Sécurité

Sommaire

Les différentes façon de tester ses applications 2

Les test unitaires 2

Les tests fonctionnels 2

Les principales failles de sécurité et comment s'en protéger 3

Top Ten OWASP 3

L'injection SQL 3

Sécuriser son application contre les injections SQL. 4

Le piratage de session 4

Se protéger des authentifications non autorisées 4

L'exposition des données sensibles 5

Comment éviter d'exposer les données sensibles en transit ? 6

Comment éviter d'exposer les données stockées ? 6

Les attaques contre le contrôle d'accès 6

Comment éviter ces attaques ? 7

Les failles XSS 7

Les bonnes pratiques pour s'en prémunir. 8

Les failles XXE 8

Se protéger contre les injection XXE 8

La désérialisation non sécurisée 9

Comment s'en protéger ? 9

Utiliser des outils dont les vulnérabilités sont connues 9

Une mauvaise configuration de sécurité. 9

La surveillance du trafic web 10

Quelques sources pour rester informé. 11

Les différentes façon de tester ses applications

Les test unitaires

Un test unitaire permet de tester le bon fonctionnement d'une partie précise d'un programme. Il permet de s'assurer que le comportement d'une application est correct.
Les avantages sont les suivants:

-Permet un retour sur le développement afin de pouvoir corriger le code rapidement.

- Permet d'avoir un filet de sécurité, si quelque chose ne fonctionne pas bien on le voit tout de suite.

-Permet d'éviter d'accumuler la dette technique, en ayant une application qui soit facilement maintenable.

Les tests fonctionnels

Un test fonctionnel est le test qui servira à tester automatiquement toutes les fonctionnalités d'une application c'est à dire toutes les fonctionnalités inscrites sur le cahier des charges du projet.
Par exemple, on pourrait tester qu'un utilisateur puisse bien s'inscrire, se connecter et se déconnecter.
L'intérêt des tests fonctionnels est de permettre de s'assurer que les modifications apportées au code n'impactent pas les autres fonctionnalités du site.

Les principales failles de sécurité et comment s’en protéger

L'Open Web Application Security Project (OWASP) est une organisation à but non lucratif fondée en 2004 pour prévenir de manière proactive les attaques sur les applications web. Il s'agit du premier effort de normalisation des pratiques de développement sécurisé.

En 2001, l'OWASP n'était pas une organisation officielle, mais plutôt un collectif qui préconisait des **pratiques de développement sécurisé**.

Ce collectif a pris de l'ampleur et est devenu l’**OWASP foundation** en 2004, avec une norme éthique pour maintenir une neutralité et l'absence de pressions commerciales.

L'OWASP n'est réglementée par aucune entreprise. Elle propose un référentiel neutre permettant d’accompagner les entreprises dans le processus de sécurisation ou d’audit de sécurité.

Top Ten OWASP

1) L’injection SQL

Cette vulnérabilité permet à un attaquant d’injecter des données non maîtrisées qui seront exécutées par l’application et qui permettent d’effectuer des actions qui ne sont normalement pas autorisées.
Ce type d’attaque s’effectue généralement grâce aux champs présents dans les formulaires.
Dans le cas d’une attaque par injection SQL, au lieu de mettre un nom d'utilisateur et un mot de passe sur une page de connexion, un utilisateur malveillant entrera des données directement interprétées par le moteur SQL, ce qui lui permettra de modifier le comportement de votre application.

Sécuriser son application contre les injections SQL.

- **Utiliser un pare feu d'application web ou WAF (Web Application Firewall)**

Ce pare-feu se place entre l'utilisateur et l'application web et permet de vérifier et d'intercepter les données envoyées. Toutefois il est également possible de sécuriser l'application directement dans le code.

- **Validez les entrées**

Cela consiste à limiter ce que l'utilisateur peut mettre dans la zone de texte.Cela n'empêchera pas l'injection, mais c'est une mesure que vous pouvez mettre en place pour limiter des attaques de base. En effet, les caractères spéciaux spécifiques à certains langages ne pourront pas être utilisés.

- **Préparez les requêtes SQL**

Ce sont des requêtes dans lesquelles les paramètres sont interprétés indépendamment de la requête elle-même. De cette manière, il est impossible d'effectuer des injections.

2) Le piratage de session

Le piratage de session (ou Broken Authentication) est une attaque qui cible un client, où l'assaillant tente de voler des informations en utilisant l'accès du client à une application Web.

