code source de la page actuelle s'active, pour Chrome, avec la commande 'Ctrl+u'.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : VilScorpion
[Logout](#)



FIGURE 6 : Connecté en tant que 'VilScorpion' après l'attaque "SSO-Based Login"

Nous visitons ensuite le site "méchant", et nous cliquons sur l'image de squelette de Hypsilophodon. Le formulaire forgé, camouflé dans l'image, est chargé par le navigateur. Pour voir le résultat de l'attaque, nous retournons sur le site "gentil", où nous pouvons observer que le compte actif a changé. Comme vous pouvez voir sur la figure 6, nous sommes maintenant connecté en tant que "VilScorpion", le compte SSO appartenant à l'attaquant.

De même que pour l'attaque de la section 3.1, dans notre maquette, ce résultat n'est pas très grave. Mais, en réalité, une attaque de ce type permet à l'attaquant d'avoir accès à l'historique de navigation ou d'achat de l'utilisateur si celui-ci ne fait pas attention au compte actuellement actif.

Nous avons décrit la mise en oeuvre de deux attaques de type Auth-CSRF décrites dans l'article [Sudhodanan et al., 2017]. Nous souhaitons dorénavant mettre en place des moyens de détection et d'analyse de ces failles.

4 Méthodes de Détection

Dans [Sudhodanan et al., 2017], l'équipe de recherche propose des méthodes de test manuel pour chacune des attaques Auth-CSRF qu'ils ont décrites. Dans cette partie, nous allons décrire ces méthodes, puis nous proposerons une généralisation pour ensuite tenter de créer une nouvelle méthodologie pouvant s'appliquer plus généralement à toute faille CSRF.

4.1 Méthodologies de détection de l'article

L'équipe de chercheurs propose dans leur article six méthodes pour tester manuellement si un site en cours de construction présente une des failles Auth-CSRF. Chaque méthode correspondant à une seule attaque CSRF, sauf la sixième méthode qui englobe à la fois les attaques

Étude des vulnérabilités applicatives : failles XSS et CSRF

CSRF visant les systèmes de changement d'email et de mot de passe.

Méthodologie de test pour "Form-Based Registration" :

1. Visiter la page d'inscription du site à tester.
2. Se créer un compte en mettant les détails souhaités par l'attaquant.
3. Intercepter la requête HTTP correspondante (à l'aide d'un proxy).
4. Récupérer les informations de cette requête (Header, URL, Content-Type et Content-Length).
5. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
6. Visiter le site à tester.
7. Envoyer une nouvelle requête HTTP avec un Referer forgé mais en gardant les attributs de la requête interceptée.
8. Vérifier : Si vous êtes connecté en tant qu'attaquant, alors il existe une faille.

Méthodologie de test pour "URL-Based Account Activation" :

1. Incrire un compte attaquant sur le site à tester.
2. Recevoir le lien d'activation de ce compte.
3. Nettoyer les cookies du navigateur.
4. Visiter le site à tester.
5. Utiliser le lien URL d'activation.
6. Vérifier : Si vous êtes connecté en tant qu'attaquant, alors il existe une faille.

Méthodologie de test pour "Form-Based Login" :

1. Visiter la page de connexion du site à tester.
2. Se connecter en utilisant le compte attaquant.
3. Intercepter la requête HTTP correspondante (à l'aide d'un proxy).
4. Récupérer les informations de cette requête (Header, URL, Content-Type et Content-Length).
5. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
6. Visiter le site à tester.
7. Envoyer une nouvelle requête HTTP avec un Referer forgé mais en gardant les attributs de la requête interceptée.
8. Vérifier : Si vous êtes connecté en tant qu'attaquant, alors il existe une faille.

Méthodologie de test pour "SSO Login" :

1. Lier le compte SSO attaquant avec le compte standard de l'attaquant.
2. Intercepter la requête HTTP correspondante (à l'aide d'un proxy).

3. Récupérer les informations de cette requête (Header, URL, Content-Type et Content-Length).
4. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
5. Visiter le site à tester.
6. Envoyer une nouvelle requête HTTP avec un Referer forgé mais en gardant les attributs de la requête interceptée.
7. Vérifier : Si vous êtes connecté en tant qu'attaquant, alors il existe une faille.

Méthodologie de test pour "SSO-Based Account Association" :

1. Se connecter au compte attaquant sur le site à tester.
2. Visiter la page de lien avec un compte SSO.
3. Lier le compte SSO de l'attaquant avec son compte standard.
4. Intercepter la requête HTTP correspondante (à l'aide d'un proxy).
5. Récupérer les informations de cette requête (Header, URL, Content-Type et Content-Length).
6. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
7. Se connecter sur le compte innocent.
8. Envoyer une nouvelle requête HTTP avec un Referer forgé mais en gardant les attributs de la requête interceptée.
9. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
10. Vérifier : Si vous pouvez vous connecter au compte innocent en utilisant les informations SSO de l'attaquant, alors il existe une faille.

Méthodologie de test pour "Email and Password Change" :

1. Se connecter au compte attaquant sur le site à tester.
2. Visiter la page de changement d'email ou de mot de passe sur le site à tester.
3. Donner un nouveau email/mot de passe au compte attaquant.
4. Intercepter la requête HTTP correspondante (à l'aide d'un proxy).
5. Récupérer les informations de cette requête (Header, URL, Content-Type et Content-Length).
6. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
7. Se connecter sur le compte innocent.
8. Envoyer une nouvelle requête HTTP avec un Referer forgé mais en gardant les attributs de la requête interceptée.
9. Nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
10. Vérifier : Si vous pouvez vous connecter au compte innocent en utilisant le nouveau email/mot de passe, alors il existe une faille.

Dans toutes ces méthodologies, nous voyons apparaître le terme "Referer". d'après [MDN, 2021], un Referer permet au serveur du site web recevant la requête de vérifier sa provenance. L'équipe cherche ici simplement à forger une requête qui sera présente sur un site attaquant, mais avec un Referer modifié de sorte à faire croire que la source de la requête CSRF est sûre. La modification d'un Referer ne présente, en général, pas de difficultés particulières. De plus, il existe des sites qui ne prennent pas en compte ces Referer. Nous décrivons ici rapidement l'usage que l'équipe fait des referer afin de ne pas laisser place aux doutes du lecteur, mais pour la suite de notre travail, nous nous focalisons essentiellement sur les paramètres des requêtes GET et POST.

