MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ SAAD DAHLEB DE BLIDA
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

MÉMOIRE DE MASTER

Spécialité : Génie des Systèmes Informatiques

Intitulé :

Cloud-based Web Application Firewall

Par Bourezane Nabil & Tounani Abdellah

Thèse présentée et soutenue à Blida le 30/06/2013 devant le jury composé de :

Président : Mme Boustia Narhimene

Examinateurs :

Mme Mancer Yasmine

Mme Toubaline Nesrine

Promoteur : M. Bennouar Djamal

Blida, juin 2013

RÉSUMÉ

Cette étude porte sur une nouvelle approche d'externalisation de la sécurité des applications Web, qui est le pare-feu applicatif Web distribué en nuage. Cette solution est un service à la demande chargé de filtrer et de surveiller le trafic et de le livrer à l'application. Simple à déployer, il tire avantage du concept de l'informatique en nuage et assure donc une haute disponibilité et de hautes performances dans la sécurité, la livraison et la mitigation des attaques DDoS du fait de ses capacités et ses ressources. La centralisation de ce dernier est un aspect stratégique pour l'évaluation continuelle et le suivi professionnel de la sécurité, en effet, elle favorise un apprentissage supervisé et commun.

Les mots clefs : informatique en nuage, pare-feu applicatif distribué en nuage, sécurité Web, piratage, Nginx, ModSecurity, proxy inverse, application Web, évolutivité horizontale, haute disponibilité, redondance, tolérance aux pannes au niveau VM, tolérance aux pannes au niveau application.

Abstract

This study focuses on a new approach to outsourcing web security, which is the cloud-based web application firewall. This solution is an on-demand service in charge of filtering and monitoring the traffic and delivering it to the application. Simple to use, it takes advantage of the concept of cloud computing and thus ensures high availability and high performance in security, delivery and mitigation of DDoS attacks because of its capabilities and resources. The centralization of the latter is a strategic issue for ongoing evaluation and professional tracking of security, in fact, it promotes a shared and supervised learning.

Key words : cloud computing, cloud-based web application firewall, Web security, hacking, Nginx, ModSecurity, reverse proxy, web application. horizontal scalability, high-availability, redundancy, VM-level fault tolerance, application-level fault tolerance

ملخص

تركز هذه الدراسة على مقاربة جديدة لتعهيد أمـن تطبيقـات الشـبكة، ألا وهـي الجـدران الناريـة المحوسبة سحابيا لتطبيقات الويب. هذا الحل هو خدمة حسب الطلب تقوم بتصـفية ومراقبـة حركـة مرور البيانات وتسليمها إلـى التطـبيق. سـهل الاسـتخدام، يسـتفيد مـن مفهـوم الحوسـبة السـحابية، وبالتالي يضمن توافرا و أداءا عاليين في مجال الحماية والتسليم والتخفيـف مـن الهجمـات الموزعـة للحرمان من الخدمة بسبب قدراته وموارده الكبيرة. مركزية هذا الأخير هي ميزة استراتيجية للتقييم المستمر والرصد المحترف للأمن، فإنه يعزز التعلم الألي المشترك و المُشرف عليه.

الكلمات الرئيسية : الحوسبة السحابية، جدار حمايـة التطبيقـات المحوسـب سـحابيا، الأمـن علـى شبكة الإنترنت، القرصنة، إنجـن إكـس ، مـود سـكيوريتي، الوكيـل العكسـي، تطـبيق ويـب، التوسّـع الأفقي، التواجدية العالية، الوفرة، تحمل الأخطـاء علـى مسـتوى الآلات الافتراضـية، تحمـل الأخطـاء على مستوى التطبيقات.

REMERCIEMENTS

Nous tenons tout d'abord à remercier Dieu le tout-puissant et miséricordieux, qui nous a donné force et connaissance.

En second lieu, nous tenons à remercier notre établissement, mais également notre promoteur Mr Bennouar pour son encadrement, sa confiance et sa patience.

À nos familles, gratitude et reconnaissance, nous leur devons, pour leur soutien et leurs encouragements ainsi qu'à nos amis et à toute personne ayant contribué d'une manière ou d'une autre à la réalisation de ce travail.

SOMMAIRE

PRÉSENTATION GÉNÉRALE

0.1 Introduction

Il faut dire que de nos jours les applications Web sont devenues omniprésentes. En décembre dernier, la toile, abondante de connaissances, d'échanges sociaux et commerciaux comptait plus de 633 millions de sites web dont environ 50 millions ajoutés au cours de l'année [1]. Ce nombre est en progression constante et ne cesse d'augmenter vis-à-vis de l'utilité et le déploiement des applications qui est de plus en plus facile et moins coûteux, mais qui dévoilent très souvent des vulnérabilités par lesquelles celles-ci, sans suivi professionnel ni remédiation, deviennent un handicap.

