ROYAUME DU MAROC

ADMINISTRATION DE LA DÉFENSE NATIONALE

DIRECTION GÉNÉRALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

GUIDE DE SÉCURITÉ DES APPLICATIONS WEB

INFORMATIONS

AVERTISSEMENT
Destiné à vous assister dans l'adoption d'une démarche cohérente et homogène pour la mise en conformité de la sécurité de vos systèmes d'information avec les règles de sécurité édictées par la Directive Nationale de la Sécurité des Systèmes d'information (DNSSI), ce guide élaboré par la DGSSI présente les bonnes pratiques de sécurité des sites web. Il est destiné à évoluer avec les usages, mais aussi avec vos contributions et retours d'expérience. Les recommandations citées dans ce guide sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, la DGSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par la DGSSI doit être soumise, au préalable, à la validation du Responsable de la Sécurité des Systèmes d'Information (RSSI) et de l'administrateur du système concerné.

PERSONNES AYANT CONTRIBUÉ À LA RÉDACTION DE CE DOCUMENT :

Rédigé par	Version	Date
DGSSI	1.0	12/12/2014

ÉVOLUTION DU DOCUMENT :

Version	Date	Nature des modifications
1.0	12/12/2014	Version initiale

PUBLIC CONCERNÉ PAR CE DOCUMENT :

RSSI
Administrateur systèmes et réseaux
Développeur des sites web

POUR TOUTE REMARQUE :

Contact	Email
DGSSI	contact@dgssi.gov.ma

Table des matières

1 Introduction

Avec le développement de l'Internet au Maroc, les administrations publiques cherchent de plus en plus d'avoir une présence sur l'Internet via des sites web ou des applications web offrant des services aux citoyens ou aux tierces entités. Cependant, les vulnérabilités de ces applications Web sont désormais le vecteur le plus important des attaques dirigées contre la sécurité des systèmes d'information de ces administrations. En effet, D'après les différents rapports publiés cette année par les observatoires et sociétés de sécurité informatiques, les attaques web sont en constante augmentation. Les conséquences peuvent être très lourdes pour les administrations victimes de cette situation :

• Atteinte à l'image de l'administration,
• Une défiguration du site pour relayer un message politique (hacktivisme), pour dénigrer ou pour revendiquer son attaque,
• Mise en danger de l'intégrité du système d'information,
• Exfiltration des données et d'information sensibles.

A cet effet, nous ne pouvons plus nous permettre de tolérer les problèmes les plus simples comme ceux présentés dans le Top 10 OWASP [1] qui sont dus principalement à un développement et un déploiement non sécurisé. Ainsi, la mise en place de méthodes et d'outils pour gérer le développement et le contrôle qualité des applications s'avère plus que nécessaire pour réduire leur vulnérabilité.

Dans ce cadre, le présent guide se propose d'aider les responsables de la sécurité des systèmes d'information, à travers la présentation des règles de sécurité devant être respectées pendant les différentes phases du cycle de vie d'une application, à mieux sécuriser leurs applications web. Ainsi, le présent document est organisé en quatre parties :
• La première partie présente les recommandations de base à respecter, notamment les clauses de sécurité à intégrer dans le cahier des spécifications spéciales ainsi que la formation ;
• La deuxième partie est consacrée aux meilleures pratiques permettant d'éviter les failles les plus connus dans le développement des applications web.
• Enfin, La troisième partie porte sur les bonnes pratiques à respecter lors du déploiement et la mise en production d'une application web. Aussi, elle explique le processus de détection des incidents ainsi que la conduite à tenir au cas où un incident se produit.

1. Open Web Application Security Project est une communauté publique permettant à des organismes de développer, acheter et maintenir des applications fiables. A l'OWASP, vous trouverez en accès libre et gratuit sur www.owasp.org/

2 Recommandations de base

2.1 Cahier des Prescriptions Spéciales (CPS)

Lors de l'élaboration du CPS, les besoins et les objectifs de sécurité doivent être clairement identifiés. Ces besoins et objectifs doivent être spécifiés et exigés dans les clauses de sécurité se rapportant aux quatre phases d'exécution du projet : analyse des besoins, conception, développement et recette.

2.1.1 Phase d'analyse des besoins

Le maitre d'œuvre (MO) et le maitre d'ouvrage (MOA) doivent définir conjointement les besoins de l'application en matière de sécurité conformément au processus ci-après :
• **Evaluation du risque :** identifier et documenter les risques auxquels l'application pourrait être confrontée, que ce soit sur les biens (infrastructures informatiques, données, etc.) ou sur les fonctions importantes fournies par cette application. A cet effet, chacun des sujets énumérés dans la section des exigences ci-après devrait être considéré.
• **Définition des exigences de sécurité :** les exigences de sécurité sont à élaborer en fonction des spécifications de l'application web à développer. Chacun des sujets suivant devrait être discuté et évalué par le développeur et le client (propriétaire métier de l'application) afin de comprendre les risques et définir les exigences de sécurité appropriés :
- Gestion des entrées et sorties ;
- Authentification et gestion de session ;
- Contrôle d'accès ;
- Gestion d'erreur ;
- Journalisation ;
- Connexions aux systèmes externes ;
- Chiffrement ;
- Disponibilité.
- Vulnérabilités spécifiques : Les exigences devront inclure un ensemble de vulnérabilités basiques qui doivent impérativement être évitées. Il s'agit principalement des défaillances décrites dans la liste « OWASP Top Ten Most Critical Web Application Vulnérabilités ».
Les exigences de sécurité de chacun des sujets énumérés en-dessus seront développées dans les parties qui suivent de ce document.

