OWASP

Qu'est-ce que l'OWASP ?
Le projet de sécurité des applications web ouvertes tient une liste régulièrement mise à jour des préoccupations les plus urgentes en matière de sécurité des applications web.

Définition de l'OWASP ?
L'Open Web Application Security Project, ou OWASP, est une organisation internationale à but non lucratif qui se consacre à la sécurité des applications web. L'un des principes fondamentaux de l'OWASP est que tous ses documents soient disponibles gratuitement et facilement accessibles sur son site web, ce qui permet à chacun d'améliorer la sécurité de ses propres applications web. Le matériel qu'ils proposent comprend de la documentation, des outils, des vidéos et des forums. Leur projet le plus connu est peut-être le Top 10 de l'OWASP.

Top 10 des l'OWASP ?
Le Top 10 de l'OWASP est un rapport régulièrement mis à jour qui expose les préoccupations en matière de sécurité des applications web, en se concentrant sur les 10 risques les plus critiques. Le rapport est élaboré par une équipe d'experts en sécurité du monde entier. L'OWASP qualifie le Top 10 de « document de sensibilisation » et recommande à toutes les entreprises d'intégrer le rapport dans leurs processus afin de minimiser et/ou d'atténuer les risques de sécurité.

Vous trouverez ci-dessous les risques de sécurité signalés dans le rapport OWASP Top 10 2017 :

1. Attaques par Injection SQL
Les attaques par injection SQL se produisent lorsque des données non fiables sont envoyées à un interpréteur de code par le biais d'une saisie de formulaire ou d'une autre soumission de données à une application web. Par exemple, un attaquant pourrait entrer du code de base de données SQL dans un formulaire qui attend un nom d'utilisateur en clair. Si la saisie de ce formulaire n'est pas correctement sécurisée, le code SQL sera exécuté. C'est ce qu'on appelle une attaque par injection SQL.
Les attaques par injection peuvent être évitées en validant et/ou en assainissant les données soumises par les utilisateurs. (La validation

signifie le rejet des données suspectes, tandis que l'assainissement consiste à nettoyer les parties suspectes des données). En outre, un administrateur de base de données peut définir des contrôles pour minimiser la quantité d'informations qu'une attaque par injection peut exposer.

2. Authentification frauduleuse

Les vulnérabilités des systèmes d'authentification (login) peuvent donner aux attaquants l'accès à des comptes d'utilisateurs et même la possibilité de compromettre un système entier en utilisant un compte d'administrateur. Par exemple, un attaquant peut prendre une liste contenant des milliers de combinaisons connues de noms d'utilisateur et de mots de passe obtenues lors d'une violation de données et utiliser un script pour essayer toutes ces combinaisons sur un système de connexion afin de voir si certaines fonctionnent.

Certaines stratégies visant à atténuer les vulnérabilités de l'authentification consistent à exiger une authentification à deux facteurs (2FA) ainsi qu'à limiter ou à retarder les tentatives de connexion répétées en utilisant la limitation du débit.

3. Exposition aux données sensibles

Si les applications web ne protègent pas les données sensibles telles que les informations financières et les mots de passe, les pirates peuvent accéder à ces données et les vendre à des fins malveillantes. Une méthode populaire pour voler des informations sensibles consiste à utiliser une attaque de type « man-in-the-middle».

Le risque d'exposition aux données peut être minimisé en cryptant toutes les données sensibles et en désactivant la mise en cache* de toute information sensible. En outre, les développeurs d'applications web doivent veiller à ne pas stocker inutilement des données sensibles.

*La mise en cache est la pratique qui consiste à stocker temporairement des données en vue de leur réutilisation. Par exemple, les navigateurs web mettent souvent en cache des pages web de sorte que si un utilisateur consulte à nouveau ces pages dans un délai déterminé, le navigateur n'a pas besoin de les récupérer sur le web.

4. Entités externes XML (XEE)

Il s'agit d'une attaque contre une application web qui analyse les entrées XML*. Cette entrée peut faire référence à une entité externe, en essayant d'exploiter une vulnérabilité dans l'analyseur. Une « entité externe » dans

ce contexte fait référence à une unité de stockage, comme un disque dur. Un analyseur XML peut être trompé en envoyant des données à une entité externe non autorisée, qui peut transmettre des données sensibles directement à un attaquant.
Le meilleur moyen de prévenir les attaques XEE est de faire en sorte que les applications web acceptent un type de données moins complexe, comme JSON**, ou tout au moins de corriger les analyseurs XML et de désactiver l'utilisation d'entités externes dans une application XML.
*XML ou Extensible Markup Language est un langage de balisage destiné à être à la fois lisible par l'homme et par la machine. En raison de sa complexité et de ses vulnérabilités en matière de sécurité, il est en train d'être progressivement abandonné dans de nombreuses applications web.
**La notation d'objet JavaScript (JSON) est un type de notation simple, lisible par l'homme, souvent utilisée pour transmettre des données sur Internet. Bien qu'il ait été créé à l'origine pour JavaScript, JSON est un langage agnostique et peut être interprété par de nombreux langages de programmation différents.

