Séance 13 :

→ REACT.JS – Concepts de sécurité Web

INF37407 – Technologie de l'inforoute

Prof. **Yacine YADDADEN**, Ph. D.

UQAR

Plan

1. Aperçu de la sécurité
2. Principes généraux
3. Filtrage des entrées & contrôle des sorties
4. Les attaques les plus communes
5. Cryptage & authentification
6. Questions et discussion

UQAR

Plan

1. **Aperçu de la sécurité**
2. Principes généraux
3. Filtrage des entrées & contrôle des sorties
4. Les attaques les plus communes
5. Cryptage & authentification
6. Questions et discussion

Définitions

- **Qu’est-ce que la sécurité :**
  - *« C’est le fait d’être protégé ou être sous la protection d’un certain danger. »*
- **Qu’est-ce que la sécurité Web :**
  - *« C’est le fait de protéger le serveur Web ainsi que les applications qui y sont hébergées de tout danger externe. »*
- La présence de **sécurité** implique la **confiance** des utilisateurs et clients,
- **Sécurité** = *Conscience et connaissance* + *Actions et protections*

UQAR

Son importance

- Facilité de création de site Web,
- **Complexité** = augmentation des **risques** d'attaque,
- Différentes *attaques* et *actions* **malveillantes**,
- Réaction en chaîne & atteinte à la réputation du service,
- **Sécurité** = Protection du *propriétaire* + Protection des *clients*

Hacker !

- Les deux faces de la force :
  - **White** hat hacker : *bienveillant* ☺
  - **Black** hat hacker : *malveillant* ☹
- Plusieurs catégories de **Black** hat hacker :
  - Curieux,
  - Soif d'adrénaline,
  - Chasseurs de trophées,
  - Activistes,
  - Professionnels.

- « Un *système impénétrable* ou une *sécurité absolue* sont des **mythes** »,
- « Un système est dit *sécurisé* s'il est *protégé* de toutes les *menaces* et *attaques connues* ».

Consciences de la sécurité

- La sécurité n'est pas juste l'affaire des développeurs et des IT,
- Nécessité de mises à jour et d'évaluations régulières,
- Nécessité d'informations et des suivis des changements,
- Documentation, procédures & politique de sécurité,
- La **sécurité** est l'affaire de **tout le monde** !

UQAR

Plan

1. Aperçu de la sécurité
2. **Principes généraux**
3. Filtrage des entrées & contrôle des sorties
4. Les attaques les plus communes
5. Cryptage & authentification
6. Questions et discussion

Moindre accès

- **Principe :** *« consiste à attribuer le moins de privilèges ou d'accès à un utilisateur, juste le minimum. »,*
- Permet d'avoir plus de contrôle sur l'accès aux données,
- Facilite la localisation de la vulnérabilité,
- Évite les réactions en chaîne.

Simplicité

- ↗ complexité ⇒ ↗ de risques d'attaques,
- Révision du code régulière pour détection de vulnérabilité,
- Éviter un code trop complexe !
- Désactiver les fonctionnalités inutiles,
- *« Qu'est-ce qui est plus sécuritaire que de verrouiller une porte, c'est de la supprimer complétement. ».*

Ne pas faire confiance

- *« L'erreur est humaine ! »,*
- Sécurité = être paranoïaque … des pirates partout !
- L'utilisateur est généralement la faille : *consciemment* ou *inconsciemment* !
- *Mise à jour régulière* des **autorisations** & **privilèges**.

UQAR

S’attendre à tout !

- En termes de sécurité, il faut être **proactif** et non **réactif** !
- Anticiper le pire des scénarios possible,
- Se mettre à la place du hacker.

UQAR

Plusieurs niveaux

- Inspiré d'une stratégie militaire,
- La défense par plusieurs couches,
- Différents types de protections par couche,
- Ralentir l'attaque pour se donner le temps de réagir.

UQAR

Sécurité par l'obscurité

- N'exposer que le minimum d'informations,
- ↘ d'informations ⟹ ↗ de sécurité,
- **Exemple :** *versions du serveur, type de SGBD, …*

Listes blanches & noires

- **Différences :**
  - Blanches : *personnes ou objets autorisés,*
  - Noires : *personnes ou objets non autorisés (bannis),*

- Appliquer la *restriction par défaut* – Liste blanches.

