LES DOSSIERS TECHNIQUES

Défense en profondeur des applications Web

Décembre 2011

CLUB DE LA SECURITE DE L'INFORMATION FRANÇAIS

11 rue de Mogador – 75009 Paris
Tél : +33 1 53 25 08 80 – Fax : +33 1 53 25 08 88
clusif@clusif.asso.fr – www.clusif.asso.fr

Table des matières

Table des illustrations

1. Remerciements

Le CLUSIF tient à mettre ici à l'honneur les personnes qui ont rendu possible la réalisation de ce document, tout particulièrement :

Le responsable du groupe de travail :

Sébastien	**GIORIA**	*Groupe Y*

Les contributeurs :

Jérôme	**CLAUZADE**	*Bee Ware*
Philippe	**LARUE**	*CBP*
Pierre-Emmanuel	**LERICHE**	*Verizon Business*
Michel	**MAXIMIN**	*Crédit Logement*
François	**RENOU**	*Informatique CDC*
Philippe	**SARAFOGLOU**	*PSA Peugeot Citroën*
Arnaud	**TREPS**	*Accor*
Johanne	**ULLOA**	*DenyAll*

Nous remercions aussi les adhérents du CLUSIF ayant participé à la relecture.

2. Introduction

Ce document s'adresse aux responsables, architectes, chefs de projets dont le rôle est d'intégrer une application Web sécurisée dans leur système d'information.

Le but de ce document est de présenter des solutions, technologies et bonnes pratiques permettant de concevoir et de déployer une application Web de manière sécurisée. Ce document s'attache à ne pas présenter de technologies propriétaires.

Ce document n'a pas comme but de présenter dans le détail les concepts de programmation sécurisée, ni la sécurisation du navigateur, ni les technologies clientes (AJAX, Flash, Silverlight,…) ni les Web Services.

Ces éléments pourront être abordés par un autre groupe de travail du CLUSIF et compléter ce document.

3. Définir le niveau de sécurité.

3.1 Qu'est-ce qu'un risque de sécurité applicatif[1]

Les attaquants peuvent potentiellement utiliser beaucoup de cheminements différents à travers une application pour porter atteinte au métier ou au système d'information de l'entreprise (qu'il soit interne ou externalisé). Chacun de ces chemins représente un scénario d'attaque plus ou moins sérieux pouvant mériter une attention particulière.

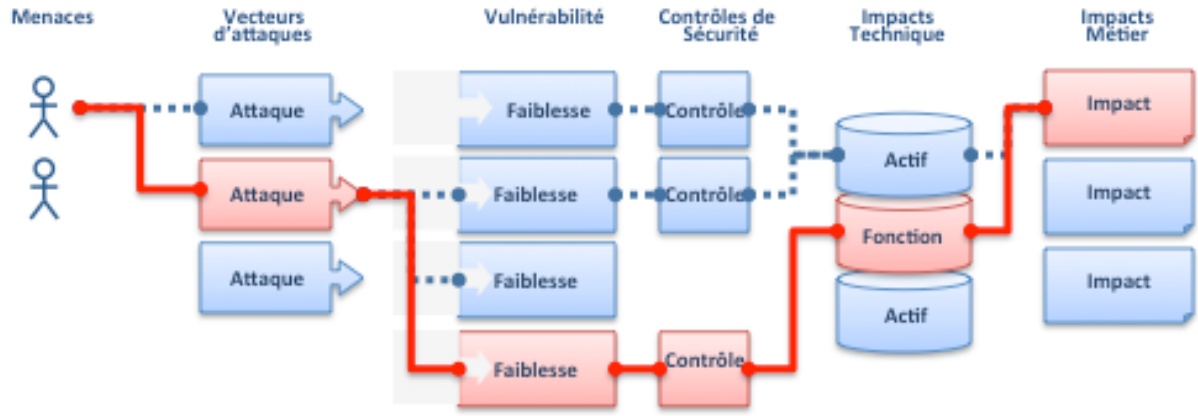


Figure 1 : Risque de sécurité applicatif

Parfois ce chemin est extrêmement simple à découvrir et à exploiter, parfois extrêmement difficile. De la même manière, le préjudice peut être très faible tout comme mettre en péril l'entreprise.

Pour déterminer le risque pour une entreprise, il est possible d'évaluer la probabilité d'occurrence associée à chaque menace, vecteur d'attaque et faiblesse de sécurité et la combiner avec une évaluation des impacts technique et métier de l'entreprise.

L'ensemble de ces facteurs détermine le risque global.

3.2 Analyse des besoins sécurité

La définition d'un niveau de sécurité d'une application Web passe comme pour toute application par une analyse des risques. Cette analyse peut être menée via les méthodologies classiques (compatibles ou non avec ISO 27005, telles que MEHARI ou EBIOS) ou via des méthodes internes à la structure.

Pour effectuer la modélisation des menaces et des vecteurs d'attaques, il est possible de se baser sur les référentiels ci-dessous :

- CWE : le référentiel le plus complet sur les failles de programmation les plus souvent rencontrées. Le CWE/25 est un extrait plus léger permettant de se focaliser sur les 25 erreurs les plus courantes. A noter que ce référentiel n'est pas dédié uniquement au Web !

[1] Le schéma ainsi que la définition sont tirés du document « OWASP Top10 2010 » (version Française)

- WASC ID : référentiel Web utilisé par les différents produits du marché pour catégoriser les attaques. Cela permet une analyse via les vulnérabilités assez poussée.
- OWASP Top10 : la particularité de ce référentiel est d'aborder des risques et non des vulnérabilités contrairement aux deux précédents. Néanmoins, les 10 risques présentés peuvent suffire à mettre en place une première passe d'analyse des risques.

Chacun des référentiels propose des techniques permettant de limiter l'exploitation des vulnérabilités ou de minimiser les risques ce qui permet donc de mettre en face de chacun des risques identifiés des solutions potentielles pour gérer ce dernier.

3.3 Exemple

3.3.1 Présentation

La société exemple.com s'appuie fortement sur Internet pour l'ensemble de son activité tant pour la communication que la vente ou les relations avec ses clients. Elle a mis en ligne pour ce faire quatre services principaux fonctionnellement indépendants :

- site Web institutionnel,
- site de support (FAQ, Forums),
- un site B2C permettant au grand public d'acquérir les solutions d'exemple.com
- un site B2B permettant aux entreprises d'utiliser les services d'exemple.com en mode hébergé via des Web Services.

3.3.2 Analyse des besoins sécurité

Soucieuse d'adapter au mieux le niveau de sécurité de ses 4 services centraux, la société exemple.com a mené une analyse d'impact dont les résultats globaux sont :

- Site Web :
  - Le principal impact considéré est l'intégrité en cas de « defacement » (possible atteinte à l'image).
  - Une indisponibilité de 2 jours est supportable.
- Site de support
  - Le principal impact considéré est l'indisponibilité (qui amènerait une augmentation significative des appels au support et à une insatisfaction possible des clients).
  - La pollution des forums par des personnes mal intentionnées est peu vraisemblable du fait de l'existence d'un modérateur.
- Site B2C
  - Le principal impact identifié est la perte de confidentialité de la base clients pour des raisons réglementaires (CNIL). Faisant appel à un service de paiement externalisé exemple.com ne dispose pas des N° de CB ni d'aucune information bancaire de ses clients grand public.
  - Une indisponibilité de 2 jours est supportable pour peu qu'une page « Site en refonte » soit présente.

