



Cours 4 Sécurisation des services Web

Un cours de Yann Fornier



Session 4 : Sécurisation des services web

- Les différentes méthodes de sécurisation des services web (authentification, autorisation, chiffrement, etc.)
- L'utilisation de HTTPS pour protéger les communications entre les services web
- **Exercice** : sécuriser un service web en utilisant les différentes techniques vues en cours
- **QCM** : connaissances sur la sécurisation des services web



Introduction

Les services web sont de plus en plus utilisés pour offrir des services en ligne, tels que des sites e-commerce, des réseaux sociaux et des applications d'entreprise. Pour protéger ces services et les données qu'ils utilisent, il est important de mettre en place des mécanismes de sécurité efficaces. Les principales méthodes de sécurisation des services web incluent l'authentification, l'autorisation, le chiffrement, et l'utilisation de HTTPS.



Authentification

L'authentification est le processus de vérification de l'identité d'un utilisateur ou d'un système. Il existe plusieurs méthodes d'authentification couramment utilisées pour les services web, notamment les mots de passe, les jetons d'accès, et les jetons d'actualisation.



Authentification

L'authentification basée sur les mots de passe est la méthode la plus courante. L'utilisateur entre un nom d'utilisateur et un mot de passe pour se connecter au service web. Ce dernier vérifie ensuite ces informations avec un serveur d'authentification pour valider l'identité de l'utilisateur.



Authentification

L'authentification basée sur les jetons d'accès utilise des jetons chiffrés pour vérifier l'identité de l'utilisateur. Lorsqu'un utilisateur se connecte au service web avec ses informations d'identification, le service web lui attribue un jeton d'accès. Ce jeton est ensuite utilisé pour vérifier l'identité de l'utilisateur lorsqu'il effectue des requêtes ultérieures au service web.



Authentification

L'authentification basée sur les jetons d'actualisation utilise des jetons chiffrés pour vérifier l'identité de l'utilisateur et pour actualiser les jetons d'accès. Lorsqu'un utilisateur se connecte au service web avec ses informations d'identification, le service web lui attribue un jeton d'accès et un jeton d'actualisation. Lorsque le jeton d'accès expire, l'utilisateur peut utiliser le jeton d'actualisation pour obtenir un nouveau jeton d'accès.



Authentification

L'authentification peut être implémentée en utilisant des mots de passe chiffrés avec des bibliothèques telles que `bcrypt` ou `passlib` pour vérifier les informations d'identification de l'utilisateur. Par exemple, voici comment vérifier un nom d'utilisateur et un mot de passe :

```
import bcrypt

# Hash a password for the first time, with a randomly-generated salt
password = b"supersecretpassword".encode('utf-8')
salt = bcrypt.gensalt()
hashed = bcrypt.hashpw(password, salt)

# Check that an unencrypted password matches one that has previously been
# hashed
password_provided = "supersecretpassword".encode('utf-8')
if bcrypt.checkpw(password_provided, hashed):
    print("The password is correct")
else:
    print("The password is incorrect")
```




Autorisation

L'autorisation est le processus de vérification des autorisations d'un utilisateur ou d'un système pour accéder à des ressources spécifiques. Il existe plusieurs méthodes d'autorisation couramment utilisées pour les services web, notamment les rôles et les politiques d'accès.



Autorisation

Les rôles sont utilisés pour attribuer des autorisations à des utilisateurs ou à des groupes d'utilisateurs. Par exemple, un administrateur peut avoir accès à toutes les fonctionnalités d'un service web, alors qu'un utilisateur standard n'a accès qu'à certaines fonctionnalités limitées. Pour implémenter les rôles dans un service web, nous pouvons utiliser des frameworks de sécurité.



