



UE L318 - Sécurité des applications Web

CSRF & XSS

Semaine 2

Ilaria Zappatore

CSRF

Cross-Site Request Forgery

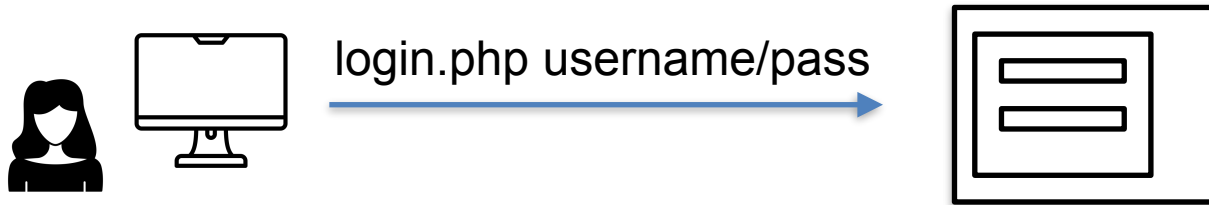
Une attaque qui permet à un attaquant de forcer un utilisateur **authentifié** sur un site ou une application web à **effectuer des actions non intentionnelles**.



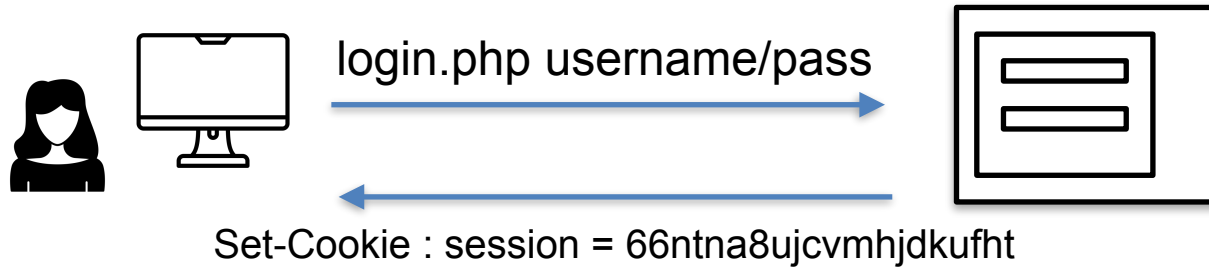
Violer l'intégrité du contenu, défigurer un site
Voler de l'argent, identité, ...

Cette attaque **n'est plus considérée comme une menace**, car les navigateurs mettent en oeuvre des mesures de sécurité pour se protéger.