Se protéger des authentifications non autorisées

Des recommandations simples peuvent être appliquées :

- exigez de vos utilisateurs qu'ils aient un mot de passe fort, c'est-à-dire contenant des majuscules, des minuscules, des chiffres et des caractères spéciaux. Il sera ainsi plus difficile pour un utilisateur malveillant de trouver le mot de passe ;
- il est également recommandé d'exiger des utilisateurs qu'ils changent régulièrement leur mot de passe en cas d'attaque de credential stuffing ;
- mettez en place le verrouillage de compte lorsqu'un utilisateur essaie de se connecter un trop grand nombre de fois sans y parvenir. Cela permet d'empêcher les attaques de force brute ;
- changez ou désactivez les comptes par défaut ;
- implémentez une authentification forte, c'est-à-dire avec plusieurs facteurs d'authentification, comme la validation par SMS ou par mail, par exemple.
- Protégez les identifiants de session au moyen d'algorithmes de chiffrement.
- Limitez la durée des sessions.

3) L'exposition des données sensibles

Lorsque vous surfez sur Internet, votre navigateur utilise le protocole HTTP (Hypertext Transfer Protocol) pour afficher les pages web, et le protocole **Transmission Control Protocol/Internet Protocol (TCP/IP)** pour les transmettre.

Si le serveur web établit la **connexion TCP** avec le navigateur, une réponse avec le code status et le fichier demandé (généralement le fichier index.html pour la page web) sera transmise. Mais dans notre cas, les données transitent en HTTP et pas en HTTPS…

Les données transitant en HTTP peuvent être interceptées, car elles transitent en clair.

Comment éviter d’exposer les données sensibles en transit ?

- Utilisez le HTTPS pour l'ensemble de votre site, même s'il ne contient pas de données sensibles.
- Utilisez les requêtes GET pour récupérer les informations et POST pour modifier les informations.
- Sécurisez vos cookies pour qu'ils soient transmis par l'en-tête et via HTTPS.
- Sécurisez vos sessions en ajoutant une date d'expiration, en sécurisant l'ID et en ne mettant pas cet ID dans l'URL.

Les données sensibles ne sont pas seulement en transit, elles sont aussi stockées en base de données. Pour protéger certaines données stockées sur une application, il est possible d’utiliser des algorithmes de hachage.

L'intérêt des algorithmes de hachage est qu’ils permettent de calculer une empreinte (ou *hash*) d’une chaîne de caractères, par exemple. Cette empreinte est utile pour éviter de stocker en clair le mot de passe dans la base de données.

Comment éviter d’exposer les données stockées ?

- Sécurisez votre base de données avec le chiffrement.
- Utilisez des algorithmes de hachage sécurisés tels que Argon5, Scrypt, Bcrypt et PBKDF2.
- Le masquage des données peut être utilisé pour sécuriser les données sensibles d’une base de données.

4) Les attaques contre le contrôle d’accès

Les attaques courantes contre le contrôle d'accès se produisent lorsqu'une URL permet de contourner l'authentification. Les pirates informatiques utilisent la connaissance des formats et des modèles pour deviner l'URL des pages privilégiées qui n'ont pas été configurées

de manière sécurisée. Pour se protéger contre ce type d'attaque, il est possible de mettre en place une restriction URL.

Comment éviter ces attaques ?

- Les applications web avec authentification ne garantissent pas que toutes les pages sont verrouillées par contrôle d’accès.
- Les références directes aux objets peuvent amener un attaquant à comprendre les modèles et la configuration des applications web.
- N'utilisez pas de noms prévisibles ou de références directes à la base de données dans l'URL.
- Utilisez des références d'objet indirectes avec des paramètres et des combinaisons clé-valeur.
- Personnalisez vos codes d'erreur pour qu'ils ne révèlent pas les attributs de la base de données.

5) Les failles XSS

Avec une attaque XSS, un attaquant va essayer de prendre le contrôle de votre navigateur en injectant un **script JavaScript** dans l'application web. Il pourra l’injecter directement dans un formulaire, mais il peut également l’injecter dans l'URL, l'en-tête HTTP ou d'autres parties du framework utilisé.

Contrairement aux injections SQL, il ne s'agit pas de requêtes et de commandes SQL sur une base de données. **Une faille XSS s’exécute dans le code de l'application web.**

Les bonnes pratiques pour s’en prémunir.

- **Appliquez la validation des données d'entrée** : pour empêcher les attaques communes, il est possible de blacklister certains caractères comme les balises script.

- **Appliquez la transformation des entrées** : vous pouvez encoder toutes vos entrées dans une entité de caractères HTML ou du texte pour qu'il n'exécute aucun script. Il existe des fonctions simples et des bibliothèques qui peuvent vous aider à encoder tout votre HTML et JavaScript.
- Configurez vos cookies avec le flag **HttpOnly**.

6) Les failles XXE

Pour comprendre ce que sont les XXE, il faut d'abord définir le langage XML. Le **langage XML (*Extensible Markup Language*)** a été créé pour **stocker, partager** et **transporter des données entre systèmes**. Il est indépendant de la plateforme et du langage.