Autorisation

Les rôles d'utilisateur peuvent être implémentés en utilisant des bibliothèques telles que **Flask-Principal** ou **Flask-Security**. Par exemple, voici comment définir des rôles pour un utilisateur et vérifier s'ils ont accès à une certaine ressource :

```
from flask_principal import RoleNeed, UserNeed, Permission

admin_permission = Permission(RoleNeed('admin'))
moderator_permission = Permission(RoleNeed('moderator'))

# assign roles to a user
user = User(username='john', roles=['admin', 'moderator'])

# check if user has admin role
if admin_permission.allows(user):
    print("User has admin role")

# check if user has moderator role
if moderator_permission.allows(user):
    print("User has moderator role")
```



Autorisation

Il est important de noter que ces exemples sont très simplifiés pour l'illustration de ces méthodes d'authentification et d'autorisation. Il est conseillé d'utiliser des bibliothèques dédiées pour gérer ces aspects de sécurité dans un environnement réel pour garantir la sécurité et fiabilité des données.



Chiffrement

Pour bien commencer lisez la page (suivante): <https://chiffrer.info/>



Chiffrement

Le chiffrement est le processus de rendre une information illisible pour toute personne qui n'est pas autorisée à y accéder. Il est utilisé pour protéger les données sensibles telles que les mots de passe, les informations bancaires et les messages confidentiels, lors de leur transmission sur des réseaux publics ou lors de leur stockage sur des systèmes non sécurisés. Il existe de nombreux algorithmes de chiffrement qui peuvent être utilisés, chacun ayant ses propres avantages et inconvénients. Certains des algorithmes les plus couramment utilisés sont :



Chiffrement

AES (Advanced Encryption Standard) : est un standard de chiffrement symétrique utilisé pour protéger les données sensibles. Il utilise une clé de 128, 192 ou 256 bits pour chiffrer les données. Il est souvent utilisé pour chiffrer les données sur les réseaux WiFi, les disques durs externes et les applications de messagerie instantanée.

```
from cryptography.fernet import Fernet  
key = Fernet.generate_key() #generates a fresh key  
cipher = Fernet(key)  
cipher_text = cipher.encrypt(b"your secret data") # b"your secret data" should be bytes
```



Chiffrement

RSA (Rivest-Shamir-Adleman) : est un standard de chiffrement asymétrique utilisé pour chiffrer les données sensibles. Il utilise une paire de clés, une clé privée pour déchiffrer les données et une clé publique pour chiffrer les données. Il est souvent utilisé pour protéger les données lors de la transmission sur des réseaux publics et pour établir des connexions sécurisées à distance.

```
import rsa

# Generate private and public key pair
(public_key, private_key) = rsa.newkeys(512)

# Encrypt a message
encrypted_data = rsa.encrypt('Secret Message'.encode(), public_key)

# Decrypt the message
decrypted_data = rsa.decrypt(encrypted_data, private_key).decode()
```




Chiffrement

Blowfish : est un algorithme de chiffrement symétrique utilisé pour protéger les données sensibles. Il utilise une clé de toute longueur de 32 à 448 bits pour chiffrer les données. Il est souvent utilisé pour chiffrer les fichiers, les disques durs et les transmissions sur des réseaux privés.

```
from pycryptodome.cipher import Blowfish

key = b'An arbitrarily long key'
cipher = Blowfish.new(key, Blowfish.MODE_ECB)
plaintext = b'Please encrypt my data'
ciphertext = cipher.encrypt(plaintext)
```



Chiffrement

Ces exemples sont donnés à titre indicatif et qu'il est important de choisir l'algorithme de chiffrement approprié en fonction des exigences de sécurité de l'application et de l'environnement dans lequel il est utilisé. Il est également important de s'assurer de tenir à jour l'algorithme et les bibliothèques utilisées pour assurer la sécurité à long terme. Il est également important de se rappeler qu'une fois que les données sont déchiffrées, il est important de les protéger avec des mécanismes de sécurité appropriés pour éviter les accès non autorisés.



Protocoles SSL & TLS

SSL et TLS sont des protocoles de sécurité utilisés pour sécuriser les communications sur Internet. Ils permettent d'établir une connexion chiffrée entre un client (généralement un navigateur web) et un serveur (généralement un site web) pour s'assurer que les données échangées entre eux restent confidentielles et intègres.