UQAR

Points d’entrés & de sorties

- Points d’entrés :
  - URLs,
  - Formulaires,
  - Cookies/Sessions,
  - Lecture de base de données, …
- Points de sorties :
  - HTML,
  - JavaScript,
  - Écriture de base de données, …

Sécurité = *Conscience et connaissance* + *Actions et protections*

UQAR

Plan

1. Aperçu de la sécurité
2. Principes généraux
3. **Filtrage des entrées & contrôle des sorties**
4. Les attaques les plus communes
5. Cryptage & authentification
6. Questions et discussion

UQAR

Contrôle des requêtes

- Différents types :
  - **GET** : URLs & lien HyperText
  - **POST** : Formulaires.
  - **CONNECT**, **DELETE**, **PUT**, …
- **N'autoriser que ce qui est attendus !**
- Formats :
  - « *Content-Type* » : *de ce qui est envoyé*
  - « *Accept* » : de ce qui est renvoyé ou retourné

Validation des entrées

- Vérification de ce qui est envoyé par l'utilisateur,
- Définir ce qui est attendu : *accepté* ou *refusé*,
- Se prévenir des attaques et des bugs,
- Compatibilité des *données entrées* et celles de la *base de données,*
- **Validations :**
  - *Présence / Longueur,*
  - *Type d'entrée,*
  - *Unicité,*
  - *Format.*

UQAR

Traitement des données

- Vérifier et traiter les données arrivant au serveur :
  - **Encodage des caractères :** remplacer `"<"` par `"<"` en HTML.
  - **Échapper des caractères :** ajouter un `\` dans `"WHERE userID=\'attack\'"`

- Utiliser des méthodes existantes fournies avec le langage,
- Faire les traitements adéquats selon la destination finale des données reçues.

UQAR

Nommage des variables

- Donner une indication sur l'état des données :
  - **Non traitées :** *raw, unsafe, …*
  - **Traitées :** *safe, clean, …*
- **Exemple :**

```
$raw_email = $_POST['email'];
$safe_email = processing($raw_email);
```

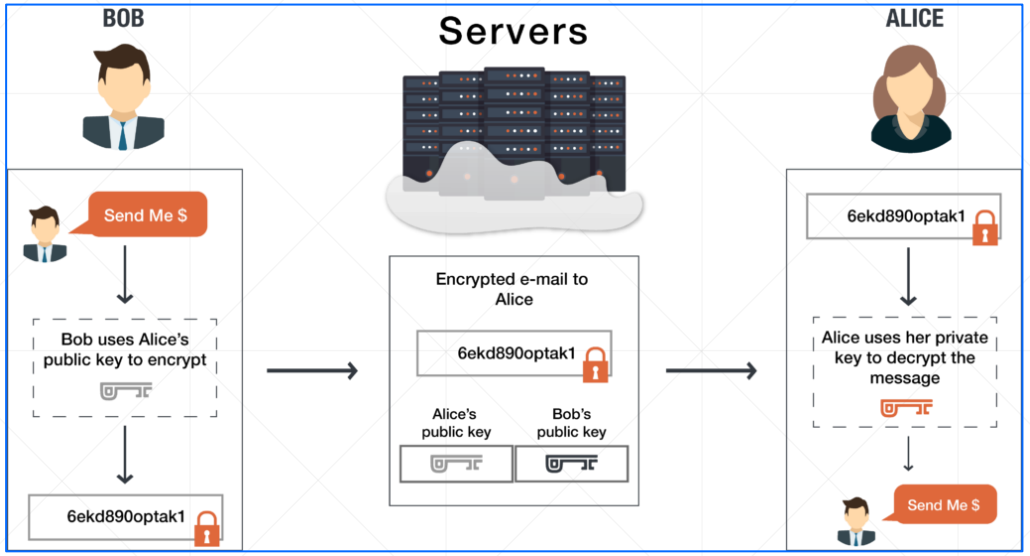
UQAR

Cacher le code

- Restreindre l'accès au code en termes de visibilité,
- Deux types de répertoires :
  - **Publique :** *code sans risques et simple*
  - **Privé :** *code complexe et sensible*
- La configuration des accès se fait au niveau du serveur Web :
  - Définir le *dossier racine*,
  - Définir les *autorisations* et les *restrictions*.
- Le fichier **`.htaccess`** : *autorisation*, *authentification*, …

Cacher les accès

- Ne pas exposer les accès dans le code : *identifiants* & *mots de passes*,
- Mettre les accès dans un fichier séparé (*contrôle de version*),
- Ne pas réutiliser les mots de passes (*serveur, base de données, …*),
- Encoder (*hacher*) le mot de passe,
- Utiliser un *cryptage* ou *chiffrement* :
  - Clé *publique*,
  - Clé *privée*.



UQAR

Rester vague dans les messages

- Renvoyer le minimum d'information à l'utilisateur,
- C'est suffisant pour un utilisateur *lambda*,
- Plus d'informations $\Longrightarrow \nearrow$ de risques d'attaques.