Gestion des sessions



Gestion des sessions



Gestion des sessions



Gestion des sessions



Backend

session = 66ntna8ujcvmhjdkufht



Le navigateur cherche dans la
liste le cookie correspondant au
site

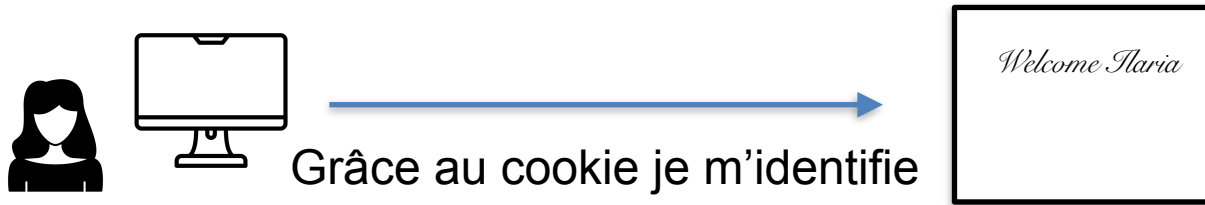
Gestion des sessions



Backend



Gestion des sessions



Backend

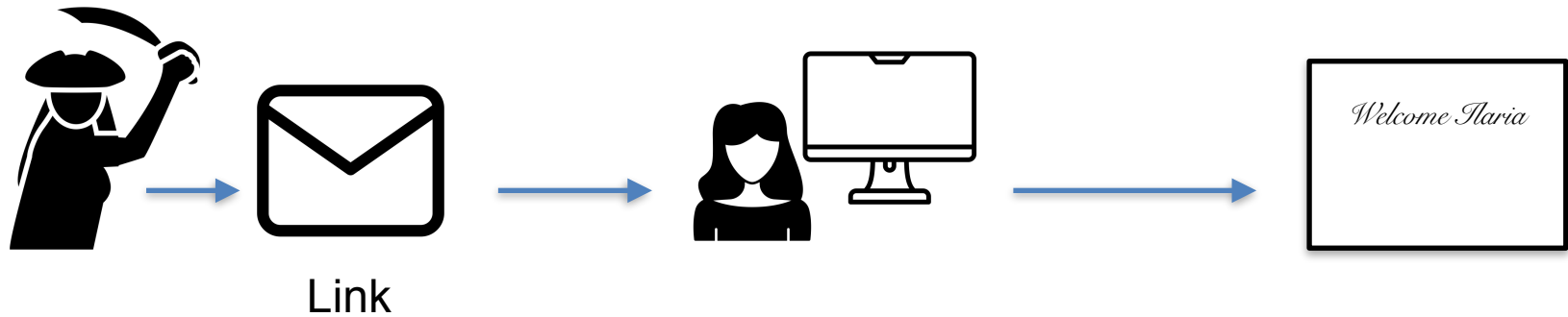
session = 66ntna8ujcvmhjdkufht



Ilaria = 66ntna8ujcvmhjdkufht
Tina = 67gtnry6jc77mhjfkuf45
Tom = 5647ghjgdhj2782bhdbg