Nous observons que ces méthodes présentent de nombreuses répétitions. De plus, notre but est de proposer une amélioration aux travaux de cet article. C'est pourquoi nous tentons par la suite de généraliser ces différentes logiques.

4.2 Notre généralisation des méthodologies

Nous souhaitons définir une méthodologie plus générale, qui pourrait servir à détecter tout type de faille CSRF dans un site web en construction.

Voici la méthodologie de test que nous avons mis en place :

1. Se connecter sur le site à tester.
2. Visiter la page à vérifier.
3. Remplir le formulaire présent sur cette page.
4. Intercepter la requête correspondante (avec un proxy).
5. Récupérer le header de cette requête.
6. Forger une nouvelle requête en conservant un maximum d'informations du header intercepté.
7. Si votre site utilise un système de défense CSRF avec token généré pseudo-aléatoirement, alors nettoyer les cookies du navigateur et réinitialiser le proxy utilisé.
8. Envoyer la nouvelle requête forgée.
9. Vérifier : Si cette requête a eu un impact sur le site, alors il existe une faille.

Notre méthode vise à vérifier l'existence tout type de faille CSRF, et pas uniquement les Auth-CSRF. Etant donné que les attaques CSRF peuvent avoir des conséquences différentes selon les fonctionnalités proposées par le site web, cette méthode demande que le testeur connaisse bien le site à tester et sache quels endroits peuvent présenter des failles potentielles. Ceci peut donc poser plus ou moins de problèmes, car il n'est pas impossible que le testeur ne soit pas capable de détecter toutes les failles CSRF présentes sur le site web.

Nous allons maintenant tenter de mettre en place cette méthodologie sur notre maquette.

```

Request to http://127.0.0.1:80
[Forward] [Drop] [Intercept is ...] [Action] [Open Brow...]
[Pretty] [Raw] [\n] [Actions ▾]

1 GET /formulaire_login.php?pseudo=bolt&mdp=123456&log=Submit HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="90"
4 sec-ch-ua-mobile: ?0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Referer: http://127.0.0.1/edsa-IR_projet/formulaire_login.php
13 Accept-Encoding: gzip, deflate
14 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
15 Cookie: PHPSESSID=djoa0ouafrmr17bnchgrbtteu3
16 Connection: close
17

```

FIGURE 7 : Header obtenu pour une requête de connexion sur notre maquette.

4.3 Mise en place du processus avec Burp Suite

La méthodologie manuelle peut être effectuée sans aucun outil lorsque le testeur connaît bien le site. Cependant, l'utilisation d'un proxy permet à toute personne d'essayer cette méthodologie sur tout site web de son choix.

Il existe plusieurs manières d'intercepter des requêtes. Un premier exemple est d'utiliser les outils de développement intégré aux navigateurs tels que Firefox ou Chrome⁹. Une autre manière est d'installer un plugin servant de proxy, comme 'Tamper Dev'¹⁰.

Dans notre cas, après plusieurs recherches, nous avons découvert Burp Suite. En utilisant cette application, nous avons accès à de multiples outils, dont un proxy. En lançant le navigateur Chromium pré-configuré pour fonctionner avec la fonctionnalité Proxy, nous pouvons capturer facilement les headers des requêtes que nous envoyons.

Sur la figure 7, nous voyons le header pour une requête de connexion sur le compte de l'utilisateur ayant pour pseudo 'bolt'. À partir de toutes ces informations, on peut choisir de copier tout ce header et de forger une requête en la modifiant. Cependant, une autre solution, est de récupérer la première ligne pour la transformer en attaque CSRF.

Par exemple, nous récupérons ici cette ligne :

```
GET /formulaire_login.php?pseudo=bolt&mdp=123456&log=Submit HTTP/1.1
```

On peut alors modifier cette ligne et créer cette requête GET qui attaque notre maquette :

9. Sur Chrome, Les outils de développement peuvent être ouverts en utilisant la commande 'Ctrl+Shift+i'. Puis, en allant dans l'onglet 'Network', nous avons accès aux informations des différentes requêtes que nous lançons, comme les headers par exemple.

10. Plugin disponible pour Chrome. Lien : <https://tamper.dev/>

```
http://127.0.0.1/edsa-IR_projet/formulaire_login.php?pseudo=xavie&mdp=facile&log=Submit
```

Créer des requêtes CSRF de cette manière est plus simple que si nous devions les créer grâce à nos connaissances de la construction du site. Mais il reste énormément de travail manuel à effectuer. Il faut, entre autres :

- Gérer l'activation et la désactivation de l'interception (sachant que l'interception bloque le trafic).
- Analyser le header et définir ce dont nous avons besoin. Ceci peut varier selon les défenses que propose le site web à tester.
- Forger une requête CSRF potentielle manuellement. Il faut analyser les champs que l'on souhaite éditer pour simuler une attaque.

Nous souhaitons donc proposer une amélioration de cette méthodologie manuelle. La première idée est d'essayer d'automatiser le travail au maximum de nos capacités.

Avant de décrire l'amélioration que nous avons envisagée, nous allons présenter Burp Suite, que nous allons utiliser pour implémenter notre amélioration.

4.3.1 Présentation de Burp Suite

Burp Suite¹¹ est une application proposée par l'entreprise "PortSwigger". Elle a pour but d'accélérer les tests de sécurité sur applications, en proposant un large panel d'outils d'analyse, de détection et de simulation.

Il existe deux versions de Burp Suite : Enterprise et Community. La version Enterprise, payante, possède des fonctionnalités supplémentaires, mais qui nous sont de peu d'utilité pour ce projet. C'est pourquoi nous utilisons la version Community, gratuite.

Burp Suite possède également une grande communauté, permettant de trouver un grand nombre de tutoriels sur son usage en ligne, afin de maîtriser l'application rapidement.

4.3.2 Principe des extensions Burp Suite

Burp Suite propose à ses utilisateurs d'ajouter ou de modifier des fonctionnalités de son application à l'aide d'extensions Burp écrites en Java, Python ou Ruby. Le système gérant ces extensions s'appelle "Burp Extender". De plus, Burp Suite Community autorise l'usage et la création d'extension, ce qui nous arrange fortement.

Il existe un catalogue d'extensions proposées par des utilisateurs en ligne. Cependant, nous n'avons pas trouvé d'extension qui gère le travail que nous souhaitons effectuer. Nous décidons alors d'essayer d'écrire une extension Burp Suite qui automatise la méthodologie que nous avons décrite dans la partie 4.2.

