API Rest - Jour 2

4 Rappels sur la sécurité

4.1 Les grands principes de la sécurité informatique. Menaces et impacts potentiels

- Confidentialité
- Intégrité
- Disponibilité

4.2 Spécificités des APIs : Farming et Throttling 1

Farming :

- Consiste à utiliser une API pour récupérer des données et les réutiliser dans une autre application.
- Exemple : récupérer les données d'un site de météo pour les afficher dans une application mobile.

4.2 Spécificités des APIs : Farming et Throttling 2

Throttling :

- Consiste à limiter le nombre de requêtes qu'un utilisateur peut faire à une API.

4.3 BFA et IA : les nouvelles menaces 1

- BFA : Brute Force Attack
- Envoi de requêtes en boucle pour tester toutes les combinaisons possibles de mots de passe.

4.3 BFA et IA : les nouvelles menaces 2

- IA : Intelligence Artificielle
- Utilisation de l'IA pour trouver des failles dans les systèmes de sécurité.

4.3 BFA et IA : les nouvelles menaces 3

- Echange autour des nouvelles menaces

4.4 Les différentes injections (XSS, BSI, XSRF, RFI, XPi,…) 1

- XSS : Cross-Site Scripting
- Attaque où des scripts malveillants sont injectés dans des sites web fiables.

4.4 Les différentes injections (XSS, BSI, XSRF, RFI, XPi,…) 2

- BSI : Blind SQL Injection
- Type d’injection SQL où l’attaquant ne peut pas voir le résultat de la requête.

4.4 Les différentes injections (XSS, BSI, XSRF, RFI, XPi,…) 3

- XSRF : Cross-Site Request Forgery
- Attaque qui force un utilisateur connecté à envoyer des requêtes non désirées.

4.4 Les différentes injections (XSS, BSI, XSRF, RFI, XPi,…) 4

- RFI : Remote File Inclusion
- Attaque permettant à un attaquant d’inclure des fichiers distants dans une application web.

4.4 Les différentes injections (XSS, BSI, XSRF, RFI, XPi,…) 5

- XPi : XML External Entity
- Injection malveillante dans un chemin XML utilisé dans une application web.

4.5 Exposition de données sensibles. Sécurisation des accès 1

- Protégé les données sensibles
- Authentification
- Autorisation

4.6 Désérialisation non sécurisée. Composants vulnérables 1

Désérialisation

Processus par lequel des données structurées, souvent stockées ou transmises sous forme de texte (comme JSON ou XML), sont converties en objets utilisables par un programme

4.6 Désérialisation non sécurisée. Composants vulnérables 2

Désérialisation - risque

Le Risque : Quand cette opération de traduction n’est pas effectuée de manière sécurisée, nous ouvrons la porte à des vulnérabilités. Si les données textuelles ne sont pas fiables (par exemple, modifiées par un attaquant), leur conversion en objets peut entraîner des comportements inattendus, voire dangereux. Dans le pire des cas, cela peut permettre à un attaquant d’exécuter du code malveillant à distance.

4.6 Désérialisation non sécurisée. Composants vulnérables 3

Désérialisation - en pratique

En Pratique : Pour prévenir ce risque, il est crucial de valider et de nettoyer les données avant la désérialisation. Imaginez que vous recevez un paquet d'un expéditeur inconnu. Avant de l'ouvrir, vous devez vous assurer qu'il ne contient rien de dangereux. De la même façon, notre application doit inspecter les données avant de les traiter.

4.6 Désérialisation non sécurisée. Composants vulnérables 4

Composants Vulnérables

Nos applications modernes sont souvent assemblées à partir de diverses bibliothèques et composants tiers. C’est comme construire une maison en utilisant des matériaux achetés auprès de différents fournisseurs.

4.6 Désérialisation non sécurisée. Composants vulnérables 5

Composants Vulnérables - problème

Le Problème : Si l’un de ces composants contient des failles de sécurité non corrigées, c’est comme si une des briques utilisées pour construire notre maison avait une fissure. Un attaquant peut exploiter cette fissure pour compromettre la sécurité de l’ensemble de l’édifice.