CSRF

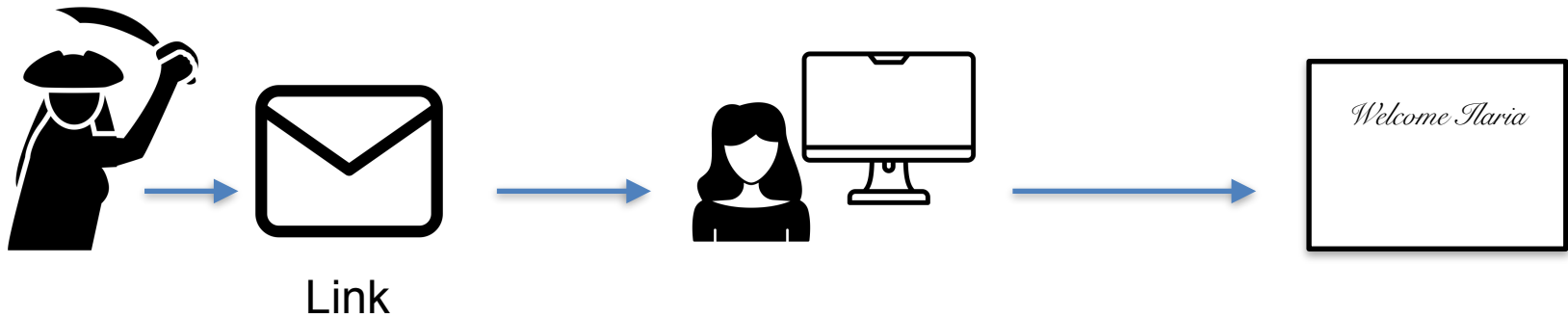
l'attaquant envoie un mail avec un lien



<https://banque.com/email/change?email=pirate@mail.com>

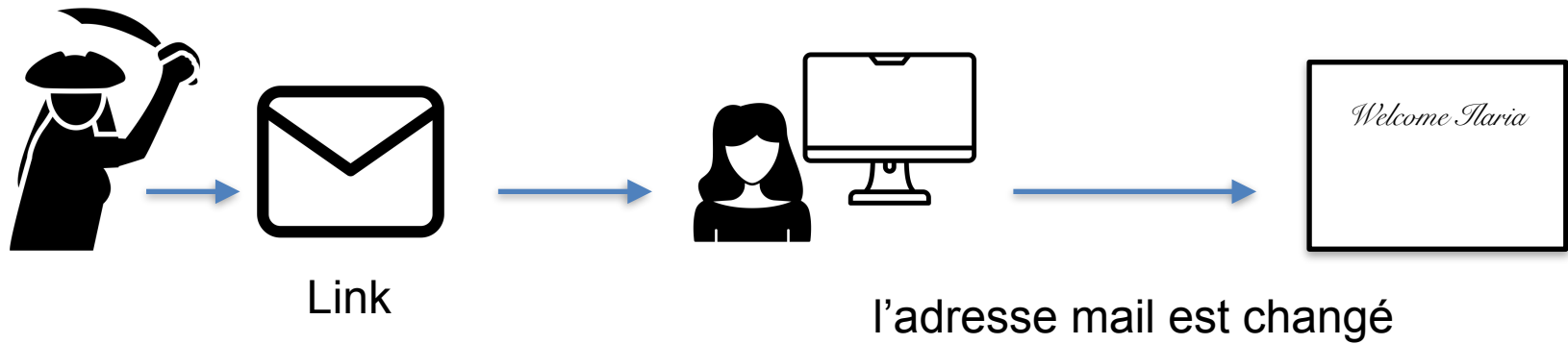
CSRF

l'utilisateur est déjà authentifié sur le site
l'utilisateur clique sur le lien



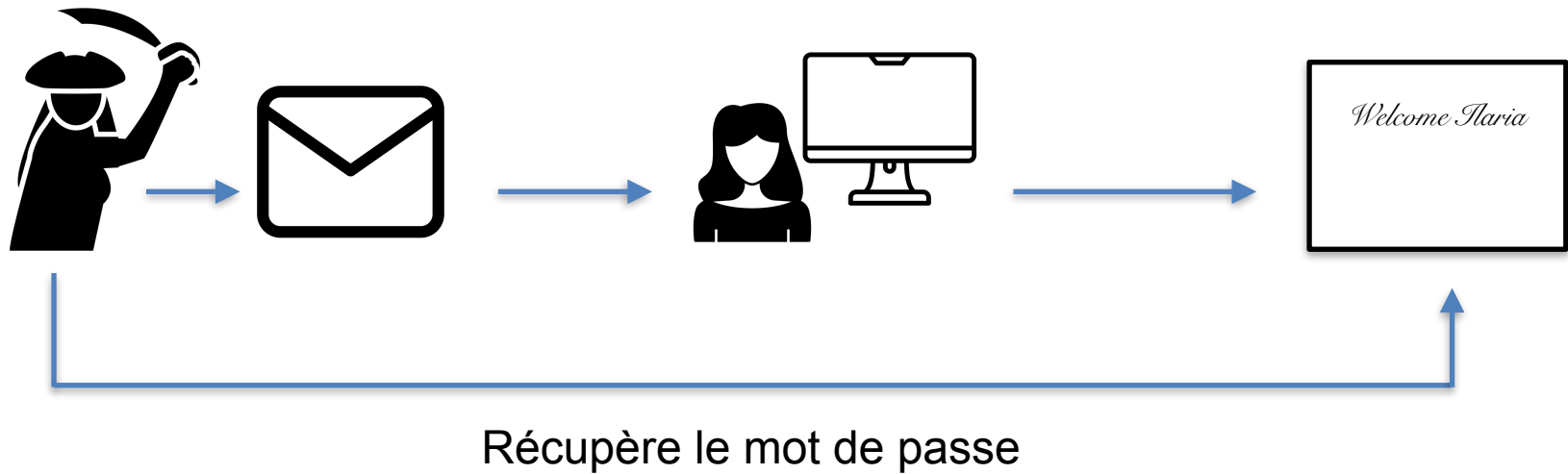
<https://banque.com/email/change?email=pirate@mail.com>