0.2 Présentation du domaine : Sécurité des applications web

Les applications Web sont sensibles du fait qu'elles soient « disponibles de partout et par tout le monde[2] », ce qui les rend « particulièrement exposées aux attaques par des personnes malveillantes qui décident de tenter de contourner les mesures de sécurité mises en place par défis technique et intellectuel, revendication ou tout simplement pour en tirer un profit [2]. »

Parmi les initiatives cherchant à classer de manière systématique les vulnérabilités au sein des applications Web figurent six catégories publiées par le consortium WASC[3] :

- L'authentification (Brute Force …)
- L'autorisation (Session hijacking, autorisation insuffisante ...)
- Attaques côté client (CS, XSS ...)
- Exécution de commandes (les injections SQL, XPath, SSI, BOF ...)
- Divulgation d'informations (Code Disclosure, Directory Indexation …)
- Attaques logiques (abus de fonctionnalités, Spamming, Flooding, DoS, DDoS …)

Et pour les enjeux stratégiques qu'impose la sécurité, il est devenu quasi-impossible aujourd'hui de concevoir une application sans penser à sa sécurité. Les développeurs et les spécialistes ont alors conçu des piliers de sécurité distincts, afin de soutenir ce besoin constant des applications comme : les frameworks de sécurité, Les IDS/IPS et les pare-feux applicatifs.

Ces solutions peuvent être implémentées différemment de l'exemple d'une API exploitée au niveau de l'application passant par des modules intégrés sur serveur à des équipements matériels spécialisés qui gèrent par exemple l'authentification et le filtrage des E/S.

0.3 Énoncé de la problématique initial : Les défis actuels du domaine

Assurer la sécurité de son application Web et la maintenir est un défi auquel sont souvent

confrontés les administrateurs des applications Web et les responsables informatiques des petites et moyennes entreprises.

L'utilisation des pare-feux d'applications Web tente de limiter ces risques, par exemple : la norme de sécurité PCI DSS[4] pour le paiement en ligne spécifie dans sa première condition de « construction et maintien d'un réseau sécurisé » : « l'installation et la gestion d'une configuration de pare-feu pour protéger les données des titulaires de cartes ».

Mais malheureusement, pour les pare-feux classiques on n'en trouve qu'un nombre très limité de solutions open-sources convaincantes dont une seule avec une communauté professionnelle (ModSecurity). Elles sont proposées comme solutions mono-serveur, généralement destinées aux experts (filtrage par expressions régulières directes) et restent inefficaces contre les attaques de masse comme le DDoS. La sécurité est une exigence vitale. L'assistance, le soutien et l'évaluation continuelle sont une responsabilité de professionnels qui n'est pas donnée à tout le monde. Un problème auquel l'externalisation de la sécurité tente d'apporter les réponses.

0.4 Énoncé sommaire des objectifs :

Pour faire face à ces problèmes, il est indispensable d'ajouter une couche indépendante de sécurité pour filtrer ces attaques, tout en étant :

- Performante ;
- Fiable : la capacité de traiter d'énormes trafics,
- Distribuée : pour assurer la haute disponibilité, pour diminuer les latences liées aux réseaux et augmenter la scalabilité du système ...
- Simple à déployer et à configurer.
- Maintenance et mises à jour professionnelles.
- La capacité de prendre des décisions dans certains cas.

Ces critères en matière de ressources et de services sont l'essence même de l'informatique en nuage. C'est ainsi que nous comptons atteindre nos objectifs de performance, fiabilité et distributivité au niveau du WAF.

Cette architecture de pare-feu distribué en nuage « CWAF » est un concept de services Cloud désigné par « Security as Service » qui aura l'avantage de :

- Résoudre un maximum de problèmes d'attaques par lots (Flood, DoS, DDoS) – avec la

centralisation des ressources.

- Simplicité de déploiement – une simple modification sur les registres DNS.
- Configuration simple grâce à une plate-forme IHM ergonomique (journalisation, rapports...).
- Gestion centrale :
  - Application rapide et professionnelle des patchs de sécurité.
  - Partage d'informations entre les sites sécurisés : l'analyse des logs et des attaques permettrait une meilleure vision de la situation et de meilleurs résultats en termes de prédictions (prévisibilité) et de traçabilité.
- Un « Business Model » pour les WAF.

0.5 Méthodologie suivie

Ce mémoire est rédigé dans le cadre d'une recherche, le sujet étant technologiquement nouveau, la documentation l'est aussi, d'ailleurs, elle ne date globalement que de ces quatre dernières années, dont une bonne partie publiée il y a quelques mois.

Principalement recueillie grâce à la toile de chez des spécialistes, organismes de standardisation comme le NIST, des projets communautaires comme OWASP et des professionnels de l'industrie comme pour les livres blancs ou les aperçus techniques. Majoritairement anglophones, on a donc pris le soin de les traduire tout en référençant ou en incluant le texte original.

