* + 1. MENACES ET RISQUES

1. TYPES D’ATTAQUES (MENACES)

Le WASC établit dans son rapport « WASC Threat Classification » une liste exhaustive des menaces qui pèsent sur la sécurité des applications Web. Elles sont regroupées en six catégories définies dans la version 2004 de ce rapport.

1. La catégorie « ***authentification*** » regroupe les attaques de sites Web dont la cible est le système de validation de l'identité d'un utilisateur, d'un service ou d'une application.
2. La catégorie « ***autorisation*** » couvre l'ensemble des attaques de sites Web dont la cible est le système de vérification des droits d'un utilisateur, d'un service ou d'une application pour effectuer une action dans l'application.
3. La catégorie « ***attaques côté client*** » rassemble les attaques visant l'utilisateur pendant qu'il utilise l'application.
4. La catégorie « ***exécution de commandes*** » englobe toutes les attaques qui permettent d'exécuter des commandes sur un des composants de l'architecture du site Web.
5. La catégorie « ***révélation d'informations*** » définit l'ensemble des attaques permettant de découvrir des informations ou des fonctionnalités cachées.
6. La catégorie « ***attaques logiques*** » caractérise les attaques qui utilisent les processus applicatifs (système de changement de mot de passe, système de création de compte…) à des fins hostiles.
7. RISQUES DE SÉCURITÉ

Contrairement au WASC qui décrit toutes les attaques possibles sur une application Web, l'OWASP ne traite que les dix plus grands risques de sécurité. Le rapport « OWASP Top 10 » permet ainsi à l'équipe projet de se focaliser sur la protection de l'application Web face aux menaces les plus importantes, ce qui est moins coûteux et plus facilement réalisable que d'essayer de se protéger de tous les dangers. L'OWASP établit le classement 2010 ci-dessous, dont chacune des failles est développée dans les chapitres suivants.

* Une faille d**'injection** se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent duper l'interpréteur afin de l'amener à exécuter des commandes inhabituelles ou à accéder à des données non autorisées.
* Les failles de **Cross-Site Scripting (XSS)** se produisent chaque fois qu'une application prend des données non fiables et les envoie à un navigateur Web sans validation. XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites Web, ou rediriger l'utilisateur vers des sites malveillants.
* Les failles de **violation de gestion d'authentification et de session** se produisent quand les fonctions correspondantes ne sont pas mises en œuvre correctement, permettant aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs.
* Une faille de **référence directe à un objet** se produit quand un développeur expose une référence à une variable interne, tels un nom de fichier, de dossier, un enregistrement de base de données ou une clé de base de données. Sans un contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées.
* Une attaque par **falsification de requête intersites (CSRF)** force le navigateur d'une victime authentifiée à envoyer une requête HTTP, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement incluse, à une application Web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes, l'application vulnérable considérant alors qu'elles émanent légitimement de la victime.
* Une faille due à une **mauvaise configuration de sécurité** se produit quand les serveurs d'application, serveurs Web, serveur de base de données, et la plate-forme n'ont pas de configuration sécurisée correctement établie et déployée. Tous ces paramètres doivent être définis, mis en œuvre, et maintenus. Ceci implique de maintenir tous les logiciels à jour, notamment toutes les bibliothèques de code employées par l'application.
* Une faille de **stockage de données non sécurisées** se produit quand une application Web ne protège pas correctement les données sensibles, telles que les numéros de cartes de crédit, de sécurité sociale, les informations d'authentification, avec un algorithme de chiffrement ou de hash approprié. Les pirates peuvent voler ou modifier ces données faiblement protégées pour perpétrer un vol d'identité et d'autres crimes, tels que la fraude à la carte de crédit.
* La défaillance dans la **restriction des accès à une URL** se produit quand une application Web ne protège pas l'accès aux URL. Les applications doivent effectuer des contrôles d'accès similaires chaque fois que ces pages sont accédées, sinon les attaquants seront en mesure de forger des URL pour accéder à ces pages cachées.
* La faille de **protection de la couche transport** se produit quand les applications ne peuvent pas chiffrer et protéger la confidentialité et l'intégrité du trafic réseau sensible. De plus, quand elles le font, elles supportent parfois des algorithmes faibles, utilisent des certificats expirés ou invalides, ou ne les emploient pas correctement.
* Une faille de **redirection et renvoi non validés** se produit quand une application Web réoriente les utilisateurs vers d'autres pages et sites Web, et utilise des données non fiables pour déterminer les pages de destination. Sans validation appropriée, les attaquants peuvent rediriger les victimes vers des sites de phishing ou de logiciel malveillant, ou utiliser les renvois pour accéder à des pages non autorisées.