Garder des *logs*

- C’est des *journaux d’événements* ou *d’actions*,
- **Rôle :** suivi, historique & analyse (*éventuelle attaque*),
- **Informations à garder :**
  - *Date et heure,*
  - *Source (utilisateur ou adresse IP),*
  - *Action réalisée,*
  - *Cible,*
  - *URL & paramètres.*
- Détecter ce qui s’est passé.

UQAR

Plan

1. Aperçu de la sécurité
2. Principes généraux
3. Filtrage des entrées & contrôle des sorties
4. **Les attaques les plus communes**
5. Cryptage & authentification
6. Questions et discussion

Cross-Site Scripting ou XSS

- **Principe :**
  - *Permet au hacker ou pirate d'injecter du code JavaScript dans une page Web,*
  - *Ensuite faire en sorte que l'utilisateur ou victime l'exécute.*
- **Exemple :**

Normal

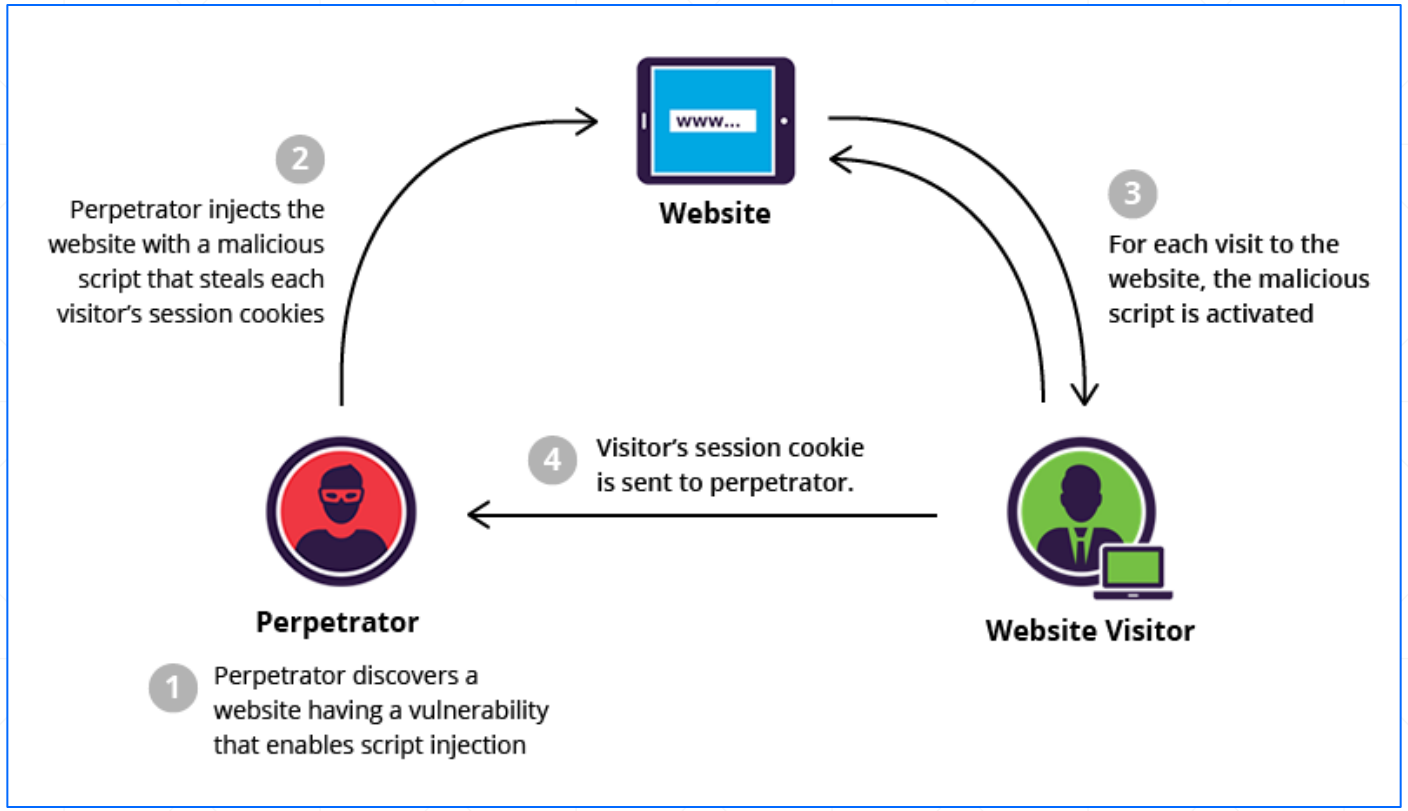
```
GET /update.jsp?email=bouchard@email.ca
Email: <%= email %>
Email: email=bouchard@email.ca
```