Fonctionnement de base des protocoles SSL et TLS

La communication SSL/TLS démarre par un échange de messages de salutation entre le client et le serveur, qui négocient les options de chiffrage et de compression à utiliser. Ensuite, le serveur envoie son certificat, qui contient des informations sur l'identité du serveur (telles que son nom et son adresse) ainsi que sa clé publique. Le client vérifie la validité de ce certificat à l'aide des autorités de certification (CA) et utilise la clé publique pour chiffrer une clé de session qui sera utilisée pour chiffrer les données échangées. Le client envoie également son propre certificat (s'il en possède un) et une clé de session chiffrée pour le serveur.

Une fois cette phase de négociation terminée, les deux parties peuvent échanger des données chiffrées en utilisant la clé de session convenues.



Certificats SSL et TLS

Un des éléments clés de SSL/TLS est le certificat, qui permet d'authentifier l'identité du serveur au client. Il est émis et signé par une autorité de certification (CA), qui est chargée de vérifier l'identité de l'entité qui demande le certificat. Les certificats peuvent être achetés auprès de CAs commerciales, ou il est possible de créer un certificat auto-signé pour une utilisation interne (ce qui n'est pas recommandé pour des connexions externes).



Certificats SSL et TLS

Les certificats SSL/TLS peuvent être de deux types :

Certificats à usage général : utilisés pour protéger les sites web courants, ces certificats peuvent être vérifiés par les navigateurs et les appareils mobiles courants. Ils sont émis pour un nom de domaine spécifique (par exemple, www.example.com).

Certificats étendus : utilisés pour protéger les sites web de commerce électronique, les services bancaires en ligne, etc. Ils sont émis pour une entreprise ou une organisation spécifique et incluent des informations supplémentaires sur l'entité telle que le nom d'entreprise, l'adresse, etc. Ces certificats requièrent une validation supplémentaire pour être émis.



Utilisation de SSL/TLS

La plupart des sites web modernes utilisent SSL/TLS pour protéger les communications entre les clients et les serveurs. Pour utiliser SSL/TLS sur un site web, il est nécessaire d'avoir un certificat SSL/TLS valide et de configurer le serveur web pour utiliser SSL/TLS (ce qui dépend de la plateforme utilisée).




Utilisation de SSL/TLS

Une fois configuré, les connexions au site web seront automatiquement dirigées vers une URL commençant par “https” au lieu de “http”. Les navigateurs modernes affichent également un cadenas ou un autre indicateur pour indiquer que la connexion est sécurisée.



Utilisation de SSL/TLS

Il est également possible d'utiliser SSL/TLS pour d'autres protocoles de communication, tels que SMTP (protocole de courrier électronique) et IMAP/POP (protocoles de messagerie). Il suffit de configurer le logiciel ou le service pour utiliser SSL/TLS pour établir des connexions sécurisées.




En résumé SSL/TLS est une technique de sécurisation des communications sur internet, en utilisant une authentification du serveur à travers un certificat émis par une autorité de certification et un échange de clés pour chiffrer les données échangées. Il est maintenant remplacé par TLS (Transport Layer Security) qui est plus sécurisé.




Protocoles Web

Protocole HTTP (HyperText Transfer Protocol)

HTTP est un protocole de communication qui permet de transférer du contenu hypertexte (par exemple, des pages web) sur un réseau, principalement sur Internet. Il utilise le modèle client-serveur : le client (par exemple, un navigateur web) envoie une requête au serveur (par exemple, un site web), qui répond en retournant le contenu demandé (par exemple, une page HTML, une image, un fichier).



HTTP utilise des méthodes (ou verbes) pour spécifier l'action à effectuer sur le contenu : GET pour récupérer le contenu, POST pour envoyer des données (par exemple, un formulaire), PUT pour remplacer le contenu, DELETE pour supprimer le contenu, etc. Il utilise également des en-têtes (ou headers) pour fournir des informations complémentaires sur la requête ou la réponse (par exemple, le type de contenu, la langue, la date, l'encodage).