Cependant, avant de nous lancer dans l'écriture du code, nous allons définir un algorithme reprenant la logique de notre méthode.

4.3.3 Algorithme général de notre extension

Comme il s'agit de notre première extension Burp Suite, nous avons choisi de rester simples : nous voulons créer un algorithme d'extension qui permet de transformer une requête GET interceptée lors de la navigation sur notre maquette en attaque CSRF potentielle.

11. Site web officiel de Burp Suite : portswigger.net/burp.

Algorithme 1 : Création automatisée de requêtes CSRF pour notre maquette

Entrées : Requête interceptée par le proxy de Burp Suite R
Sorties : LogCSRF : Un log des requêtes CSRF générées à partir des headers des requêtes interceptées R

```
1 Créer un Listener Listen de Proxy Burp Suite;
2 tant que Listen est actif faire
3   si le message intercepté est une requête alors
4     Récupérer le header;
5     si la requête est de type GET alors
6       String capture = Récupérer la première ligne du header capturé;
7       Parser et modifier capture pour produire une requête CSRF;
8       Afficher le résultat CSRF;
9       Ajouter CSRF à LogCSRF;
10    fin
11  fin
12 fin
13 Retourner LogCSRF
```

Maintenant que nous avons posé la base de notre algorithme, nous passons au développement.

5 Développement et Résultats

Dans cette partie, nous décrivons le déroulement du développement de l'algorithme [Morin-Docter and Pedrotti, 2021], comment utiliser notre extension, ainsi que les résultats produits.

5.1 Détails techniques

Le langage utilisé pour coder l'extension est Java. Nous avons utilisé l'environnement de développement Eclipse. L'extension fonctionne sur nos deux machines respectives.

Détails de la machine de Rüdiger Morin-Docter :
Processeur : AMD Ryzen 5 2600X - Six Coeurs - 3.60GHz
RAM : 16,0Go DDR4
Système d'exploitation : Windows 10 Famille

Détails de la machine de Thomas Pedrotti :
Processeur : Intel Core i7-8700K - Six Coeurs - 3.70GHz
RAM : 16,0Go DDR4
Système d'exploitation : Windows 10 Famille

Le code source de notre extension nommée "CSRF Attender" est disponible sur GitHub¹².

12. Extension CSRF Attender pour Burp Suite, lien Github : github.com/RudigerMorinDocter/CSRF-Attender.

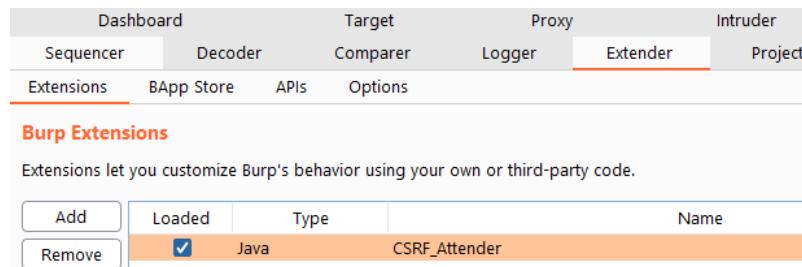


FIGURE 8 : Activation de 'CSRF Attender' dans Burp Suite.

5.2 Développement de l'extension pour Burp Extender

Afin de créer une extension pour Burp Extender, il faut récupérer le package qui est mis à disposition en ligne. Ce package 'burp'¹³ contient un grand nombre d'interfaces reprenant diverses fonctionnalités de Burp Suite. Le développeur doit utiliser ces interfaces dans son code, afin de créer ou modifier des fonctionnalités de Burp.

Nous avons importé le package 'burp' dans notre projet Eclipse. Puis nous créons une nouvelle classe nommée BurpExtender.java. Il ne nous reste plus qu'à explorer les interfaces dont nous disposons, de définir celles qui peuvent nous être utiles, et de coder les fonctionnalités que l'on souhaite ajouter.

Quand nous avons fini d'écrire nos nouvelles fonctionnalités, nous exportons notre projet au format JAR, puis nous démarrons Burp Suite. Si vous souhaitez savoir comment installer une extension, nous proposons un guide en Annexe D.

5.3 Exécution et Résultats

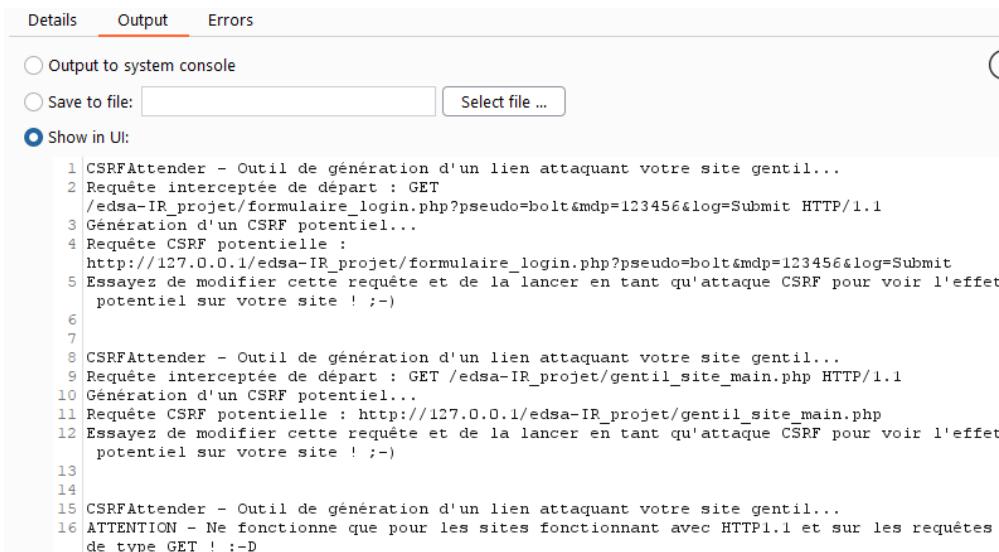
Pour faire fonctionner l'extension, il faut démarrer le navigateur chromium proposé par Burp Suite en allant dans 'Proxy' → 'Open Browser'. Puis, retourner dans l'onglet 'Extender', et cliquer sur 'Output' dans la partie secondaire de l'onglet. Il ne reste plus qu'à naviguer sur des sites web avec le navigateur chromium, et vous verrez s'afficher le log des requêtes CSRF générées par l'extension, à condition que la requête soit de type GET et que le site utilise HTTP1.1.

