Les Dix Vulnérabilités de Sécurité Applicatives Web les Plus Critiques

Mise à jour 2004

27 Janvier 2004

Commentaires

« Avec une fréquence d'annonce de nouvelles vulnérabilités presque hebdomadaire, beaucoup d'entreprises peuvent se sentir submergées à essayer de se tenir à jour. Mais une aide existe sous la forme de listes de consensus de vulnérabilités et de défense.

« Le projet « Open Web Application Security Project » a produit une telle liste des 10 vulnérabilités de sécurité les plus critiques inhérentes aux applications web et aux bases de données, et les façons les plus efficaces de les répertorier. Les vulnérabilités applicatives sont souvent négligées, mais sont tout aussi importantes à traiter que des problèmes réseau. Si chaque société éliminait ces vulnérabilités courantes, leur travail ne serait pas fait, mais celles-ci et l'Internet, seraient significativement plus sûrs. »

- ***J. Howard Beales, III, Directeur du Bureau de la Commission Commerciale Fédérale de Défense du Consommateur, avant le Comité Associatif de Technologie de l'information de la Politique Internet de l'Amérique, vendredi 12 décembre 2003***

« Une mauvaise configuration, l'inattention et un logiciel défectueux peuvent semer le désastre sur Internet. Un des principaux secteurs de vulnérabilité se fait à travers les connexions WWW. Par conception, les services WWW sont destinés à être ouverts, et agissent habituellement comme une interface vers de précieuses ressources. En tant que tels, il est critique que ces services soient sécurisés. Mais avec des centaines de vulnérabilités potentielles il peut être excessivement intimidant de décider où commencer à appliquer des mesures défensives. Le Top 10 de l'OWASP fournit une vue consensuelle des vulnérabilités les plus significatives et probables dans les applications web personnalisées. Les entreprises devraient l'utiliser pour concentrer leurs efforts, leur donnant l'assurance d'identifier les secteurs qui auront le plus d'impact sur la sécurisation de leurs applications. »

- ***Eugene H. Spafford, Professeur de Sciences Informatiques, Purdue Université et Directeur du Centre Universitaire Purdue pour l'Éducation et la Recherche sur l'Assurance de l'information et la Sécurité (CERIAS)***

« Cette liste des « dix plus désirés » égratigne vivement le bout d'un énorme iceberg. La réalité sous-jacente est honteuse : la plupart des systèmes et logiciels applicatifs Web sont écrits sans soucis des principes de sécurité, d'ingénierie logicielle, des implications opérationnelles et à vrai dire du bon sens. »

- ***Docteur Peter G. Neumann, Scientifique Principal, Laboratoire International de Science Informatique SRI, Modérateur du forum ACM Risques, Auteur de « Risques concernant l'Ordinateur»***

« Cette liste est un développement important pour les consommateurs tout autant que pour les vendeurs. Elle éduquera les vendeurs afin éviter les mêmes erreurs qui ont été répétées d'innombrables fois dans d'autres applications web. Mais elle fournit aussi aux consommateurs une façon de demander aux vendeurs de répondre à une série d'attentes minimales sur la sécurité des applications web et, d'une façon toute aussi importante, identifier les vendeurs ne se montrant pas à la hauteur de ces attentes »

- ***Steven M. Christey, Ingénieur Sécurité de l'Information Principal et Rédacteur CVE, Mitre***

« Le Top 10 de l'OWASP braque un projecteur directement sur l'un des risques les plus sérieux et le plus souvent rencontré par les instances gouvernementales et entreprises commerciales. La cause première de ces risques ne provient pas de logiciels défectueux, mais des processus de développements logiciels qui prêtent peu ou pas attention à la sécurité. La première étape la plus efficace vers la création d'une prise de conscience de la sécurité dans votre entreprise est d'adopter immédiatement le Top 10 comme standard minimum de sécurité pour la sécurité des applications web. »

- ***Jeffrey R. Williams, PDG d'Aspect Security et Leader du projet OWASP Top 10.***

« Il n'y a aucune solution miracle pour la sécurité web malgré ce que quelques sociétés de technologie voudraient vous faire croire. La résolution de ce complexe et stimulant problème réside dans le fait de disposer de personnes de valeur, de grandes connaissances, de bonne éducation, de bons processus commerciaux et de l'utilisation d'une technologie avancée. Le Top 10 de l'OWASP fournit une liste acceptable aussi bien ordonnée que sérieusement réfléchie à partir de laquelle vous pouvez commencer à comprendre votre attitude en terme de sécurité (ou celle de vos fournisseurs de service) et planifier l'utilisation de vos précieuses ressources en conséquence. »

- ***Mark Curphey, Fondateur de l'OWASP et Directeur du cabinet Application Security Consulting, Sécurité Stratégique Foundstone.***

« Des pages web au moindre bureau back office, presque toutes les entreprises prennent goût au code applicatif, beaucoup de personnes l'écrivant et chacun l'utilisant. Mais des failles continuent à être trouvées dans les applications, même après presque cinquante ans d'expérience de programmation. Pire, les mêmes sortes de failles apparaissent à maintes reprises. Cet échec d'apprentissage non seulement de nos erreurs mais aussi de ceux de la génération de nos parents crée bien trop de vulnérabilités pour une attaque potentielle. Il n'est donc pas étonnant que les attaques contre les applications soient en hausse.

En compilant cette liste des dix vulnérabilités de code applicatifs les plus critiques, l'OWASP a accompli / fourni un réel service tant aux développeurs qu'aux utilisateurs en focalisant l'attention sur les faiblesses courantes et ce qui peut y remédier. C'est maintenant aux entreprises de développement de logiciels, aux programmeurs et aux utilisateurs d'appliquer les prudents conseils présentés ici. »

- ***Docteur Charles P. Pfleeger, CISSP, Architecte de Sécurité, Cable & Wireless, Auteur de « Security in Computing »***

« Le nouveau ROI est la sécurité. Les sociétés doivent être capables de fournir des applications web de confiance et des services internet à leurs associés qui exigent tant une technologie sûre qu'un indice traditionnel de sécurité financière, comme l'assurance. Cela ne s'arrête pas là quoique ; une mauvaise sécurité met les données d'une société et les applications en danger et remet donc en cause sa viabilité en tant qu'entité commerciale. Décevoir vos clients est une chose, essayer de se remetttre de la perte ou de la corruption de vos propres systèmes en est une autre. »

- ***Robert A. Parisi, Jr, Senior VP et Chief Underwriting Officer, AIG Solutions de Risque de commerce électronique***

« Les développeurs web doivent savoir que le degré de sécurisation inhérent aux applications de gestion et aux données clients qui sont protégées de l'Internet hostile est directement déterminé par le niveau de programmation sécurisée de leur code. La liste du Top 10 de l'OWASP est une vraie manière de comprendre comment programmer de façon sécurisée et éviter les pièges de sécurité qui mettent à mal les applications web. »

- ***Chris Wysopal, Directeur Recherche & Développement, @stake, Inc.***

« La fourniture d'applications web sûres protégeant la vie privée des utilisateurs est un besoin critique pour l'industrie médicale. Le Top 10 de l'OWASP aide les entreprises médicales à évaluer la sécurité des logiciels applicatifs et solutions web. La conformité aux règlements HIPAA peut être difficile pour toute entreprise médicale utilisant des applications contenant des failles. »

- ***Lisa Gallagher, VP Senior, Accréditations d'Informations et de Technologie, URAC***

« De plus en plus de sociétés se tournant vers l'externalisation extra-territoriale pour le développement d'applications web (développement off-shore), le respect d'une programmation sécurisée doit figurer en tête de la liste prioritaire de la Direction. Le Projet Top 10 de l'OWASP fait clairement état des vulnérabilités applicatives majeures, fournissant aux cadres dirigeants une mine d'or en terme d'informations utiles permettant la mise en place d'une politique de sécurité pour les applications web. »

- ***Stuart McClure, President et CTO de Foundstone Inc.***

« Les considérations de sécurité applicatives sont souvent traitées comme le domaine des spécialistes, à être appliquées après que la programmation soit faite et que toutes les autres exigences d'affaires sont satisfaites. Chez Oracle nous avons constaté que la réussite de la tâche de sécuriser des applications est d'autant plus probable si la sécurité est conçue dès le début. Les développeurs doivent avoir une compréhension de base des vulnérabilités de sécurité et de la façon de les éviter, et les contrôleurs de versions doivent explicitement vérifier la prise en compte des exigences de sécurité avant que le produit ne soit expédié. La liste du Top 10 de l'OWASP est un excellent point de départ pour les sociétés pour mettre en place une prise de conscience de la sécurité parmi leurs développeurs, et préparer des critères de sécurité pour le déploiement d'application. »

- ***John Heimann, Directeur de la Sécurité chez Oracle Corp.***

« Beaucoup de personnes ont vu des listes Top 10 détailler «les plus grands risques de sécurité de réseau ». Ces listes font seulement état de failles dans des logiciel tiers que vous pouvez ou ne pouvez pas faire fonctionner sur votre réseau. L'application des informations de cette liste tiendra le logiciel que vous développez à l'écart de ces autres listes! »

- ***John Viega, Scientifique en Chef, Secure Software Inc., co-auteur de « Conception de Logiciel Sécurisé »***

Table des matières

Introduction

Le Projet Sécurité des Applications Internet de l'OWASP (Open Web Application Security Project) est dédié à aider les entreprises à comprendre et améliorer la sécurité de leurs applications et services internet. Cette liste a été créée pour focaliser l'attention des sociétés et des agences gouvernementales sur les plus sérieuses de ces vulnérabilités. La sécurité des applications web est devenue un sujet d'actualité brûlant en regard de la course que se livre les sociétés pour rendre du contenu et des services accessibles par le web. Dans le même temps, les pirates informatiques focalisent leur attention sur les faiblesses communément créées par les développeurs d'applications.

Quand une entreprise héberge une application web, elles invitent le monde à leur envoyer des requêtes HTTP. Les attaques cachées dans ces requêtes traversent les coupe-feux (firewalls), les filtres, les plate-formes renforcées et les systèmes de détection d'intrusion sans avertissement parce qu'elles sont encapsulées à l'intérieur des demandes HTTP légales. Même les sites web « secure » qui utilisent SSL acceptent les demandes qui arrivent par le tunnel chiffré sans examen minutieux. **Cela signifie que votre code d'application web fait partie de votre périmètre de sécurité**. Votre périmètre d'exposition va de pair avec l'augmentation du nombre, de la taille et de la complexité de vos applications web.

Les questions de sécurité évoquées ici ne sont pas nouvelles. En fait, certaines ont été bien comprises pendant des décennies. Pour toute une variété de raisons encore, des projets majeurs de développement de logiciel font toujours ces erreurs et mettent en danger non seulement la sécurité de leurs clients, mais aussi la sécurité de l'Internet tout entier. Il n'y a aucune solution miracle à ces problèmes. Les technologies d'évaluation et de protection d'aujourd'hui s'améliorent, mais peuvent actuellement seulement traiter avec un sous-ensemble limité de publications au mieux. Pour aborder les questions décrites dans ce document, les entreprises devront changer leur culture de développement, former les développeurs, mettre à jour leurs processus de développements logiciel, et utiliser la technologie quand cela s'avèrera nécessaire.

Le Top 10 de l'OWASP est une liste des vulnérabilités qui exigent une remédiation immédiate. Le code de programmation existant devrait être immédiatement vérifié pour ces vulnérabilités, ces failles étant activement ciblées par les attaquants. Les projets de développement devraient répertorier ces vulnérabilités dans leurs documents d'exigences et concevoir, construire et évaluer leurs applications pour s'assurer de leur conformité. Les chefs de projet devraient inclure le temps et le budget pour les activités de sécurité applicatives comprenant la formation des développeurs, le développement d'une politique de sécurité applicative, la conception de mécanisme de sécurité et de développement, des tests d'intrusion et l'audit de sécurité de code.