2.1.2 Phase de conception

Une fois les besoins de sécurité et les menaces identifiés, il convient de concevoir la sécurité de l'application Web, c'est-à-dire de définir précisément les mécanismes de sécurité qui sont en mesure de répondre aux exigences fixées. Un document de conception doit décrire formellement ces mécanismes, en fournissant une bonne visibilité sur la manière dont les menaces seront gérées. La conception devrait clairement préciser les versions des logiciels et la plateforme utilisée ainsi que tous composants de tierce partie.

2.1.3 Phase de développement

Le développeur convient de fournir et de suivre un ensemble de lignes directrices de codage sécurisé et d'utiliser un ensemble de bibliothèques de sécurité. L'équipe de développement peut se référer au guide de conception et d'implémentation d'applications Web sécurisées de l'OWASP.

2.1.4 Phase de la recette

Cette phase vise à assurer que l'application est conforme aux exigences définies pendant l'étape d'analyse des besoins. Elle implique le déroulement rigoureux de procédures de tests et d'évaluation préalablement décrits. Elle doit aussi inclure une évaluation des vulnérabilités et tests de pénétration, et une analyse du code source conformément aux exigences de vérification d'une norme arrêté d'avance (par exemple la norme OWASP ASVS).
Le développeur doit fournir le code source, les différentes composantes de l'application (les librairies utilisées, la version du CMS, ...). Ces différentes composantes doivent être livrées à leur dernière version.
L'étape d'élaboration du cahier de charges est primordiale, il est recommandé de suivre le document d'OWASP spécifiant les points à préciser lors de l'élaboration du cahier de charge :
https ://www.owasp.org/index.php/File :OWASP_Secure_Software_Contract_Annex-FR.doc

2.2 Formation et sensibilisation

Tous les membres de l'équipe-projet doivent être d'une part, sensibilisés aux enjeux et risques de sécurité et d'autre part, formés aux mécanismes de sécurité de base :
• La maîtrise d'ouvrage doit être en mesure d'identifier les enjeux de sécurité pour exprimer les besoins ;
• Le chef de projet doit connaitre les normes de sécurité à respecter lors du déploiement ;
• Les développeurs doivent être en mesure d'implémenter les règles de sécurité générales ainsi que celles spécifiques aux technologies Web utilisés ;
• En cas de développement externe, le prestataire doit justifier le fait qu'il dispose

des compétences en développement sécurisé pour répondre convenablement au besoin.
Malgré le respect des bonnes pratiques lors du développement des applications web, de nouveaux types de vulnérabilités peuvent apparaitre. Il est donc important d'assurer une veille de sécurité pour remédier à ces vulnérabilités et éventuellement revoir les mécanismes de sécurité initialement mis en place.

3 Recommandations au cours de développement

Cette partie liste les meilleures pratiques permettant de sensibiliser et aider les équipes de développement à créer des applications plus sécurisées. C'est une première étape vers la construction d'une base de connaissances autour de la sécurité des applications web. Cette liste permet d'identifier les règles minimales nécessaires pour neutraliser les failles dans les applications les plus critiques.

3.1 Authentification et gestion de session

Les fonctions applicatives relatives à l'authentification et à la gestion de session ne sont souvent pas correctement mises en œuvre. Ceci permet aux attaquants de compromettre les mots de passe ainsi que les clés et les jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs.

3.1.1 Authentification

R 1	Ne pas coder en dur les identifiants c.-à-d. ne jamais laisser les informations d'identification stockées directement dans le code de l'application.

R 2	Développer un système de réinitialisation des mots de passe fort. Ces systèmes sont en général basés sur les réponses données par les utilisateurs suite à des questions personnelles et qui permettent de deviner leurs identités et réinitialiser leurs mots de passe. Le système doit être basé sur des questions qui sont à la fois difficiles à deviner et à casser. En outre, une option de réinitialisation de mots de passe ne doit pas révéler si un compte est valide ou pas afin d'empêcher l'énumération des noms des utilisateurs.

R 3	Gérer correctement les messages d'erreurs d'authentification afin de prévenir l'énumération des utilisateurs. A titre d'exemple, les messages d'erreurs qui révèlent que l'identifiant de l'utilisateur est valide et que le mot de passe est incorrect confirme que ce compte existe sur le système.

R 4	Favoriser l'utilisation d'un annuaire LDAP.