Les requêtes CSRF générées prennent la forme de chaînes de caractères dans la console. Pour les utiliser, il suffit de les copier/coller dans le navigateur. Vous pouvez manuellement modifier les paramètres des requêtes CSRF si vous le souhaitez.

Par exemple, sur la figure 9, nous avons une attaque CSRF générée (ligne 4) à partir de la requête interceptée (ligne 2). Cette requête CSRF est de la forme :

```
http://127.0.0.1/edsa-IR\_projet/formulaire\login.php?pseudo=bo1t\&mdp=123456\&log=Submit
```

13. Les explications de ces différentes interfaces sont disponibles en ligne, grâce à ce lien : portswigger.net/burp/extender/api/burp/package-summary.html.



The screenshot shows the 'Output' tab of the 'CSRF Attender' tool. It has three options: 'Output to system console' (radio button), 'Save to file:' (checkbox with a browse button 'Select file ...'), and 'Show in UI:' (radio button, which is selected). Below these options is a large text area containing 16 numbered lines of generated CSRF requests. Lines 1 through 15 show various GET requests to 'edsa-IR_projet/formulaire_login.php' with parameters like 'pseudo=bolt&mdp=123456&logSubmit'. Line 16 is a warning: 'ATTENTION - Ne fonctionne que pour les sites fonctionnant avec HTTP1.1 et sur les requêtes de type GET ! :-D'.

```

1 CSRFAttender - Outil de génération d'un lien attaquant votre site gentil...
2 Requête interceptée de départ : GET
3 /edsa-IR_projet/formulaire_login.php?pseudo=bolt&mdp=123456&logSubmit HTTP/1.1
4 Génération d'un CSRF potentiel...
5 Requête CSRF potentielle :
6 http://127.0.0.1/edsa-IR_projet/formulaire_login.php?pseudo=bolt&mdp=123456&log=Submit
7 Essayez de modifier cette requête et de la lancer en tant qu'attaque CSRF pour voir l'effet
8 potentiel sur votre site ! ;)
9
10 CSRFAttender - Outil de génération d'un lien attaquant votre site gentil...
11 Requête interceptée de départ : GET /edsa-IR_projet/gentil_site_main.php HTTP/1.1
12 Génération d'un CSRF potentiel...
13 Requête CSRF potentielle : http://127.0.0.1/edsa-IR_projet/gentil_site_main.php
14 Essayez de modifier cette requête et de la lancer en tant qu'attaque CSRF pour voir l'effet
15 potentiel sur votre site ! ;)
16 CSRFAttender - Outil de génération d'un lien attaquant votre site gentil...
ATTENTION - Ne fonctionne que pour les sites fonctionnant avec HTTP1.1 et sur les requêtes
de type GET ! :-D

```

FIGURE 9 : Affichage du journal de requêtes CSRF générées par 'CSRF Attender'.

Elle correspond à l'utilisateur 'bolt' qui se connecte au site "gentil". Si nous le souhaitons, nous pouvons modifier le pseudo et le mot de passe en paramètres de la requête GET pour définir une nouvelle attaque CSRF :

```
http://127.0.0.1/edsa-IR\projet\formulaire\_login.php?pseudo=xavier\&mdp=facile\&log=Submit
```

Cependant, si l'on souhaite générer directement une attaque CSRF, il suffit d'intercepter une requête où nous nous connectons au compte attaquant, et la requête CSRF générée représentera alors l'attaquant sans que l'on ait de besoin de modifier manuellement celle-ci.

5.4 Améliorations et perspectives futures

Notre extension est encore améliorable. Une première amélioration serait de reconnaître les paramètres des requêtes, afin de proposer à l'utilisateur d'altérer leurs valeurs. Pour intégrer cette amélioration, on pourrait imaginer utiliser Hawk et les expressions régulières, mettre en place une nomenclature pour nommer les variables et permettre à l'algorithme de détecter les champs, ou bien créer une intelligence artificielle capable de différencier les champs d'email et autres.

Une deuxième amélioration serait de gérer les requêtes POST. Nous n'avons eu le temps que de couvrir les requêtes GET, mais il est possible de faire de même avec les requêtes POST. Burp Suite Pro propose un outil qui permet de créer des 'Proof-of-Concept' CSRF à partir de requêtes POST interceptées. La logique que cet outil suit est d'intercepter la requête, de retrouver le formulaire correspondant à cette requête, de créer un fichier HTML avec un formulaire reprenant les variables du véritable formulaire, et de proposer à l'utilisateur de télécharger ce

fichier. L'utilisateur peut ensuite altérer les valeurs du formulaire et lancer la requête forgée grâce à un bouton.

Une autre perspective pour faire évoluer ce projet serait de se tourner vers XSS. Les attaques CSRF et XSS représentent un vaste domaine, et posent encore aujourd'hui de nombreux problèmes de sécurité. On pourrait imaginer une extension Burp Suite capable de détecter des failles XSS dans un site web.

6 Conclusion

Après plusieurs recherches bibliographiques, nous avons créé une maquette afin de tester des attaques CSRF visant les mécanismes d'authentification. Puis, nous avons mis en place des méthodes de détection de failles CSRF existantes avant de proposer un algorithme permettant d'automatiser la détection. Enfin, nous avons développé une extension Burp Suite que nous avons nommée 'CSRF Attender' capable de générer automatiquement des attaques CSRF lors de la navigation sur une application web à tester. Nous avons également proposé plusieurs perspectives d'avenir à ce travail, que pourront suivre nos potentiels successeurs.

Ce travail nous aura permis de découvrir le monde de la sécurité des applications web, et d'utiliser divers outils permettant d'analyser et de découvrir des failles tels que Tamper Dev ou Burp Suite. Nous sommes satisfaits du travail que nous vous présentons, et nous pensons qu'il est fort possible que de futurs étudiants reprennent notre flambeau afin de proposer une suite à ce travail. Ce que nous retiendrons de ce projet, est que les problèmes de sécurité de petite envergure peuvent grandir au cours des années, devenant des obstacles à la sécurité globale. Les failles XSS et CSRF en sont un bon exemple, car en 2008 ils n'étaient pas très représentés, alors qu'en 2021 ils représentent une grande part des attaques enregistrées par la CWE. La sécurité informatique représente donc un domaine où les développeurs et les chercheurs doivent toujours s'adapter, se former et évoluer.