0.6 Subdivision du mémoire

Outre la présentation générale servant à définir le sujet, ce mémoire est composé de quatre chapitres :

Au cours du **premier chapitre**, nous étudions quelques notions de la sécurité des applications Web. Ainsi que l'appui qu'apporte l'informatique en nuage aux pare-feux en regard des autres solutions.

Le **deuxième chapitre** décrit l'architecture de la solution que nous proposons, son infrastructure et ses différents composants.

Dans le **troisième chapitre**, nous développons plus sur les stratégies de défense et la mitigation des attaques.

Le **dernier chapitre** sera consacré à la réalisation et à l'aspect technique de l'implémentation, un

test de performance et de pénétration sera fait pour évaluer la solution réalisée.

Enfin, une conclusion générale pour terminer ce travail de mémoire.

CHAPITRE I: SÉCURITÉ DES APPLICATIONS WEB

La première question logique qui pourrait se poser, c'est pourquoi en faire un pare-feu applicatif Web distribué en nuage ?

Et pour y répondre, nous allons présenter dans ce chapitre le concept de la sécurité Web et où l'on est aujourd'hui avec les solutions proposées puis nous expliquerons, « pourquoi le pare-feu d'application Web ? » et « pourquoi le nuage informatique ? ».

I.1 Introduction

« Les technologies déployées dans les environnements Web ont fortement évolué ces dernières années. L'avènement du Web 2.0 a changé la manière de présenter et d'accéder aux ressources accessibles via le protocole HTTP. Ces nouvelles architectures ont considérablement augmenté la surface d'attaque et rendu bon nombre de serveurs plus vulnérables [5]. »

De nombreuses études publiques, comme celles des communautés OWASP, SANS, WASC ou CLUSIF nous révèlent que la grande majorité des vulnérabilités sont d'ordres applicatifs.

Top 10 OWASP
A1 - Injection
A2 - Broken Authentication and Session Management
A3 - Cross-Site Scripting (XSS)
A4 - Insecure Direct Object References
A5 - Security Misconfiguration
A6 - Sensitive Data Exposure
A7 - Missing Function Level Access Control
A8 - Cross-Site Request Forgery (CSRF)
A9 - Using Components with Known Vulnerabilities
A10 - Unvalidated Redirects and Forwards

Illustration 2: Liste des dix risques de sécurité

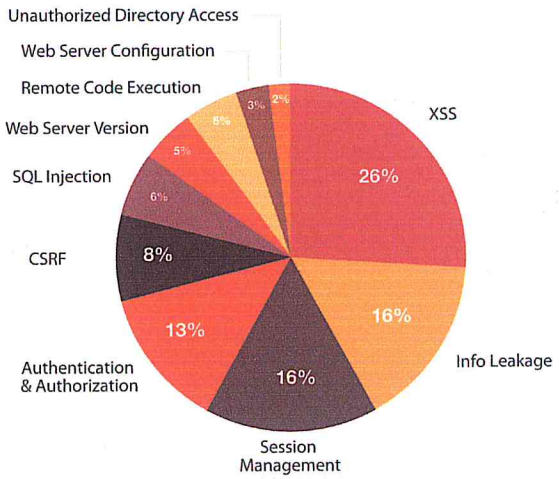


Illustration 1: Rapport de Cenzic, Inc spécialisé dans les solutions de sécurité et contributeur dans plusieurs organismes comme OWASP, WASC, SANS sur les vulnérabilités d'application Web (2013)