XML est utilisé pour transmettre des données d'une plateforme à l'autre. Beaucoup d'applications web utilisent XML pour gérer les données. Il ressemble à HTML, mais fonctionne différemment car HTML est utilisé pour présenter les données en hypertexte sur un navigateur.

Une attaque XXE ou *injection XXE* est un type d'attaque contre une application qui utilise XML. Cette attaque se produit lorsque l'entrée XML contenant une référence à une entité externe est traitée par un analyseur XML mal configuré. Cette attaque peut entraîner la divulgation de données confidentielles, un déni de service, une falsification de requête côté serveur et d'autres impacts sur le système.

Se protéger contre les injection XXE

Il existe un moyen de désactiver les entités externes dans tous les langages. Il s'agit généralement d'une balise binaire vrai/faux.

Par exemple, dans un analyseur XML PHP, le code ressemblerait à ceci :

libxml_disable_entity_loader (true) ;

7) La désérialisation non sécurisée

La désérialisation non sécurisée est une vulnérabilité qui se produit lorsque des données non fiables sont utilisées pour abuser de la logique d'une application, causer un déni de service (DoS, Deny of Service), ou même exécuter du code arbitraire.

Comment s'en protéger ?

La désérialisation non sécurisée peut être évitée en créant des contrôles sur l'état du code.

8) Utiliser des outils dont les vulnérabilités sont connues

Il est important de faire des recherches sur les outils de votre environnement de développement pour être au courant de toutes les vulnérabilités et des méthodes d'atténuation. Documentez tous les modules, API et bibliothèques utilisés, ainsi que les vulnérabilités connues afin de garder une trace des actions effectuées.

La bonne pratique à avoir dans ce cas là est de Pensez à consulter les bases de données sur le web. La base de données **Common Vulnerabilities and Exposures** (en anglais), créée par le MITRE, est un excellent outil à utiliser.

9) Une mauvaise configuration de sécurité.

Il est très important d'avoir une bonne configuration de sécurité pour l'environnement qui héberge votre application web afin d'assurer sa disponibilité et son intégrité. Par ailleurs, il faudra également mettre en place des couches de sécurité telles qu'un pare-feu (WAF). Ce pare-feu devra être configuré de manière optimale pour s'assurer que seuls les ports requis sont ouverts afin de renforcer la sécurité. Un outil comme

un SIEM (*Security information management system)* peut détecter les événements et assurer la gestion de la sécurité du serveur.

En dehors du serveur d'application web et du pare-feu, se trouve le **serveur physique** lui-même. En règle générale, les services cloud qui fournissent cet espace ont déjà mis en place des mesures de sécurité. Assurez-vous que les **normes de conformité pour le service cloud** utilisées sont conformes aux **exigences de la politique de sécurité** de votre application web. La configuration de sécurité du serveur physique et du pare-feu pour la plateforme de cloud nécessitent une configuration stricte.

Le contrôle d'accès pour les composants externes de l'environnement, tels que la plateforme et l'administration du serveur, sont d'autres moyens par lesquels un attaquant peut obtenir un contrôle non autorisé à une application web. Ces facteurs doivent être correctement configurés et documentés.

10) La surveillance du trafic web

Le trafic web devrait également être surveillé pour déceler les anomalies. Imaginez si quelqu'un commençait à effectuer des scans de ports pour rechercher périodiquement les ports ouverts. Il y a des outils qui peuvent détecter de tels événements pour avertir les administrateurs qu'il y a une tentative possible de piratage du site.
Tous les aspects **administratifs** tels que les répertoires et les fichiers de configuration pour toutes les applications doivent être protégés des regards indiscrets. Protégez le code source et la structure des fichiers en vous assurant que le serveur web est verrouillé.

Toutes vos applications sont hébergées sur des systèmes d'exploitation. Assurez-vous que les applications ont un accès **minimal** au système d'exploitation, car le piratage du système d'exploitation est un moyen sûr de prendre le contrôle de votre application.

Vous disposez également de jetons d'accès, de mots de passe d'authentification et de clés de chiffrement ! Protégez-les également.

Enfin, les **exceptions** et les **codes d'erreur** en disent long sur votre application, votre code source, le fonctionnement du réseau, la structure des fichiers de votre base de données, les répertoires de fichiers et plus encore. Les attaquants cherchent ce type d'information. Une stack trace peut être utile lorsque vous déboguez votre application ; cependant, si elle est utilisée à des fins malveillantes, la quantité d'informations qu'elle peut révéler peut être dangereuse !

Quelques sources pour rester informé.

oswap.org

livehacking.com

cert-fr