Remerciements : Remerciements à Mme Herrmann qui nous a guidés pour ce projet, Mr Michel qui gère la matière d'initiation à la recherche et Mr Blansche qui nous a fourni un template pour ce rapport.

Références

- [CWE, 2020] (2020). 2020 cwe top 25 most dangerous software weaknesses.
- [MDN, 2021] (2021). Referer <http://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Referer>.
- [Eric Dalci, 2021a] Eric Dalci, M. (2021a). Cwe-352 : Cross-site request forgery (csrf).
- [Eric Dalci, 2021b] Eric Dalci, M. (2021b). Cwe-79 : Improper neutralization of input during web page generation ('cross-site scripting').
- [Morin-Docter and Pedrotti, 2021] Morin-Docter, R. and Pedrotti, T. (2021). Csrf attender. github.com/RudigerMorinDocter/CSRF-Attender.
- [Sudhodanan et al., 2017] Sudhodanan, A., Carbone, R., Compagna, L., Dolgin, N., Armando, A., and Morelli, U. (2017). Large-scale analysis & detection of authentication cross-site request forgeries. In *2017 IEEE European symposium on security and privacy (EuroS&P)*, pages 350–365. IEEE.
- [Zeller and Felten, 2008] Zeller, W. and Felten, E. W. (2008). Cross-site request forgeries : Exploitation and prevention. *Bericht, Princeton University*.

Annexes

A Les différentes réunions du semestre

Date	Sujet(s) évoqué(s)
08/02/2021	Réunion générale avec tous les groupes de Mme Herrmann. - Explications générales sur la matière d'initiation à la recherche. - Comment rechercher des articles en ligne (Google Scholar + Bibliothèques universitaires). - Comment organiser notre travail. Pour la prochaine fois, faire un résumé des articles proposés dans les sujets d'IR fournis.
10/02/2021	Présentation et explication de l'article de Zeller ainsi que des articles de la CWE. Travail de la semaine suivante : trouver un article qui nous plaît, et en faire un résumé pour le présenter.
25/02/2021	Présentation de l'article de Sudhodanan et al et des différentes attaques CSRF décrites. Pour la semaine suivante : essayer de tester différentes méthodes d'attaques et voir si elles sont « scriptables ».
05/03/2021	Explications des différentes attaques XSS et CSRF mises en pratique à l'aide de Gruyère. Pour la suite, déployer une maquette sur laquelle nous ferons plus de tests.
30/03/2021	Présentation de la maquette encore incomplète. Quatre des sept attaques type Auth-CSRF sont fonctionnelles. Pour la prochaine fois, mettre en place les trois dernières attaques, mettre en place une attaque avec POST et réfléchir à des contre-mesures.
09/04/2021	La maquette est complète et fonctionnelle, avec les sept attaques GET type Auth-CSRF et une attaque POST. Pour la suite, étudier des outils de détection existants pour tenter de proposer des améliorations.
21/04/2021	Mise en place d'une méthodologie générale basée sur l'usage de l'application Burp Suite. Pour la semaine suivante : transformer cette méthodologie en script d'extension Burp Extender.
29/04/2021	Présentation de notre proposition d'amélioration finale. Mme Herrmann nous propose de nous aider à mettre en place un plan pour notre rapport.
06/05/2021	Réunion rapide pour relire notre rapport et s'assurer que nous utilisons correctement Latex.

B Sujet initial

Sujet 4: Etude des vulnérabilités applicatives: failles XSS et CSRF

Proposé par Francine Herrmann

Ce sujet peut être réalisé par 2 étudiants

L'énumération des vulnérabilités les plus fréquentes en 2020 (CWE Commun Weakness Enumeration) a pour objectif d'élaborer chaque année le Top 25 des failles logicielles les plus dangereuses (CWE Top 25). C'est une liste des vulnérabilités applicatives les plus courantes et les plus marquantes rencontrées au cours des deux années précédentes. Ces faiblesses sont dangereuses car elles sont souvent faciles à trouver, à exploiter et peuvent permettre à des adversaires de s'emparer complètement d'un système, de voler des données ou d'empêcher une application de fonctionner. Le CWE Top 25 est une ressource communautaire précieuse qui peut aider les développeurs, les testeurs et les utilisateurs - ainsi que les chefs de projet, les chercheurs en sécurité et les enseignants - à fournir un aperçu des faiblesses de sécurité les plus graves et les plus actuelles.

En étudiant les sources [1, 2, 3, 4] et en particulier les CWE 79 et CWE 352, étudier et comparer les failles XSS et CSRF.

Le plan du travail à réaliser sera le suivant :

- a. Etude Bibliographique du système de classement CWE
- b. Etude bibliographique des failles XSS et CSRF
- c. Présenter les définitions et les conséquences de ces failles
- d. Présenter des exemples conceptuels
- e. Présenter des exemples pratiques dans de réelles CVE (concept et mise en œuvre),
- f. Analyse des contremesures (concept et mise en œuvre).
- g. Méthodes de détection (concept et mise en œuvre),
- h. Imaginez de nouvelles contre-mesures et de nouvelles méthodes de détection

Point de départ de l'étude de bibliographique :

[1] William Zeller and Edward W. Felten, Cross-Site Request Forgery: Exploitation and Prevention, 2008, <http://citp.princeton.edu/csrf/>

[2] Publication du MITRE, 2020 CWE Top 25 Most Dangerous Software Weaknesses, https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html#limitations_of_the_methodology

[3] CWE-79: Improper Neutralization of Input During Web Page Generation, <https://cwe.mitre.org/data/definitions/79.html>

[4] CWE-352: Cross-Site Request Forgery , <https://cwe.mitre.org/data/definitions/352.html>

C Apparence de notre maquette

Dans cette section, nous allons simplement montrer l'apparence de notre maquette. La maquette est composée de deux sites web : un site "gentil" et un site "méchant".

Le site "gentil" est un site de fleurs. Lorsque l'on est connecté en tant qu'anonyme, quatre images de fleurs sont affichées. Les utilisateurs connectés à leur compte peuvent voir huit images de fleurs. Nous avons également intégré un affichage du pseudonyme de l'utilisateur actuellement connecté afin de simplifier la visibilité des résultats des attaques CSRF.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : Anonymous

[Login](#)

[Inscription](#)

[Login VIP](#)

[Inscription VIP](#)

[Login SSO](#)

[Login SSO avec POST](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



Le site "méchant" est un site contenant des images de squelettes de dinosaures. Chaque image correspond à un type d'attaque CSRF comme indiqué dans la partie 3 de ce document.