HTTP est un protocole sans état : il ne conserve pas d'informations sur les requêtes précédentes ou suivantes, ce qui signifie que chaque requête doit être autonome et indépendante. Cependant, il existe des mécanismes (comme les cookies) qui permettent de maintenir une session entre le client et le serveur, en stockant des informations sur le côté du client (par exemple, dans un fichier sur l'ordinateur de l'utilisateur).



Méthodes HTTP

HTTP utilise des méthodes pour décrire l'action à effectuer sur les données. Voici quelques méthodes courantes :

GET : récupère une ressource à partir de l'URI (Uniform Resource Identifier) spécifié.

HEAD : récupère les informations de l'en-tête de la ressource à partir de l'URI spécifié, sans le corps de la ressource.

POST : envoie les données à l'URI spécifié pour créer une nouvelle ressource.

PUT : remplace la ressource à l'URI spécifié avec les données envoyées.

DELETE : supprime la ressource à l'URI spécifié.



Code de statut HTTP

HTTP utilise des codes de statut pour indiquer l'état de la réponse. Voici quelques codes de statut courants :

1xx (Information) : les codes de statut de cette classe indiquent une action en cours ou une demande reçue avec succès.

2xx (Succès) : les codes de statut de cette classe indiquent que la requête a été traitée avec succès.

3xx (Redirection) : les codes de statut de cette classe indiquent que la requête doit être redirigée vers une autre URI pour être traitée.

4xx (Erreur de client) : les codes de statut de cette classe indiquent une erreur dans la requête, comme une syntaxe incorrecte ou une autorisation manquante.

5xx (Erreur de serveur) : les codes de statut de cette classe indiquent une erreur sur le serveur, comme une erreur de traitement ou une ressource indisponible.



En-têtes HTTP

HTTP utilise des en-têtes pour transmettre des informations supplémentaires avec les requêtes et les réponses. Voici quelques en-têtes courants :

Accept : indique les types de contenu acceptés par le client.

Accept-Language : indique les langues acceptées par le client.

Accept-Encoding : indique les encodages acceptés par le client.

Authorization : authentifie le client auprès du serveur avec un nom d'utilisateur et un mot de passe.

Cache-Control : contrôle le comportement du cache du client et du serveur.

Connection : spécifie si la connexion doit être maintenue ouverte ou fermée après la réponse.

Content-Encoding : indique l'encodage utilisé pour le corps de la requête ou de la réponse.



En-têtes HTTP

Content-Length : indique la longueur du corps de la requête ou de la réponse en octets.

Content-Type : indique le type de contenu du corps de la requête ou de la réponse.

Cookie : envoie ou reçoit les cookies associés à la requête ou à la réponse.

Host : indique le nom de domaine du serveur cible de la requête.

If-Modified-Since : indique si la ressource doit être renvoyée uniquement si elle a été modifiée depuis la date spécifiée.

User-Agent : indique le navigateur web et les informations système du client.



Cookies

Les cookies sont de petits fichiers de données (texte, images, etc.) qui sont stockés sur l'ordinateur de l'utilisateur par un site web visité. Ils permettent de stocker des informations sur la session de l'utilisateur (par exemple, pour maintenir une connexion ou un panier d'achats), de personnaliser les pagesweb en fonction de ses préférences, de suivre sa navigation et de lui proposer des publicités ciblées.



Cookies

Les cookies sont gérés par le navigateur web et sont envoyés et reçus avec chaque requête et réponse HTTP. Ils peuvent être créés et modifiés par le site web, mais également par des tiers (comme des réseaux publicitaires ou des réseaux sociaux). Ils ont une durée de vie limitée (par défaut, jusqu'à la fermeture du navigateur) et peuvent être supprimés ou désactivés par l'utilisateur.



Cookies

Il est important de prendre en compte que les cookies peuvent être utilisés à des fins de tracking et de ciblage publicitaire, ce qui peut être perçu comme intrusif par les utilisateurs. Ils peuvent également bloquer l'accès à certaines pages web si l'utilisateur les a désactivés ou les a supprimés. Enfin, ils peuvent être volés par des tiers et utilisés à des fins malveillantes.