CSRF



<https://banque.com/email/change?email=pirate@mail.com>

CSRF



CSRF

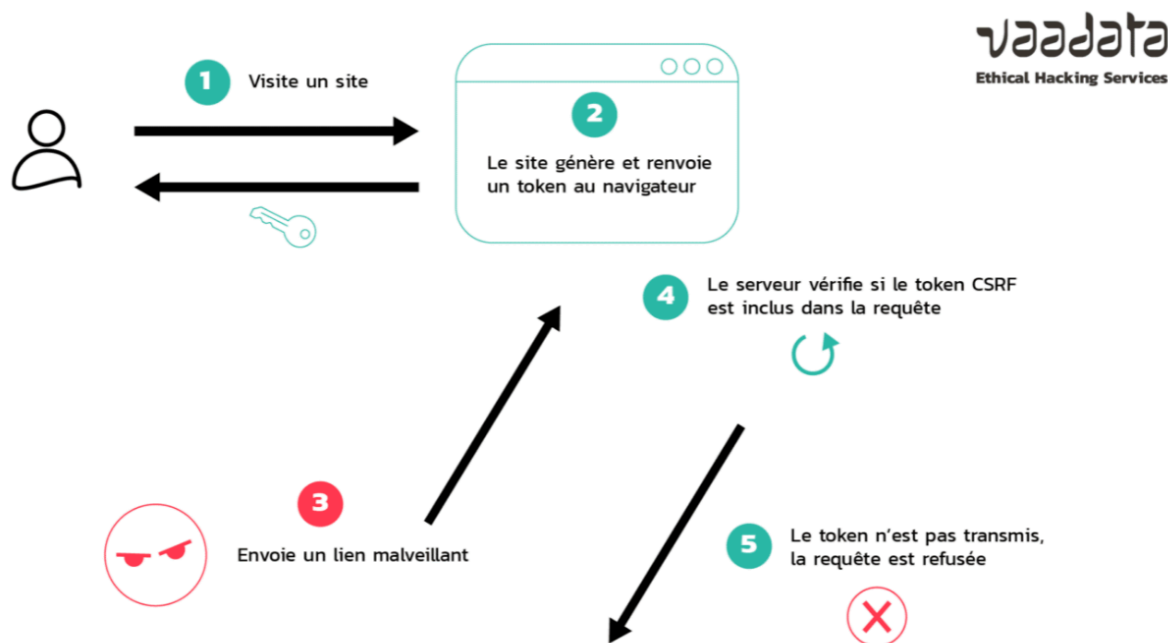


L'attaque CSRF est **possible** grâce à :

1. une **action** de la part de l'utilisateur,
2. l'**authentification basée** sur les **cookies**,
3. au fait qu'il n'y a pas de **paramètres de requête non prévisibles** (l'attaquant peut deviner les valeurs)

Comment se protéger contre les attaques CSRF ?

1. Implémenter des **jetons CSRF** ← Séquence random
2. Utiliser l'attribut SameSite sur les cookies



<https://www.vaadata.com/blog/fr/attaques-csrf-principes-impacts-exploitations-bonnes-pratiques-securite/>

Comment se protéger contre les attaques CSRF ?

1. Implémenter des **jetons CSRF**
2. Utiliser l'attribut **SameSite** sur les cookies

Set-Cookie : session = test; SameSite=Strict

Set-Cookie : session = test; SameSite=Lax

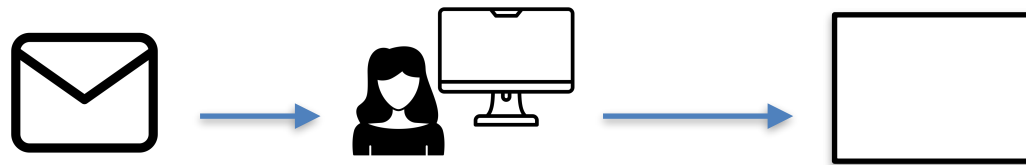
Entête d'une requête http



Comment se protéger contre les attaques CSRF ?