Les dinosaures étaient les prédateurs d'un autre temps...

Attaque 1 : Form-based Registration



D Installation d'une extension Burp Extender

Naviguer dans l'onglet "Extender". Puis cliquer sur le bouton "Add".

The screenshot shows the Burp Suite interface with the 'Extender' tab selected. In the 'Extensions' section, there is a table with three columns: 'Loaded', 'Type', and 'Name'. To the left of the table is a vertical sidebar containing four buttons: 'Add', 'Remove', 'Up', and 'Down'. The 'Add' button is highlighted.

Un nouvel onglet est affiché, dans la section 'Extension Details', mettre 'Extension Type' comme étant 'Java', et indiquer le chemin vers 'CSRFAttender.jar'.

The screenshot shows the 'Load Burp Extension' dialog box. The 'Extension Details' section has 'Extension type:' set to 'Java' and 'Extension file (.jar):' set to 'rs\rudig\Downloads\CSRFAttender.jar'. The 'Standard Output' section has 'Show in UI' selected. The 'Standard Error' section also has 'Show in UI' selected. At the bottom right are 'Cancel' and 'Next' buttons.

Cliquer sur 'Next', puis un autre onglet s'affiche. Si tout est bon, cet onglet n'affichera aucune erreur.
Cliquer sur 'Close'.

The screenshot shows the Burp Suite Extender interface. At the top, a green header bar displays the text "Load Burp Extension" with a left arrow icon. To the right are standard window control buttons for minimize, maximize, and close. Below the header, a message states: "The extension loaded successfully. Any output or error messages generated are shown below." A tab bar at the top of the main area has two tabs: "Output" and "Errors", with "Errors" being the active tab. The main pane is currently empty, showing only the number "1" in the top-left corner. At the bottom of the main pane, there is a toolbar with icons for help, settings, clear, search, and a search input field containing "Search...". To the right of the search field, it says "0 matches". A large orange "Close" button is located at the bottom right of the main pane. Below the main pane, a status message reads: "L'extension est dorénavant intégrée à Burp Suite et est activée par défaut." A navigation bar below the status message includes tabs for Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Logger, and Extender, with "Extender" being the active tab. Under the "Extender" tab, there is a section titled "Burp Extensions" with the sub-instruction: "Extensions let you customize Burp's behavior using your own or third-party code." On the left, there is a vertical toolbar with buttons for Add, Remove, Up, and Down. On the right, a table lists extensions. The table has columns for "Loaded", "Type", and "Name". One row is highlighted in orange, showing "Java" under Type and "CSRF_Attender" under Name. There is also a checked checkbox next to the "Java" entry.

Étude des vulnérabilités applicatives : failles XSS et CSRF

Si ce n'est pas le cas, sélectionner 'Output' au centre de la page. S'assurer que 'Show in UI' est sélectionné.

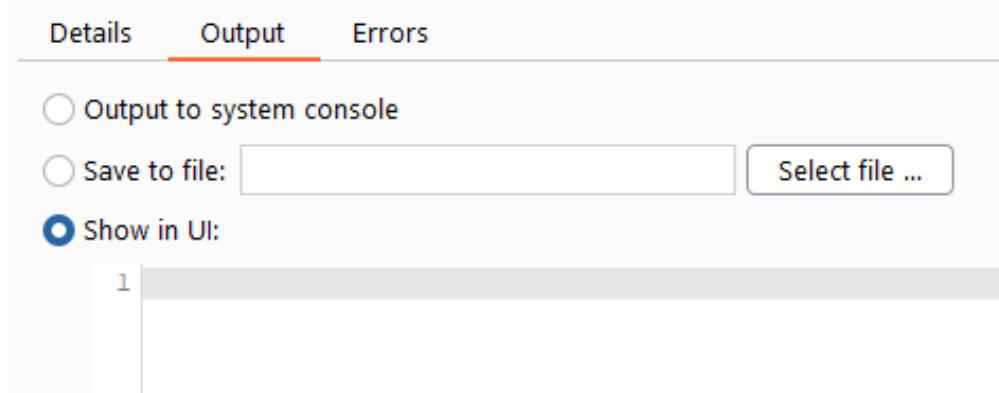
Details **Output** Errors

Output to system console

Save to file:

Show in UI:

1



E Démonstration des attaques Auth-CSRF

Ci-dessous, nous faisons une description des six autres attaques que nous avons testées avec des requêtes GET et que nous n'avons pas décrites dans la partie principale. Si vous souhaitez voir le code source de notre maquette, voici un lien Gitlab : gitlab.com/RudigerMorinDocter/ir_site_test_csrf.

La maquette a été mise en place à l'aide de EasyPhp, afin de mettre en place une base de données. La base de données que nous utilisons n'est pas disponible sur le lien Gitlab ci-dessus, mais si vous souhaitez la recréer, voici à quoi ressemblent les tables que nous avons définies.

Hiérarchie de la base de données :

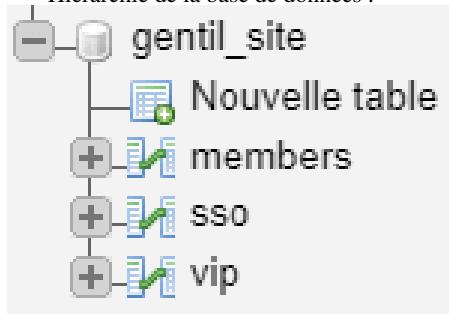


Table "members" représentant les comptes normaux :

nom	prenom	pseudo	mdp	email	telephone	pseudoSSO
Test	Bob	bob	getHacked	bob@gmail.com	645982641	VilScorpion
Abeille	Maya	abeille	456	email_mechant@gmail.fr	748956212	
Veilleur	Jacques	pingpong	789	jacques@gmail.com	345986251	
Usain	Bolt	bolt	123456	usain@gmail.com	645456532	
Adams	Bryan	BryanAdams	loveyou	bryan@gmail.com	648953265	
Mechant	Xavier	xavier	facile	xavier@gmail.fr	658476952	
Harley	David	davidson	753	davidson@hotmail.fr	648953212	
Hacker	Bertrand	plngpong	hacked	hacked@gmail.com	678542356	