Attaque

```
GET /update.jsp?email=<script>alert("attaque!");</script>
Email: <%= email %>
Email: <script>alert("attaque!");</script>
```

- **Solution :** *Vérifier ce qui est envoyé au serveur.*

Illustration



Cross-Site Request Forgery (CSRF)

- **Principe :**
  - *Permet au hacker ou pirate d'envoyer une requête **HTTP** falsifiée,*
  - *Faire en sorte de faire exécuter une action par un utilisateur ou victime à son insu.*
- **Exemple :**

Lien dans un forum

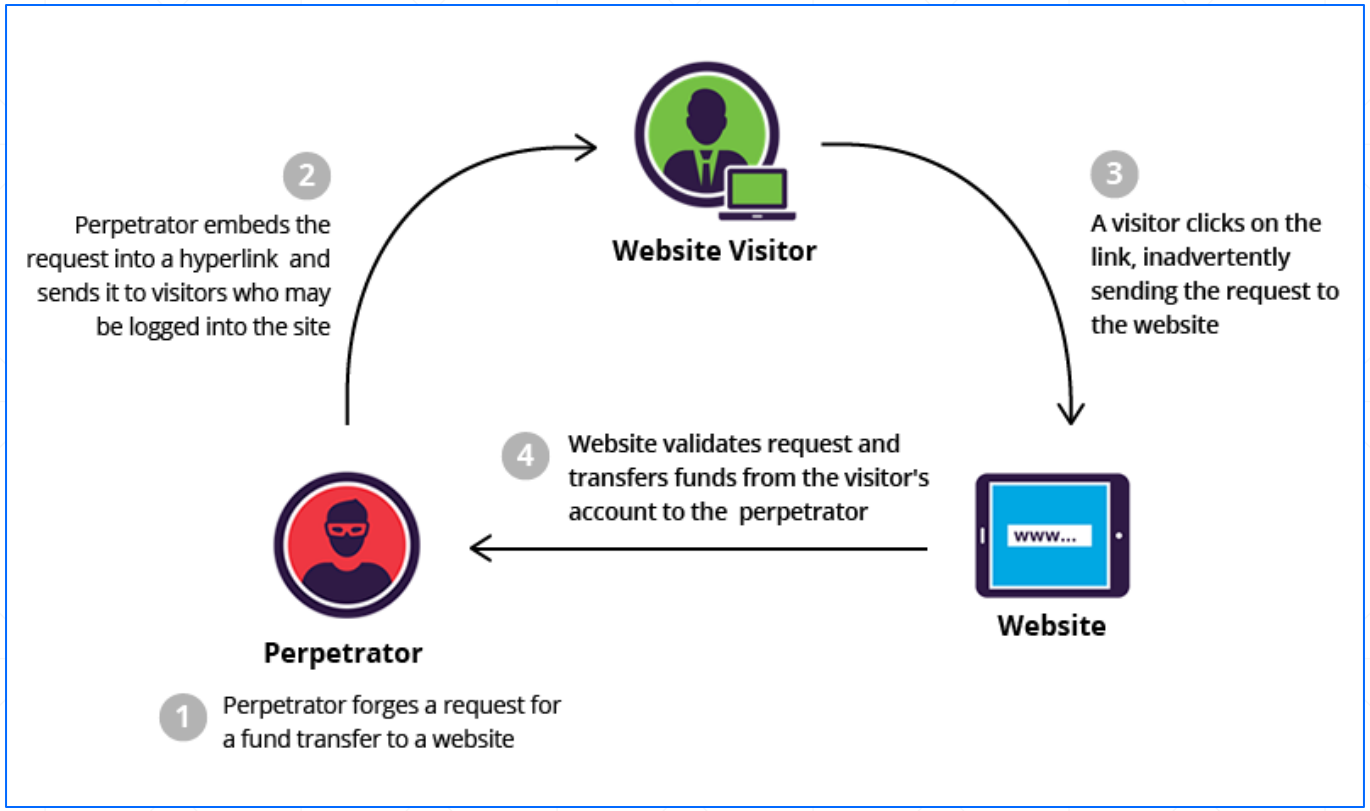
Vote en ligne dans le cadre d'un concours

Bouton de vote pour un utilisateur :
https://site-de-vote.ca/vote?User=1008

```
<img src="https://site-de-vote.ca/vote?User=1008" />
```