5. Contrôle d'accès interrompu
Le contrôle d'accès désigne un système qui contrôle l'accès à des informations ou à des fonctionnalités. Les contrôles d'accès brisés permettent aux attaquants de contourner l'autorisation et d'effectuer des tâches comme s'ils étaient des utilisateurs privilégiés tels que les administrateurs. Par exemple, une application web pourrait permettre à un utilisateur de changer le compte auquel il est connecté en changeant simplement une partie d'une url, sans autre vérification.
Les contrôles d'accès peuvent être sécurisés en s'assurant qu'une application web utilise des jetons d'autorisation* et les soumet à des contrôles stricts. De nombreux services émettent des jetons d'autorisation lorsque les utilisateurs se connectent. Toute demande privilégiée faite par un utilisateur nécessitera la présence du jeton d'autorisation. C'est un moyen sûr de s'assurer que l'utilisateur est bien celui qu'il prétend être, sans avoir à entrer constamment ses identifiants de connexion.

6. Mauvaise configuration de la sécurité
La mauvaise configuration de la sécurité est la vulnérabilité la plus courante de la liste, et est souvent le résultat de l'utilisation de configurations par défaut ou de l'affichage d'erreurs excessivement verbeuses. Par exemple, une application peut présenter à l'utilisateur des

erreurs trop descriptives qui peuvent révéler des vulnérabilités dans l'application. Il est possible d'atténuer ce problème en supprimant toutes les fonctionnalités inutilisées dans le code et en veillant à ce que les messages d'erreur soient plus généraux.

7. Scénario de site croisé

Les vulnérabilités des scripts intersites se produisent lorsque des applications web permettent aux utilisateurs d'ajouter un code personnalisé dans un chemin d'accès ou sur un site web qui sera vu par les autres utilisateurs. Cette vulnérabilité peut être exploitée pour exécuter un code JavaScript malveillant sur le navigateur d'une victime, par exemple, un attaquant pourrait envoyer un courriel à une victime qui semble provenir d'une banque de confiance, avec un lien vers le site web de cette banque. Ce lien pourrait contenir un code JavaScript malveillant marqué à la fin de l'url. Si le site de la banque n'est pas correctement protégé contre le cross-site scripting, alors ce code malveillant sera exécuté dans le navigateur web de la victime lorsqu'elle cliquera sur le lien.

Les stratégies d'atténuation pour les scripts intersites comprennent l'évitement des requêtes HTTP non fiables ainsi que la validation et/ou l'assainissement du contenu généré par les utilisateurs. L'utilisation de cadres de développement web modernes comme ReactJS et Ruby on Rails offre également une protection intégrée des scripts intersites.

8. Désérialisation incertaine

Cette menace vise les nombreuses applications web qui sérialisent et désérialisent fréquemment les données. La sérialisation consiste à prendre des objets dans le code de l'application et à les convertir dans un format qui peut être utilisé à d'autres fins, comme le stockage des données sur disque ou la diffusion en continu. La sérialisation est un peu comme emballer des meubles dans des boîtes avant un déménagement, et la désérialisation est comme déballer les boîtes et assembler les meubles après le déménagement. Une attaque de désérialisation non sécurisée revient à demander aux déménageurs de modifier le contenu des boîtes avant de les déballer.

Un exploit de désérialisation non sécurisé est le résultat de la désérialisation de données provenant de sources non fiables, et peut avoir des conséquences graves comme des attaques DDoS et des attaques d'exécution de code à distance. Bien que des mesures puissent être prises pour essayer d'attraper les attaquants, comme la surveillance de la

désérialisation et la mise en œuvre de contrôles de type, la seule façon sûre de se protéger contre les attaques de désérialisation non sécurisées est d'interdire la désérialisation de données provenant de sources non fiables.

9. Utilisation de composants présentant des vulnérabilités connues

De nombreux développeurs web modernes utilisent des composants tels que des bibliothèques et des frameworks dans leurs applications web. Ces composants sont des logiciels qui aident les développeurs à éviter le travail redondant et à fournir les fonctionnalités nécessaires ; les exemples les plus courants sont les cadres frontaux comme React et les petites bibliothèques qui permettent d'ajouter des icônes de partage ou des tests a/b. Certains attaquants recherchent des vulnérabilités dans ces composants qu'ils peuvent ensuite utiliser pour orchestrer des attaques. Certains des composants les plus populaires sont utilisés sur des centaines de milliers de sites web ; un attaquant trouvant une faille de sécurité dans l'un de ces composants pourrait laisser des centaines de milliers de sites vulnérables à exploiter.

Les développeurs de composants proposent souvent des correctifs de sécurité et des mises à jour pour remédier aux vulnérabilités connues, mais les développeurs d'applications web ne disposent pas toujours des versions les plus récentes ou des correctifs des composants qui s'exécutent sur leurs applications. Pour réduire au minimum le risque d'exécuter des composants présentant des vulnérabilités connues, les développeurs doivent supprimer les composants inutilisés de leurs projets, ainsi que s'assurer qu'ils reçoivent des composants d'une source fiable et qu'ils sont à jour.

10. Insuffisance de l'enregistrement et de la surveillance

De nombreuses applications web ne prennent pas suffisamment de mesures pour détecter les violations de données. Le délai moyen de découverte d'une brèche est d'environ 200 jours après qu'elle se soit produite. Cela donne aux attaquants beaucoup de temps pour causer des dommages avant qu'il n'y ait une réponse. L'OWASP recommande aux développeurs web de mettre en place un système de journalisation et de surveillance ainsi que des plans de réponse aux incidents afin de s'assurer qu'ils sont informés des attaques dont leurs applications font l'objet.