- Site B2B
  - Les principaux impacts sont l'indisponibilité même de courte durée (entraînant mécontentement client et pénalités) et la perte d'intégrité des données des clients B2B qui en altérant les valeurs produites pourraient les induire en erreur ; d'où pertes financières potentiellement importantes et pénalités en conséquence. La menace de la concurrence est forte.
  - Les données des clients sont publiques et donc ne sont pas confidentielles.

3.3.3 Exemples de solutions

Une analyse des risques avec quantification des conséquences a permis d'établir les préconisations décrites ci-dessous.

3.3.3.1 Solutions communes

Architecture nTiers multi DMZ, surveillance de tous les composants, application industrialisée des correctifs de sécurité, gestion des habilitations, développement « organisé », existence de plateformes de recette, validation.

3.3.3.2 Solutions spécifiques

- Site Web :
  - Firewall (filtrage niveau 3), IPS, WAF
  - Configuration du serveur « en lecture seule »
  - Interface d'administration inaccessible depuis Internet.
  - Pas de redondance.
- Site de support
  - Firewall (filtrage niveau 3),
  - 2 serveurs redondants sont mis en place
  - Le serveur Web s'exécute dans un environnement isolé (bac à sable, chroot,…)
  - Pour éviter la saturation par des robots, des CAPTCHA sont utilisés.
- Site B2C
  - Firewall (filtrage niveau 3), IPS,
  - L'accès à l'interface d'administration de ce site nécessite une authentification forte.
  - Base de données chiffrée, utilisation exclusive de procédures stockées, limitation du nombre de requêtes et du nombre d'entrées retournées pour 1 requête.
- Site B2B
  - Firewall (filtrage niveau 3), IPS,
  - 2 serveurs redondants sont mis en place
  - Tous les accès des administrateurs sont authentifiés fortement et finement tracés
  - Utilisation des standards WS-Security pour la sécurisation des Web Services (notamment par une authentification mutuelle à base de certificats délivrés par exemple.com)

- Intégration de fonctions de hachage dans les développements pour permettre la détection de valeurs altérées

N.B. Ces préconisations rendent le niveau de risque acceptable pour la société exemple.com mais pourraient s'avérer moins appropriées dans un contexte différent.

4. Défense en profondeur

4.1 Principes et concepts

La défense en profondeur, terme emprunté au domaine militaire, est destinée à retarder l'ennemi. Elle consiste à faire appel à plusieurs dispositifs de sécurité afin de réduire la probabilité de la survenance d'un incident de sécurité et son impact. Le cas échéant, elle est particulièrement adaptée aux applications Web notamment celles utilisant des technologies multi-tiers.

Ce concept fait appel aux notions suivantes :

- **Barrière** : un moyen de sécurité capable de protéger une partie du système d'information contre au moins une menace, elle doit bénéficier d'un moyen de contrôle de son état ; une barrière peut être humaine, organisationnelle ou technique ; ce document fait uniquement référence aux barrières techniques,
- **Ligne de défense** : un ensemble de barrières dont le franchissement provoque un incident avec une gravité qui dépend du nombre de barrières restant à franchir par la ou les menaces, pour atteindre le ou les biens à protéger, et de la valeur de ces biens.

Dans le domaine de la protection des applications web, la mise en œuvre du concept de la défense en profondeur fait appel à des barrières techniques qui composent des lignes de défense. Ces lignes de défense permettent de définir une architecture d'hébergement sécurisée d'application web

Principes: Une politique de défense en profondeur est régie par cinq fondamentaux :

1. Cloisonner le SI par des lignes de défense autonomes et successives, chaque zone ainsi créée ayant un niveau de sécurité homogène et cohérent. Ces zones sont appelées DMZ (DeMilitarized Zone : zone démilitarisée), dans la suite du document.
2. Assurer la continuité et la reprise en cas d'incident de chaque service essentiel.
3. Assurer la défense multi-niveaux des services : au niveau réseau, au niveau système d'exploitation et au niveau applicatif.
4. Ne permettre que le juste nécessaire au fonctionnement du Système d'Information.
5. Ne mettre en œuvre que des barrières très bien maitrisées tant sur le plan technique qu'organisationnel.

4.2 Les lignes de défense et barrières

Le présent chapitre présente un éventail de différentes barrières techniques utilisées dans la construction des lignes de défense. Cet inventaire ne se veut pas exhaustif, le choix ayant été fait de ne retenir que les technologies arrivées à maturité et maintenant bien maitrisées.

4.2.1 Le Pare-feu (firewall)

Ce composant permet de bloquer des attaques en implémentant un filtrage protocolaire des couches 3 et 4 du modèle OSI, tracer voire détecter des comportements précurseurs d'attaques.

Principalement utilisée contre les tentatives d'intrusion et l'exploitation de failles système, c'est la barrière devant être mise en œuvre a minima dans toute défense en profondeur.

4.2.2 Le Pare-feu applicatif (Web Application Firewall)

Ce composant permet :

- De bloquer des attaques en implémentant des filtres au niveau des données transportées par HTTP/HTTPS (HTML et XML).
- De limiter l'exposition à des vulnérabilités détectées dans la logique applicative en attendant une correction en profondeur (par exemple : escalade horizontale, modification du comportement prévu de l'application Web, …).
- D'authentifier et/ou d'autoriser l'accès à tout ou partie de l'application.
- De tracer pour apporter de la visibilité supplémentaire sur l'environnement applicatif (facilite l'investigation, apporter des preuves de tentatives d'intrusions, donner des informations sur les temps de réponses des serveurs Web).
- De gérer la couche SSL en un point central.

Principalement utilisée contre les tentatives d'exploitation de failles applicatives notamment les injections de code, cette barrière peut également être utilisée pour prévenir le déni de service par limitation de nombre de sessions applicatives et préserver des attaques de type « brute force ».

4.2.3 L'IDS/IPS (Intrusion Detection System/ Intrusion Prevention System)

Ce composant permet :

- d'implémenter la détection ou la prévention des activités anormales ou suspectes,
- de détecter et tracer les tentatives réussies ou avortées d'intrusion en s'appuyant sur des bases de signatures ou le comportement de l'attaquant.

Principalement utilisée contre les tentatives d'intrusion, cette barrière offre une grande réactivité face à des attaques car s'appuyant sur des bases de signatures régulièrement mises à jour, en contrepartie le risque des faux positifs perturbant la disponibilité du service n'est pas négligeable et un paramétrage fin ainsi qu'une forte maîtrise sont nécessaires.

4.2.3.1 Choix du modèle de sécurité de la barrière

Un modèle de sécurité négative (voir 0 page 22) devrait systématiquement être utilisé pour protéger par exemple des attaques à structure courte, c'est-à-dire dont le vecteur tient en quelques caractères et contre lesquels une liste blanche est généralement inefficace.

La qualité d'un modèle de sécurité négative dépend de la richesse, de la régularité de mise à disposition des signatures d'attaque et de sa capacité à décoder les requêtes avant de filtrer.

A l'opposé, les paramètres à prendre en considération avant de mettre en œuvre un modèle de sécurité positive (voir 0) sont principalement : le contrôle sur l'application Web, l'évolutivité du site Web dans le temps et l'impact d'un faux positif sur la production (impact métier).