4.6 Désérialisation non sécurisée. Composants vulnérables 6

Composants Vulnérables - solution

La Solution : Il est donc impératif de s’assurer que les composants que nous utilisons sont à jour et sécurisés. Cela implique de suivre les actualités de sécurité concernant ces composants, d’appliquer régulièrement les mises à jour fournies par les développeurs, et de tester notre application pour détecter les vulnérabilités potentielles. En bref, c’est comme inspecter et entretenir régulièrement notre maison pour garantir sa sécurité et sa solidité.

4.7 Logging et monitoring 1

- Logging : Enregistrement des événements d'un système informatique.
- Monitoring : Surveillance des performances d'un système informatique.

4.8 Présentation de l’OWASP TOP 10 1

- OWASP : Open Web Application Security Project
- TOP 10 : Liste des 10 principales failles de sécurité dans les applications web.

4.8 Présentation de l’OWASP TOP 10 2

OWASP

4.9 Découvrir le Pentesting 1

- Pentesting : Test d'intrusion
- Processus d'évaluation de la sécurité d'un système informatique ou d'un réseau en simulant une attaque de cybercriminels.

4.10 Introduction à Restler-Fuzzer 1

Outil spécifiquement conçu pour le fuzz testing des APIs REST, permettant de détecter automatiquement les failles de sécurité en envoyant des requêtes modifiées à l'API.

4.10 Introduction à Restler-Fuzzer 2

Gitlab - What is fuzz testing

Travaux pratiques : Présentation de quelques solutions de sécurisation de sites web

Atelier :

- Présenter des solutions de sites web
- Echanger autour de la sécurité des API web
- Mettre en place des solutions de sécurisation pour l'API

5.1 Sécurité de l’authentification 1

Concept Clé: L’authentification est le processus par lequel un système vérifie l’identité d’un utilisateur.

5.1 Sécurité de l’authentification 2

Méthodes Communes: Mot de passe, authentification à deux facteurs

5.1 Sécurité de l’authentification 3

- Authentification
- Autorisation

5.1 Sécurité de l’authentification 4

Atelier : Créer un système d’authenification simple

5.2 Système de logging 1

- Définition : Enregistrement des activités des utilisateurs et du système dans des journaux.
- Utilité : Permet de détecter les activités suspectes et d’analyser les incidents de sécurité.
- Bonnes pratiques : Les journaux doivent être sécurisés, régulièrement revus et archivés.

5.2 Système de logging 2

Atelier : Créer un système de logging simple

5.3 Sécurité côté serveur 1

- Enjeux : Protection des serveurs contre les intrusions et les exploits.
- Stratégies : Bonnes pratiques, robustesse des composants, tests de sécurité, monitoring.
- Sécurisation des données : Chiffrement des données stockées et en transit.

5.3 Sécurité côté serveur 2

Atelier : Sécuriser le serveur de l'API

5.4 CORS (Cross-Origin Resource Sharing) et CSRF (Cross-Site Request Forgery) 1

- CORS (Cross-Origin Resource Sharing) : Mécanisme permettant de sécuriser les requêtes entre différents domaines.
- CSRF (Cross-Site Request Forgery) : Type d'attaque où un site malveillant effectue des actions sur un autre site en utilisant les droits de l'utilisateur.
- Prévention : Politiques strictes de CORS, tokens anti-CSRF.

5.4 CORS (Cross-Origin Resource Sharing) et CSRF (Cross-Site Request Forgery) 2

A quoi sert CORS ?

- CORS est un mécanisme de sécurité qui permet à un serveur de restreindre les ressources qu'il autorise à être accessibles par un navigateur web à partir d'une origine différente.