1. **FAILLES ET PARADES**
2. INJECTION

**Principe**

Pour réaliser une attaque de ce type, il faut injecter dans les zones de saisie classiques présentées à l'utilisateur du code malicieux. Ces données seront interprétées comme des instructions par un des composants de l'application Web. Les champs de formulaires peuvent être protégés par JavaScript pour vérifier que les valeurs saisies correspondent à ce qui est attendu.

* L’attaque par injection SQL par exemple consiste à injecter du code SQL qui sera interprété par le moteur de base de données. Le code malicieux le plus répandu est d'ajouter une instruction pour faire en sorte que la requête sous-jacente soit toujours positive. Cela permet par exemple d'usurper une identité pour se connecter à une application Web, de rendre l'application inutilisable ou de supprimer toutes les données de la table visée, voire de la base de données complète.
* L'attaque par injection de XPath suit le même principe que pour SQL. En effet, XPath est un langage de requête pour gérer les données stockées au format XML, comme le fait SQL pour les bases de données relationnelles. XPath et Xquery, dont XPath est un sous-ensemble, souffrent donc des mêmes vulnérabilités face à l'injection de code malicieux.
* L'attaque par injection LDAP permet d'accéder à des informations privées qui sont enregistrées dans l'annuaire d'entreprise. En modifiant le comportement du filtrage dans la requête LDAP qui sera générée, il est possible de récupérer la liste exhaustive des adresses de courrier électronique d'une entreprise pour les saturer de spam par exemple.
* L'attaque par injection de commandes est surtout principalement possible sur les scripts CGI écrits en Perl, PHP et Shell. Il est possible de prendre le contrôle du serveur. Il faut pour cela faire en sorte que la commande initiale soit exécutée sans problème et ajouter des commandes du système d'exploitation du serveur qui seront exécutées par le serveur.
* L'attaque par traversée de répertoire permet d'accéder à des fichiers présents sur le serveur. Les fichiers cibles privilégiés étant ceux contenant des informations de sécurité comme le fichier des mots de passe ou les fichiers contenant les clés privées de chiffrement pour SSL par exemple. Cette attaque est rendue possible si l'application Web inclut du contenu de fichier en passant l'adresse de ce fichier en paramètres de la requête.
* Les attaques XXE (XML eXternal Entity) sont un dérivé des attaques par traversée de répertoire. Les conséquences vis-à-vis des fichiers présents sur les serveurs sont donc les mêmes. Ce type d'attaque est basé sur la fonctionnalité de XML « entités externes ». Les entités sont des substituts pour des séquences d'information. Elles sont équivalentes aux variables dans les langages de programmation. Les entités externes permettent de déclarer des documents dont le contenu sera affiché lors de l'utilisation de l'entité. Si l'entité pointe sur un fichier existant sur le serveur, son contenu pourra être divulgué à l'attaquant. Cette fonctionnalité peut être exploitée en plaçant un fichier XML au format RSS sur un site et de l'intégrer à un agrégateur en ligne. Si ce dernier est vulnérable, il sera alors possible de voir le contenu des fichiers demandés par l'attaquant.

**Parade:**

Les différentes attaques citées précédemment reposent principalement sur l'utilisation de caractères spécifiques qui permettent de mettre en commentaire des portions de code et d'insérer du code frauduleux. Il est cependant rare que l'application ait besoin d'accepter les caractères suivants :



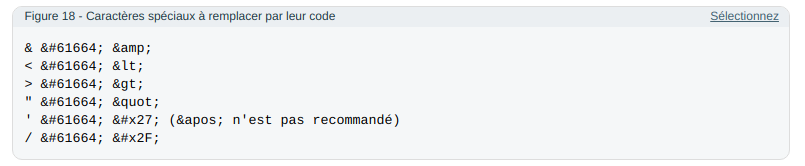
1. CROSS-SITE SCRIPTING (XSS)

Il en existe deux (02) types:

1. L'attaque ***XSS par réflexion (reflected XSS)*** s'appuie sur le fait que l'application Web affiche ce que l'utilisateur vient de saisir dans un formulaire dans une page de résultat. Le navigateur de la victime exécute alors le code frauduleux généré dans la page de résultat. Tous les champs de formulaire sont donc une faille de sécurité potentielle que l'attaquant peut exploiter par XSS. L'attaquant crée un lien déguisé vers l'application Web dont un des paramètres contient du code JavaScript frauduleux. En utilisant ce lien, la victime fait exécuter par son navigateur le code JavaScript.
2. L'attaque ***XSS stockée (stored XSS)*** s'appuie sur le fait que l'attaquant réussisse à stocker dans la base de données du code frauduleux qui sera exécuté par la victime lorsqu'elle tentera d'afficher la donnée malveillante. Cette attaque est plus dangereuse que la première car le code fait partie intégrante des données de l'application Web et peut atteindre plusieurs victimes.