R 5	Mettre en oeuvre une politique complexe de mot de passe : – La longueur minimale d'un mot de passe doit être de 12 caractères au minimum ; – Les types de caractères qui doivent être utilisés dans un mot de passe sont : lettre (majuscule et minuscule), chiffre, caractère spécial ; – Le mot de passe ne doit pas comporter le login ; – Le nouveau mot de passe doit comporter au moins 4 différentes lettres par rapport à l'ancien mot de passe ; – Procéder au changement régulier des mots de passe ; – Une fois un mot de passe est expiré, cinq changements sont requis avant la réutilisation de ce mot de passe ; – La communication des mots de passe aux utilisateurs doit se faire d'une manière sécurisée. Il est recommandé de ne pas utiliser à cette fin un courrier électronique non protégé ; – Les mots de passe par défaut des constructeurs et des éditeurs doivent être modifiés après l'installation ; – Les comptes par défauts doivent être renommés ou désactivés.

R 6	Implémenter des règles contre les attaques de "Brute Force" : – En cas d'utilisation de LDAP, il est recommandé de verrouiller pendant une période déterminée tout compte utilisateur qui subit plus de trois tentatives d'authentification erronées. Cette période ne doit pas dépasser 5 min afin d'éviter une attaque par dénis de service. – Dans le cas le contraire (LDAP non utilisé) : il faut implémenter un CAPTCHA.

R 7	Envisager pour des applications sensibles, l'utilisation d'authentification forte.

3.1.2 Gestion des sessions

Une session est une connexion, limitée dans le temps, entre un client et un serveur. Elle est définie par un jeton de session (ou identifiant de session) qui est généré aléatoirement et souvent enregistré du côté client dans un cookie. Dans ce cadre, il faut :

R 8	S'assurer que les identificateurs de session sont suffisamment "aléatoires" : Les jetons de session doivent être générés par des fonctions aléatoires sécurisées et avoir une longueur suffisamment longue pour résister à l'analyse et à la prévision (entropie de 128 bits minimum).

R 9	Régénérer les jetons de session : Les jetons de session doivent être régénérés chaque fois que l'utilisateur s'authentifie auprès d'une application ou change de niveau de privilège. De même, une fois le statut de cryptage change, le jeton de session doit obligatoirement être régénéré.

R 10	Implémenter un délai d'attente de session inactive : Quand un utilisateur est inactif, l'application doit le déconnecter automatiquement. Les applications "Ajax" peuvent faire des appels récurrents à l'application qui réinitialise automatiquement le compteur d'inactivité.

R 11	Implémenter un temps d'expiration de la session : Après une période de connexion relativement longue (de 4 à 8 heures), les utilisateurs doivent être déconnectés. Ceci contribue à atténuer le risque qu'un attaquant utilise une session détournée pendant une longue durée.

R 12	Détruire la session si un signe d'altération est détecté : Si l'application nécessite plusieurs sessions simultanées pour un seul utilisateur, il est nécessaire d'implémenter des fonctionnalités pour détecter les tentatives de clonage de session. Une fois un signe de clonage est détecté, la session doit être détruite afin d'obliger l'utilisateur à s'authentifier de nouveau.

R 13	Rendre la session invalide après la déconnexion : Lorsque l'utilisateur se déconnecte de l'application, la session et les données correspondantes sur le serveur doivent être détruites. Ceci garantit que la session ne soit accidentellement rétablie.

R 14	Placer un bouton de déconnexion sur chaque page : Le bouton ou le lien de déconnexion doit être facilement accessible sur chaque page après l'authentification.

R 15	Utiliser des attributs de cookie sécurisés (les flags httponly et secure) : Le cookie de session doit être défini à la fois avec les flags HttpOnly et Secure. Ceci garantit que l'identifiant de session ne soit accessible aux scripts côté client et soit transmis via SSL respectivement.

R 16	Définir correctement le domaine et le chemin du cookie : Le domaine du cookie et l'étendue du chemin doivent être réglés sur les paramètres les plus restrictifs de votre application.

R 17	Définir le délai d'expiration du cookie : Le cookie de session doit avoir une date d'expiration raisonnable si non, ces cookies doivent être évités.

3.2 Protection des données

Dans beaucoup d'applications web, les données sensibles telles que les identifiants et les informations d'authentification ne sont pas correctement protégées. Ces données sensibles doivent être chiffrées en local et en transit. En général les précautions ci-après doivent être prises :

3.2.1 Protection des données locales

R 18	Stocker les mots de passe des utilisateurs avec une fonction de hash forte, itérative et salée : Les mots de passe utilisateurs doivent être stockés en utilisant des techniques de hachage sécurisées tel que l'algorithme SHA-256. En plus, plusieurs itérations de hachage et un sel (salt) aléatoire sont à implémenter afin de protéger efficacement un mot de passe.

R 19	Mettre en place des processus sécurisés de gestion des clés de cryptage et des certificats : Les clés doivent être correctement sécurisées lorsqu'elles sont stockées dans le système et ne doivent être accessibles qu'aux personnes habilitées.