1. Implémenter des **jetons CSRF**
2. Utiliser l'attribut **SameSite** sur les cookies

Set-Cookie : session = test; SameSite=Strict



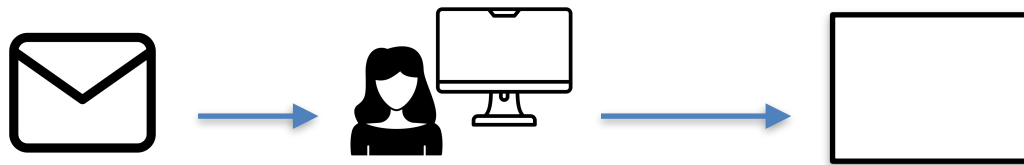
Utilisateur clique sur le lien

Requête HTTP au serveur

Comment se protéger contre les attaques CSRF ?

1. Implémenter des **jetons CSRF**
2. Utiliser l'attribut **SameSite** sur les cookies

Set-Cookie : session = test; SameSite=**Strict**



Le serveur vérifie que la
requête provienne du même
site



Rejette la requête

Comment se protéger contre les attaques CSRF ?

1. Implémenter des **jetons CSRF**
2. Utiliser l'attribut **SameSite** sur les cookies

Set-Cookie : session = test; SameSite=**Strict**



Attribut pas très utilisé

Comment se protéger contre les attaques CSRF ?

1. Implémenter des **jetons CSRF**
2. Utiliser l'attribut **SameSite** sur les cookies

Set-Cookie : session = test; SameSite=**Lax**







Le serveur envoie les cookies
sur les requêtes GET,
Navigation **inter-site**



Cliquer sur un lien du même site

Travail de la semaine 2

Partie 1

-  Créez un compte sur le site https://www.root-me.org/?var_hasard=14928371965e5d3c41e07e
-  Faites le challenge **CSRF - 0 protection**
-  Vous devez rendre un **rapport** expliquant **comment vous avez résolu le challenge** et contenant **des captures d'écran** montrant que vous avez résolu le défi sur votre machine.
-  (Bonus) - **CSRF - contournement de jeton**

XSS

Cross Site Scripting

Une attaque permettant d'injecter du contenu malveillant dans une page



en déclenchant des actions dans le navigateur

But : phishing, récupérer/voler les cookies/sessions, etc

- XSS **stockée** (persistante)
- XSS **reflétée** (non persistante)
- XSS **basée sur le DOM**

XSS

Cross Site Scripting

Une attaque permettant d'injecter du contenu malveillant dans une page



en déclenchant des actions dans le navigateur

But : phishing, récupérer/voler les cookies/sessions, etc

- XSS **stockée** (persistante)  **Très forte !**

Page stocke (dans la bd) et affiche le contenu malveillant

Moyens : messages dans un forum, formulaires, ...

XSS

Cross Site Scripting

Une attaque permettant d'injecter du contenu malveillant dans une page



en déclenchant des actions dans le navigateur

But : phishing, récupérer/voler les cookies/sessions, etc

- XSS **reflétée** (non persistante)  **La plus courante !**

Page affiche le contenu malveillant, interprété par le navigateur

Moyens : liens externes (mails, sites, etc)

XSS

Cross Site Scripting

Une attaque permettant d'injecter du contenu malveillant dans une page



en déclenchant des actions dans le navigateur

But : phishing, récupérer/voler les cookies/sessions, etc

- XSS basée sur le DOM → **Difficile à détecter !**

Faible observée dans le DOM

XSS

Cross Site Scripting

Une attaque permettant d'injecter du contenu malveillant dans une page



en déclenchant des actions dans le navigateur

But : phishing, récupérer/voler les cookies/sessions, etc

- XSS **stockée** (persistante)
- XSS **reflétée** (non persistante)
- XSS **basée sur le DOM**

Comment on se protège ?

Travail de la semaine 2

Partie 2

- 📌 Choisissez un challenge sur la **faille XSS** (root me)
- 📌 Vous devez rendre un **rapport** expliquant **comment vous avez résolu le challenge** et contenant **des captures d'écran** montrant que vous avez résolu le défi sur votre machine.
- 📌 Expliquez les **recommandations ANSSI** contre les vulnérabilités XSS.