**Parade**:

Les recommandations faites précédemment pour se prémunir des risques d'injection sont valables pour XSS. Cependant, transformer les six caractères douteux suivants suffit:



1. VIOLATION DE GESTION D’AUTHENTIFICATION ET DE SESSION

**Principe**:

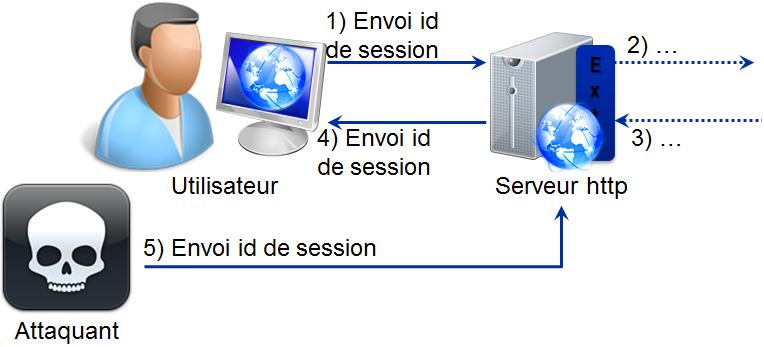
Cette faille de sécurité regroupe toutes les vulnérabilités pouvant mener à une usurpation d'identité. Ces points de faiblesse dans les applications Web peuvent ouvrir à des attaquants des accès à des fonctionnalités des applications Web auxquelles ils n'ont pas le droit normalement. Cela peut donc leur permettre de voler des informations ou d'endommager le bon fonctionnement de l'application. La protection des accès à l'application repose généralement sur un système d'authentification. La plupart du temps, le système d'authentification est redéveloppé pour chaque application, ce qui implique que ces systèmes ne bénéficient pas de l'expérience acquise sur le développement d'autres applications.

Pour comprendre comment les attaques peuvent être menées, il faut comprendre le mécanisme d'authentification le plus commun des applications Web :

1. L'utilisateur non authentifié demande l'accès à une page Web ;
2. Le serveur renvoie une page d'authentification ;
3. L'utilisateur remplit le formulaire en fournissant un identifiant et un mot de passe et revoie ces informations au serveur Web ;
4. Le serveur Web fait appel à un service pour vérifier la validité du couple identifiant/mot de passe ;
5. Si la validité est avérée, le serveur Web fournit un identifiant de session à l'utilisateur. Comme expliqué précédemment HTTP est un protocole déconnecté, donc entre deux requêtes HTTP la connexion entre le navigateur et le serveur HTTP est coupée. Donc le serveur HTTP ne peut pas reconnaître un utilisateur qui s'est déjà authentifié et qui a ouvert une session de travail dans l'application Web. Pour remédier à cela, la plupart des systèmes d'authentification reposent sur un identifiant de session. Celui-ci est envoyé à chaque page entre le client et le serveur par le biais d'un cookie, d'un paramètre d'adresse ou d'un champ de formulaire invisible pour l'utilisateur ;
6. L'utilisateur peut utiliser l'application Web tant que la session est ouverte.

Les attaques pour usurper une identité peuvent être regroupées en deux catégories :

* les attaques contre le système d'authentification qui cherchent à obtenir un droit d'accès ;
* les usurpations de session qui permettent de s'affranchir de l'étape d'authentification.

[](https://web.developpez.com/tutoriels/web/failles-securite-application-web/images/failles_de_securite_v1-3_Page_22_Image_0001.jpg)

**Exemples**:

Parmi les attaques contre les systèmes d'authentification, la plus répandue est l'utilisation de la force brute. Pour cela l'attaquant va bombarder la page d'authentification avec des valeurs d'identifiant et de mots de passe jusqu'à ce qu'il se fasse accepter. L'attaque est facilitée si le message d'erreur de l'échec de l'authentification donne l'origine de l'erreur. Ainsi « l'utilisateur n'existe pas » permet à l'attaquant de ne pas tenter d'entrer des mots de passe pour cet utilisateur absent de la base de compte. « Mot de passe incorrect » permet à l'attaquant de se concentrer sur cet utilisateur, ce qui lui fait gagner beaucoup de temps. De même si l'application Web offre un service pour créer un compte par lui-même et qu'au moment de la saisie de l'identifiant ce système indique si le compte existe déjà ou non, l'attaquant dispose d'un moyen pour trouver des comptes attaquables. L'impact de ce type d'attaque n'est pas seulement limité à une usurpation d'identité pour l'application Web. Un internaute utilise souvent les mêmes valeurs d'identifiant et de mot de passe pour de nombreuses applications présentes sur le Web. L'attaquant peut donc tenter d'utiliser ces valeurs sur différentes applications Web.