Nous encourageons les entreprises à rejoindre la liste croissante des sociétés qui ont **adopté le Top 10 de l'OWASP comme standard minimum** et sont engagées dans la conception d'applications web qui ne contiennent pas ces vulnérabilités.

Nous avons choisi de présenter cette liste dans un format similaire à la liste du Top 20 du SANS/FBI qui a rencontré un très large succès, pour faciliter son utilisation et sa compréhension. La liste du SANS se concentre sur des failles particulières largement utilisées sur un réseau et des produits d'infrastructure. Parce que chaque site web est unique, le Top 10 de l'OWASP est organisé autour de types particuliers de catégories de vulnérabilités fréquemment rencontrées dans les applications web. Ces catégories sont standardisées dans le projet XML « OASIS Web Application Security » (WAS).

Cette liste représente la vision combinée des experts de l'OWASP, dont l'expérience est le fruit de nombreuses années de travail sur la sécurité applicative pour des gouvernements, des services financiers, des entreprises de logiciels et de fabrication pharmaceutique, aussi bien que des outils de développement et de technologie. Ce document est conçu pour présenter les vulnérabilités des applications web les plus sérieuses. Beaucoup de livres et notes indicatives décrivent ces vulnérabilités plus en détail et fournissent des conseils détaillés sur la façon de les éliminer. Une de ces directives est le Guide de l'OWASP, disponible à http://www.owasp.org .

Le Top 10 de l'OWASP est un document en constante évolution. Il comprend des instructions étape par étape et des indicateurs sur l'information utile complémentaire permettant de corriger les failles de sécurité. Nous mettrons à jour la liste et les instructions au fur et à mesure de l'identification de la criticité des menaces et des pratiques méthodologiques, vos commentaires étant les bienvenus. Ce document est basé sur un consensus communautaire — votre expérience à contrer les attaquants et à l'élimination des vulnérabilités peut aider ceux qui viennent après vous. Merci de soumettre vos suggestions par courrier électronique à topten@owasp.org avec le sujet « Commentaires Top 10 de l'OWASP ».

OWASP est reconnaissant à la contribution spéciale d'**Aspect Security** pour son rôle dans la recherche et la préparation de ce document.

http://www.aspectsecurity.com

Historique

Le challenge d'identifier le « Top » des vulnérabilités des applications web est une tâche pratiquement impossible. Il n'y a pas même un accord universel sur ce qui est exactement inclus dans le terme « sécurité des applications web ». Certains ont soutenu que nous devrions seulement nous focaliser sur les problèmes de sécurité qui affectent les développeurs écrivant du code applicatif web personnalisé. D'autres ont argumenté pour une définition plus expansive qui couvre la couche applicative entière, y compris les bibliothèques, la configuration de serveur et les protocoles de la couche Application. Dans l'espoir de répertorier les risques les plus sérieux rencontrés par les entreprises, nous avons décidé de donner une interprétation relativement large de la sécurité des applications web, tout en restant toujours clairs avec les problèmes de sécurité réseau et d'infrastructure.

Un autre challenge à cet effort consiste en ce que chaque vulnérabilité spécifique est unique au site web d'une entreprise particulière. Il y aurait peu d'intérêt à montrer du doigt des vulnérabilités spécifiques dans les applications web d'entreprises individuelles, spécialement depuis qu'elles ont bon espoir d'être bientôt résolues après qu'un large auditoire ait connaissance de leur existence. Nous avons donc voulu nous concentrer sur les classes majeures, les types, ou catégories de vulnérabilités des applications web.

Dans la première version de ce document, nous avions décidé de classer un grand nombre de problèmes relatifs aux applications web dans des catégories significatives. Nous avons étudié une variété de classification de vulnérabilité et avons inventé un jeu de catégories. Des failles étroitement relatées, pouvant être répertoriées avec des contre-mesures adéquates, et qui se produisent fréquemment dans des architectures d'applications web types sont les facteurs qui caractérisent une bonne catégorie de vulnérabilité. Dans cette version, nous présentons un schéma amélioré. Ce développement est le fruit de notre travail continu sur le comité technique de l'OASIS WAS dans lequel nous décrirons un Thesaurus de questions dont les chercheurs en sécurité peuvent décrire des signatures dans un format XML.

Choisir un Top 10 à partir d'une grande liste de candidats est une gageure. Il n'y a simplement aucune source fiable de statistiques sur les problèmes de sécurité des applications web. Dans l'avenir, nous voudrions recueillir des statistique sur la fréquence de certaines failles dans la programmation de code applicatif web et utiliser cette métrique pour aider à prioriser le Top 10. Cependant, pour nombre de raisons, ce type de mesure ne va probablement pas arriver dans un proche avenir.

Nous reconnaissons qu'il n'y a aucune réponse « juste » justifiant de la présence des catégories de vulnérabilité dans le Top 10. Chaque entreprise devra penser à évaluer le risque de probabilité d'avoir une de ces failles et les conséquences spécifiques inhérentes à leur périmètre. En attendant, nous mettons cette liste en avant comme un référentiel des problèmes représentant une quantité significative de risque pour la plupart des entreprises. Le Top 10 lui-même ne respecte pas d'ordre particulier, car il serait presque impossible de déterminer la faille la plus représentative en terme de risque global.

Le Projet Top 10 de l'OWASP est un effort en cours afin de rendre l'information sur les principales failles de sécurité applicatives internet disponibles à un large public. Nous pensons mettre à jour ce document annuellement en fonction des discussions sur les listes de diffusion de l'OWASP et des réactions à topten@owasp.org.

Nouveautés

OWASP a parcouru du chemin depuis la parution du dernier Top sorti en janvier 2003. Cette mise à jour incorpore l'ensemble des discussion et avis les plus récents, opinions et débats dans la communauté OWASP des 12 derniers mois. Au total, cela n'a donné lieu qu'à des améliorations mineures sur toutes les parties du Top 10, et seulement quelques changements majeurs :

- **L'alignement WAS-XML** – Un des nouveaux projets initié en 2003 est le Comité Technique de Sécurité des Applications Web (WAS TC - Web Application Security Technical Committee) à l'OASIS. Le but de WAS TC est d'établir une étude de classification des vulnérabilités de sécurité internet, un modèle fournissant des conseils relatifs à la menace initiale, à l'impact et par conséquent des indices de risque, et un schéma XML pour décrire les conditions de sécurité web qui peuvent être utilisées tant par des outils de protection que d'évaluation. Le projet Top 10 de l'OWASP a utilisé le WAS TC comme une référence pour ré-établir le profil du Top 10 pour fournir une approche standardisée à la classification des vulnérabilités de sécurité des applications web. Le dictionnaire WAS définit un langage standard pour discuter de la sécurité des applications web, et nous adoptons ce vocabulaire ici.
- **Ajout de Déni de Service** – La seule catégorie majeure qui a changé était l'ajout à la liste de la catégorie « A9 - Déni de Service ». Notre recherche a montré qu'un large panel d'entreprises sont sensibles à ce type d'attaque. En se basant sur la probabilité d'une attaque par déni de service et des conséquences si l'attaque réussit, nous avons déterminé que cela justifiait son inclusion dans le Top 10. Pour concilier cette nouvelle entrée, nous avons combiné les failles d'Administration à distance A9 de l'année dernière dans la catégorie A2 de Violation de Contrôle d'Accès, ceci constituant un cas spécial de cette catégorie. Nous pensons que cela est approprié, les types de failles dans A2 étant typiquement les mêmes que celles de A9 et exigeant les mêmes types de remédiation.

La table ci-dessous met en évidence le rapport entre le nouveau Top 10, le Top 10 de l'année dernière, et le dictionnaire WAS TC.

Nouveau Top 10 2004	Top 10 2003	Nouveau Dictionnaire WAS
A1 Paramètre non validé	A1 Paramètres non validés	Validation de paramètre
A2 Violation de Contrôle d'Accès	A2 Violation de Contrôle d'Accès (A9 Failles d'Administration à distance)	Contrôle d'Accès
A3 Violation de Gestion d'Authentification et de Session	A3 Violation de Gestion d'Authentification et de Session	Gestion d'Authentification et de Session
A4 Failles Cross Site Scripting (XSS)	A4 Failles Cross Site Scripting (XSS)	Validation de paramètres ->Cross site scripting
A5 Débordement de tampon	A5 Débordement de tampon	Débordement de tampon
A6 Failles d'Injection	A6 Failles d'Injection de Commande	Validation de paramètres ->Injection
A7 Traitement d'erreur incorrect	A7 Problèmes de traitement d'erreur	Traitement d'erreur
A8 Stockage non sécurisé	A8 Usage non sécurisé de la cryptographie	Protection de donnée
A9 Déni de Service	N/A	Disponibilité
A10 Gestion de Configuration non sécurisée	A10 Mauvaise configuration de serveur web et applicatif	Gestion de configuration applicative Gestion de configuration d'infrastructure

La Liste du Top 10

Ce qui suit est un court résumé des plus significatives vulnérabilités de sécurité des applications web. Chacune d'entre elles est décrite plus en détail dans les sections suivantes.

Les vulnérabilités majeures dans les Applications Web		
A1	**Paramètre non validé**	L'information des requêtes web n'est pas validée avant d'être utilisée par une application web. Les attaquants peuvent utiliser ces failles pour attaquer des composants secondaires à travers une application web.
A2	**Violation de Contrôle d'Accès**	Les restrictions sur ce que l'on permet de faire aux utilisateurs authentifiés ne sont pas correctement mises en application. Les attaquants peuvent exploiter ces failles pour avoir accès aux comptes d'autres utilisateurs, voir des fichiers sensibles, ou utiliser des fonctions non autorisées.
A3	**Violation de Gestion d'Authentification et de Session**	Les accréditations de compte et jetons de session ne sont pas correctement protégées. Les attaquants qui peuvent compromettre des mots de passe, des clés, des cookies de session ou autres jetons, peuvent faire échouer les restrictions d'authentification et s'approprier les identités d'autres utilisateurs.
A4	**Failles Cross Site Scripting (XSS)**	L'application web peut être utilisée comme un mécanisme pour véhiculer une attaque vers le navigateur d'un utilisateur final. Une attaque couronnée de succès peut révéler le jeton de session de l'utilisateur final, attaquer la machine locale, ou imiter du contenu pour tromper l'utilisateur.
A5	**Débordement de tampon**	Les composants applicatifs web dans quelques langages qui ne valident pas correctement les paramètres en entrée peuvent être « crashés » et, dans certains cas, utilisés pour prendre le contrôle d'un processus. Ces composants peuvent inclure des CGI, des bibliothèques, des pilotes et des composants de serveur applicatif web.
A6	**Injection de commandes**	Les applications web passent des paramètres quand elles accèdent à des systèmes externes ou au système d'exploitation local. Si un attaquant peut insérer des commandes malveillantes dans ces paramètres, le système externe peut exécuter ces commandes provenant de l'application web.
A7	**Mauvaise gestion des erreurs**	Les conditions d'erreur qui se produisent pendant une opération normale ne sont pas traitées correctement. Si un attaquant peut arriver à ce que des erreurs non prises en compte par l'application web aient lieu, elles peuvent obtenir des informations système détaillées, causer un déni de service, faire échouer des mécanismes de sécurité, ou « crasher » le serveur.
A8	**Stockage non sécurisé**	Les applications web utilisent fréquemment des fonctions cryptographiques pour protéger des informations et des références. Ces fonctions et le chiffrement permettant de les intégrer ont prouvé la difficulté de chiffrer correctement, aboutissant fréquemment à une protection faible.
A9	**Déni de Service**	Les attaquants peuvent consommer les ressources des applications web au point où d'autres utilisateurs légitimes ne peuvent plus avoir accès ou utiliser l'application. Les attaquants peuvent aussi empêcher les utilisateurs d'utiliser leurs comptes ou même mettre l'application entière en échec.