Il est possible de diminuer les contraintes d'utilisation d'une liste blanche en ne prenant en compte que les éléments les plus sensibles (formulaires spécifiques, pages dynamiques, …)

Le meilleur niveau de sécurité est atteint en utilisant conjointement les deux modèles.

4.3 Gestion de la QoS/Haute Disponibilité

La disponibilité des applications Web est également liée à leur accessibilité depuis Internet, réseau par essence non maitrisable en termes de disponibilité et de charge. Ainsi sans qu'une activité malveillante soit en cause une application peut devenir inaccessible. Un certain nombre de barrières permettent d'améliorer cette accessibilité. Parmi les solutions, il est possible d'utiliser les éléments suivants :

Equilibreurs de liens/charge : ces équipements permettent une répartition des flux soit entre les différents liens Internet soit entre serveurs des étages supérieurs, notamment en cas de coupure d'un des liens, ils sont généralement couplés avec le service DNS. Ces éléments permettent aussi d'assurer la haute disponibilité des briques frontales.

Lissage de trafic : ces équipements permettent la gestion des priorités des flux (QoS). Ainsi en cas de congestion des liens Internet certains trafics peuvent être rendus prioritaires par ces équipements. En général, le trafic transactionnel est priorisé par rapport au transfert de fichiers.

Web Applications Firewalls : ils permettent d'améliorer le « rendu utilisateur » via des mécanismes de cache, de compression à la volée et d'optimisation des connexions TCP.

4.4 Les DMZ

Délimitée par 2 lignes de défense, une DMZ est une partie du Système d'Information présentant un niveau de sécurité homogène. Ses composants participent également à la défense en profondeur tant par la réduction de la probabilité de survenance d'un incident que dans la limitation des impacts.

4.4.1 Serveurs HTTP

Les serveurs HTTP apportent leur contribution à la défense en profondeur par :

- Leurs configurations système, réseaux et applicative renforcée limitant au maximum les risques de failles inhérentes à ces composants.
- La traçabilité d'une partie des actions menées par les utilisateurs pour une analyse à posteriori ou dynamique.
- Leur positionnement en coupure qui interdit un accès direct aux bases de données situées en DMZ Données, imposant une rupture de protocole.
- Leur séparation des bases de données par une ligne de défense voire par les serveurs applicatifs de l'étage supérieur.
- Leur capacité d'adresser différents serveurs applicatifs ou serveurs de données en fonction de leur disponibilité.
- Leur capacité à intégrer des fonctionnalités de pare-feu applicatifs (par exemple *mod_security*).
- Leur capacité à gérer le protocole SSL (par exemple en l'absence de pare-feu applicatif).

Serveurs les plus exposés de la chaîne, leurs apports essentiels à la défense en profondeur sont : de la rupture de protocole et l'isolation entre l'utilisateur et les serveurs applicatifs/SGBD.

4.4.2 Serveurs Applicatifs

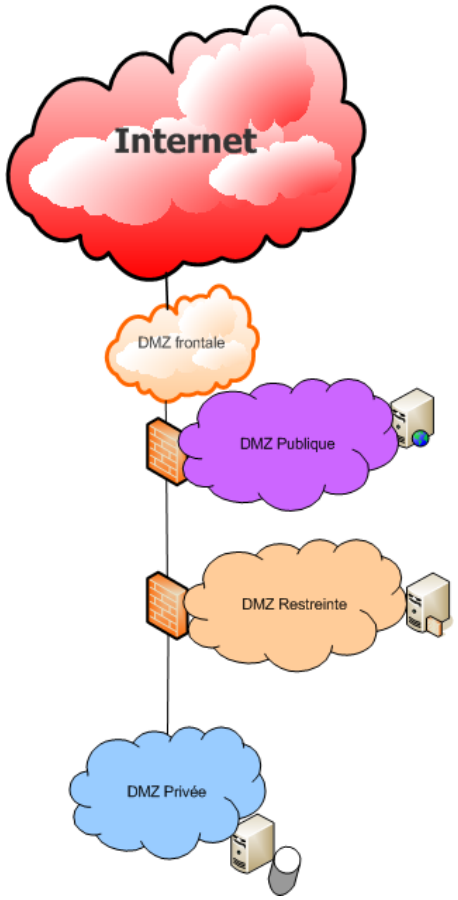
Ces serveurs, dont le rôle principal est de traiter les requêtes reçues par les serveurs de la DMZ publique, participent également à la sécurisation du dispositif global.

- Leurs configurations système, réseaux et applicative doivent être particulièrement renforcées afin de limiter au maximum les risques de failles inhérentes à ces composants.
- Ils permettent de tracer finement les actions menées par les utilisateurs pour une analyse a posteriori ou dynamique.
- Seuls processus autorisés à accéder aux bases de données de l'étage supérieur, ils en interdisent l'accès direct en imposant une nouvelle rupture de protocole.

4.4.3 Serveurs de données :

Ces serveurs qui hébergent les données proprement dites sont protégés par de multiples barrières. Pour autant, eux aussi doivent participer à la défense en profondeur par :

- Des configurations système, réseaux et applicative renforcées afin de limiter au maximum les risques de failles inhérentes à ces composants.
- Des traces fines des actions menées par les utilisateurs pour une analyse temps réel ou a posteriori.



Note : Nombre des éléments figurant sur le schéma précédent sont redondés pour participer à la haute disponibilité. Les équipements « de secours » n'ont pas été représentés.

4.5 Infrastructure standard

4.5.1 DMZ frontale

Cette DMZ a pour rôle principal de garantir la qualité du trafic en provenance d'Internet de façon à ce que les étages inférieurs n'aient à traiter qu'un trafic « propre ».

Au delà de l'aspect sécurité, elle contribue à l'amélioration des performances en évitant que les étages inférieurs ne soient encombrés par les traitements/rejets de flux non légitimes et par la répartition de trafic entre les serveurs des étages inférieurs.

Elle ne contient que des composants techniques dédiés à la sécurité.

Au delà de sa fonction de protection des étages inférieurs, cette zone participe fortement à la haute disponibilité et à la QoS de par la fonction de répartition de trafic qu'elle peut intégrer.

4.5.1.1 Nettoyage des flux

La fonctionnalité de nettoyage des flux assurée par cette ligne de défense peut-être effectuée par la mise en place des barrières techniques suivantes :

- IPS (Intrusion Prevention System).
- Firewalls(Firewalls furtifs ou classiques).
- Reverses-proxies.
- Répartiteurs de charge.
- Firewalls XML/applicatifs (WAF).

4.5.1.2 Répartition de trafic

La répartition de trafic vise uniquement l'aspect Disponibilité/QoS et peut être réalisée par plusieurs équipements :

- Equilibreurs de liens/charge.
- Lisseurs de trafic.

4.5.2 DMZ publique

Dans le cadre d'une architecture sécurité pour l'hébergement d'une application Web, cette zone met en œuvre généralement les serveurs de présentation, ce sont généralement des serveurs HTTP pour des applications Web avec IHM. La sécurité est assurée par ces deux barrières techniques suivantes :

- IPS : Si la zone frontale a déchiffré les flux HTTPS, leur analyse par les IPS devient possible : les risques de faux positifs sont moindres qu'en zone frontale, leurs règles peuvent donc être plus strictes.
- Serveurs HTTP.