Il est possible pour l'attaquant de chercher des couples identifiant/mot de passe sans faire appel à la force brute. L'attaquant va simplement tenter d'utiliser des comptes généralement présents dans les applications Web, comme ceux d'administration. Ceci est surtout possible lorsque les applications sont basées sur des outils Open Source qui ont des comptes créés automatiquement avec des mots de passe par défaut connus du domaine public. L'attaquant n'a plus qu'à consulter le code source pour trouver une liste restreinte d'identifiants/mots de passe valides.

Les pages Web qui permettent de réinitialiser les mots de passe sont une faille importante pour l'usurpation d'identité. En effet, pour s'assurer de l'identité du demandeur, la plupart d'entre elles demandent une information que seule la personne est censée connaître. Or avec les réseaux sociaux, les internautes partagent leur vie privée. Ces informations personnelles visibles de tous peuvent être les mêmes que celles demandées dans les pages de réinitialisation de mot de passe. Dans ce cas l'attaquant peut définir un nouveau mot de passe qu'il pourra utiliser pour se connecter à l'application.

Pour voler un identifiant de session, l'attaque par la force brute est également possible. Dans ce cas l'attaquant va générer des valeurs et tenter de les utiliser comme identifiant de session. S'il réussit à trouver une valeur valide, il pourra utiliser l'application Web sans s'être authentifié. Par ailleurs, une attaque XSS peut permettre de récupérer un identifiant de session présent dans le cookie de l'internaute. L'identifiant de session peut également être découvert par un attaquant en écoutant la transmission de données sur le réseau, en consultant les fichiers de journalisation sur le serveur par injection de commandes systèmes par exemple.

Une autre technique consiste à fournir un identifiant de session à la victime, par hameçonnage par exemple. L'utilisateur se connecte à l'application Web et s'authentifie. La session est créée en utilisant l'identifiant fourni. L'attaquant peut alors accéder au site en fournissant l'identifiant fixé. L'identifiant peut être généré ou obtenu en émettant une requête vers l'application cible de l'attaque si l'application retourne un identifiant de session pour toute requête avant même l'authentification de l'utilisateur.

Si l'identifiant de session n'est pas généré aléatoirement mais est un nombre incrémenté à chaque ouverture de session, par exemple, l'attaquant peut arriver à deviner un identifiant valide.

**Parade**:

Concernant les systèmes d'authentification, l'application ne doit accepter que des mots de passe suffisamment forts pour éviter d'être devinés rapidement par la force brute. Une longueur minimale de huit caractères est ce qui recommandé par l'OWASP pour les applications critiques. De plus il doit comporter au moins un chiffre, une lettre en minuscule et une lettre en majuscule.

Le message affiché lors d'un problème de validité de l'identifiant ou du mot de passe doit être générique et ne doit pas donner d'indice quant à l'origine de l'erreur.

Pour contrer les attaques de force brute, le compte ciblé par l'attaque doit être verrouillé après cinq tentatives consécutives infructueuses de connexion. La procédure de déverrouillage peut être automatique après un laps de temps prédéfini ou manuelle par un administrateur de l'application.

Dans la mesure du possible il est conseillé de ne pas développer son propre mécanisme d'authentification. Il est préférable d'utiliser un système existant éprouvé.

Concernant les identifiants de session, les applications Web doivent limiter la durée de vie d'une session. Une période d'inactivité maximale doit être définie. Si l'utilisateur n'utilise pas l'application pendant ce laps de temps, la session devient inutilisable et l'utilisateur doit se reconnecter. Une session doit également avoir une durée de vie maximale au-delà de laquelle la session expire, même si la période d'inactivité autorisée n'était pas dépassée. Ces précautions permettent de limiter le temps d'action d'un attaquant.

Du côté client, du code JavaScript doit fermer la session lorsque l'utilisateur ferme le navigateur. Cela permet de simuler une action de l'utilisateur pour se déconnecter de l'application. Pour que l'utilisateur évite de perdre des saisies non sauvegardées, du code JavaScript peut prévenir l'utilisateur que sa session va bientôt expirer.

L'identifiant de session doit être généré automatiquement et être suffisamment long pour se prémunir des vols par prédiction.

Pour détecter des attaques de force brute, il faut régulièrement consulter les journaux d'activité à la recherche d'événements inhabituels, comme un nombre important de requêtes utilisant des identifiants de sessions invalides.

1. RÉFÉRENCE DIRECTE NON SÉCURISÉE À UN OBJET
2. FALSIFICATION DES REQUÊTES INTERSITES (CSRF)
3. MAUVAISE CONFIGURATION DE SÉCURITÉ
4. STOCKAGE DE DONNÉES NON SÉCURISÉES
5. PROTECTION INSUFFISANTE DE LA COUCHE DE TRANSPORT
6. REDIRECTIONS ET RENVOIS NON VALIDÉS