A10	**Gestion de configuration non sécurisée**	Disposer d'un standard de configuration serveur robuste est un besoin critique pour une application web sécurisée. Ces serveurs ont beaucoup d'options de configuration qui affectent la sécurité et ne sont pas sûres hors contexte.

A1 Paramètre non validé

A1.1 Description

Les applications web utilisent les demandes d'entrée des requêtes HTTP (et occasionnellement des fichiers) pour déterminer comment répondre. Les attaquants peuvent falsifier n'importe quelle partie d'une requête HTTP, y compris l'URL, la chaîne de requête, les en-têtes, les cookies, les champs de formulaire et les champs cachés, pour essayer de contourner les mécanismes de sécurité du site. Des noms courants pour les attaques types de falsification des entrées incluent:: le *browsing* forcé, l'insertion de commande, le cross site scripting, le débordement de tampon, les attaques de chaîne de format, l'injection SQL, la corruption de cookie (*poisoning*) et la manipulation de champs cachés. Chacun de ces types d'attaque est décrit plus en détail un peu plus loin dans ce document.

- « A4 – Failles Cross Site Scripting » (ou *programmation inter-site*), parle d'entrée qui contient des scripts à exécuter sur les navigateurs d'autre utilisateurs
- « A5 – Débordement de tampon », parle d'entrée qui a été conçue pour réécrire dans un espace d'exécution de programme
- « A6 - Failles d'Injection », parle d'entrée qui est modifiée pour contenir des commandes exécutables

Certains sites essayent de se protéger par filtrage de l'entrée malveillante. Le problème est qu'il existe beaucoup de manières différentes d'encoder l'information. Ces formats d'encodage ne ressemblent pas au chiffrement, puisqu'ils sont triviaux à décoder. Cependant, les développeurs oublient souvent de décoder tous les paramètres dans leur forme la plus simple avant de les utiliser. Les paramètres doivent être convertis sous la forme la plus simple avant qu'ils ne soient validés, autrement, l'entrée malveillante peut être masquée et peut passer les filtres sans être détectée. Le processus de simplification de ces encodages s'appelle la « canonisation ». Puisque presque toute entrée HTTP peut être représentée dans des formats multiples, cette technique peut être employée pour masquer n'importe quelle attaque visant les vulnérabilités décrites dans ce document. Ceci rend le filtrage très difficile.

Un nombre surprenant d'applications web utilise seulement des mécanismes côté client pour valider l'entrée. Les mécanismes de validation côté client sont facilement contournables, laissant l'application web sans aucune protection contre les paramètres malveillants. Les attaquants peuvent générer leurs propres requêtes HTTP en utilisant des utilitaires aussi simples que Telnet. Ils n'ont pas à prêter attention à ce que le développeur ait eu l'intention de produire du côté client. Notez que la validation du côté client est une excellente idée pour la performance et la rentabilité, mais toutefois sans représenter aucun profit sécurité. Les contrôles du côté serveur sont requis pour se défendre contre les attaques de manipulation de paramètre. Une fois ceux-ci en place, la vérification côté client peut aussi être implémentée pour améliorer les contrôles utilisateur, pour les utilisateurs légitimes et/ou réduire le taux de trafic invalide à destination du serveur.

Ces attaques deviennent de plus en plus probables en fonction de la croissance du nombre d'outils qui supportent le paramètre « fuzzing », la corruption et la force brute. L'impact d'une utilisation de paramètre d'entrée non validé ne devrait pas être sous-estimée. Un immense nombre d'attaques deviendraient difficiles ou impossibles si les développeurs validaient simplement le paramètre d'entrée avant utilisation. À moins qu'une application web ne dispose d'un fort mécanisme centralisé pour valider toute entrée émanant de requêtes HTTP (et de toutes autres sources), les vulnérabilités basées sur des paramètres d'entrée malveillants vont très vraisemblablement persister.

A1.2 Environnements Affectés

Tous les serveurs web, serveurs applicatifs et les environnements applicatifs web sont sensibles à la falsification de paramètre.

A1.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services web sûrs », Chapitre 8 : Validation de données http://www.owasp.org/documentation/guide/

- Projet modsecurity (Module Apache pour validation HTTP) http://www.modsecurity.org
- Comment construire un moteur de validation de requête HTTP (validation J2EE avec Stinger) http://www.owasp.org/columns/jeffwilliams/jeffwilliams2
- « Prenez votre gâteau et mangez-le aussi » (validation .NET) http://www.owasp.org/columns/jpoteet/jpoteet2

A1.4 Comment déterminer si vous êtes vulnérable

Toute partie d'une requête HTTP qui est utilisée par une application web sans être soigneusement validée est connue comme un paramètre « infecté » (*tainted*). La façon la plus simple de trouver une utilisation de paramètre infecté est d'effectuer un audit de code détaillé, cherchant tous les appels où les informations sont extraites d'une requête HTTP. Par exemple, dans une application J2EE, ceux-ci constituent les méthodes dans la classe « HttpServletRequest ». Vous pouvez alors suivre le code pour voir où cette variable va être utilisée. Si la variable n'est pas vérifiée avant qu'elle ne soit utilisée, il y a très probablement un problème. Dans Perl, vous devriez envisager d'utiliser l'option « taint » (-T) (i.e. *infecté*).

Il est aussi possible de trouver une utilisation de paramètre infecté en utilisant des outils comme WebScarab de l'OWASP. En soumettant des valeurs inattendues dans des requêtes HTTP et en regardant les réponses de l'application web, vous pouvez identifier les endroits où les paramètres infectés sont utilisés.

A1.5 Comment vous protéger

La meilleure façon d'empêcher la falsification de paramètre est de s'assurer que tous les paramètres sont validés avant qu'ils ne soient utilisés. Un composant centralisé ou une bibliothèque vont probablement être les plus efficaces, de façon que le programme exécutant la vérification se trouve en une place unique. Chaque paramètre devrait être vérifié contre un format strict qui spécifie exactement quel paramètre d'entrée sera autorisé. Les approches « négatives » qui impliquent le filtrage en sortie de certaines mauvaise entrées ou les approches qui comptent sur des signatures ne vont probablement pas être efficaces et peuvent être difficiles à maintenir.

Les paramètres devraient être validés contre une spécification « positive » qui définit :

- le type de données (chaîne, entier, réel, etc…)
- le jeu de caractère autorisé
- la longueur minimale et maximale
- si le nul est permit
- si le paramètre est exigé ou non
- si les duplicatas sont permis
- l'étendue numérique
- les valeurs légales spécifiques (énumération)
- les modèles spécifiques (expressions régulières)

Une nouvelle classe de dispositifs de sécurité connus sous le noms de coupe-feux (firewalls) applicatifs web peut fournir quelques services de validation de paramètre. Cependant, afin qu'il puisse être efficace, le dispositif doit être configuré avec une définition stricte de ce qui est valide pour chaque paramètre de votre site. Cela inclut la protection correcte de tous les types d'entrée de requêtes HTTP, y compris URLs, les formulaires, les cookies, les chaînes de requêtes, les champs cachés et les paramètres.

Le projet de filtrage de l'OWASP fournit des composants réutilisables dans plusieurs langages pour aider à empêcher beaucoup de formes de falsification de paramètre. Le moteur de validation de requête HTTP Stinger (stinger.sourceforge.net) a aussi été développé par l'OWASP pour les environnements J2EE.

A2 Violation de Contrôle d'Accès

A2.1 Description

Le contrôle d'accès, parfois appelé autorisation, consiste en la façon dont une application web accorde l'accès à du contenu et à des fonctions, à des utilisateurs et pas à d'autres. Ces contrôles sont exécutés après authentification et régissent ce que les utilisateurs « autorisés » ont la permission de faire. Le contrôle d'accès ressemble à un problème simple, mais est insidieusement difficile à mettre en oeuvre correctement. Un modèle de contrôle d'accès d'application web est étroitement lié au contenu et aux fonctions que le site fournit. De plus, les utilisateurs peuvent tomber dans un certain nombre de groupes ou rôles avec des capacités ou des privilèges différents.

Les développeurs sous-estiment fréquemment la difficulté de mise en oeuvre d'un mécanisme de contrôle d'accès fiable. Beaucoup de ces schémas n'ont délibérément pas été conçus, mais se sont simplement développés avec le site web. Dans ces cas, les règles de contrôle d'accès sont insérées dans des emplacements divers à travers tout le programme. Le site approchant de sa phase de déploiement, la collection des règles ad hoc devient si peu maniable qu'elle est presque impossible à comprendre.

Beaucoup de ces failles de contrôle d'accès défectueux ne sont pas difficiles à découvrir et exploiter. Fréquemment, tout ce qui est nécessaire est de forger une requête pour des fonctions ou du contenu qui ne devrait pas être accordée. Une fois la faille découverte, les conséquences d'un schéma de contrôle d'accès défectueux peuvent être dévastatrices. En plus de l'observation de contenu non autorisé, un attaquant pourrait être capable de changer ou supprimer du contenu, exécuter des fonctions non autorisées, ou même prendre le contrôle de l'administration du site.

Un type spécifique de problème de contrôle d'accès est inhérent aux interfaces administratives qui permettent aux administrateurs de site de gérer un site sur l'Internet. De telles fonctions sont fréquemment utilisées pour permettre aux administrateurs de site de gérer efficacement les utilisateurs, les données et le contenu sur leur site. Dans beaucoup de cas, les sites supportent toute une variété de rôles administratifs pour en permettre une granularité d'administration plus fine. En raison de leur puissance, ces interfaces sont fréquemment des cibles principales d'attaque tant par des utilisateurs externes qu'internes.

A2.2 Environnements Affectés

Tous les serveurs web connus, serveurs applicatifs et environnements d'application web sont sensibles à au moins certaines de ces failles. Même si un site est complètement statique, s'il n'est pas configuré correctement, les pirates informatiques pourraient obtenir un accès à des fichiers sensibles et « défacer » le site, ou exécuter d'autres méfaits.

A2.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services Web sûrs », Chapitre 8 : Validation de Données: http://www.owasp.org/guide/
- Contrôle d'Accès (i.e. Autorisation) dans votre Application J2EE http://www.owasp.org/columns/jeffwilliams/jeffwilliams3
- http://www.infosecuritymag.com/2002/jun/insecurity.shtml

A2.4 Comment déterminer si vous êtes vulnérable

Pratiquement tous les sites disposent de quelques exigences de contrôle d'accès. Par conséquent, une politique de contrôle d'accès devrait être clairement documentée. La documentation de conception devrait aussi disposer d'une approche pour imposer cette politique. Si cette documentation n'existe pas, le site va donc probablement être vulnérable.

Le programme (le code) qui met en application la politique de contrôle d'accès devrait être vérifié. Un tel code devrait être bien structuré, modulaire et plus vraisemblablement centralisé. Un audit de code détaillé devrait être exécuté pour valider

l'exactitude de la mise en oeuvre du contrôle d'accès. De plus, la mise en oeuvre de tests de pénétration peut s'avérer utile pour déterminer s'il y a des problèmes de contrôle d'accès.