5.4 CORS (Cross-Origin Resource Sharing) et CSRF (Cross-Site Request Forgery) 3

Atelier : Mettre en place CORS sur l'API

5.5 Canonicalization, Escaping et Sanitization 1

- Canonicalization : Transformation des données en un format standard.
- Escaping : Neutralisation des caractères spéciaux dans les entrées utilisateur pour prévenir l'injection.
- Sanitization : Nettoyage des données entrantes pour éliminer les éléments malveillants.

5.6 Gestion des permissions : Role-Based Acces vs. Resource-based access 1

- Role-Based Access Control (RBAC) : Accès basé sur les rôles assignés aux utilisateurs.
- Resource-Based Access Control (RBAC) : Accès déterminé par la relation de l'utilisateur avec les ressources.
- Comparaison : RBAC simplifie la gestion des permissions, tandis que RBAC offre une granularité plus fine.

5.7 Authentification avec OAuth2 et OpenID Connect : vocabulaire et workflow 1

- OAuth2 : Protocole permettant l'autorisation sécurisée dans un environnement simple et standard.
- OpenID Connect : Couche d'identité construite sur OAuth2, fournissant des informations d'authentification.
- Workflow : Processus d'authentification et d'autorisation, incluant l'obtention et l'utilisation de tokens.

Travaux pratiques : Recherche et exploitation de vulnérabilités d’authentification et d’autorisation 1

Atelier :

- Recherche et exploitation de vulnérabilités d’authentification et d’autorisation
- Mise en place de solutions de sécurisation pour l’API

6.1 Rappels sur la cryptographie 1

La cryptographie est l'art de protéger les informations en les transformant en un format sécurisé, connu uniquement de l'expéditeur et du destinataire.

6.1 Rappels sur la cryptographie 2

Elle inclut le chiffrement des données pour les sécuriser et le déchiffrement pour les rendre à nouveau lisibles.

6.1 Rappels sur la cryptographie 3

Utilisée pour sécuriser les communications, authentifier les utilisateurs, et garantir l’intégrité des données.

6.1 Rappels sur la cryptographie 4

Cryptographie symétrique (même clé pour chiffrer et déchiffrer) et asymétrique (clés publiques et privées).

6.1 Rappels sur la cryptographie 5

- https://chiffrer.info/

6.2 Les grands principes de JWT 1

- JWT (JSON Web Token) : Un standard ouvert (RFC 7519) pour la création de tokens d'accès qui peuvent être vérifiés de manière fiable grâce à une signature numérique.

6.2 Les grands principes de JWT 2

- Structure : Un JWT est composé de trois parties : un en-tête (header), une charge utile (payload), et une signature.

6.2 Les grands principes de JWT 3

- Utilisation : Souvent utilisé pour l'authentification et l'échange d'informations entre deux parties de manière sûre.

6.2 Les grands principes de JWT 4

Avantages : Facilité d’utilisation, interopérabilité, et la capacité d’être facilement transmis via des URL, des paramètres POST, ou dans un en-tête HTTP.

6.3 Risques et vulnérabilités intrinsèques 1

- Exposition des Informations : Les JWTs peuvent être vulnérables à l'exposition d'informations si la charge utile est mal sécurisée.

6.3 Risques et vulnérabilités intrinsèques 2

- Vol de Token : Comme tout autre système d’authentification, ils sont sujets au risque de vol de token, surtout s’ils sont transmis sans chiffrement approprié.

6.3 Risques et vulnérabilités intrinsèques 3

- Failles de Sécurité dans l'Implémentation : Les erreurs dans la mise en œuvre de JWT, comme l'utilisation de signatures faibles ou l'absence de validation, peuvent conduire à des vulnérabilités.

- Gestion des Tokens : La gestion des tokens JWT, en particulier leur révocation, peut être complexe et nécessite une attention particulière pour éviter les failles de sécurité.

Middeleware

- Dans l'architecture informatique, un middleware désigne un logiciel qui agit comme un intermédiaire entre des applications ou des composants logiciels distincts.

Travaux pratiques : Challenge sur une API non sécurisée

Atelier :

- Challenge sur une API non sécurisée
- Implémentation des notions de sécurité vues précédemment

Conclusion