La question du chiffrement des flux entre WAF/Reverse Proxy et DMZ publique ne peut être tranchée dans l'absolu et les deux alternatives doivent être pesées

4.5.3 DMZ Restreinte

Cette DMZ contient les serveurs applicatifs et notamment ceux placés en aval d'une authentification effectuée au niveau de la DMZ Publique. Sa participation à la sécurité globale s'articule essentiellement autour de deux points :

- Firewalls : Cette barrière de sécurité n'autorise que les flux strictement nécessaires en provenance des serveurs de présentation de la DMZ Publique.
- Serveurs Applicatifs.

4.5.4 DMZ Privée

Cette zone n'héberge que des bases de données ou des fichiers de données. Sa participation à la sécurité globale s'articule essentiellement autour de plusieurs points :

- Firewalls : Cette barrière de sécurité n'autorise que les flux strictement nécessaires depuis les serveurs applicatifs à destination des serveurs de bases de données.
- Serveurs de données.

4.6 Exemples d'architectures pour le déploiement des WAF

Les WAF peuvent être déployés selon différentes architectures qui pourront être choisies en fonction de différents critères comme :

- La facilité de déploiement.
- Le niveau de sécurité attendu.
- Le niveau de haute-disponibilité attendu.
- La capacité de l'architecture à évoluer en terme de charge.

Voici quelques exemples possibles

4.6.1 Mode Transparent

4.6.1.1 Principes

Dans le mode transparent, le WAF se comporte comme un fil au niveau réseau. Il peut par conséquent être déployé sans avoir aucun impact sur l'architecture, tant au niveau physique qu'au niveau IP.En mode transparent le WAF ne réalise aucune modification de l'adresse IP du client et du serveur WEB. Il est donc transparent à la fois pour le client et pour le serveur. La facilité et la rapidité de déploiement de ce type de WAF sont ses principaux avantages.

En outre, le fait qu'il n'ait aucun rôle actif dans l'infrastructure permet de le déployer en mode « fail-open ». Cela signifie qu'en cas de dysfonctionnement ou de surcharge le WAF met automatiquement en œuvre un mécanisme désactivant l'intégralité de ses fonctions et permettant le transfert des données sans effectuer aucune opération et par conséquent sans aucun impact sur le trafic.

En revanche ce mode de fonctionnement interdit la mise en œuvre d'une rupture protocolaire. Cette limitation implique en particulier :

- Qu'il n'y a pas de rupture protocolaire entre le client et le serveur Web, contrairement au déploiement en reverse proxy ;
- qu'il analyse moins finement les caractéristiques du protocole HTTP, contrairement au déploiement en reverse proxy qui utilise un véritable moteur HTTP ;
- que le WAF ne peut terminer un tunnel SSL. Les conséquences de cette dernière limitation sont :
  - que l'authentification par certificat SSL n'est pas possible sur le WAF ;
  - que le WAF ne peut pas mettre en œuvre un mécanisme d' « offload » SSL, la charge liée au traitement des transactions chiffrées restant du ressort du serveur protégé.

La haute-disponibilité peut être prise en défaut par ce mécanisme. Aucune adresse IP n'étant définie sur les interfaces effectuant l'analyse, il n'est pas possible de mettre en œuvre des protocoles réseau standard de haute-disponibilité tels que VRRP (Virtual Router Redundancy Protocol). Par conséquent, les WAF en mode transparent doivent se reposer sur des protocoles locaux, tel que Spanning-Tree, qui reste limité à un segment Ethernet et présente des temps de convergence de l'ordre de la minute.

Une alternative consiste à positionner les WAF en cascade l'un derrière l'autre. Si le premier est défaillant, il se positionne en « fail-open» et laisse passer le trafic vers le deuxième. Il est à noter qu'un attaquant qui réussirait à faire tomber la première barrière pourra venir à bout de la deuxième de la même façon. Dans ce cas les serveurs Web se trouveront sans protection.

Enfin leur rôle passif sur le réseau leur interdit la mise en œuvre de technologies souvent présentes dans les autres solutions de ce type, en particulier l'interaction avec des serveurs tiers pour l'authentification par exemple, ou la répartition de charge des serveurs web protégés. Pour les mêmes raisons les fonctions d'accélération telles que le cache ou la compression, mises en œuvre pour compenser la latence induite par le filtrage, ne peuvent être implémentées sur ce type de WAF.

En dehors du filtrage applicatif, les WAF en mode transparent restent limités en termes de fonctionnalités « annexes» et leur principal avantage est la simplicité de déploiement et l'absence d'impact sur l'architecture réseau.

4.6.1.2 Architectures

Le mode transparent impose que le WAF soit mis en œuvre sur un lien physique supportant l'intégralité du trafic à destination des serveurs à protéger. Cela impose de laisser les équipements protégés dans des zones de sécurité accessibles depuis l'extérieur.

Idéalement ces équipements seront déployés sur les liens de concentration des flux avant leur distribution vers les liens serveurs.

Une architecture de ce type peut se définir comme dans les figures ci-dessous :

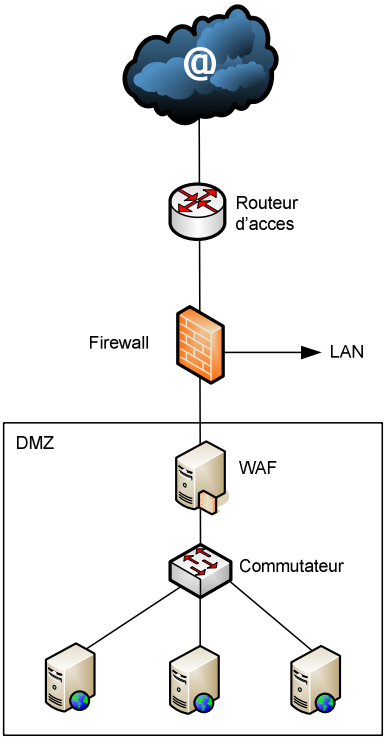


Figure 2 : Déploiement de WAF en mode transparent sans haute disponibilité

Les figures ci-dessous détaillent ces deux types d'architecture.

4.6.2 Mode reverse-proxy

4.6.2.1 Principes

Le mode reverse-proxy est le mode le plus commun de déploiement des WAF. Il consiste à faire apparaitre le WAF comme le serveur Web du point de vue de l'utilisateur.

Schématiquement, un WAF fonctionnant en mode reverse-proxy reçoit les requêtes de l'utilisateur, les analyse, puis les transmet au serveur réel si aucune menace n'est identifiée. Il apparaît alors pour le serveur comme le client. Les réponses lui sont retournées, de nouveau analysées puis envoyées au client réel.

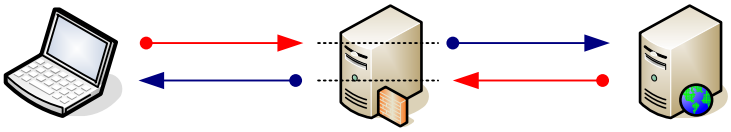


Figure 3 : Architecture en reverse-proxy

Ce mode de fonctionnement présente de nombreux avantages tant en termes de sécurité qu'en termes de performances ou d'intégration dans des architectures complexes.