Découvrez comment votre site web est administré. Vous voulez découvrir comment les changements sont effectués sur les pages web, et comment ils sont répliqués au serveur de production. Si les administrateurs peuvent faire des changements à distance, vous voulez savoir comment ces canaux de communications sont protégés. Passez soigneusement en revue chaque interface afin de vous assurer que l'accès est seulement permit aux administrateurs autorisés. De plus, s'il y a différents types ou groupements de données qui peuvent être consultées par l'interface, assurez-vous aussi que seules les données autorisées peuvent être consultées. Si de telles interfaces emploient des commandes externes, passez en revue l'utilisation de telles commandes pour vous assurer qu'elles ne sont pas sujettes aux failles d'injection de commande décrites dans ce document.

A2.5 Comment vous protéger

L'étape la plus importante est de bien réfléchir aux exigences de contrôle d'accès d'une application et de les formaliser dans une politique de sécurité d'application web. Nous recommandons fortement l'utilisation d'une matrice de contrôle d'accès pour définir les règles de contrôle d'accès. Sans documentation de la politique sécurité, il y a aucune définition sur ce qui doit être sécurisé pour ce site. La politique devrait documenter quels types d'utilisateurs peuvent accéder au système, et les permission d'accès aux fonctions et contenus inhérentes à chacun de ces types d'utilisateurs. Le mécanisme de contrôle d'accès devrait être fortement évalué pour être sûr qu'il n'y a aucune manière de le contourner. Ce test nécessite toute une variété de comptes et de tentatives pour accéder à du contenu non autorisé ou à des fonctions.

Quelques thèmes de contrôle d'accès spécifiques comprennent :

- Identifications non sécurisées - La plupart des sites web utilisent certaines forme d'identification, clef, ou index pour référencer les utilisateurs, les rôles, le contenu, les objets, ou des fonctions. Si un attaquant peut deviner ces identités et que les valeurs fournies ne sont pas validées pour s'assurer qu'elles sont autorisées pour l'utilisateur en cours, l'attaquant peut librement exercer une modification de contrôle d'accès pour voir à quoi elles peuvent avoir accès. En terme de protection, les applications web ne devraient compter sur le secret d'aucune identité.
- Vérifications du contrôle d'accès de *browsing* - beaucoup de sites demandent que les utilisateurs passent certains contrôles avant d'être autorisés à accéder à certaines URLs qui sont typiquement « plus en profondeur » dans le site. Ces contrôles ne doivent pas être contournables par un utilisateur qui passe simplement sur la page avec le contrôle de sécurité.
- Traversée de path (cheminement) - Cette attaque implique de fournir une informations relative au « path » (par exemple, « ../../target_dir/target_file ») intégrée à une demande d'informations. De telles attaques essayent d'avoir accès à des fichiers qui ne sont normalement pas directement accessibles par qui que ce soit, ou dont l'accès seraient autrement refusés en cas de requête directe. De telles attaques peuvent être soumises aussi bien dans les URLs que dans une autre entrée qui a en fin de compte accès à un fichier (i.e. appels système et commandes shell).
- Permissions de Fichier - Beaucoup de serveurs web applicatifs comptent sur des listes de contrôle d'accès fournies par le système de fichiers de la plate-forme sous-jacente. Même si presque toutes les données sont stockées sur des serveurs secondaires, il y a toujours des fichiers stockés localement sur le serveur web applicatif qui ne devraient pas être publiquement accessibles, particulièrement des fichiers de configuration, des fichiers par défaut et des scripts qui sont installés sur la plupart des serveurs web applicatifs. Seuls les fichiers qui sont spécifiquement destinés à être présentés aux utilisateurs web devraient être marqués comme lisibles en utilisant le mécanisme de permissions de l'OS, la plupart des répertoires ne devrait pas être lisibles, et très peu de fichiers, s'il en est, devraient être marqués comme exécutables.
- Mise en cache côté Client - Beaucoup d'utilisateurs accèdent aux applications web à partir d'ordinateurs mutualisés dans des bibliothèques, des écoles, des aéroports et autres points d'accès publics. Les navigateurs mettent fréquemment en cache les pages web qui peuvent être accédées par des attaquants pour obtenir un accès aux parties autrement inaccessibles des sites. Les développeurs devraient utiliser des mécanismes multiples, y compris les en-têtes HTTP et les « meta tags », pour être sûr que les pages contenant des informations sensibles ne sont pas mises en cache par les navigateurs des utilisateurs.

Il existe quelques composants de sécurité de la couche Application qui peuvent aider dans l'exécution appropriée de quelques aspects de votre schéma de contrôle d'accès. De nouveau, quant à la validation de paramètre, pour être efficace,

le composant doit être configuré avec une définition stricte de la validation des requêtes d'accès pour votre site. En utilisant un tel composant, vous devez être prudents dans la façon de comprendre exactement quelle assistance de contrôle d'accès le composant peut vous fournir en fonction de la politique sécurité de votre site, quelle partie de votre politique de contrôle d'accès le composant ne peut pas traiter, et doit donc être correctement traité dans votre propre code personnalisé.

Concernant les fonctions administratives, la recommandation principale est de ne jamais permettre l'accès Administrateur par la porte d'entrée de votre site dans la mesure du possible. Étant donné la puissance de ces interfaces, la plupart des entreprises ne devraient pas accepter le risque de rendre ces interfaces disponibles à une attaque extérieure. Si l'accès d'administration à distance est absolument nécessaire, cela peut être accompli sans ouvrir la porte d'entrée du site. L'utilisation de technologie VPN pourrait être utilisée pour fournir un accès d'administration extérieur au réseau interne (ou au site) de la société à partir duquel un administrateur peut alors avoir accès au site par une connexion secondaire protégée.

A3 Violation de gestion d'Authentification et de Session

A3.1 Description

La gestion d'authentification et de session inclue tous les aspects de traitement d'authentification utilisateur et la gestion des sessions actives. L'Authentification est un aspect critique de ce processus, mais même des mécanismes d'identification solides peuvent être altérés par des fonctions de gestion d'accréditation défectueuses, y compris le changement de mot de passe, oubli de mot de passe, rappel de mot de passe, mise à jour de compte et d'autres fonctions liées. Du fait de la probabilité de « la voie par » les attaques pour beaucoup d'applications web, toutes les fonctions de gestion de compte devraient exiger une réauthentification même si l'utilisateur a un ID de session valide.

L'authentification utilisateur sur le web implique typiquement l'utilisation d'un identifiant utilisateur et d'un mot de passe. Des méthodes d'authentification plus fortes sont disponibles dans le commerce telles les solutions logicielles et matérielles a base de « tokens » cryptographiques ou biométriques, mais de tels mécanismes sont coûteux pour la plupart des applications web. Un large panel de failles de gestion de compte et de session peut aboutir à la compromission de comptes utilisateurs ou d'administration système. Les équipes de développement sous-estiment fréquemment la complexité de concevoir un système de gestion d'authentification et de session qui protège en juste proportion les droits d'accès au site sous tous ses aspects.

Les applications web doivent établir des sessions pour maintenir le flot des demandes de chaque utilisateur. HTTP ne fournit pas cette capacité, les applications web devant donc les créer elles-mêmes. Fréquemment, l'environnement web applicatif fournit une capacité de session, mais beaucoup de développeurs préfèrent créer leurs propres jetons de session. Dans un cas ou dans l'autre, si les jetons de session ne sont pas correctement protégés, un attaquant peut détourner une session active et assumer l'identité d'un utilisateur. La création d'un système pour créer des jetons de session forts et les protéger pendant tout leur cycle de vie a été jugée évasive pour beaucoup de développeurs.

À moins que toutes les accréditations d'authentification et que les identifiants de session ne soient protégés avec SSL à tout moment et protégés contre la divulgation d'autres failles comme le cross site scripting, un attaquant peut détourner la session d'un utilisateur et assumer son identité.

A3.2 Environnements Affectés

Tous les serveurs web connus, les serveurs d'application, et les environnements d'application web sont sensibles à la violation de gestion d'authentification et de session.

A3.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services web sûrs », Chapitre 6: Authentification et Chapitre 7: Gestion de session: http://www.owasp.org/guide/
- Libre blanc sur la vulnérabilité de fixation de session dans les applications web: http://www.acros.si/papers/session_fixation.pdf
- Livre blanc sur la récupération de mot de passe pour les applications web - http://fishbowl.pastiche.org/archives/docs/PasswordRecovery.pdf

A3.4 Comment déterminer si vous êtes vulnérable

L'audit de code et les tests de pénétration peuvent être employés pour diagnostiquer des problèmes de gestion d'authentification et de session . Passez soigneusement en revue chaque aspect de vos mécanismes d'identification pour vous assurer que les accréditations de l'utilisateur sont protégées à tout moment, pendant qu'ils sont au repos (par exemple, sur le disque), et pendant qu'ils sont en transit (par exemple, pendant la procédure de connexion). Passez en revue votre mécanisme de gestion de session pour vous assurer que les identifiants de session sont toujours protégés et sont utilisés de telle façon à réduire au minimum la probabilité d'exposition accidentelle ou hostile.

A3.5 Comment vous protéger

L'utilisation prudente et appropriée des mécanismes de gestion d'authentification et de session personnalisés devrait significativement réduire la probabilité d'un problème dans cette zone. Définir et documenter la politique de votre site en ce qui concerne les qualifications de gestion utilisateurs est une bonne première étape. S'assurer que votre implémentation impose uniformément cette politique est clef pour disposer d'un mécanisme de gestion d'authentification et de session sûr et robuste. Quelques secteurs critiques incluent:

- Robustesse des mots de passe - Les mots de passe devraient avoir des restrictions qui exigent une taille minimale et une complexité pour le mot de passe. La complexité exige typiquement l'utilisation de combinaisons minimales de caractères alphabétiques, numériques, et/ou non-alphanumériques dans le mot de passe d'un utilisateur (par exemple, au moins un de chacun). Le changement de mot de passe périodique devrait être une exigence envers tous les utilisateurs, et la réutilisation de mots de passe antérieurs prohibée.
- Utilisation de mot de passe - Les utilisateurs devraient être limités à un nombre défini de tentatives d'établissement de connexion par unité de temps et les échecs de connexion devraient être enregistrées. Les mots de passe fournis pendant les échecs de tentatives d'établissement de connexion ne devraient pas être enregistrés, car cela peut exposer le mot de passe d'un utilisateur à quiconque peut accéder à cet enregistrement. Le système ne devrait pas indiquer s'il s'agissait du nom de l'utilisateur ou du mot de passe en cas d'échec de tentative de connexion. Les utilisateurs devraient être informés de la date/temps de leur dernière connexion et du nombre d'échecs de tentatives d'accès à leur compte depuis cette date.
- Contrôle de changement de mot de passe : Un mécanisme unique de changement de mot de passe devrait être utilisé partout où on permet aux utilisateurs de changer un mot de passe, indépendamment de la situation. Les utilisateurs devraient toujours être obligés de fournir leur ancien mot de passe ainsi que le nouveau lors du changement de leur mot de passe (comme toute information de compte). Si les mots de passe oubliés sont envoyés par courrier électronique aux utilisateurs, le système devrait obliger l'utilisateur à se ré-authentifier chaque fois qu'il change son adresse électronique, autrement un attaquant qui a temporairement accès à leur session (par exemple, en s'approchant de leur ordinateur tandis qu'ils sont connectés) peut changer simplement leur adresse électronique et demander à ce qu'un mot de passe « oublié » leur soit envoyé.
- Stockage de mot de passe - Tous les mots de passe doivent être stockés sous forme « hashée » ou chiffrée pour les protéger d'une exposition, indépendamment d'où ils sont stockés. La forme hashée est préféré puisqu'elle n'est pas réversible. Le chiffrage devrait être utilisé quand le mot de passe en texte est nécessaire, comme l'utilisation de mot de passe lors de l'établissement d'une connexion à un autre système. Les mots de passe ne devraient jamais être codés en dur dans aucun code source. Les clefs de déchiffrement doivent être fortement protégées pour s'assurer qu'elles ne peuvent pas être saisies et utilisées pour déchiffrer le fichier de mot de passe.
- Protection des accréditations en transit - La seule technique efficace est de chiffrer entièrement la transaction d'établissement de connexion en utilisant quelque chose comme SSL. Les transformations simples du mot de passe comme le hasher sur le client avant sa transmission ne fournit que à peu de protection car la version hashée peut être interceptées et retransmise simplement bien que le mot de passe texte actuel ne soit peut-être pas connu.
- Protection de l'ID de session - Idéalement, la session entière d'un utilisateur devrait être protégée via SSL. Si c'est fait, l'ID de session (par exemple, le cookie de session) ne peut pas être intercepté du réseau, ce qui représente le plus grand risque d'exposition pour un ID de session. Si SSL n'est pas viable pour la performance ou d'autres raisons, alors les IDs de session eux-mêmes doivent être protégés par d'autres façons. D'abord, ils ne devraient jamais être inclus dans l'URL car ils peuvent être mis en cache par le navigateur, envoyé dans l'en-tête « referrer », ou a accidentellement expédié à un « ami ». Les IDs de session devraient être constitués de nombres aléatoires longs, compliqués, ne pouvant pas être facilement devinés. Les IDs de session peuvent aussi être fréquemment changés pendant une session pour réduire leur temps de validité. Les IDs de session doivent être changés lors d'un basculement vers SSL, une authentification, ou autres transitions majeures. Les IDs de session choisi par un utilisateur ne devrait jamais être acceptés.
- Listes de compte - Les Systèmes devraient être conçus pour éviter de permettre aux utilisateurs d'accéder à une liste des noms de compte sur le site. Si les listes utilisateurs doivent être présentées, il est recommandé de n'afficher à la place qu'une certaine forme de pseudonyme (nom à l'écran) en adéquation au compte réel. Ce faisant, le pseudonyme ne peut pas être utilisé pendant une tentative de connexion ou un quelconque autre piratage à l'encontre d'un compte utilisateur.
- La fonction de mise en cache du navigateur - L'authentification et les données de session ne devraient jamais être soumises comme une partie d'un GET, POST devant toujours être utilisé à la place. Les pages d'authentification devraient être marquées avec toutes les variétés d'étiquettes de non mise en cache afin d'empêcher quelqu'un d'utiliser le bouton de retour arrière dans le navigateur d'un utilisateur pour revenir à la page d'authentification

(login) et re-soumettre les caractères précédents dans les accréditations. Beaucoup de navigateurs supportent maintenant le flag « autocomplete=false » pour empêcher le stockage des accréditations dans les caches automatiques.

- Les relations d'approbation - Votre architecture de site devrait éviter l'approbation (trust) implicite entre des composants chaque fois que possible. Chaque composant devrait s'authentifier à tout autre composant avec qui il interagit à moins qu'il n'y ait une forte raison de ne pas le faire (comme la performance ou le manque d'utilisation d'un mécanisme). Si des relations d'approbation sont nécessaires, de forts mécanismes procéduraux et d'architecture devraient être mis en place pour s'assurer qu'une telle relation de confiance ne peut pas être flouée en fonction de l'évolution de l'architecture du site avec le temps.

A4 Failles Cross-Site Scripting (XSS)

A4.1 Description

Les vulnérabilités de type Cross-site scripting, ou plus littéralement « *programmation inter-site* », (parfois mentionnées comme XSS) arrivent quand un attaquant utilise une application web pour transmettre du code malveillant, généralement sous la forme d'un script, à un autre utilisateur. Ces failles sont assez répandues et arrivent n'importe où lors de l'utilisation d'une entrée utilisateur par une application web dans le résultat généré sans qu'il soit validé.

Un attaquant peut utiliser le Cross-site scripting pour envoyer le script malveillant à un utilisateur peu soupçonneux. Le navigateur de l'utilisateur final n'a aucune façon de savoir qu'il ne devrait pas faire confiance au script et exécutera celui-ci. Parce qu'il (le navigateur de l'utilisateur) pense que le script émane d'une source de confiance, le script malveillant peut avoir accès à n'importe quels cookies, jetons de session, ou autres informations sensibles conservées par votre navigateur et utilisées avec ce site. Ces scripts peuvent même réécrire le contenu de la page HTML.

Les attaques XSS peuvent généralement être répertoriées en deux catégories : par « stockage » (*stored*), et par « réflexion » (*reflected*). Les attaques dites par stockage sont celles où le code injecté est stocké de manière permanente sur les serveurs cibles, comme dans une base de données, dans un forum de message, enregistrement (log) de visite, champ de commentaire, etc... La victime récupère alors le script malveillant du serveur quand il demande les informations stockées. Les attaques par réflexion correspondent à celles où le code injecté est « retransmit » par le serveur web, comme dans un message électronique, le résultat d'une recherche, ou toute autre réponse incluant tout ou partie du paramètre d'entrée envoyé au serveur en tant que requête. Les victimes subissent les attaques par réflexion par l'intermédiaire d'autre voie, tel un message e-mail, ou sur un quelconque autre serveur web. Quand un utilisateur est duppé en cliquant sur un lien malveillant ou en soumettant un formulaire spécifiquement forgé, le code injecté va jusqu'au serveur web vulnérable, qui renvoie ensuite l'attaque au navigateur de l'utilisateur. Le navigateur exécute alors le code parce qu'il émane d'un serveur « de confiance ».

La conséquence d'une attaque XSS est identique, qu'elle soit de type « par stockage » ou « par réflexion ». La différence réside dans la façon où la charge utile parvient au serveur. Ne soyez pas induits en erreur en pensant qu'un site en « lecture seule » ou en « brochureware » n'est pas vulnérable aux sérieuses attaques XSS « par réflexion ». XSS peut causer toute une variété de problèmes pour l'utilisateur final, d'un désagrément sévère à la complète compromission de compte. Les attaques XSS les plus sévères impliquent la divulgation du cookie de session de l'utilisateur, permettant à un attaquant de détourner la session utilisateur et de prendre la main sur le compte. D'autres attaques destructrices incluent la divulgation de fichiers utilisateur, l'installation de programmes de type Cheval de Troie, redirigeant l'utilisateur vers une autre page ou un autre site, et modifiant la présentation du contenu. Une vulnérabilité XSS permettant à un attaquant de modifier un communiqué de presse ou une information pourrait affecter le prix de l'action d'une société ou diminuer la confiance des consommateurs. Une vulnérabilité XSS sur un site pharmaceutique pourrait permettre à un attaquant de modifier des informations de dosage, aboutissant ainsi à un surdosage.

Les attaquants utilisent fréquemment toute une variété de méthodes pour coder la partie malveillante de « l'étiquette » (i.e. le *tag*), comme l'utilisation d'Unicode, la demande paraissant donc moins soupçonneuse du point de vue de l'utilisateur. Il existe des centaines de variantes de ces attaques, y compris des versions qui n'exigent même pas de symboles < >. Pour cette raison, essayer de « filtrer » ces scripts ne va probablement pas réussir. Au lieu de cela nous recommandons de valider l'entrée en fonction d'une spécification positive rigoureuse de ce à quoi on s'attend. Les attaques XSS se traduisent d'habitude sous la forme de Script Java embarqué. Quoi qu'il en soit, tout contenu actif embarqué est une source potentielle de danger, y compris : ActiveX (OLE), VBscript, Shockwave, Flash et autre.

Les problèmes XSS peuvent aussi être présents dans les serveurs web applicatifs sous-jacents. La plupart des serveurs web applicatifs génèrent des pages web simples à montrer en cas d'erreurs diverses, telles que 404 « page non trouvée » ou 500 « erreur interne de serveur ». Si ces pages renvoient des informations à la requête de l'utilisateur, comme l'URL à laquelle ils essayaient d'avoir accès, ils peuvent être vulnérables à une attaque XSS par réflexion.

La probabilité qu'un site contienne des vulnérabilités XSS est extrêmement élevée. Il existe de nombreuses façons pour duper les applications web dans le relayage de scripts malveillants. Les développeurs qui essayent de filtrer les parties malveillantes de ces requêtes vont très probablement laisser échapper des attaques ou des encodages possibles. La découverte de ces failles n'est pas très difficile pour les attaquants, tout ce dont ils ont besoin étant un navigateur et du

temps. Il existe de nombreux outils gratuits disponibles qui aident les pirates informatiques à trouver ces failles aussi bien qu'à soigneusement forger et injecter des attaques XSS dans un site cible.

A4.2 Environnements Affectés

Tous les serveurs web, les serveurs d'applications et les environnements applicatifs web sont sensibles au Cross Site Scripting.

A4.3 Exemples et Références

- La FAQ Cross Site Scripting: http://www.cgisecurity.com/articles/xss-faq.shtml
- Avis consultatif du CERT sur les Étiquettes HTML malveillantes: http://www.cert.org/advisories/CA-2000-02.html
- CERT « Comprendre la réduction de contente Malveillant » http://www.cert.org/tech_tips/malicious_code_mitigation.html
- Résumé de l'exposé de sécurité Cross-Site Scripting: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/ExSumCS.asp
- Comprendre la cause et l'effet des vulnérabilités CSS: http://www.technicalinfo.net/papers/CSS.html
- Guide de l'OWASP « Concevoir des Applications et Services web sûrs », Chapitre 8 : Validation de données http://www.owasp.org/documentation/guide/
- Comment construire un moteur de validation de requête HTTP (validation J2EE avec Stinger) http://www.owasp.org/columns/jeffwilliams/jeffwilliams2
- « Prenez votre gâteau et mangez-le aussi » (validation .NET) http://www.owasp.org/columns/jpoteet/jpoteet2

A4.4 Comment déterminer si vous êtes vulnérable

Les failles XSS peuvent être difficiles à identifier et à supprimer d'une application web. La meilleure façon de trouver des failles est d'exécuter un audit de sécurité du programme et de chercher touts les endroits où l'entrée d'une requête HTTP pourrait éventuellement réussir à produire du HTML. Notez qu'une variété d'étiquettes HTML différentes peut être utilisée pour transmettre un Script Java malveillant. Nessus, Nikto et quelques autres outils disponibles peuvent aider à scanner un site web pour rechercher ces failles, mais ne peuvent qu'effleurer seulement la surface. Si une partie d'un site web est vulnérable, il y a une forte probabilité qu'il y ait aussi d'autres problèmes.

A4.5 Comment vous protéger

La meilleure façon de protéger une application web des attaques XSS est de s'assurer que votre application exécute une validation de toutes les en-têtes, des cookies, des chaînes de requêtes, des formulaires de champs et des champs cachés (c'est-à-dire, tous les paramètres), en fonction d'une spécification rigoureuse de ce qui devrait être permis. La phase de validation ne devrait pas essayer d'identifier le contenu actif, mais le supprimer, le filtrer, ou l'épurer. Il y a trop de types de contenus actifs et trop de façons d'encoder cela afin de contourner les filtres pour un tel contenu. Nous recommandons fortement une politique de sécurité « positive » qui spécifie ce que l'on permet. Les politiques « négatives » ou basées sur des signatures d'attaques sont difficiles à maintenir et sont probablement susceptibles d'être incomplètes.