R 20	Désactiver la mise en cache des données avec « des en-têtes de contrôle de cache » et le paramètre « autocomplete » : La mise en cache des données du navigateur doit être désactivée en utilisant les en-têtes de contrôle de cache HTTP ou les balises méta dans la page HTML. En outre, les champs de saisie sensibles (ex. formulaire de connexion) doivent avoir comme configuration dans le formulaire HTML « autocomplete = off » pour obliger le navigateur à ne pas mettre en cache les informations d'identification.

R 21	Limiter l'utilisation et le stockage des données d'authentification : Procéder à une évaluation afin de s'assurer que les informations d'authentifications ne sont pas inutilement transportées ou stockées. Si possible, utiliser la tokénisation (Un concept de cryptage qui permet de séparer l'information du code qui la recouvre en le remplaçant par un jeton) pour réduire les risques d'exposition de données.

R 22	Affiner les droits d'accès de l'application web au serveur de base de données.

3.2.2 Protection des données en transit

La sécurisation des données sensibles durant leur transport est très importante. A cet effet, il faut :

R 23	Utiliser le protocole SSL : Le protocole SSL doit être utilisé à tous les niveaux de l'application. En cas de contraintes ou de limitation d'utilisation de ce protocole, il doit être appliqué à toutes les pages d'authentification ainsi que toutes les pages une fois l'utilisateur est authentifié.

R 24	Désactiver l'accès HTTP pour toutes les ressources SSL activées : Pour toutes les pages nécessitant une protection par le protocole SSL, leurs URLs ne doivent pas être accessibles via un canal non-SSL.

R 25	Utiliser l'entête ″Strict-Transport-Security″ : L'entête ″Strict-Transport-Security″ garantit que le navigateur ne communique pas avec le serveur via un canal non-SSL. Ceci permet de réduire le risque des attaques de type ″SSL stripping″ mises en œuvre par l'outil ″ sslsniff ″.

R 26	Echanger d'une manière sécurisée les clés de chiffrement : Si ces clés sont échangées ou prédéfinies au niveau de l'application, leur échange doit emprunter un canal sécurisé.

R 27	Désactiver au niveau des serveurs les protocoles de chiffrements SSL faibles.

R 28	Utiliser des certificats SSL validés par une autorité de certification digne de confiance et reconnue par le navigateur. Le nom sur le certificat doit correspondre au nom complet du domaine du site et la date d'expiration du certificat doit être encore valide.

3.3 Gestion des entrées et sorties

La validation des paramètres en « entrée » et en « sortie » requiert une importance capitale. L'équipe de développement doit faire très attention à ce niveau afin d'éviter des problèmes de sécurité non négligeables au niveau applicatif. A ce propos, les attaques les plus répandues et les plus dangereuses sont les attaques de type ″XSS″ et ″injections SQL″. Ces deux types d'attaques résident parmi les tops ″10 d'OWASP″.

En général, il est recommandé de créer un mécanisme centralisé de validation des entrées en tant que partie intégrante des applications web. Pour pallier aux problèmes précités, il est important de :

R 29	Préférer les listes blanches aux listes noires : Pour chaque champ de saisie, il faut appliquer un processus de validation de contenu. Le choix d'une liste blanche est privilégié. Le principe consiste à n'accepter que les données qui répondent à un certain nombre de critères. Parfois et pour plus de flexibilité, une liste noire est utilisée. Dans ce cas, il faut bloquer les mauvaises entrées de patterns et certains caractères utilisés dans les attaques d'injection.

R 30	Valider la source de l'entrée : La source de l'entrée doit être validée. Par exemple, si l'entrée est prévue à partir d'une requête POST, il ne faut pas accepter la variable d'entrée à partir d'une requête GET.

R 31	Utiliser des requêtes SQL paramétrées : Les requêtes SQL doivent être conçues de façon à véhiculer les entrées des utilisateurs à travers des variables de liaison afin de se prémunir contre les attaques de type ”injection SQL”. Il faut aussi interdire les requêtes SQL qui peuvent être créées d'une manière dynamique en utilisant la concaténation de chaînes de caractères. De même, la chaîne de requête SQL utilisée, liée ou paramétrée, ne doit jamais être construite de façon dynamique à partir des entrées d'un utilisateur.

R 32	Utiliser des jetons pour empêcher les attaques de type ”Cross-Site Request Forgery” (CSRF) : Il s'agit d'ajouter un paramètre obligatoire correspondant à un identifiant d'accès unique et non prédictible, régénéré pour chaque action utilisateur et par utilisateur (le plus efficace), ou pour chaque session (le moins efficace). Cet identifiant d'accès (ou “laisser-passer”) est nommé “jeton CSRF”. Ce mécanisme doit être implémenté sur chaque action qui dépend de l'utilisateur qui l'exécute (déconnexion, modification/suppression de données en BDD, etc.) et doit avoir une durée de validité limitée dans le temps.