D'un point de vue de la sécurité, ce mode opératoire permet en premier lieu de masquer l'infrastructure hébergeant l'application Web. En effet, le seul point d'accès étant le reverse-proxy, l'utilisateur n'a par conséquent aucune visibilité de cette infrastructure, et ce :

- Du point de vue réseau, car la seule adresse IP qui est visible et accessible depuis l'extérieur est l'adresse du WAF, lequel se trouve localisé dans une zone de sécurité spécifique et isolée du reste du réseau interne par un équipement de filtrage tel qu'un firewall réseau ;

- Du point de vue applicatif, dans la mesure où toutes les informations pouvant laisser la possibilité d'identifier la nature du serveur web (en-têtes, comportement dans les cas d'erreurs etc.) sont positionnées par le WAF.

Le fonctionnement en reverse-proxy permet également d'effectuer une rupture protocolaire dans la mesure où la session HTTP ou HTTPS est terminée par le WAF, lequel en initie une nouvelle vers le serveur réel. Il est donc nécessaire que le WAF « comprenne » l'intégralité des éléments de la requête, et que par conséquent cette dernière se doit d'être correctement construite, conformément aux spécifications du protocole. Une requête malformée sera donc naturellement rejetée par le WAF.

En outre, les différents éléments soumis par le client doivent être compris par le WAF, ce qui impose que ce dernier décode les requêtes du client. Ainsi les tentatives de contournement des filtres basées sur l'encodage (généralement en Hexadécimal ou en Unicode) seront également rejetées si elles ne peuvent être décodées.

En ce qui concerne les performances, le positionnement en coupure permet non seulement de mettre en œuvre des fonctions de cache et de compression mais également d'appliquer des mécanismes de multiplexage des connexions : plusieurs requêtes provenant de clients différents pouvant être soumises au serveur réel via une unique connexion TCP établie vers ce dernier.

Le positionnement logique comme unique point d'entrée vers les infrastructures Web offre la possibilité de masquer plusieurs serveurs ou applications derrière une adresse unique, la différenciation s'effectuant au niveau de paramètres telles que le nom du serveur (en-tête Host:), le répertoire ou encore la valeur d'un argument soumis dans une requête GET ou POST.

Enfin le mode reverse-proxy offre de nombreuses possibilités de déploiement sécurisé telles que la mise en place d'architectures multi-DMZ.

Le mode reverse proxy est plus difficile à mettre en œuvre que le mode transparent. Toutefois, il offre une grande richesse en termes de fonctionnalités « annexes » et permet de disposer d'une architecture apte à grandir si la charge augmente de façon conséquente.

4.6.2.2 Architectures

Le principal avantage du déploiement en reverse-proxy est que le WAF peut être mis dans une DMZ publique alors que le reste de l'infrastructure Web est localisé dans une zone privée à laquelle aucun système externe n'est autorisé à accéder.

Dans le schéma le plus simple, c'est-à-dire sur une unique plate-forme et sans système de haute disponibilité une architecture type peut se définir comme dans la figure ci-dessous.

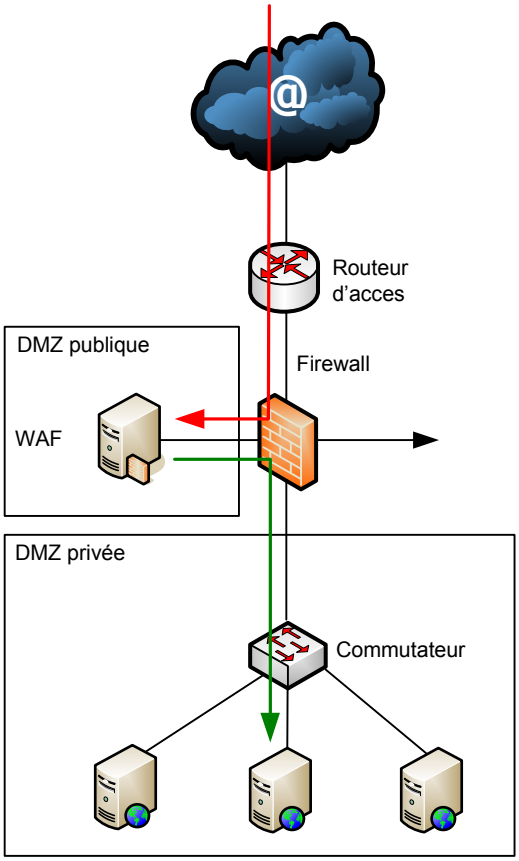


Figure 4 : Déploiement de WAF en reverse-proxy

4.7 Règles de filtrage à implémenter dans les barrières

S'il n'est pas possible de définir des jeux de règles standards pour les applications Web du fait de leur diversité, quatre principes doivent toutefois être systématiquement implémentés. Ces règles doivent être :

Adaptées aux applications

L'analyse de risques de l'application et son besoin en sécurité permettent de définir l'ensemble des règles à mettre en place dans les barrières, voire les barrières techniques à implémenter. Elles sont spécifiques aux menaces identifiées et doivent être détaillées en connaissance de l'application.

Aussi génériques que possible

La mise en place d'une architecture dédiée pour chaque application n'est généralement pas possible, pour des raisons de coûts et d'exploitabilité.

De ce fait une définition d'architecture générique doit être mise en place pour protéger les applications classées selon le niveau de criticité de l'application défini selon les critères Disponibilité, Intégrité, Confidentialité, Preuve (DICP).

Non suffisantes
La mise en œuvre d'une architecture d'hébergement utilisant le principe de protection en profondeur permet de retarder un attaquant. Elle ne se substitue pas à la démarche d'intégrer la sécurité au sein du cycle de développement des applications web.

Intégrées dans les processus d'évolution applicative.
Enfin les évolutions applicatives devront être menées en intégrant les barrières de sécurité et les livraisons applicatives synchronisées avec les évolutions des barrières. Ceci est particulièrement important concernant les pare-feu applicatifs.

Pour la mise en place de ces règles il est possible de s'appuyer sur l'un ou les modèles de sécurité suivants :

Modèle de sécurité positive
Le modèle de sécurité positive consiste à n'accepter que ce qui a été spécifiquement autorisé. Pour mettre en œuvre un modèle de sécurité positive, on utilise une liste blanche. Toutes les requêtes qui présenteront des aspects différents de ce qui est contenu dans cette liste seront rejetées.

Lorsqu'une application est utilisée par un utilisateur, ce dernier envoie à l'application un certain nombre de données. Généralement, ces données (aussi appelées entrées utilisateur) ne sont que peu ou pas du tout contrôlées par l'application.

Le modèle de sécurité positive permet de combler ce manquement et préserve le site Web de certaines utilisations non prévues. Ce modèle est souvent présenté comme le plus fort en termes de sécurité applicative. Toutefois, il est difficile à mettre en œuvre, car la liste blanche doit être l'exact reflet du site Web. Si un changement est réalisé sur le site Web, ce changement doit être reporté sur la liste blanche. La liste blanche demande donc à être construite et doit évoluer parallèlement au site Web. Des procédures doivent donc être mises en place pour fournir une nouvelle liste blanche à chaque évolution du site Web.

Une liste blanche permet de se prémunir efficacement contre la plupart des attaques. Pour ce faire, la liste blanche devra être la plus restrictive possible. Dans le cas contraire, son efficacité sera très limitée. Il convient donc de porter une attention particulière à la réalisation de la liste blanche pour que des règles trop permissives ne soient pas introduites par erreur

Modèle de sécurité négative
Le modèle de sécurité négative consiste à bloquer les requêtes qui contiennent des similitudes avec des signatures d'attaques. Ces signatures d'attaques sont stockées dans des listes noires.