La fourniture d'encodage utilisateur peut aussi faire échouer les vulnérabilités XSS en empêchant les scripts insérés d'être transmis aux utilisateurs sous une forme exécutable. Les applications peuvent significativement gagner en protection contre les attaques Java Script en convertissant les caractères suivants dans tout résultat généré vers l'entité HTML d'encodage appropriée:

From:	To:
<	&lt;
>	&gt;
(	&#40;
)	&#41;

#	&#35;
&	&#38;

Le projet Filtres de l'OWASP fournit des composants réutilisables dans plusieurs langues pour aider à empêcher beaucoup de formes de falsification de paramètre, y compris l'injection d'attaques XSS. L'OWASP a aussi sorti CodeSeeker, un coupe-feu (firewall) de niveau Applicatif. De plus, le programme de formation WebGoat de l'OWASP intègre des cours sur le Cross Site Scripting et l'encodage des données.

A5 Débordement de tampon

A5.1 Description

Les attaquants utilisent les débordements de tampon pour corrompre la pile d'exécution d'une application web. En envoyant une entrée soigneusement forgée à une application web, un attaquant peut faire exécuter du code arbitraire par l'application - efficacement prendre le contrôle de la machine. Les débordements de tampon ne sont pas facile à découvrir et même quand on en découvre un, il est généralement extrêmement difficile à exploiter. Néanmoins, les attaquants ont réussi à identifier les débordements de tampon sur un panel stupéfiant de produits et des composants. Une autre classe de failles très semblable est connue sous le nom d'attaques des chaînes de format.

Les failles de débordements de tampon peuvent être présentes tant dans les serveur web que dans les logiciels de serveur applicatifs en charge des aspects statiques et dynamiques du site, ou l'application web elle-même. Les débordements de tampon trouvés dans les logiciels serveur largement utilisés vont probablement devenir largement connu et peuvent poser un risque significatif aux utilisateurs de ces produits. Quand les applications web utilisent des bibliothèques, comme une bibliothèque graphique pour produire des images, elles s'exposent à de potentielles attaques de débordements de tampon.

Les débordements de tampon peuvent aussi être trouvés dans un code d'application web personnalisé, et peuvent même être plus probablement dûs au manque d'examen minutieux dont les applications web ne font typiquement pas l'objet. Les failles de débordements de tampon dans les applications web personnalisées vont probablement être moins détectées parce qu'il y aura normalement moins de pirates informatiques qui essaieront de trouver et exploiter de tels défauts dans une application spécifique. En cas de découverte dans une application personnalisée, la capacité d'exploiter la faille (autrement que de crasher l'application) est significativement réduite par le fait que le code source et les messages d'erreur détaillés pour l'application ne sont normalement pas disponibles au pirate.

A5.2 Environnements Affectés

Presque tous les serveurs web connus, les serveurs applicatifs et les environnements d'applicatifs web sont sensibles au débordement de tampon, l'exception notable étant Java et les environnements J2EE, qui sont immunisés à ces attaques (à part les débordements dans la JVM elle-même).

A5.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services Web sûrs », Chapitre 8 : Validation de Données: http://www.owasp.org/documentation/guide/
- Aleph One, "Smashing the Stack for Fun and Profit", http://www.phrack.com/show.php?p=49&a=14
- Mark Donaldson, "Inside the Buffer Overflow Attack: Mechanism, Method, & Prevention", http://rr.sans.org/code/inside_buffer.php

A5.4 Comment déterminer si vous êtes vulnérable

Pour les produits serveur et les bibliothèques, tenez-vous informés sur les rapports de bugs les plus récents pour les logiciels que vous utilisez. Pour un logiciel applicatif personnalisé, tout code qui accepte une entrée utilisateurs via la requête HTTP doit être passée en revue pour s'assurer qu'il peut correctement manipuler une entrée arbitrairement large.

A5.5 Comment vous protéger

Tenez-vous informés sur les rapports de bugs les plus récents relatifs à vos logiciels serveurs web et applicatifs ainsi qu'aux autres produits dans votre infrastructure Internet. Appliquez les derniers patches à ces produits. Scannez périodiquement votre site web avec un (ou plus) des scanners couramment disponibles qui cherchent les failles de débordements de tampon dans vos produits serveur et vos applications web personnalisées.

Pour votre programme applicatif personnalisé, vous devez passer en revue tout code qui accepte un paramètre en entrée émanant d'utilisateurs via la requête HTTP et vous assurer que le code effectue une vérification de taille appropriée sur toutes les entrées de ce type. Cela devrait être fait même pour les environnements qui ne sont pas vulnérables à de telles attaques, sachant que les entrées excessivement grandes qui ne sont pas stoppées peuvent toujours causer un déni de service ou d'autres problèmes opérationnels.

A6 Failles d'Injection

A6.1 Description

Les failles d'injection permettent aux attaquants de retransmettre du code malveillant par une application web à un autre système. Ces attaques comprennent des appels au système d'exploitation via des appels système, l'utilisation de programmes externes via des commandes shell, aussi bien que des appels à des bases de données secondaires par SQL (i.e. Injection SQL). Tout un ensemble de scripts écrits en Perl, Python et d'autres langages peuvent être injectés et exécutés dans des applications web mal conçues. A chaque fois qu'une application web utilise un interprète de n'importe quel type, il y a danger d'une attaque par injection.

Beaucoup d'applications web utilisent des caractéristiques du système d'exploitation et des programmes externes pour exécuter leurs fonctions. Sendmail est probablement le programme externe le plus fréquemment invoqué, mais beaucoup d'autres programmes sont aussi utilisés. Quand une application web envoie une information d'une requête HTTP en tant que partie d'une requête externe, elle doit être soigneusement « nettoyée ». Autrement, l'attaquant peut injecter des (meta) caractères spéciaux, des commandes malveillantes, ou ordonner des modificateurs dans l'information et l'application web transmettra aveuglément ceux-ci au système externe pour l'exécution.

L'injection SQL est une forme d'injection particulièrement répandue et dangereuse. Pour exploiter une faille d'injection SQL, l'attaquant doit trouver un paramètre que l'application web envoie à une base de données. En intégrant soigneusement des commandes SQL malveillantes dans le contenu du paramètre, l'attaquant peut leurrer l'application web par la transmission d'une requête malveillante à la base de données. Ces attaques ne sont pas difficiles à tenter et de plus en plus d'utilitaires dédiés à la découvertes de ces failles émergent. Les conséquences sont particulièrement préjudiciables, un attaquant pouvant obtenir, corrompre, ou détruire le contenu d'une base de données.

Les attaques par injection peuvent être très faciles à découvrir et à exploiter, mais peuvent aussi être extrêmement obscures. Les conséquences peuvent aussi exécuter une gamme entière de sévérité, de la plus insignifiante à la complète compromission ou destruction du système. En tout cas, l'utilisation d'appels externes est largement répandue, la probabilité qu'une application web soit vulnérable à une faille d'injection de commande devant donc être considérée comme haute.

A6.2 Environnements Affectés

Tout environnement d'application web permettant l'exécution de commandes externes tels des appels système, des commandes shell et des requêtes SQL. La prédisposition de vulnérabilité d'un appel externe à l'injection de commande dépend de la façon dont l'appel est fait et du composant spécifique qui est appelé, mais presque tous les appels externes peuvent être attaqués si l'application web n'est pas correctement programmée.

A6.3 Exemples et Références

- Exemples : un paramètre malveillant pourrait modifier les actions prises par un appel système qui récupère normalement le fichier de l'utilisateur actuel pour avoir accès au fichier d'un autre utilisateur (par exemple, en incluant les caractères de chemin « ../ » en tant que partie d'une requête de nom de fichier). Des commandes complémentaires pourraient être ajoutées à la fin d'un paramètre qui est passé à un shell script pour exécuter une commande shell supplémentaire (e.g « ; rm-r * ») avec la commande désirée. Les requêtes SQL pourraient être modifiées en ajoutant « des contraintes » complémentaires à une clause de lieu (par exemple, « OR 1=1 ») pour obtenir l'accès ou modifier des données non autorisées.
- Guide de l'OWASP « Concevoir des Applications et Services web sûrs », Chapitre 8 : Validation de données http://www.owasp.org/documentation/guide/
- Comment construire un moteur de validation de requête HTTP (validation J2EE avec Stinger) http://www.owasp.org/columns/jeffwilliams/jeffwilliams2
- « Prenez votre gâteau et mangez-le aussi » (validation .NET) http://www.owasp.org/columns/jpoteet/jpoteet2
- Livre Blanc sur l'Injection SQL: http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf

A6.4 Comment déterminer si vous êtes vulnérable

La meilleure façon de déterminer si vous êtes vulnérables aux attaques par injection de commande est de rechercher dans le code source tous les appels aux ressources externes (e.g. système, exécutable, exécutable Runtime, requêtes SQL, ou quoi que la syntaxe soit pour faire des demandes aux interpréteurs dans votre environnement). Notez que beaucoup de langages ont des possibilités multiples d'exécuter des commandes externes. Les développeurs devraient passer leur code en revue et chercher touts les endroits où une entrée émanant d'une requête HTTP pourrait le cas échéant faire son chemin à travers n'importe lequel de ces appels. Vous devriez soigneusement examiner chacun de ces appels afin d'être sûr que les étapes de protection décrites ci-dessous sont suivies.

A6.5 Comment vous protéger

La façon la plus simple de se protéger contre l'injection est d'éviter d'avoir accès aux interpréteurs externes partout où cela est possible. Pour beaucoup de commandes shell et quelques appels système, il existe des bibliothèques spécifiques de langages qui exécutent les mêmes fonctions. L'utilisation de telles bibliothèques n'implique pas l'interpréteur shell du système d'exploitation et évite donc un grand nombre de problèmes avec les commandes shell.

Pour ces appels que vous devez encore employer, tels les appels aux bases de données secondaires, vous devez soigneusement valider les données fournies pour vous assurer qu'elles ne contiennent pas de contenu malveillant. Vous pouvez aussi structurer beaucoup de requêtes de telle façon de s'assurer que tous les paramètres fournis sont traités comme des données, plutôt que comme du contenu potentiellement exécutable. L'utilisation de procédures stockées ou d'états préparés fournira une protection significative, en s'assurant que l'entrée fournie est traitée en tant que donnée. Ces mesures réduiront, mais n'élimineront pas complètement le risque induit par ces appels externes. De plus, vous devez toujours valider une telle entrée pour vous assurer qu'elle répond aux attentes de l'application en question.

Une autre protection forte contre l'injection de commande est de s'assurer que l'application web fonctionne uniquement avec les privilèges absolument nécessaires à l'exécution de sa fonction. Vous ne devriez donc pas faire fonctionner un serveur web en tant que « root » ou accéder à une base de données en tant qu'Administrateur, un attaquant pouvant utiliser ces privilèges administratifs accordés à l'application web. Certains environnements J2EE permettent l'utilisation du bac à sable Java, qui peut empêcher l'exécution de commandes système.

Si une commande externe doit être utilisée, toute informations utilisateur qui est insérées dans la commande devrait être rigoureusement vérifiée. Les mécanismes devraient être mis en place pour manipuler toutes erreurs possibles, timeouts, ou blocages pendant l'appel.

Tout résultat, codes de retour et codes d'erreur de l'appel devraient être vérifiés pour s'assurer que le traitement attendu a véritablement eu lieu. Au minimum, cela vous permettra de déterminer que quelque chose a mal tourné. Autrement, l'attaque peut avoir lieu et ne jamais être détectée.

Le projet de Filtres de l'OWASP fournit des composants réutilisables dans plusieurs langages pour aider à empêcher beaucoup de formes d'injection. L'OWASP a aussi sorti CodeSeeker, un coupe-feu (firewall) de niveau applicatif.