Table "vip" représentant les comptes créés avec vérification par email :

nom	prenom	pseudo	mdp	email	telephone	cle	actif
Kenobi	ObiWan	obi	hellothere	obiwani@coruscant.com	845653285	a65ef6c4ce785360dbfc2157d448639b	1
Skywalker	Luke	Jedi	456	luke@tatooine.com	785652312	e6f6d19869fa60bf05a961a673754d0b	1
Clone	Rex	Captain	789	rex@republique.cor	866774523	1e2d89f5495229b8fd480f9d80b915c2	1
Tano	Ashoka	Padawan	753	tano@temple.com	945789835	afb897dba3aefa771b170b5235eb98ba	1
Dark	Vador	darkvador	666	Vador@Sith.com	845759535	fc61917f4c80af3e397d21abd487a1ac	1

Table "sso" représentant les comptes SSO pré-existants :

nom	prenom	pseudo	mdp	email	telephone
Rhino	Petit	PetitRhino	123	rhino.savane@gmail.com	785698574
Girafe	Courte	girafe	456	girafe.afrique@hotmail.fr	759481526
Gazelle	Rapide	FastGazelle	789	gazelle.turbo@hotmail.fr	915264859
Lion	Grand	lion57	741	lion.afrique@gmail.com	648512695
Scorpion	Vil	VilScorpion	159	scorpion.poison@gmail.com	948958426

E.1 Attaque type "Form-Based Login"

Pour que cette attaque ait lieu , il faut que le site "gentil" fournis un système de connexion avec formulaire.

Un utilisateur innocent se crée un compte. Dans cet exemple, l'utilisateur a pour pseudo "pingpong". L'attaquant se crée un compte ressemblant à celui d'un utilisateur. Dans cet exemple, il a pour pseudo "pIngpong".

Etape 1 : l'utilisateur innocent se connecte sur son compte sur le site "gentil".

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : pingpong
[Logout](#)
[Changement d'email](#)
[Lier son compte SSO](#)



Etape 2 : l'utilisateur innocent visite le site "méchant" et charge l'image correspondante à l'attaque CSRF.

Attaque 2 : Form-based Login



Etape 3 : l'utilisateur innocent est dorénavant connecté sur le compte de l'attaquant, avec un pseudo très ressemblant.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : p1ngpong

[Logout](#)

[Changement d'email](#)

[Lier son compte SSO](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



Résultat : Si l'utilisateur ne se rend pas compte de la différence, l'attaquant aura accès à son historique de navigation ou d'achats.

E.2 Attaque type "URL-Based Confirmation"

Pour que cette attaque ait lieu, il faut que le site "gentil" propose un système d'inscription avec vérification de l'email avec une URL d'activation. De plus, cette URL d'activation nous connecte directement une fois utilisée.

Un utilisateur navigue sur le site "gentil" en tant qu'anonyme (ou bien en étant connecté). L'attaquant se crée un compte, ici nommé "darkvador", et reçoit un mail de confirmation.

Sur cette image, nous voyons le lien d'activation de son compte :

Nouveau membre créé avec succès !

Inscrivez-vous sur notre Gentil Site ! :D

Inscription VIP

Login

Nom:	<input type="text"/>
Prenom:	<input type="text"/>
Pseudo:	<input type="text"/>
Mot de Passe:	<input type="password"/>
Confirmation Mot de Passe:	<input type="password"/>
Email:	<input type="text"/>
Telephone:	<input type="text"/>
<input type="button" value="Submit"/>	

Un mail de confirmation vous a été envoyé ! :D

Sujet : Activer votre compte

From : inscription@gentiliste.com

Bienvenue sur Gentil Site,

Pour activer votre compte, veuillez cliquer sur le lien ci-dessous

ou copier/coller dans votre navigateur Internet :

http://127.0.0.1/edsa-IR_projet/activation_mail.php?log=darkvador&cle=fc61917f4c80af3e397d21abd487a1ac

Ceci est un mail automatique. Merci de ne pas y répondre.

Etape 1 : L'attaquant cache cette URL d'activation dans une image de son site "méchant".

```
<h3>Attaque 3 : URL-based Confirmation</h3>
<a href="http://127.0.0.1/edsa-IR_projet/activation_mail.php?log=darkvador&cle=fc61917f4c80af3e397d21abd487a1ac">
    
</a>
```

Etape 2 : L'utilisateur innocent visite le site "méchant" et charge l'URL d'activation avec son navigateur sans s'en rendre compte.

Attaque 3 : URL-based Confirmation



Etape 3 : L'utilisateur active donc le compte de darkvador et est directement connecté sur ce compte.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : darkvador
[Logout](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



Si l'utilisateur ne se rend pas compte de la différence, l'attaquant aura accès à son historique de navigation ou d'achats.

E.3 Attaque type "SSO-Based Login" avec GET

Dans la partie 3.2 de ce document, nous avons vu cette attaque avec POST. Nous l'avons également faites avec GET.

Pour que cette attaque ait lieu, il faut que le site "gentil" propose un système de connexion SSO. Pour rappel, un compte SSO est un compte d'une plateforme autre, comme par exemple Google ou Facebook.

Etape 1 : Un utilisateur navigue sur le site "gentil" en tant qu'anonyme (ou connecté avec son compte SSO). Dans notre exemple, l'utilisateur a pour pseudo "PetitRhino".

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : PetitRhino
[Logout](#)



L'attaquant possède un compte SSO et veut s'en servir pour effectuer cette attaque. Dans notre exemple, l'attaquant a pour pseudo "VilScorpion".

Avec un outil simple¹⁴, l'attaquant explore et découvre la requête permettant de se connecter avec un compte SSO. Il la récupère et la modifie avec les paramètres correspondants à son propre compte. Il cache ensuite cette requête dans une image de son site "méchant".

Etape 2 : L'utilisateur visite le site "méchant" et charge l'image contenant la requête forgée par l'attaquant.

¹⁴. Il existe plusieurs outils pour récupérer les requêtes, comme par exemple un proxy, ou bien l'attaquant peut explorer le code source de la page web.