Le modèle de sécurité négative peut être mis en œuvre facilement, car il est dissocié de l'application (contrairement au modèle de sécurité positive).

Si l'on souhaite bloquer de manière exhaustive l'ensemble des variantes d'attaques avec un mécanisme de liste noire classique, le taux de faux positifs généré (trafic légitime bloqué) peut être important.

Il existe une méthode hybride a base de mécanismes de liste noire pondérée. Une même requête sera examinée de façon multiple pour un type d'attaque donné, contrairement au mécanisme de liste noire classique qui ne fait qu'une seule évaluation pour un type d'attaque.

Renforcement du contrôle d'accès applicatif

Les mécanismes d'authentification des applications peuvent être déportés au niveau de la barrière de sécurité. Celle-ci permet de :

- Recueillir et contrôler, avec l'aide éventuelle d'un serveur tiers (Radius, LDAP, …), les éléments d'authentification des utilisateurs à travers, par exemple, un formulaire ou un certificat client.
- Renforcer les conditions de l'accès à l'application (anti-brute force, suivi de session, détection d'injection de contenu offensif, etc.)
- Ne présenter l'application qu'aux utilisateurs autorisés

La mise en place d'un mécanisme de Web Single Sign On (Web SSO) intégré à la barrière de sécurité permet la mutualisation de plusieurs authentifications applicatives.

4.8 SSL (Secure Sockets Layer) ou TLS (Transport Layer Security)

Netscape est à l'origine du protocole SSL. Suite à des travaux, l'IETF a fait évoluer SSL en TLS.

On parle de SSL pour désigner indifféremment SSL ou TLS. Son utilisation permet de répondre aux besoins d'intégrité et de confidentialité par l'utilisation d'une session chiffrée. On peut également gérer un mécanisme d'authentification forte du client.

La gestion des certificats et des clefs doit faire l'objet d'une attention particulière pour éviter aux utilisateurs tout message intempestif :

- Certificat non reconnu, expiré ou révoqué.
- Nom du certificat ne correspondant pas au nom du serveur.

Références

- Le premier texte formel définissant le protocole TLS est la RFC 2246 publiée par l'IETF en 1999.
- D'autres textes ont suivi sur ce protocole :
  - RFC 2712 : Addition de la suite Kerberos à TLS ;
  - RFC 2817 : Passage à TLS lors d'une session HTTP 1.1 ;
  - RFC 2818 : HTTP sur TLS ;
  - RFC 3268 : Utilisation du système de chiffrement AES pour TLS.
  - RFC 5246 : TLS version 1.2 (publication août 2008)

4.8.1.1 De l’importance de déployer un SSL de qualité.

Il ne suffit pas d’utiliser le protocole SSL pour obligatoirement être protégé. SSL est un protocole fiable qui sert de fondement à la sécurité d’Internet. Mais qui n’est pas toujours configuré correctement sur de nombreux sites Internet.

La bonne implémentation du service SSL est primordiale.

Cette notion évolue constamment dans le temps selon deux facteurs :

- Avec l'augmentation progressive de la puissance de calcul des ordinateurs, le niveau global des dispositifs techniques doit être régulièrement réajusté à la hausse.
- Des vulnérabilités sont régulièrement découvertes dans le protocole SSL lui-même.

Ainsi, en 2011, on peut donner quelques exemples de bonnes pratiques SSL du moment:

- On considère qu’un certificat de serveur Web doit reposer sur une clé RSA de minimum 2048 bits.
- On considère qu’il faut un chiffrement AES des connexions sur minimum 128 bits. La configuration serveur ou le type de certificat utilisé peuvent imposer le niveau minimum requis.
- Il faut utiliser du TLS SSL V3, à jour au niveau des patchs et proscrire SSL V1 et 2.
- Le chainage du certificat avec les autorités racines est aussi un vecteur d’attaque à verrouiller.

Pour la génération du trousseau de clés, la qualité de l’entropie est également importante car c’est un biais d’attaque récurent.

Note : Par ailleurs, des scanners de vulnérabilité sont capables de tester ce niveau de sécurité et pointer les éventuelles faiblesses ou vulnérabilités.

5. Architecture applicative / Développements sécurisés

La défense en profondeur d'une application Web ne dispense pas d'effectuer de la sécurisation dans le développement. Il est impensable de faire reposer l'ensemble de sa sécurité uniquement sur des briques d'infrastructures.

En effet, certaines attaques ne pourront être que difficilement traitées par l'infrastructure (attaques sur la logique applicative).

Les principes de développements sécurisés sont donc essentiels.

5.1 Le cycle de vie logiciel et la sécurité

Une application de par son principe n'est pas figée : correctifs, ajouts de fonctionnalités sont des éléments classiques rencontrés lors de la vie d'une application.

Il est donc indispensable de mettre en place un ensemble de bonnes pratiques pour implémenter et maintenir un niveau de sécurité adapté aux risques tout on long du cycle de vie logiciel.

5.1.1 Les outils et méthodologies disponibles

Pour cela, il existe différentes boites à outils sur le marché permettant d'ajouter de la sécurité lors du cycle de vie logiciel. A chaque outil, ses avantages et ses inconvénients.

Parmi les outils méthodologiques disponibles il est possible de citer :

- SDL Microsoft[2].
- La méthode de codage sécurisé du CERT.
- La méthode de l'ANSSI (ex DCSSI/SCSSI).
- La méthode de Cigital.
- L'OWASP CLASP.

5.1.2 Les grandes étapes dans le cycle de vie

Chacune des méthodes de développements sécurisés qu'il est possible d'utiliser dispose de quatre grandes étapes qui sont communes et qui peuvent se rendre indépendantes, les unes des autres.

- La formation des intervenants : il ne faut pas uniquement former les développeurs, mais aussi les architectes, chefs de projets et testeurs !
- L'analyse des menaces : globalement assez proche d'une analyse des risques de type ISO 27005 telles que MEHARI, EBIOS,…

[2] Il existe une version de la SDL simplifiée et une version sur les méthodes Agiles

- La conception, le développement : il est possible de s'appuyer pour cela sur les différents guides OWASP (ASVS, Building Guide, …).
- Les tests sécurité avant mise en production (tests d'intrusion, revues de code)
- Les tests et contrôles sécurité récurrents en production.

5.2 Les grands principes de développements sécurisés

5.2.1 Valider les entrées

Les failles sont exploitées en abusant d'une fonctionnalité offerte par l'application. L'exploitation s'appuie donc toujours sur un point d'entrée, tel qu'un champ de saisie d'un formulaire (apparent ou caché), un paramètre d'une page ou des en-têtes HTTP.

Ainsi, il est recommandé de considérer toutes les données provenant du client comme « non sûres ». La mise en place de contrôles forts sur les différents points d'entrée offrira un premier niveau de protection efficace face à une grande partie des attaques web.

Ces contrôles seront effectués conformément aux règles suivantes:

- Vérifier le nombre de paramètres et pour chacun d'eux contrôler leur nom et leur valeur (longueur, type de données, intervalle de valeurs).
- De préférence, n'autoriser qu'une liste finie de possibilités (« liste blanche »).
- Le contrôle doit être effectué côté serveur, puisque les mécanismes clients, tels qu'une validation dans du code javascript, peuvent être facilement contournés.

5.2.2 Limiter la surface d'attaque

Plus un programme ou une architecture est complexe, plus le nombre de vulnérabilités potentielles augmente. En effet, chaque service exposé, chaque composant constitue un point d'entrée pour l'attaquant.