R 33	Définir l'encodage de l'application : Pour chaque page de l'application, il faut définir l'encodage des en-têtes HTTP ou des méta-tags à l'intérieur de l'HTML. Le navigateur n'aura pas besoin de déterminer le type d'encodage. Un paramétrage cohérent d'un codage, comme UTF-8, permet de réduire les attaques de type ”Cross-Site Scripting”.

R 34	Utiliser la Politique de Sécurité de Contenu (CSP) ou les en-têtes de protection ”X-XSS” : ces deux techniques aident à se protéger contre les attaques réfléchies ”cross-site scripting” (XSS).

R 35	Effectuer l'encodage contextuel des données produites (à la sortie) : Toutes les fonctions de sortie doivent contextuellement encoder les données produites avant de les présenter à l'utilisateur. La présentation des données produites « sortie » doit être codée différemment selon l'emplacement sortie dans la page HTML. Par exemple dans une page HTML, les données placées dans le contexte de l'URL doivent être encodées différemment de ceux placées dans le contexte ”JavaScript”.

R 36	Valider les fichiers soumis à l'application : Avant d'accepter une action ”upload” des fichiers d'un utilisateur, il faut vérifier la taille, le type, le contenu et le chemin de destination du fichier.

R 37	Utiliser l'en-tête de réponse HTTP ”X-Frame-Options” qui demande au navigateur de ne pas permettre l'utilisation de ‘ Frame ‘ d'autres domaines. Ceci pour empêcher le contenu d'être chargé par un site étranger dans une frame. Ceci atténue les attaques Clickjacking.

3.4 Contôle d'accès

A l'instar des vérifications des droits d'accès au niveau des applications web avant de rendre visible une fonctionnalité donnée sur l'interface de l'utilisateur, les mêmes contrôles doivent s'effectuer sur le serveur pour autoriser l'accès à une fonction donnée. Sinon, les attaquants peuvent forger des demandes pour accéder à une fonctionnalité non autorisée. A titre d'exemple, l'attaquant peut outrepasser (by passe) la phase d'authentification et accéder directement à la page souhaitée en tapant directement son URL.
Pour éviter ce genre de problèmes, les contrôles d'accès ci-après sont nécessaires :

R 38	Appliquer systématiquement les vérifications de contrôle d'accès : Pour garantir le déclenchement des contrôles d'accès d'un utilisateur qui s'est authentifié ou pas, il faut toujours appliquer le principe de la médiation complète.

R 39	Appliquer le principe du moindre privilège : Toute décision d'accès à une donnée ou à une ressource doit répondre au principe du moindre privilège. Si elle n'est pas explicitement permise, l'accès doit être refusé. Si un utilisateur n'a d'autres besoins que de lire une colonne dans une table spécifique, l'administrateur de base de données ne doit lui octroyer que ce droit.

R 40	Accorder le minimum de privilèges pour le fonctionnement des applications et des middlewares : Si une application est compromise, il est important que cette application ainsi que tous les services de middleware ne puissent exécuter des taches autres que celles qui lui ont été attribuées. A titre d'exemple, si les couches "application ou métier" ont la capacité de lire et d'écrire des données dans une des bases de données, ces droits ne doivent pas lui permettre de toucher à d'autres tables ou bases de données.

R 41	Ne pas utiliser les références d'objet directes pour les contrôles d'accès : Ne pas laisser les références directes à des fichiers ou des paramètres qui peuvent être manipulés pour accorder l'accès excessif. Les décisions de contrôle d'accès devraient être basées sur l'identité de l'utilisateur qui s'authentifie ainsi que sur l'information de confiance du côté serveur.

R 42	Ne pas utiliser des renvois ou des redirections non validés : Une redirection non validée peut permettre à un attaquant d'accéder sans authentification au contenu privé. En outre, les redirections non validées permettent à l'attaquant de renvoyer les victimes vers des sites malveillants. Il faut empêcher l'apparition de ces redirections en effectuant les vérifications de contrôles d'accès appropriés.

3.5 Gestion des erreurs

La gestion des erreurs garantit qu'en cas d'erreur, aucune information importante n'est présentée à l'utilisateur. Il est à noter, que lors de la phase de reconnaissance, un attaquant stresse l'application pour recueillir le maximum d'information à travers les messages d'erreurs. Pour s'y protéger, ci-dessous les recommandations à suivre :

R 43	Afficher des messages d'erreurs génériques : Les messages d'erreurs ne doivent pas révéler des détails sur l'état interne de l'application. Par exemple, il ne faut jamais exposés à l'utilisateur via des messages d'erreurs le chemin du système de fichiers ou la pile d'informations.

R 44	Gérer toutes les exceptions : Les gestionnaires d'erreurs doivent être configurés pour gérer les erreurs inattendues et contrôler minutieusement toutes les sorties possibles.