Attaque 4 : SSO Based Login

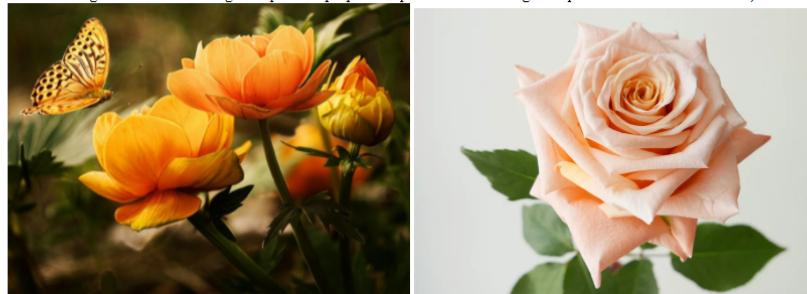


Etape 3 : L'utilisateur est maintenant connecté sur le compte SSO de l'attaquant.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : VilScorpion
[Logout](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



Résultat : Si l'utilisateur ne se rend pas compte de la différence, l'attaquant aura accès à son historique de navigation ou d'achats.

E.4 Attaque type "SSO Account Link"

Pour que cette attaque ait lieu, il faut que le site propose de lier un compte standard avec un compte SSO existant. Pour rappel, un compte SSO est un compte d'une plateforme autre, comme par exemple Google ou Facebook.

Etape 1 : Un utilisateur innocent navigue sur le site "gentil" en étant connecté à son compte. Dans notre exemple, l'utilisateur a pour pseudo "bob".

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : bob
[Logout](#)
[Changement d'email](#)
[Lier son compte SSO](#)



L'attaquant possède un compte SSO qu'il souhaite utiliser pour effectuer cette attaque. Dans notre exemple, l'attaquant a pour pseudo "VilScorpion".

Avec un outil simple, l'attaquant explore et découvre la requête permettant de son compte à un compte SSO. Il la récupère et la modifie avec les paramètres correspondants à son propre compte. Il cache ensuite cette requête dans une image de son site "méchant".

Etape 2 : L'utilisateur visite le site "méchant" et charge l'image contenant la requête forgée par l'attaquant.

Attaque 5 : SSO Account Link



Etape 3 : Le compte de l'utilisateur est maintenant lié au compte SSO de l'attaquant.

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : bob - VilScorpion
[Logout](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



Résultat : Selon la puissance qu'octroie le site "gentil" au compte SSO lié, les actions que peut prendre l'attaquant sur le compte de l'utilisateur peut aller du simple espionnage jusqu'à la prise de contrôle totale.

E.5 Attaque type "Password Reset"

Pour que cette attaque ait lieu, il faut que le site "gentil" propose de un système du type "Mot de passe oublié ?".

Etape 1 : Un utilisateur possède un compte sur le site "gentil". Dans cet exemple, nous regardons le mot de passe de Bob, qui est "123".

L'attaquant connaît, dans notre exemple, l'email de cet utilisateur ciblé. Il découvre la requête permettant de changer de mot de passe et s'en sert pour forger une requête CSRF. Il la cache dans une image de son site "méchant".

Etape 2 : L'utilisateur visite le site "méchant" et charge l'image correspondante à cette attaque.
Attaque 6 : Password reset



Etape 3 : Sans s'en rendre compte, la requête de modification de mot d epasse forgée par l'attaquant est chargée par le navigateur de l'utilisateur et son mot de passe est modifié par le mot de passe choisi par l'attaquant.

nom prenom pseudo mdp email telephone pseudoSSO
Test Bob bob getHacked bob@gmail.com 645982641 VilScorpion

Résultat : Cette attaque, lorsqu'elle réussit, est une des plus graves, car dorénavant l'attaquant connaît l'email et le mot de passe du compte de l'utilisateur, ce qui signifie qu'il en a le contrôle total.

E.6 Attaque type "Mail change"

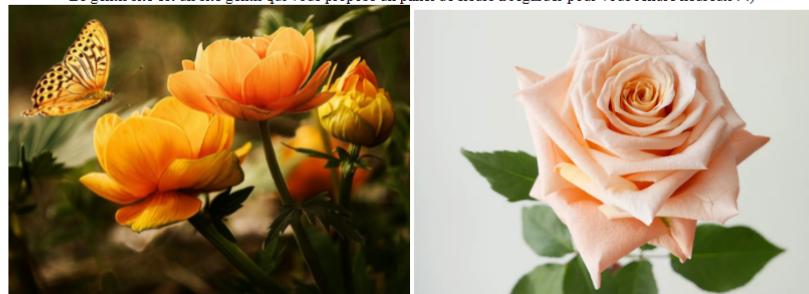
Pour que cette attaque ait lieu, il faut que le site "gentil" propose un système pour mettre à jour notre email principal.

Etape 1 : Un utilisateur innocent navigue sur le site "gentil" en étant connecté à son compte. Dans notre exemple, l'utilisateur a pour pseudo "abeille".

Bienvenue sur notre Gentil Site plein de fleurs !

Connecté en tant que : abeille
[Logout](#)
[Changement d'email](#)
[Lier son compte SSO](#)

Le gentil site est un site gentil qui vous propose un panel de fleurs à regarder pour vous rendre heureux ! :)



Etape 2 : Nous observons que son mail actuel est "maya@hotmail.fr".

nom	prenom	pseudo	mdp	email	telephone	pseudoSSO
Test	Bob	bob	getHacked	bob@gmail.com	645982641	VilScorpion
Abeille	Maya	abeille	456	maya@hotmail.fr	748956212	

L'attaquant explore et découvre la requête permettant de changer d'email. Il la modifie et la camoufle dans une image de son site "méchant".

Étude des vulnérabilités applicatives : failles XSS et CSRF

Etape 3 : L'utilisateur visite le site "méchant" et charge la requête forgée par l'attaquant.

Attaque 7 : Mail change



Etape 4 : Le compte de l'utilisateur est mis à jour sans que celui-ci ne soit au courant. Nous pouvons voir que la base de données a bien été mise à jour.

nom	prenom	pseudo	mdp	email	telephone	pseudoSSO
Test	Bob	bob	getHacked	bob@gmail.com	645982641	VilScorpion
Abeille	Maya	abeille	456	email_mechant@gmail.fr	748956212	

Résultat : Cette attaque est également très dangereuse, car la plupart des sites web proposent un système de "mot de passe oublié". Comme le compte de l'attaquant a pour email principal celui choisi par l'attaquant, ce dernier peut réinitialiser son mot de passe et en choisir un nouveau. Il a ainsi accès à l'historique de navigation et d'achats de l'utilisateur innocent, avec potentiellement les informations bancaires ou autres.