Il convient de réduire le nombre de ces vecteurs d'attaque, ce qui contribue à limiter le risque global et permet de concentrer les efforts de sécurisation sur les éléments qui restent exposés. Par exemple, il est préférable de ne pas exposer sur Internet l'interface d'administration d'un site même si celle-ci est protégée par un mécanisme de sécurité.

5.2.3 Principe du moindre privilège

Lorsqu'un attaquant exploite une vulnérabilité, les actions possibles sur l'environnement d'exécution sont limitées aux privilèges donnés à l'application. En restreignant ces privilèges au maximum, l'impact d'une vulnérabilité devient plus faible puisque les scénarii d'exploitation avancée ne seront plus réalisables. Par exemple, la connexion à la base de données se fait avec un utilisateur spécifique ayant uniquement les droits d'accès aux données de l'application.

5.2.4 Une bonne gestion des erreurs techniques

Lorsqu'une erreur se produit, le comportement d'une application ne correspond pas au comportement prévu par ses concepteurs. Tout d'abord, les messages d'erreur standards fournissent des informations techniques que l'attaquant va utiliser pour détecter ou exploiter une vulnérabilité. Par exemple, un message d'erreur remontant de la base de données permet

de déterminer la présence d'une faille dite d'injection SQL. Ensuite, la page d'erreur pourra servir lors de la phase d'exploitation : l'attaquant construira des requêtes conçues de façon à ce que les données qu'il souhaite récupérer soient visibles dans le message d'erreur retourné.

Ainsi, la gestion généralisée des erreurs est une brique importante d'une application sécurisée. Non seulement pour éviter que celles-ci servent les intérêts des attaquants, mais aussi dans une optique de détection d'intrusion (une attaque ou une recherche de failles sur une application web engendrant un volume d'erreurs suspect).

Quels mécanismes mettre en place ?

- Intercepter toutes les erreurs techniques et rediriger l'utilisateur sur une page d'erreur banalisée (sans information technique).
- Conserver la trace complète du problème dans un fichier de trace spécifique pouvant servir au debug et mettre en place le processus organisationnel de gestion de ces erreurs.

5.2.5 Une bonne gestion des traces techniques

Mettre en place des fichiers de traces spécifiques : par exemple il n'est pas utile qu'un administrateur dispose des traces de debug. Il est conseillé de mettre en place plusieurs types de fichiers de traces tels que :

- Fichiers de trace des accès (typiquement les logs HTTP).
- Fichiers de trace des erreurs des serveurs.
- Fichiers de trace des accès réussis et infructueux.
- Fichiers de trace destinés au debug.
- Fichiers de trace destinés aux besoins de traçabilité réglementaire (transactions comptable, traçabilité financière, ...).

Enfin, des précautions particulières doivent être prises pour que des données sensibles telles que les mots de passe ou les numéros de cartes de crédit n'apparaissent pas dans les fichiers de traces.

Pour éviter que ces traces techniques soient altérées pendant une attaque, une bonne pratique consiste à les déporter sur un autre serveur.

De plus, une validation systématique des entrées utilisateur contribuera à réduire le nombre de cas non prévus et donc d'erreurs.

5.2.6 Ne pas dépendre de la sécurité par l'obscurité

L'approche consistant à cacher des composants vulnérables en espérant que les attaquants ne les trouveront pas, ne rend pas les applications plus sécurisées. En effet, il existe aujourd'hui des outils ou techniques d'attaque qui permettent de trouver ces composants.

Par exemple, si une catégorie d'utilisateurs n'a normalement pas accès à une page du site, il ne suffit pas de supprimer le lien vers cette page dans le menu de navigation. Un contrôle des droits d'accès devra être effectué dans l'applicatif pour que le mécanisme d'habilitation soit correctement effectué.

Cependant, il reste utile (mais non suffisant) de ne pas dévoiler aux utilisateurs les informations dont ils n'ont pas besoin.

5.2.7 Ne pas confondre fonction ou outil de sécurité et fonctionnalité sécurisée

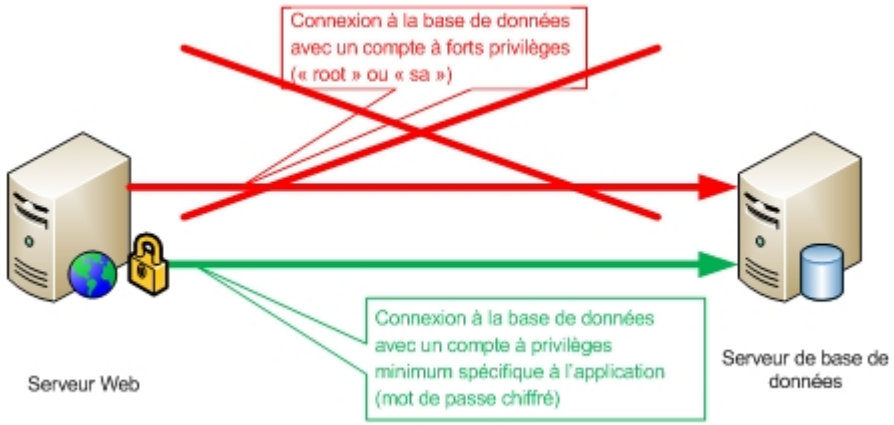
L'utilisation de mécanismes de sécurité ne rend pas nécessairement une application plus sécurisée. Par exemple, l'utilisation du protocole HTTPS n'apportera aucune protection contre les failles d'injection, ayant pourtant un impact très important. Cette fonction de sécurité assure simplement le chiffrement des données lors de la transmission sur le réseau, ce qui aura comme effet de bord le chiffrement des tentatives d'intrusion et donc rendra leur détection plus difficile.

Une architecture applicative sécurisée n'est donc pas l'accumulation de technologies de sécurité. Au contraire, cette architecture doit être pensée globalement, dans le respect des grands principes et en ayant conscience des vulnérabilités classiques des applications web. Les outils ou fonction de sécurité viennent ensuite comme autant de moyens de répondre à une problématique précise.

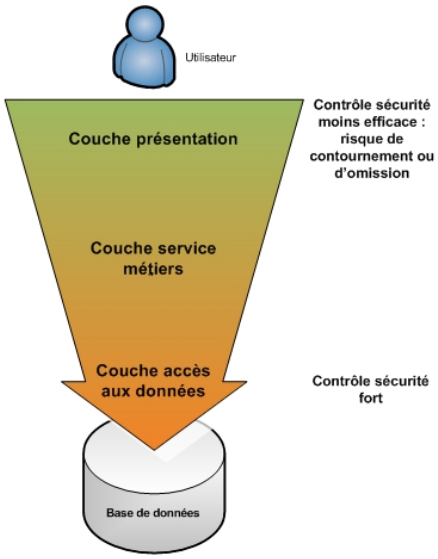
5.2.8 L'accès aux données

La sécurité des accès aux données s'appuie d'une part sur une sécurisation des comptes techniques utilisés pour effectuer ces accès (utilisateur de base de données, permissions sur les systèmes de fichier) et d'autre part sur une architecture applicative assurant un contrôle systématique des droits d'accès aux données manipulées par l'application (habilitations « métier »).

- Le compte technique de connexion aux données doit appliquer le principe du moindre privilège. De plus il est conseillé de protéger les identifiants de connexion à la base, par exemple en chiffrant la chaîne de connexion dans le fichier de configuration.