R 45	Gérer les erreurs générées par les Framework : la plateforme de développement peut générer des messages d'erreurs par défaut qui révèlent parfois des informations importantes. Ces messages devraient être remplacés par des messages d'erreurs personnalisés.

3.6 Journalisation

La journalisation est un composant important de la sécurité d'une application web. A cet effet, il est nécessaire de définir une politique de journalisation précisant notamment les modalités et les durées de conservation des différents journaux.
Une bonne gestion des journaux (logs) passe par :

R 46	L'enregistrement de toutes les activités d'authentification dans les logs : Toute activité d'authentification, réussie ou non, doit être enregistrée.

R 47	L'enregistrement de toutes les modifications des droits dans les logs : Toute activité de changements de niveau de privilège d'un l'utilisateur doit être enregistrée.

R 48	L'enregistrement de toutes les actions d'administration dans les logs.

R 49	L'enregistrement de tout accès à des données sensibles dans les logs.

R 50	L'interdiction de l'enregistrement des données confidentielles (mot de passe, numéro de carte bancaire..) dans les logs. Pour des raisons d'audit, ces données doivent être enregistrées d'une manière chiffrée.

R 51	L'enregistrement du contenu des champs "referer et user-agent" pour assurer une traçabilité de toutes les activités : configurer le format de log pour inclure les champs « referer » et « user-agent » dans les logs générées par les serveurs web.

R 52	Stocker les logs en toute sécurité : Les logs doivent être entreposés et entretenus de manière appropriée afin d'éviter leur perte ou leur compromission. Il est recommandé de rediriger les logs vers un autre système. La rétention des logs devrait également suivre la politique de rétention énoncée par l'organisation pour répondre aux exigences réglementaires ou pour fournir suffisamment d'informations à des fins d'investigation.

3.6.1 Recommandations se rapportant au format des logs

R 53	Les journaux doivent être lisibles et facile à analyser via des scripts : utiliser des délimiteurs, des types et des formats de données bien définis.

R 54	Le groupe date-heure du journal est important pour l'analyse des logs. Le groupe date-heure doit être inscrit au début de la ligne, en deux formats : un format « epoch » et un format lisible par les utilisateurs tout en indiquant le fuseau horaire, de préférence GMT ou UTC.

R 55	Chaque log doit avoir un identifiant.

R 56	Pour faciliter le filtrage, une ligne de log doit contenir le maximum d'informations utiles et éviter les logs multi-lignes.

R 57	La catégorie de log doit être introduite parmi les premiers champs de chaque entrée de log.

4 Recommandations liées au déploiement et à la mise en production

Un grand nombre d'attaques sont dues à la négligence de la sécurité lors du déploiement d'une application web. La non suppression des dossiers d'installation, l'absence d'une politique de mise à jour, l'utilisation de composantes vulnérables, l'absence de segmentation et ségrégation du réseau, l'adoption d'une architecture réseau non sécurisé, etc. sont autant de problèmes qu'il faut absolument prendre au sérieux.

4.1 Endurcissement de l'infrastructure

Il s'agit essentiellement de mettre en place certaines bonnes pratiques et apporter des modifications aux paramètres de la plateforme web afin de rendre difficile la tâche de compromission de l'application web. A cet effet, il faut :

R 58	Installer le serveur Web sur une machine dédiée. L'installation d'autres services (DNS, SMTP, BD...) avec le service web sur le même serveur est à proscrire ;

R 59	Désinstaller les services et les composantes inutiles ;

R 60	Utiliser un disque dur physique dédié ou au moins une partition différente pour le contenu du site Web ;

R 61	Réserver une partition dédiée pour les fichiers « uploadés » par les utilisateurs ;

R 62	Personnaliser toutes les pages d'erreurs de telle façon à ne divulguer aucune information qui puisse aider un hacker ;

R 63	Supprimer les traces et configurer le serveur de manière à ne pas divulguer les technologies et les versions utilisées. Exemples : – Apache : mettre la variable server « tokens » dans le fichier « http.conf » à la valeur « prod » et la variable « server signature off ». – php : mettre la variable « expose_php » à la valeur « off » dans le fichier » php.ini ».

R 64	Réécrire des URLs pour cacher les technologies utilisées.

R 65	Supprimer du système d'exploitation les comptes issus de l'installation par défaut.

R 66	Supprimer toute la documentation fournie, les pages et applications d'exemples « sample applications », les dossiers d'installation, les dossiers de backup et les dossiers inutiles sur le serveur ;

R 67	Utiliser des outils de vérification de l'intégrité des fichiers au niveau du serveur web « Tripwire, Integrit… » ;

R 68	Installer une solution antivirale et la mettre à jour ;

R 69	Tester les mises à jour sur une copie du serveur principale en vue de les valider avant de leur installation définitive sur le serveur de production ;

R 70	Respectr le principe du moindre privilège lors de la définition des droits des utilisateurs et l'exéction des services (exemple : enlever le droit d'exécution dans la partition "upload fichier") ;

R 71	Désactiver l'affichage du contenu des répertoires. Pour désactiver la fonction d'indexation des répertoires dans le serveur Apache, par exemple, il faut effacer « Indexes» dans la ligne « Option » du répertoire concerné au niveau du fichier « httpd.conf ».