- **Solutions :**
  - *Ne pas utiliser les requêtes GET pour réaliser des opérations,*
  - *Utiliser des **tokens** dans des formulaires pour plus de sécurité.*

Illustration



UQAR

SQL Injection

- **Principe :**
  - *Permet au hacker d'exécuter des requêtes **SQL**,*
  - *Altération, destruction et vol de données.*
- **Exemple 01 :**

Normal

```
SELECT * FROM users WHERE username='${username}' AND password='${password}';
username = "sbouchard"
password = "motDePasse"
SELECT * FROM users WHERE username='sbouchard' AND password='motDePasse';
```

Attaque

```
username = "sbouchard' OR 1 = 1; --"
password = "vide"

SELECT * FROM users WHERE username='sbouchard OR 1 = 1; --' AND password='motDePasse';
```

SQL Injection

- **Exemple 02 :**

```
SELECT * FROM articles WHERE title = '${query}'

query = "q'; DROP TABLE clients; --"
```

- **Solutions :**
  - *Définir des privilèges spécifiques à l'utilisateur de l'application,*
  - *Traitement des entrées utilisateur (échappement des caractères),*
  - *Utiliser des **requêtes préparées** ou **prepared statements**.*

URL manipulation

- **Principe :**
  - *Permet au hacker ou pirate d'éditer et de modifier l'**URL**,*
  - *Avoir d'avoir de l'information privée ou réaliser des actions non autorisées.*
- **Exemples :**

```
http://site-vulnerable.ca?facture=F-7884

http://site-vulnerable.ca?UserID=54789651

http://site-vulnerable.ca/images/small/paysage.jpg
```

- **Solutions :**
  - *Rester vague dans les messages d'erreurs (pas de feedbacks),*
  - *Éviter que les requêtes GET puissent effectuer des actions.*

Fakes forms

- **Principe :**
  - *Permet au hacker ou pirate de récupérer un formulaire et de l'adapter,*
  - *Exécuter certaines opérations qui ne sont pas forcément autorisées.*

- **Solutions :**
  - *Ne pas se baser uniquement sur la validation côté client (JavaScript),*
  - *Utiliser des **tokens** ou jetons pour chaque formulaire.*

UQAR

Cookies visibility & theft

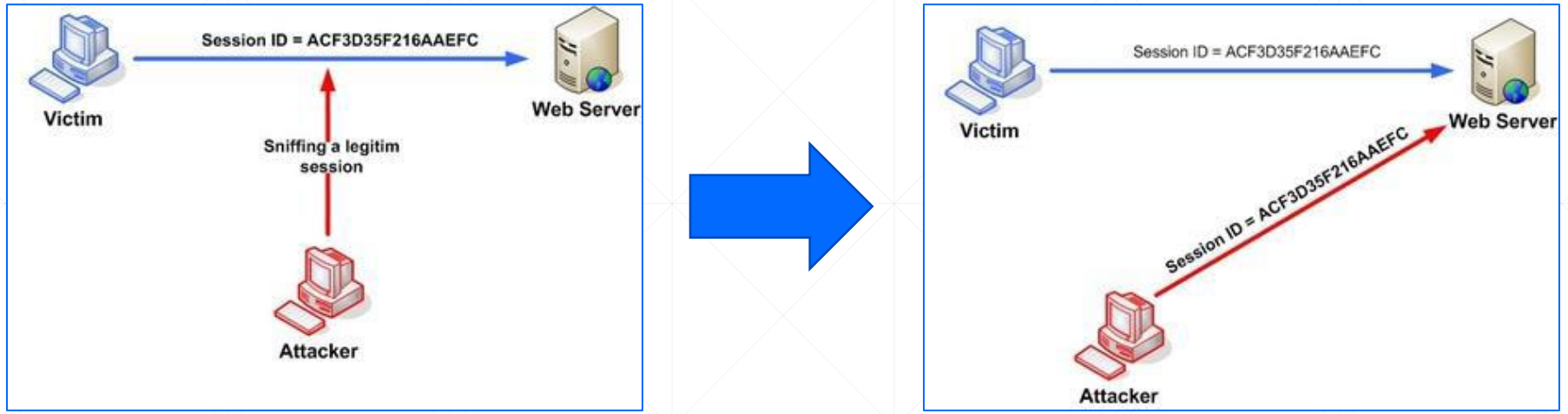
- **Principe :**
  - *Permet au hacker ou pirate de voir les informations stockées dans les cookies,*
  - *Ils peuvent être volés grâce à une attaque XSS ou sniffé dans le réseau.*