- Les habilitations « métier » doivent être vérifiées lors de chaque accès aux données. Une architecture applicative qui inclut les contrôles d'autorisation au niveau de la couche d'accès aux données permettra de garantir que ces contrôles seront effectués. Les fonctions d'accès aux données métier sensibles doivent contrôler les habilitations de l'utilisateur avant toute utilisation de la donnée.



5.3 OWASP ASVS

En complément des besoins de sécurité observés/édictés par des méthodes d'analyse des risques classiques, il est possible d'approcher la sécurité via des niveaux de maturité de l'application (sur le même principe que le COBIT). Pour cela, il existe un guide technique OWASP. L'OWASP Application Security Verification Standard (ASVS) permet d'aider le concepteur dans sa tâche. Ce guide peut être utilisé de différentes manières :

La définition d'une politique de codage sécurisé permettant de définir l'objectif de sécurité d'une application.

- Une liste des éléments à vérifier lors d'une revue de code.
- Une liste des éléments de sécurité à mettre en place dans un appel d'offre.
- Un guide de maturité d'une application vis-à-vis de la sécurité.

L'ASVS est organisé en 3 parties principales :

- La définition des niveaux de sécurité (de 1 à 4) et les méthodes de vérifications associées.
- Les différentes exigences de sécurité, utilisant le principe d'approche en liste blanche.
- Les exigences de rapport, nécessaires pour avoir le bon niveau de détail ainsi que les éléments pour une revue lors de l'évolution de la maturité de l'organisation ou du projet.

5.3.1 Les niveaux de sécurité de l'ASVS

L'approche employée par l'ASVS permet d'établir 4 niveaux de sécurité (ou de confiance) d'une application. Ces niveaux permettent d'effectuer une revue progressive et de plus en plus détaillée :

- Le niveau 1 : il correspond à une vérification automatisée de l'application. Par l'utilisation de scanners de vulnérabilités ou d'outils de revue automatisée de code. Il constitue le niveau minimum que devrait avoir toute application.
- Le niveau 2 : ce niveau correspond à une revue de code manuelle et/ou accompagnée de tests d'intrusions manuels. Il n'est pas obligatoire d'effectuer des scans automatisés pour que l'application soit à ce niveau. Une revue manuelle suffit.
- Le niveau 3 : pour atteindre ce niveau, il est nécessaire de mettre en place une modélisation des menaces et des vulnérabilités dans le but de vérifier la conception de l'application. Cela peut s'apparenter à une analyse de risques (dans le principe de la norme ISO 27005).
- Le niveau 4 : ce dernier niveau est orienté recherche de codes malveillants au sein de l'application. Par la recherche de virus, portes dérobées….

Les deux premiers niveaux sont globalement indépendants et il est possible d'obtenir un niveau 2 de manière très simple. Les autres niveaux nécessitent de se baser sur la mise en place préalable d'une vérification plus faible (de type 1 ou 2).

5.3.2 Les exigences

Les exigences de sécurité sont décrites sur un modèle positif. Ainsi, lors d'une vérification selon le standard ASVS, il ne s'agit pas de chercher au sein de l'application un type de faille mais au contraire de s'assurer que les techniques de protection appropriées sont systématiquement utilisées.

Exemple : Faille XSS

- L'approche négative : Rechercher les morceaux de code pouvant mener à des failles XSS.
- L'approche positive : Vérifier que toutes les données disposent d'un échappement correct des entrées fournies.

Ces exigences sont regroupées en 14 catégories :

- Architecture de sécurité
- Authentification
- Gestion des Sessions
- Contrôle d'accès
- Validations d'entrées
- Encodage et échappement de sorties
- Cryptographie
- Gestion des erreurs et de la journalisation
- Protection des données
- Sécurité des communications
- HTTP Sécurisé
- Configuration de la sécurité
- Recherche de codes malveillants
- Sécurité interne

Chacune des catégories est indépendante des autres et les contrôles présentés sont clairs, simples et précis. Les différentes exigences sont ensuite positionnées sur l'échelle de quatre niveaux, et pour qu'une application soit « déclarée » de niveau X, il est nécessaire qu'elle dispose de tous les contrôles de niveau X.

6. Références

6.1 Défense en profondeur :

- http://fr.wikipedia.org/wiki/D%C3%A9fense_en_profondeur

La défense en profondeur appliquée aux systèmes d'information

- http://www.ssi.gouv.fr/fr/bonnes-pratiques/outils-methodologiques/la-defense-en-profondeur-appliquee-aux-systemes-d-information.html

6.2 Web Application Firewall

- http://www.owasp.org/index.php/Web_Application_Firewall
- http://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion

6.3 Référentiels de vulnérabilités :

- WASC-ID : http://projects.webappsec.org/w/page/13246974/Threat%20Classification%20Reference%20Grid
- OWASP Top10 : https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- CWE25 : http://cwe.mitre.org/top25/

6.4 Méthodologie

Expression des Besoins et Identification des Objectifs de Sécurité (EBIOS) :

- http://www.ssi.gouv.fr/fr/bonnes-pratiques/outils-methodologiques/ebios-expression-des-besoins-et-identification-des-objectifs-de-securite.html

GISSIP (Guide d'Intégration de la Sécurité des Systèmes d'Information dans les Projets)

- http://www.ssi.gouv.fr/fr/bonnes-pratiques/outils-methodologiques/gissip-guide-d-integration-de-la-securite-des-systemes-d-information-dans-les.html

Méthode Harmonisée d'Analyse des Risques (MEHARI) :

- http://www.clusif.asso.fr/fr/production/mehari/

ISO 27005 :

- http://www.iso.org/iso/catalogue_detail?csnumber=56742

Microsoft Security Development Lifecycle :

- http://www.microsoft.com/security/sdl/default.aspx

OWASP Application Security Verification Standard

- https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

Cert Secure Coding

- http://www.cert.org/secure-coding/

7. Glossaire

AJAX :	Asynchronous Javascript And XML
B2B :	Business to Business.
B2C :	Business to Customer.
IPS :	Intrusion Prevention System
WAF :	Web Application Firewall
XSS :	Cross Site Scripting
OWASP :	Open Web Application Security Project
COBIT :	Control Objectives for Information and related Technology
CAPTCHA :	Completely Automated Public Turing test to tell Computers and Humans Apart : test de Turing permettant de différencier de manière automatisée un utilisateur humain d'un ordinateur.
VRRP :	Virtual Router Redundancy Protocol
Offload-SSL :	Gestion du SSL
IETF :	Internet Engineering Task Force
RFC :	Request For Comment
RSA :	Algorithme de chiffrement asymétrique de Rivest Shamir Adleman
AES :	Advanced Encryption Standard
CERT :	Computer Emergency Response Team
ANSSI :	Agence Nationale de la Sécurité des Systèmes d'Information
WS-Security :	Web Services Security ; ensemble de normes de sécurité pour les Web Services.
Fail-open :	Mode de fonctionnement d'un système qui devient passant en cas de défaillance.

L'ESPRIT DE L'ÉCHANGE

CLUB DE LA SÉCURITÉ DE L'INFORMATION FRANÇAIS

11, rue de Mogador
75009 Paris
✆ 01 53 25 08 80
clusif@clusif.asso.fr

Téléchargez les productions du CLUSIF sur

www.clusif.asso.fr