R 72	Interdire toute possibilité de remonter au répertoire racine afin d'empêcher la possibilité de redescendre vers les répertoires systèmes et exécuter des commandes non autorisées.

R 73	Renforcer la sécurité des différents OS, serveurs, technologies et logiciels en appliquant les guides d'endurcissement de sécurité relatifs à chaque produit.

Liens utiles pour l'endurcissement des serveurs Apache et IIS :

• **serveur apache**

- http ://httpd.apache.org/docs/current/fr/misc/security_tips.html
- http ://www.symantec.com/connect/articles/securing-apache-step-step
- http ://www.nsa.gov/ia/_files/webs/archived/apacheWS.pdf

• **serveur IIS 7 :**

- http ://technet.microsoft.com/fr-fr/library/cc731278%28v=ws.10%29.aspx

4.2 Administration

Les mécanismes d'administration des sites web sont par nature des éléments sensibles dont l'exposition doit être limitée.

R 74	Pour administrer les applications web via des protocoles sécurisés, Il est recommandé d'utiliser les protocoles sécurisés tels que SSH, SFTP et HTTPS.

R 75	Il est préférable de restreindre l'accès à l'administration des sites web aux seuls postes d'administration autorisés. Il est souhaitable de doter le serveur web d'une autre interface réseau réservée pour les connexions d'administration.

R 76	Eviter d'exposer les interfaces d'administration (par exemple : phpmyadmin, webmin, cpanel…) à l'extérieur. Au cas où cette solution est nécessaire, il est important de : – Utiliser un VPN avec chiffrement ; – Utiliser des mots de passe fort ; – Activer un système de connexion à la demande ; – Journaliser l'activité des accès distants ; – Contrôler des tentatives d'accès.

Dans le cas d'un hébergement externalisé, il est recommandé d'administrer exclusivement l'application web qu'à partir d'une adresse IP fixe

4.3 Gestion des sauvegades

Les procédures de sauvegarde et de récupération des données doivent être formalisées. Les configurations des différentes composantes de la plateforme web doivent être sauvegardées et stockées dans un coffre ignifuge. Ce coffre doit être placé dans un local autre que celui qui abrite les serveurs afin d'éviter les dommages liés aux incendies par exemple. Ces sauvegardes doivent être mises à jour et périodiquement vérifiées pour s'assurer du bon fonctionnement des médias de stockage.

4.4 Gestion de mise à jour

Pour assurer une gestion efficiente des patches « correctifs » de sécurité, il est important de :

R 77	Utiliser des outils automatisés pour la détection des patches de sécurité manquants (ex. utiliser Microsoft Baseline Security Analyzer pour les systèmes d'exploitation Windows) ou assurer une veille pour être au courant des nouveaux patchs.

R 78	Vérifier la source du patch avant l'installation ;

R 79	Vérifier la source du patch avant l'installation ;

R 80	Examiner attentivement les documents se rapportant au patch avant son application ;

R 81	Installer les correctifs dans un environnement de test avant de l'appliquer sur l'environnement de production ;

En général, appliquer les patchs le plus tôt possible.

4.5 Supervision des performances

Le test et le réglage des performances est un processus continu. Pour tester et régler les performances de manière efficace, la première étape consiste à évaluer le niveau des performances actuelles. Étant donné que les performances de l'application web peuvent varier considérablement d'un instant à un autre, une analyse suffisamment longue doit être opérée pour obtenir une image fidèle de l'activité de l'application web.
Pour améliorer les performances de l'application web, les différentes composantes de la plateforme web doivent être examinées afin de détecter les goulots d'étranglement potentiels. Ces goulots peuvent être dus à l'utilisation d'équipements inappropriés ou mal configurés.
Après avoir évalué les performances du serveur, les modifications doivent être apportées d'une manière séquentielle (une seule modification à la fois).
Après chaque modification, il faut vérifier si la modification a eu l'effet escompté ou si des effets secondaires indésirables ont été constatés. Après avoir évalué l'impact d'une modification, il faut déterminer si d'autres modifications sont encore nécessaires.

4.6 Evaluation des vulnérabilités

La vérification continue de la sécurité permet de gérer l'adéquation entre le niveau de sécurité existant et celui recherché. Ces contrôles permettent également d'analyser les vulnérabilités résiduelles au niveau du système d'information. Au fur et à mesure que les équipements évoluent, de nouvelles failles apparaissent, d'où l'importance de mener des vérifications périodiques de la sécurité des plateformes Web à travers :

- Des audits de sécurité suite à tout changement majeur dans la plateforme ;
- Le contrôle régulier du système d'information en utilisant des scanners de vulnérabilités (exemple : Nessus, Nikto, Wapiti, ZAP. . .).