- **Solutions :**
  - *Ne pas stocker des données sensibles dans les cookies,*
  - *Utiliser le mode* ***`HttpOnly`*** *rendant les cookies inaccessibles par le JavaScript,*
  - *Utiliser des cookies sécurisées (**HTTPS**),*
  - *Définir une date d'expiration,*
  - *Préférer utiliser des **sessions serveur** que les **cookies**.*

Session hijacking

- **Fonctionnement :**
  - *Les informations sensibles sont sur le serveur,*
  - *Tandis qu'un cookie est envoyé au client avec l'ID de la session.*
- **Principe :**
  - *Permet au hacker ou pirate de détourner et de prendre possession de l'ID de la session,*
  - *Puis l'utiliser pour paraître connecté en tant que la victime.*
- **Solutions :**
  - *Double vérification avec les entêtes ou bien l'adresse IP,*
  - *Utiliser le mode* ***`HttpOnly`****,*
  - *Regénérer les sessions périodiquement et mettre une date d'expiration,*
  - *Utiliser une connexion* ***SSL****.*

Illustration

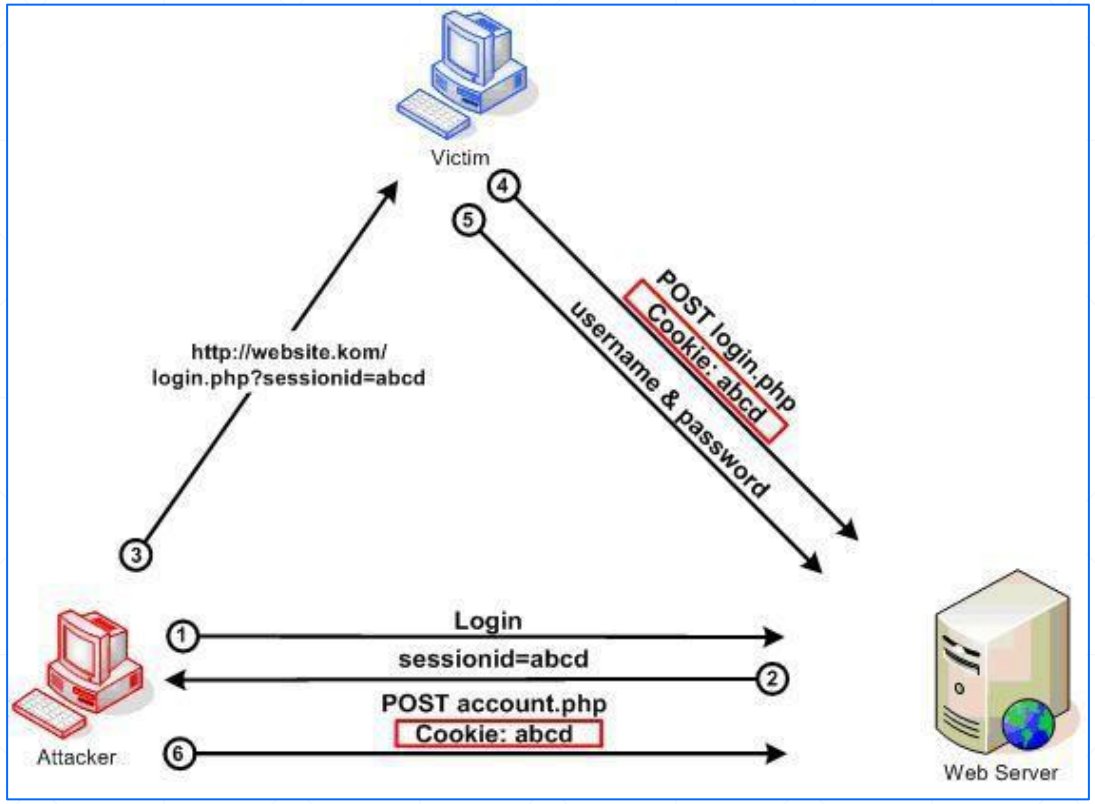


Session fixation

- **Principe :**
  - *Permet au hacker ou pirate de donner à la victime son ID de la session,*
  - *Puis de pouvoir se connecter au compte de la victime.*

- **Solutions :**
  - *Ne pas passer les IDs de session dans des requêtes GET ou POST,*
  - *Regénérer les sessions périodiquement et mettre une date d'expiration,*

Illustration



UQAR

Remote command execution

- **Principe :**
  - *Permet au hacker ou pirate d'exécuter des commandes système sur le serveur Web,*
  - *Accès à toutes les fonctionnalités du système d'exploitation,*
  - *La plus critique et dangereuse des attaques, mais la plus difficile à mettre au point.*