A7 Traitement d'erreur incorrect

A7.1 Description

Un traitement d'erreurs incorrect peut introduire toute une variété de problèmes de sécurité pour un site web. L'affichage de messages d'erreur internes détaillés comme des traces de pile, des dumps de base de données et des codes d'erreur à l'utilisateur (au pirate informatique) constitue le problème le plus courant. Ces messages révèlent des détails d'implémentation qui ne devraient jamais être révélés. De tels détails peuvent fournir aux pirates informatiques des indices importants sur des failles potentielles dans le site et de tels messages dérangent aussi les utilisateurs normaux.

Les applications web génèrent fréquemment des conditions d'erreur pendant une opération normale. Insuffisante de mémoire, exceptions d'indicateur nul, échec d'appel système, base de données indisponible, timeout réseau et des centaines d'autres conditions courantes peuvent causer la génération d'erreurs. Ces erreurs doivent être traitées selon un processus sérieusement réfléchi qui fournira un message d'erreur significatif à l'utilisateur, des informations de diagnostics aux superviseurs du site, et aucune information utile à un attaquant.

Même quand les messages d'erreur ne fournissent pas beaucoup de détail, l'inconsistance dans de tels messages peut toujours révéler des indices importants sur la façon dont un site fonctionne, et quelle information sous-jacente est présente. Quand un utilisateur essaye, par exemple, d'avoir accès à un fichier qui n'existe pas, le message d'erreur indique typiquement « fichier non trouvé ». En accédant à un fichier sur lequel l'utilisateur n'est pas autorisé, il indique « accès refusé ». L'utilisateur n'est pas même supposé savoir que le fichier existe, mais une telle inconsistance révélera aisément la présence ou l'absence de fichiers inaccessibles ou la structure de répertoire du site.

Un problème de sécurité courant causé par un traitement d'erreur incorrect est le contrôle de sécurité qui est laissé de côté. Tous les mécanismes de sécurité devraient refuser un accès jusqu'à ce qu'il soit spécifiquement autorisé, ne pas accorder d'accès en attendant qu'il soit refusé, raison courante qui explique pourquoi la mise de côté des erreurs a lieu. D'autres erreurs peuvent causer un crash du système ou une consommation significatives de ressources, stoppant efficacement ou réduisant le service aux utilisateurs légitimes.

Les bons mécanismes de traitement d'erreurs devraient être capable de manipuler tous les type d'entrées possibles, tout en appliquant une sécurité appropriée. Des messages d'erreur simples devraient être produits et enregistrés pour que leurs causes, en cas d'erreur dans le site ou tentative de piratage, puisse être réexaminées. Le traitement d'erreur ne devrait pas uniquement se focaliser sur l'entrée fournie par l'utilisateur, mais devrait aussi prendre en compte toutes les erreurs pouvant être produites par des composants internes tels les appels système, les requêtes à des bases de données, ou autres fonctions internes.

A7.2 Environnements Affectés

Tous les serveurs web, serveurs d'application et environnements applicatifs web sont sensibles aux problèmes de traitement d'erreurs.

A7.3 Exemples et Références

- Débat de l'OWASP sur la génération de codes d'erreur: http://www.owasp.org/documentation/guide/

A7.4 Comment déterminer si vous êtes vulnérable

Typiquement, un simple test de mise à l'épreuve peut déterminer comment votre site répond aux différentes sortes d'erreurs d'entrée. Un test plus minutieux est habituellement requis pour causer des erreurs internes et voir comment le site se comporte.

Une autre - importante - approche consiste à effectuer un audit de code détaillé à la recherche d'erreur de traitement logique dans le code. Le traitement d'erreur devrait être cohérent pour l'ensemble du site et chaque composante devrait être une

partie d'un schéma bien conçu. Un audit de code révélera comment le système a l'intention de manipuler les divers types d'erreurs. Si vous constatez qu'il n'y a aucune organisation dans le plan de traitement d'erreur ou qu'il semble y avoir plusieurs schémas différents, il y a fort probablement un problème.

A7.5 Comment vous protéger

Une politique spécifique inhérente au traitement d'erreurs devrait être documentée, y compris les types d'erreurs à être traités, et pour chacun, les informations devant être présentées à l'utilisateur et celles devant être enregistrées. Tous les développeurs doivent comprendre la politique et s'assurer que leur code la respecte.

En production, assurez-vous que le site est conçu pour traiter aisément toutes les erreurs possibles. Quand des erreurs se produisent, le site devrait répondre par un résultat spécifiquement conçu qui est utile à l'utilisateur sans révéler de détails internes inutiles. Certaines classes d'erreurs devraient être enregistrées pour aider à détecter des failles de mise en oeuvre dans le site et/ou les tentatives de piratage.

Très peu de sites disposent de capacités de détection d'intrusion dans leur application web, mais il est certainement concevable qu'une application web puisse suivre à la trace des échecs de tentatives répétés et générer des alertes. Notez que la plus grande majorité des attaques d'application web n'est jamais détectée parce que bien peu de sites ont la capacité de les détecter. De fait, la fréquence des attaques de sécurité d'application web doit sérieusement être sous-estimée.

Le projet de Filtres de l'OWASP fournit des composants réutilisables dans plusieurs langages pour aider à empêcher la fuite de codes d'erreur dans les pages web de l'utilisateur par filtrage des pages quand elles sont dynamiquement construites par l'application.

A8 Stockage non sécurisé

A8.1 Description

La plupart des applications web ont besoin de stocker de l'information sensible, dans une base de données ou quelque part sur un système de fichiers. L'information peut concerner des mots de passe, des numéros de carte de crédit, des relevés de compte, ou une information de marque déposée. Fréquemment, des techniques de chiffrement sont utilisées pour protéger cette information sensible. Le chiffrement étant devenu relativement facile à mettre en oeuvre et à utiliser, les développeurs font toujours fréquemment des erreurs en l'intégrant dans une application web. Les développeurs peuvent être amenés à surestimer la protection gagnée en utilisant le chiffrement et ne pas être aussi prudents dans la sécurisation d'autres aspects du site. Quelques uns des domaines où les erreurs sont généralement faites incluent :

- Manque de chiffrement des données critiques
- Stockage peu sûr de clefs, certificats et mots de passe
- Stockage incorrect de secrets dans la mémoire
- Mauvaises sources d'incohérence
- Mauvais choix d'algorithme
- Tentative d'invention d'un nouvel algorithme de chiffrement
- L'absence de support pour le changement de la clef de chiffrement et autres procédures de maintenance requises

L'impact de ces faiblesses peut être dévastateur pour la sécurité d'un site web. Le chiffrement est généralement utilisé pour protéger les actifs les plus sensibles d'un site, qui peuvent être totalement compromis par une faiblesse

A8.2 Environnements Affectés

La plupart des environnements applicatifs web disposent d'une certaine forme de support cryptographique. Dans le cas rare où un tel support n'est pas déjà disponible, il existe une grande variété des logiciels tiers qui peuvent être implémentés. Seuls les sites web qui utilisent le chiffrement pour protéger une information archivée ou en transit sont sensibles à ces attaques. Notez que cette section ne couvre pas l'utilisation de SSL, sujet couvert dans la section A10 « Gestion de configuration non sécurisée ». Cette section traite seulement de la programmation de chiffrement de donnée de la couche Application.

A8.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services web sûrs » http://www.owasp.org/documentation/guide/
- Bruce Schneier, « Cryptographie Appliquée », 2ème édition, John Wiley & Sons, 1995

A8.4 Comment déterminer si vous êtes vulnérable

La découverte de failles cryptographiques sans accéder au code source peut être extrêmement longue. Il est toutefois possible d'examiner les jetons de sessions, les IDs de session, les cookies et droits pour voir s'ils ne sont pas clairement aléatoires. Toutes les approches traditionnelles de cryptanalyse peuvent être utilisées pour essayer de découvrir comment un site web utilise les fonctions cryptographiques.

L'approche la plus facile, et de loin, est de passer le code en revue pour voir comment les fonctions cryptographiques sont mises en oeuvre. Un examen minutieux de la structure, de la qualité et de la mise en oeuvre des modules cryptographiques devrait être exécuté. L'auditeur devrait avoir une forte expérience dans l'utilisation de la cryptographie et des failles courantes. L'audit devrait aussi couvrir la façon dont les clefs, les mots de passe et autres secrets sont stockés, protégés, chargés, traités et vidés de la mémoire.

A8.5 Comment vous protéger

La façon la plus facile de se protéger contre les failles cryptographiques est de réduire au minimum l'utilisation du chiffrement et de ne seulement garder que l'information qui est absolument nécessaire. Par exemple, plutôt que de chiffrer des numéros de carte de crédit et les stocker, exigez simplement que les utilisateurs ressaisissent les numéros. De plus, au lieu de stocker des mots de passe chiffrés, utilisez une fonction à sens unique, comme SHA-1, pour « hasher » les mots de passe.

Si la cryptographie doit être utilisée, choisissez une bibliothèque qui a été soumise à un examen public minutieux et assurez-vous qu'il n'y a aucune vulnérabilité en cours. Encapsulez les fonctions cryptographiques qui sont utilisées et passez soigneusement le code en revue. Assurez-vous que les secrets, comme les clefs, les certificats et les mots de passe sont stockés de façon « secure ». Pour rendre la tâche plus difficile à un attaquant, le « secret maître » (i.e. *Master Secret*) devrait être réparti dans au moins deux emplacements et assemblé seulement lors de la requête d'exécution. Un fichier de configuration, un serveur externe, ou dans le code lui-même pourraient constituer de tels emplacements.

A9 Déni de Service

A9.1 Description

Les applications web sont particulièrement sensibles aux attaques de déni de service. A noter que les attaques de déni de service réseau, comme « *SYN floods* » (littéralement « *inondation de paquets SYN* »), est un problème particulier hors scope de ce document.

Une application web ne peut facilement faire la différence entre une attaque et un trafic ordinaire. Beaucoup de facteurs contribuent à cette difficulté, mais un des plus importants est que, pour un certain nombre de raisons, les adresses IP ne sont pas aussi utiles qu'une référence d'identification. Parce qu'il n'y a aucune façon fiable de déterminer d'où émane une requête HTTP, il est très difficile de filtrer le trafic malveillant. Pour les attaques distribuées, comment une application ferait-elle la différence entre une vraie attaque, de multiples utilisateurs effectuant un « reload » tous en même temps (ce qui pourrait arriver s'il y a un problème temporaire avec le site), ou le fait de se faire littéralement réduire au silence?

La plupart des serveurs web peuvent traiter plusieurs centaines d'utilisateurs simultanés en utilisation normale. Un seul attaquant peut générer assez de trafic à partir d'un hôte unique pour saturer (i.e. submerger) beaucoup d'applications. La répartition de charge peut rendre ces attaques plus difficiles, mais loin de les rendre impossibles, particulièrement si les sessions sont liées à un serveur particulier. C'est une bonne raison pour que la donnée de session d'une application soit aussi réduite que possible et de faire en sorte de lui rendre le démarrage d'une nouvelle session quelque peu difficile.

Une fois qu'un attaquant peut consommer la totalité des ressources requises, il peut empêcher les utilisateurs légitimes d'utiliser le système. Quelques ressources qui sont limitées comprennent la bande passante, les connexions aux bases de données, le stockage disque, la CPU, la mémoire, des threads, ou des ressources applicatives spécifiques. Toutes ces ressources peuvent être consommées par des attaques qui les ciblent. Par exemple, un site qui permet aux utilisateurs non authentifiés de demander le panneau d'affichage de trafic peut initier beaucoup de requêtes de base de données pour chaque demande HTTP qu'ils reçoivent. Un attaquant peut facilement envoyer autant de requêtes qu'il faut pour que le pool de connexion à la base de données soit saturé et qu'aucune ressource ne soit disponible pour le service aux utilisateurs légitimes.