HTTPS (HyperText Transfer Protocol Secure)

HTTPS est une variante de HTTP qui utilise un protocole de sécurisation des données (par exemple, SSL/TLS) pour chiffrer la communication entre le client et le serveur. Cela permet de protéger les données sensibles (par exemple, des mots de passe, des données bancaires) contre la lecture et la modification par des tiers.



HTTPS (HyperText Transfer Protocol Secure)

HTTPS utilise également des certificats numériques pour authentifier l'identité du serveur et éviter les attaques de type "homme du milieu" (man-in-the-middle). Un certificat est un document électronique qui associe une clé publique (utilisée pour chiffrer les données) à un nom de domaine (par exemple, "www.example.com") et à une autorité de certification (qui valide la possession du certificat par le serveur).



HTTPS (HyperText Transfer Protocol Secure)

HTTPS est de plus en plus utilisé sur les sites web qui traitent des données sensibles ou qui gèrent des transactions financières (par exemple, des sites de e-commerce, des sites de banques en ligne). Cependant, il peut être plus lent que HTTP en raison de la nécessité de chiffrer et de déchiffrer les données, et il peut être coûteux en termes de certificats et de maintenance.



Différences entre HTTP et HTTPS

Sécurisation : HTTP n'utilise pas de protocole de sécurisation des données, tandis que HTTPS utilise SSL/TLS pour chiffrer la communication.

Authentification : HTTP n'authentifie pas l'identité du serveur, tandis que HTTPS utilise des certificats numériques pour authentifier l'identité du serveur.

Confidentialité : HTTP peut être lu et modifié par des tiers, tandis que HTTPS protège les données contre la lecture et la modification par des tiers.

Intégrité : HTTP ne garantit pas l'intégrité des données (elles peuvent être altérées en transit), tandis que HTTPS garantit l'intégrité des données grâce à la sécurisation de la communication.

Performances : HTTP peut être plus rapide que HTTPS en raison de la suppression de la sécurisation des données, mais HTTPS peut être plus lent en raison de la nécessité de chiffrer et de déchiffrer les données.



Techniques de sécurité

Pare-feux

Un pare-feu est un dispositif ou un logiciel qui contrôle les communications entrantes et sortantes d'un réseau ou d'un ordinateur. Il peut être utilisé pour protéger un site Internet en contrôlant les connexions entrantes et en bloquant les connexions malveillantes. Il existe deux types de pare-feux :

Pare-feux réseau : ces pare-feux sont physiques et sont généralement installés à l'entrée d'un réseau. Ils peuvent être configurés pour bloquer les connexions entrantes en fonction de règles de filtrage basées sur des critères tels que l'adresse IP, le port et le protocole.

Pare-feux logiciels : ces pare-feux sont installés sur un ordinateur individuel et peuvent être configurés pour bloquer les connexions entrantes et sortantes en fonction des mêmes critères que les pare-feux réseau.



Sondes d'intrusion

Les sondes d'intrusion (ou IDS, Intrusion Detection System) sont des dispositifs ou des logiciels qui surveillent un réseau ou un ordinateur pour détecter les activités malveillantes. Les sondes d'intrusion peuvent utiliser des règles de filtrage pour détecter les attaques connues ou des algorithmes de détection de comportement pour détecter les activités anormales. Les sondes d'intrusion peuvent être configurées pour envoyer des alertes en cas d'activités suspectes ou pour bloquer les connexions malveillantes.



Authentification MultiFactorielle (MFA)

L'authentification à plusieurs facteurs (ou MFA) est une technique de sécurité qui permet de s'assurer que l'utilisateur qui se connecte au site est bien celui qu'il prétend être. Elle repose sur l'utilisation de plusieurs méthodes d'authentification, tels que la combinaison d'un nom d'utilisateur et d'un mot de passe avec un code de sécurité envoyé par SMS ou un jeton physique. Cela rend plus difficile pour un attaquant de pirater un compte car ils doivent posséder non seulement les informations de connexion, mais également les informations de l'autre méthode d'authentification.