- **Solutions :**
  - *Éviter l'utilisation de certaines commandes (**`call`**, **`exec`**, **`sh`**, …),*
  - *Traiter les données entrées par l'utilisateur,*
  - *Comprendre le fonctionnement des commandes et ajout de validation.*

File-upload abuse

- **Principe :**
  - *Permet au hacker ou pirate d'abuser des fonctionnalités de téléversement,*
  - *Téléverser beaucoup de fichier causant une saturation de l'espace,*
  - *Téléversement de fichiers malicieux ou malveillants.*

- **Solutions :**
  - *Imposer une authentification avant de faire un téléversement,*
  - *Limiter la taille maximale du téléversement,*
  - *Définir les formats de fichiers autorisés,*
  - *Faire attention lors de l'ouverture de fichiers ou imposer une vérification.*

Denial of Service (DoS)

- **Principe :**
  - *Permet au hacker ou pirate de rendre le serveur inaccessible,*
  - *En le submergeant de requêtes,*
  - *En surutilisant l'espace disque, le processeur ou la bande passante.*
- **Distributed** DoS ou **DDoS**,
- **Facile** à mettre en place et **difficile** à prévenir.
- **Solutions :**
  - *Configurer les pare-feu et les équipements réseaux,*
  - *Configurer une haute disponibilité,*
  - *Il n'y pas de vraie solution.*

UQAR

Plan

1. Aperçu de la sécurité
2. Principes généraux
3. Filtrage des entrées & contrôle des sorties
4. Les attaques les plus communes
5. **Cryptage & authentification**
6. Questions et discussion

UQAR

Chiffrement des mots de passe

- **Attention :**
  - Ne jamais stocker les mots de passes dans un format lisible,
- **Chiffrement non-réversible :**
  - Stocker le mot de passe chiffré (*algorithme de **hachage***),
  - Réutiliser le même algorithme pour la vérification.
- **Algorithmes de chiffrement :**
  - ~~MD5~~,
  - SHA-1,
  - SHA-2 (SHA-256 ou SHA-512),
  - AES,
  - …

UQAR

Salage des mots de passe

- **Menace :**
  - Les ***rainbow tables*** ou *table arc-en-ciel* sont des tables de haches précalculés,
  - Elles sont utilisées pour trouver le bon mot de passe.
- **Le salage :**
  - **Rôle :** *Renforcer la sécurité en ajoutant une chaine de caractère au mot de passe.*
  - **Exemple :** `"salage de mot de passe pour {$password}"`
  - **Méthodes :**
    - Salage *unique* pour chaque utilisateur,
    - Salage *aléatoire,*

Chiffrement **Blowfish**

  - Il faut également *hacher* le *salage* avant de le stocker dans la base de données.

UQAR

Robustesse du mot de passe

- Ne pas limiter la longueur du mot de passe,
- Utiliser des caractère spéciaux,
- Toujours demander une confirmation du mot de passe,
- Donner un feedback sur la robustesse du mot de passe,
- Ne pas donner d'indicateur pour le mot de passe,
- Les questions de sécurité peuvent représenter une faille.

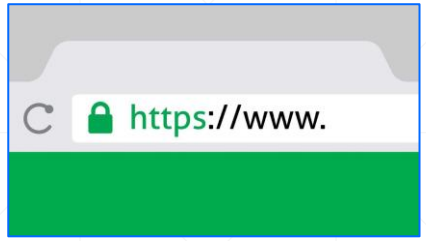
- **Astuce :** *Utiliser des générateurs de mots de passes.*