D'autres variantes de ces attaques ciblent les ressources système liées à un utilisateur particulier. Par exemple, un attaquant pourrait être capable d'empêcher un utilisateur légitime d'entrer en envoyant des accréditations invalides jusqu'à ce que le système verrouille le compte. Ou l'attaquant pourrait demander un nouveau mot de passe pour un utilisateur, le forçant à avoir accès à son compte e-mail pour regagner l'accès. Alternativement, si le système verrouille des ressources pour un utilisateur unique, un attaquant pourrait donc potentiellement les occuper afin que d'autres ne puissent les utiliser.

Quelques applications web sont même sensibles à des attaques qui les rendront immédiatement off-line. Les applications qui ne traitent pas correctement les erreurs peuvent même descendre le conteneur d'application web. Ces attaques sont particulièrement dévastatrices parce qu'elles empêchent immédiatement tous les autres utilisateurs d'utiliser la l'application.

Une grande variété de ces attaques existe, dont la plupart peuvent être facilement lancées avec quelques lignes de code Perl à partir d'un ordinateur peu puissant. Sachant qu'il n'existe pas de défenses parfaites à ces attaques, il est possible de leur rendre la tâche plus difficile.

A9.2 Environnements Affectés

Tous les serveurs web, les serveurs d'application et les environnements applicatifs web sont sensibles aux attaques de déni de service.

A9.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services web sûrs »
  http://www.owasp.org/documentation/guide/

A9.4 Comment déterminer si vous êtes vulnérable

Une des parties les plus dures des attaques de déni de service est de déterminer si vous y êtes vulnérables. Les outils de test de charge, comme *Jmeter* , peuvent générer du trafic web pour que vous puissiez évaluer certains aspects de la façon dont votre site fonctionne en cas de forte charge. Un test assurément important consiste à voir combien de requêtes par seconde votre application peut gérer. Effectuer le test à partir d'une adresse IP unique est utile en ce sens où cela vous donnera une idée du nombre de requêtes qu'un attaquant devra produire pour endommager votre site.

Pour déterminer si des ressources peuvent être utilisées pour créer un déni de service, vous devriez analyser chacune d'entre elles pour voir s'il existe un moyen de les épuiser. Vous devriez particulièrement vous concentrer sur ce qu'un utilisateur non authentifié peut faire, mais à moins que vous ayez confiance en tous vos utilisateurs, vous devriez aussi examiner ce qu'un utilisateur authentifié peut faire.

A9.5 Comment vous protéger

Se défendre contre les attaques de déni de service est difficile, aucune façon de se protéger parfaitement contre ces attaques n'existant.

En règle générale, vous devriez limiter les ressources allouées à chaque utilisateur à un strict minimum. Pour les utilisateurs authentifiés, il est possible d'établir des quotas pour que vous puissiez limiter la quantité de charge qu'un utilisateur particulier peut soumettre à votre système. En particulier, vous pourriez envisager de ne traiter seulement qu'une requête par utilisateur à la fois en synchronisant sur la session de l'utilisateur. Vous pourriez aussi songer à supprimer toutes requêtes que vous traitez actuellement pour un utilisateur quand une autre demande de cet utilisateur arrive.

Pour les utilisateurs non authentifiés, vous devriez éviter tout accès inutile aux bases de données ou autres ressources coûteuses. Essayez d'architecturer le flux de votre site pour qu'un utilisateur non authentifié ne soit pas capable d'invoquer des opérations onéreuses. Vous pourriez considérer de mettre le contenu reçu par les utilisateurs non authentifiés en cache au lieu de le générer ou d'accéder aux bases de données pour le récupérer.

Vous devriez aussi vérifier votre schéma de traitement d'erreur pour vous assurer qu'une erreur ne puisse pas affecter l'opération globale de l'application.

A10 Gestion de configuration non sécurisée

A10.1 Description

Les configurations de serveur web et de serveur applicatif jouent un rôle clef dans la sécurité d'une application web. Ces serveurs sont chargés de fournir du contenu et de faire appel aux applications qui produisent du contenu. De plus, beaucoup de serveurs d'application fournissent un certain nombre de services que les applications web peuvent utiliser, y compris le stockage de données, des services de renseignements, le courrier, la messagerie et plus. Un défaut de gestion de la configuration appropriée de vos serveurs peut mener à une grande variété de problèmes de sécurité.

Fréquemment, l'équipe de développement web est séparée de l'équipe faisant fonctionner le site. En fait, il y a souvent un grand vide entre ceux qui écrivent l'application et ceux qui sont responsables de l'environnement opérationnel. Les inquiétudes inhérentes à la sécurité d'application web comblent souvent ce vide et requièrent que les membres des deux équipes précitées assurent correctement la sécurité de l'application d'un site.

Il existe une grande variété de problèmes de configuration serveur qui peuvent mettre à mal la sécurité d'un site. Ceux-ci incluent :

- Les failles de sécurité non patchées dans le logiciel de serveur
- Les failles du logiciel serveur ou les mauvaises configurations qui permettent de lister les répertoires et les attaques dites « Directory Traversal » (i.e. passer de répertoire en répertoire)
- Fichiers par défaut, sauvegardes, ou fichiers d'exemples inutiles, y compris les scripts, applications, fichiers de configuration et pages web
- Fichier incorrect et permissions de répertoires
- Services inutiles activés, y compris la gestion de contenu et l'administration à distance
- Les comptes par défaut avec leurs mots de passe par défaut
- Les fonctions administratives ou de débogage qui sont activées ou accessible
- Messages d'erreur trop instructifs (plus de détails dans la section « Traitement d'erreur »)
- Mauvais paramétrages de certificats SSL et de chiffrement
- L'utilisation de certificats auto-signés pour parvenir à l'authentification et à la protection contre « man-in-the-middle »
- L'utilisation de certificats par défaut
- Authentification incorrecte avec des systèmes externes

Certains de ces problèmes peuvent être détectés avec des outils de scanning de sécurité aisément disponibles. Une fois détectés, ces problèmes peuvent être facilement exploités, la compromission totale d'un site web en résultant alors. Des attaques réussies peuvent aussi aboutir à la compromission de systèmes secondaires incluant des bases de données et des réseaux d'entreprise. La disponibilité d'un logiciel et d'une configuration sécurisés est un pré-requis pour avoir un site sécurisé.

A10.2 Environnements Affectés

Tous les serveurs web, serveurs d'applicatifs et les environnements d'application web sont sensibles à une mauvaise configuration.

A10.3 Exemples et Références

- Guide de l'OWASP « Concevoir des Applications et Services web sûrs »
  http://www.owasp.org/documentation/guide/

- Best Practices de Sécurité des serveurs web: http://www.pcmag.com/article2/0,4149,11525,00.asp
- Sécuriser les Serveurs Web Publics (du CERT): http://www.cert.org/security-improvement/modules/m11.html

A10.4 Comment déterminer si vous êtes vulnérable

Si vous n'avez pas fait d'effort concerté pour verrouiller vos serveurs web et applicatifs vous êtes plus que probablement vulnérables. Peu de logiciels serveur, s'il en est, sont sûrs d'ordinaire. Une configuration sûre pour votre plate-forme devrait être documentée et mise à jour fréquemment. Un examen manuel du guide de configuration devrait être exécuté régulièrement pour s'assurer que cela a été tenu à jour et est cohérent. Une comparaison aux systèmes actuellement déployés est aussi recommandée.

De plus, il existe un certain nombre de produits de scanning disponibles qui scanneront extérieurement un serveur web ou applicatif à la recherche de vulnérabilités connues, y compris *Nessus* et *Nikto*. Vous devriez utiliser ces outils régulièrement, au moins mensuellement, pour trouver les problèmes aussitôt que possible. Les outils devraient être utilisés tant en interne qu'en externe. Les scans externes devraient être utilisés à partir d'un hôte placé en externe au réseau du serveur. Les scans internes devraient être utilisés à partir du même réseau que les serveurs cibles.

A10.5 Comment vous protéger

La première étape est de créer un « guideline » (une note) de durcissement pour votre configuration de serveur web et applicative particulière. Cette configuration devrait être utilisée sur tous les hôtes sur lesquelles fonctionne l'application, aussi bien que dans l'environnement de développement. Nous vous recommandons de commencer par n'importe quels conseils existants que vous pouvez trouver auprès de votre vendeur ou ceux disponibles auprès des diverses organisations de sécurité existantes comme l'OWASP, le CERT et le SANS et de les adapter ensuite à vos besoins particuliers. La note de durcissement devrait inclure les sujets suivants :

- Configuration de tous les mécanismes de sécurité
- Désactivation de tous les services inutilisés
- Paramétrage des rôles, des permissions et des comptes, y compris la mise hors de service de tous comptes par défaut ou le changement de leurs mots de passe
- Enregistrement et alertes

Une fois que votre note a été établie, utilisez-la pour configurer et maintenir vos serveurs. Si vous avez un grand nombre de serveurs à configurer, envisagez un processus de configuration semi-automatique ou complètement automatique. Utilisez un outil de configuration existant ou développez le vôtre. Un certain nombre de tels outils existent déjà. Vous pouvez aussi utiliser des outils de réplication de disque comme *Ghost* pour faire une image d'un serveur renforcé existant et reproduire ensuite cette image à de nouveaux serveurs. Un tel processus peut ou ne pas pouvoir fonctionner en fonction de la particularité de votre environnement.

Tenir une configuration serveur sécurisée exige de la vigilance. Vous devriez être sûrs que la responsabilité du maintien de la configuration serveur à jour est assignée à une personne ou à une équipe. Le processus de maintenance devrait inclure :

- Le contrôle des dernières vulnérabilités de sécurité publiées
- L'application des derniers patchs de sécurité
- La mise à jour de la note de configuration de sécurité
- Des scans de vulnérabilité réguliers, tant internes qu'externes
- Des examens internes réguliers de la configuration sécurité du serveur en comparaison à votre guide de configuration
- Un rapports d'état réguliers pour la Direction documentant le niveau de sécurité global

Conclusions

L'OWASP a assemblé cette liste pour susciter une prise de conscience plus accrue de la sécurité inhérente aux applications web. Les experts de l'OWASP ont conclu que ces vulnérabilités représentent un risque sérieux aux agences et sociétés qui ont orienté leur logique commerciale à l'Internet. Les problèmes de sécurité applicatifs web sont tout aussi sérieux que les problèmes de sécurité réseau, bien qu'elles aient traditionnellement reçu considérablement moins d'attention. Les attaquants ont commencé à se focaliser sur les problèmes de sécurité applicatifs web et développent activement des outils et des techniques pour les détecter et les exploiter.

Cette liste Top 10 est seulement un point de départ. Nous croyons que ces failles représentent les risques de sécurité applicatifs web les plus sérieux, mais il y a beaucoup d'autre secteurs critiques de sécurité qui ont été considérés pour la liste et qui représentent aussi un risque significatif pour les entreprises déployant des applications web. Ceux-ci incluent des failles dans les domaines de:

- Code Inutile et Malveillant
- Violation de Sécurité de Thread et Programmation Simultanée
- Recensement d'information non autorisé
- Problèmes de Responsabilité et Enregistrement Faible
- Corruption de Données
- Violation de Cache, Assemblage et Réutilisation

Vos réactions sur cette liste Top 10 sont les bienvenues. Merci de prendre part à la liste de diffusion de l'OWASP et d'aider à améliorer la sécurité des applications web. Visitez http://www.owasp.org pour commencer.