4.7 Détection des incidents

Le maintien de la sécurité d'une application web dépend de la rapidité de détection des incidents de sécurité. Un simple parcours des principales pages du site permet de constater les défigurations les plus élémentaires. Toutefois, certaines défigurations sont très discrètes et peuvent subsister longtemps avant qu'elles soient découvertes.
A cet effet, il est recommandé de vérifier régulièrement l'intégrité des répertoires de l'application web ainsi que celle de la configuration. Les changements légitimes devraient être enregistrés dans des fichiers spécifiques telle que la base de données et survenir à des moments précis, comme à l'occasion des mises à jour du site par exemple. L'apparition soudaine de nouveaux fichiers ou la modification de la configuration du serveur doivent déclencher une alerte. Dans ce cas une investigation s'impose.
Par ailleurs, il faut s'assurer que les points de contact techniques associés au site sont valides et maintenus à jour. Le fournisseur d'adresses IP doit mettre à jour les informations des bases WHOIS. Ces informations doivent comprendre une adresse électronique valide pour permettre, en cas de problème, de contacter le responsable du site.

4.8 Conduite à tenir en cas d'incidents

En cas d'incident de sécurité, il convient de prendre un certain nombre de précautions pour rétablir au plus vite le fonctionnement normal de l'application web. Il est important de rassembler et de préserver toutes les informations utiles pour mener le cas échéant des investigations. Au cas où un site est hébergé chez un prestataire extérieur, il faut prendre les démarches appropriées pour obtenir les journaux nécessaires à des fins d'investigations technique ou légale.
En cas de remise en service d'un site, il faut s'assurer que le site ne comporte plus d'éléments malveillants et ou les vulnérabilités exploitées par l'attaquant.
En cas d'incident, le responsable du système devra suivre le processus suivant :

- Déclarer l'incident auprès du maCERT.
- Collecter les logs des différentes composantes de la plateforme web, en vue d'une analyse interne ou externe.
- Après avoir déterminé la nature de l'attaque et identifié sa source, il faut prendre les mesures appropriés pour la remise en service de la plateforme.

Les logs à collecter et analyser sont :

- En ce qui concerne le serveur Web Apache, deux types de fichiers log sont à collecter : « access.log » qui contient les requêtes traités par Apache et « error.log » qui contient les messages d'erreurs générés lors du traitement.
- Par défaut, ces fichiers de logs se situent sous le répertoire « /var/log/apache2 » ou « /var/log/httpd ». Les données envoyées dans les requêtes POST ne sont pas journalisées. Pour activer leur journalisation, il faut installer le module « mod_dumpost » (https ://github.com/danghvu/mod_dumpost).

- Quant aux logs de Microsoft IIS, ils sont stockés sous le répertoire « C :
  system32
  LogFiles
  W3SVC » pour Windows Server 2003 ou sous « C :
  inetpub
  logs
  LogFiles » pour Windows Server 2008.
- Vérifier les logs d'uploade et de téléchargement des fichiers dans le serveur FTP.
- Vérifier les logs d'authentification des protocoles et des solutions d'administration (Active directory, SSH, cPanel, etc. . .)

En plus de l'analyse des logs, il est opportun de vérifier :

- Les services en cours d'exécution, les sessions ouvertes, les connexions établies et les fichiers ouverts dans le partage.
- Exemple de commandes : netstat, nbtsat, net view et net session.
- Les taches planifiées : at et crontab.
- Les comptes d'utilisateurs et leurs droits (au niveau : système, BD. . .), pour vérifier continuellement s'il y a eu création d'un nouveau compte administrateur.
- L'espace occupé du disque dur.
- La base de registre.
- L'intégrité des fichiers de configuration.

Références

[1] Sécurité des applications Web : comment maîtriser les risques liés à la sécurité des applications Web, 2009, CLUSIF,
https ://www.clusif.fr/fr/production/ouvrages/pdf/CLUSIF-2009-Securite-des-applications-Web.pdf

[2] Défense en profondeur des applications Web, 2011, CLUSIF,
http ://www.clusif.fr/fr/production/ouvrages/pdf/CLUSIF-2011-Defense-en-profondeur-des-applications-Web.pdf

[3] Note technique : Recommandations pour la sécurisation des sites web, 2013, ANSSI,
http ://www.ssi.gouv.fr/IMG/pdf/NP_Securite_Web_NoteTech.pdf

[4] Guideline on securing public web servers.SP800-44V2, 2007, NIST,
http ://csrc.nist.gov/publications/nistpubs/800-44-ver2/SP800-44v2.pdf

[5] SANS Securing Web Application Technologies [SWAT] Checklist,
http ://www.securingthehuman.org/developer/swat

[6] Les dix risques de sécurité applicatifs web les plus critiques, 2013, OWASP,
https ://www.owasp.org/index.php/Top10#OWASP_Top_10_for_2013

[7] Annexe à l'entente de développement de logiciel sécurisé, 2013, OWASP,
https ://www.owasp.org/index.php/File :OWASP_Secure_Software_Contract_Annex-FR.doc, February 2002.