Attaque par force brute

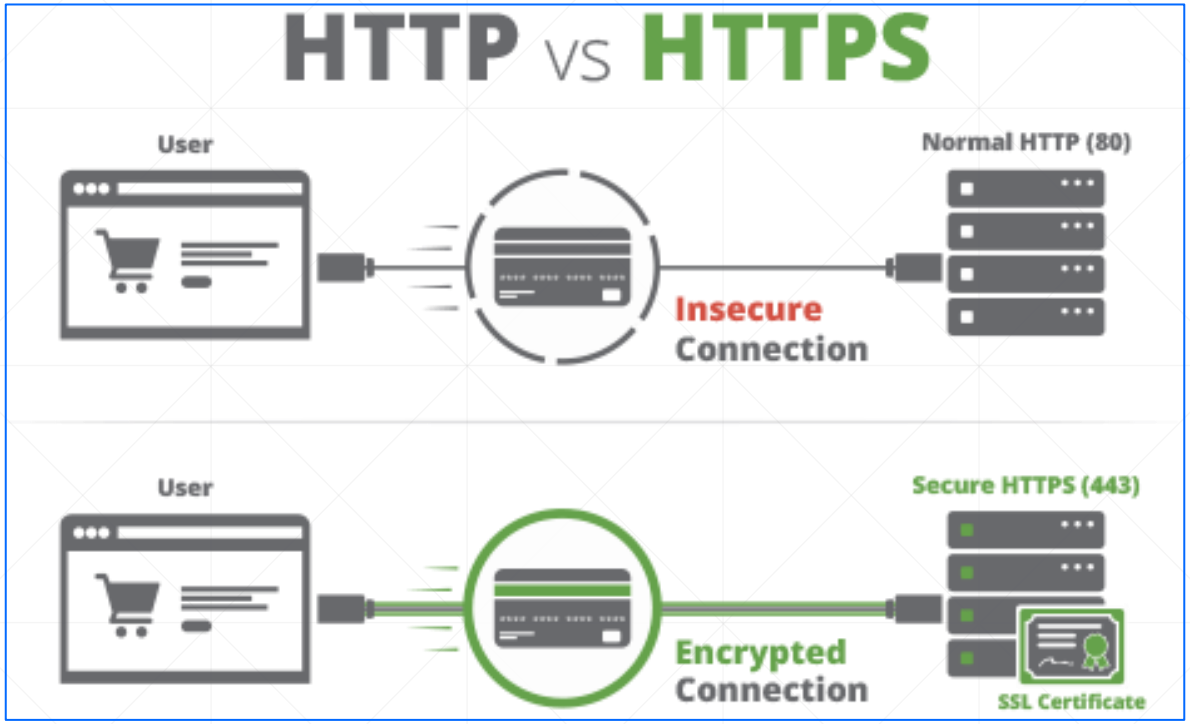
- **Principe :**
  - *Tester toutes les combinaisons possibles pour s'authentifier,*
  - *Utiliser des dictionnaires pour réduire le temps de recherche.*
- **Temps requis :**
  - $temps\ total = \ nombre\ de\ caractères^{taille\ du\ mot\ de\ passe} \times temps\ pour\ une\ tentative$
  - Faire de sorte que le $temps\ total$ soit le plus élevé possible.
- **Astuce :**
  - *Demander aux utilisateurs un mot de passe robuste,*
  - *Utiliser un algorithme de chiffrement lent,*
  - *Faire des pauses entre chaque tentative,*
  - *Utiliser les journaux d'événements.*

Utilisation du SSL

- Ou **Secure Sockets Layer**,
- **Rôle :** *permet de fournir une communication sécurisée.*
- **Fonctionnalités :**
  - *Vérifie et confirme l'authenticité du serveur distant,*
  - *Chiffre les données échangées.*



Utilisation du SSL



UQAR

Gestion des accès

- Respecter le principe du moindre privilège,
- Être organisé et établir une bonne documentation,
- Faciliter la gestion des accès,
- Définir des rôles et des niveaux de sécurité,
- Restreindre l’accès aux outils de gestion et administration.

UQAR

Réinitialiser un mot de passe

- **Principe :**
  - *Se baser sur une information qui ne peut être connue que par l'utilisateur,*
  - *Utilisation d'une adresse email.*
- **Fonctionnement :**
  1. *Envoyer un email pour la réinitialisation,*
  2. *Toujours demander le nom d'utilisateur,*
  3. *Utilisation de **tokens** avec une date d'expiration dans un lien,*
  4. *Le compte rester actif et sans changement tant que l'utilisateur ne le l'a pas changé.*

Authentification multimodales

- Requière au *minimum deux facteurs* d'authentification,
- **Facteurs :**
  - Ce que l'utilisateur **connait** : *mot de passe.*
  - Ce que l'utilisateur **possède** : *carte d'accès, carte de crédit ou débit.*
  - Ce qui **fait partie** de l'utilisateur : *biométrie (empreintes digitales, voix, visage, …).*
- **En pratique :**
  - *Utilisation de l'adresse IP de l'ordinateur,*
  - *Envoyer un email ou sms avec un code de confirmation.*

UQAR

Questions & Discussion

UQAR

Bibliographie

1. Ebel, F., Baudru, S., Crocfer, R., Puche, D., Hennecart, J., Lasson, S., & Agé, M. (2006). Sécurité informatique-Ethical Hacking-Apprendre l'attaque pour mieux se défendre. *Editions ENI*.

2. Hoffman, A. (2020). Web Application Security: Exploitation and Countermeasures for Modern Web Applications. *O'Reilly Media, Inc.*