Authentification MultiFactorielle (MFA)

En résumé, pour protéger un site internet, on peut utiliser des techniques comme les pare-feux, les sondes d'intrusion, le chiffrement, la sécurité des applications et l'authentification à plusieurs facteurs. Ces mécanismes ajoutent des couches de sécurité pour protéger les données sensibles et limiter les accès non autorisés. Il est également important de maintenir ces mécanismes à jour pour s'assurer que les vulnérabilités connues sont corrigées et de surveiller en permanence les activités sur le site pour détecter tout comportement suspect.



Authentification MultiFactorielle (MFA)

Il est important de noter que la sécurité d'un site web ne repose pas uniquement sur la mise en place de ces mécanismes de sécurité, mais également sur la mise en place de bonnes pratiques de sécurité dans la conception, le développement et l'exploitation du site. Il est important de suivre les meilleures pratiques de sécurité telles que :

La sécurisation des données sensibles en utilisant des algorithmes de chiffrement forts et en stockant les données chiffrées dans des bases de données sécurisées.

La réduction des privilèges pour les utilisateurs et les applications pour limiter les accès non nécessaires.

La mise en place d'une politique de mots de passe forte pour empêcher les attaques par dictionnaire.

La restriction de l'accès aux réseaux internes sensibles en utilisant des techniques de sécurité réseau telles que les pare-feux et les VPN.


La réduction des vulnérabilités en effectuant régulièrement des tests d'intrusion et des audits de sécurité pour détecter les faiblesses.

La surveillance en continu du site web pour détecter les activités malveillantes et les comportements suspects.



Conclusion


En résumé, pour protéger efficacement un site web, il est important de combiner des mécanismes de sécurité tels que les pare-feux, les sondes d'intrusion, le chiffrement, la sécurité des applications et l'authentification à plusieurs facteurs avec des bonnes pratiques de sécurité dans la conception, le développement et l'exploitation du site web. Cela permet de renforcer les défenses pour protéger les données sensibles et limiter les accès non autorisés. Il est également important de maintenir ces mécanismes à jour et de surveiller en permanence les activités sur le site pour détecter tout comportement suspect.



Exercice : Scénario : Hack d'un compte Instagram

Vous faites partie d'une équipe de défense en cybersécurité travaillant pour une entreprise de médias sociaux ciblant les réseaux sociaux populaires pour protéger les utilisateurs contre les cyberattaques. Vous avez appris qu'un groupe de pirates informatiques est sur le point de lancer une attaque pour accéder à des comptes Instagram populaires pour voler des informations sensibles ou utiliser les comptes à des fins malveillantes.

Votre mission, si vous l'acceptez, est de vous mettre dans la peau d'un groupe de pirates informatiques, et de planifier une attaque sur un compte Instagram populaire, puis utiliser vos connaissances en matière de sécurité pour identifier les vulnérabilités de sécurité et les mécanismes de protection existants pour protéger les utilisateurs contre cette attaque potentielle.



Exercice : Scénario : Hack d'un compte Instagram

Pour accomplir cette mission, vous devrez :

Imaginé un scénario plausible d'attaque pour accéder à un compte Instagram populaire (par exemple, utiliser des techniques de phishing pour obtenir les informations de connexion ou utiliser des vulnérabilités de sécurité pour accéder au compte).

Identifier les vulnérabilités de sécurité existantes dans le système utilisé par Instagram (par exemple, les paramètres de sécurité insuffisants, les faiblesses de l'authentification).

Proposer des mesures de sécurité pour protéger les comptes Instagram contre ces vulnérabilités (par exemple, l'utilisation de l'authentification à plusieurs facteurs, la sensibilisation des utilisateurs aux techniques de phishing).

Il est important de rappeler que cet exercice est à but pédagogique et ne doit en aucun cas être une raison pour attaquer les systèmes ou les individus. Cet exercice a pour but de permettre aux étudiants de comprendre les mécanismes d'attaque pour mieux se protéger contre eux.



QCM en ligne

Lien a fournir