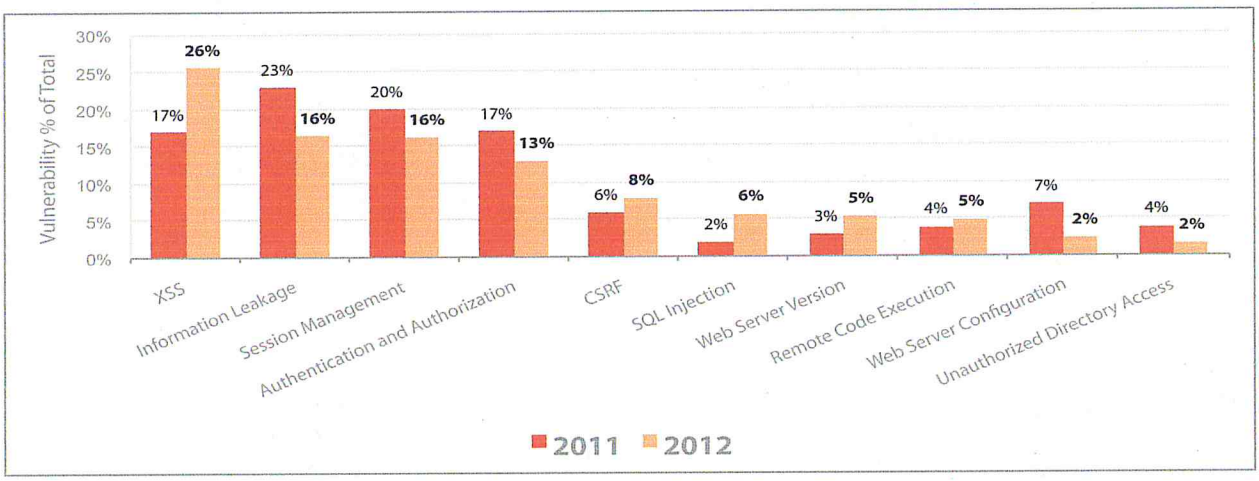


Illustration 3: Comparatif des taux de vulnérabilités en 2011 et 2012 (Cenzic Application Vulnerability Trends Report, 2013)

I.2 Concepts de la sécurité des applications Web

« … there are only two types of companies: those that have been hacked and those that will be [6] ». Il y a deux sortes d'entreprises : celles qui ont été hackées et celles qui le seront, disait Robert S. Mueller directeur du FBI.

I.2.1 Notions de la sécurité

La sécurité informatique est l'ensemble des techniques mises en place pour préserver, rétablir et assurer le bien fonctionnement du SI et la sûreté de ses ressources. Elle est définie en tant qu'une gestion des risques. « Or, comme on ne se protège efficacement que contre les risques qu'on connaît, il importe de mesurer ces risques [7]. »

Certains professionnels de la sécurité définissent le risque selon des formules comme :

$$Risque = \frac{Menace * Vulnérabilité}{Contre-mesure} \quad \text{ou} \quad Risque = Conséquence * Probabilité$$

La **menace** représente le type d'action susceptible de nuire dans l'absolu, tandis que la **vulnérabilité,** appelée faille ou brèche, représente le niveau d'exposition face à la menace dans un contexte particulier. Enfin, la **contre-mesure** est l'ensemble des moyens mis en œuvre en prévention de la menace. Il est nécessaire d'identifier les menaces potentielles, et donc de connaître et de prévoir la façon de procéder de l'ennemi afin de mieux comprendre comment il est possible de limiter les risques [8].

La sécurité est un investissement et une chaîne où chaque composant et chaque acteur apportent des risques [9], sur un type de menace donné. Il est nécessaire d'estimer les conséquences de ces risques, car l'application aura la sécurité de son maillon le plus faible. Une politique de sécurité reflète la stratégie définie pour ce système, une brèche par exemple ne doit pas emporter le système.

Il existe de nombreuses méthodes normalisées permettant l'analyse, la gestion des risques et la mise au point d'une politique de sécurité, l'exemple de MEHARI ou EBIOS.

I.2.2 Critères de sécurité

La sécurité peut s'évaluer suivant plusieurs critères :

- **Intégrité :** garantit que les données sont exactes et complètes et ne sont pas altérées.
- **Confidentialité :** consistant à assurer que seules les personnes autorisées ont accès aux ressources qui leur sont destinées.

- **Disponibilité :** le bon fonctionnement du système doit être maintenu et l'accessibilité aux services garantie.

- **Traçabilité :** assure que les interactions prises en considération sont tracées et que ces traces sont conservées et exploitables. Ce qui inclut la responsabilité des personnes [10] (preuve de non-répudiation et imputation).

I.2.3 Cibles et impacts des attaques Web

Johanne Ulloa[11] résume les éléments constituant l'interaction des WebApp :

- **L'utilisateur :** essentiellement, les attaques visant l'utilisateur cherchent à récupérer des éléments d'authentification afin d'usurper son identité et réaliser toutes les actions que ce dernier aurait pu faire.

- **Le serveur Web :** occupe une importance stratégique et les impacts les plus fréquents sont :

    - La modification de contenu : les conséquences du défacement sont variables en fonction de la sensibilité de l'image de l'organisme ;

    - L'ajout de contenu illicite ;

    - L'accès à des données confidentielles présentes ;

    - La prise de contrôle du serveur Web (exécution de commandes), le cas des zombis.

- **L'application Web :** dans ce cas, le but sera de faire en sorte que l'application Web ait un comportement différent de celui attendu. Par exemple : modifier le prix d'un produit.

- **Les bases de données :** elles sont la cible privilégiée des hackers. Les impacts vont du vol, à la suppression ou l'altération de données. Notons que bien des crimes passent complètement inaperçus, du fait que la victime ne s'en rend pas compte de l'effraction, l'exemple du vol.

- **Les services Web :** n'échappent pas aux attaques. De plus en plus déployés, ils ont une forte interaction avec les bases de données, ce qui fait d'eux une cible de choix.

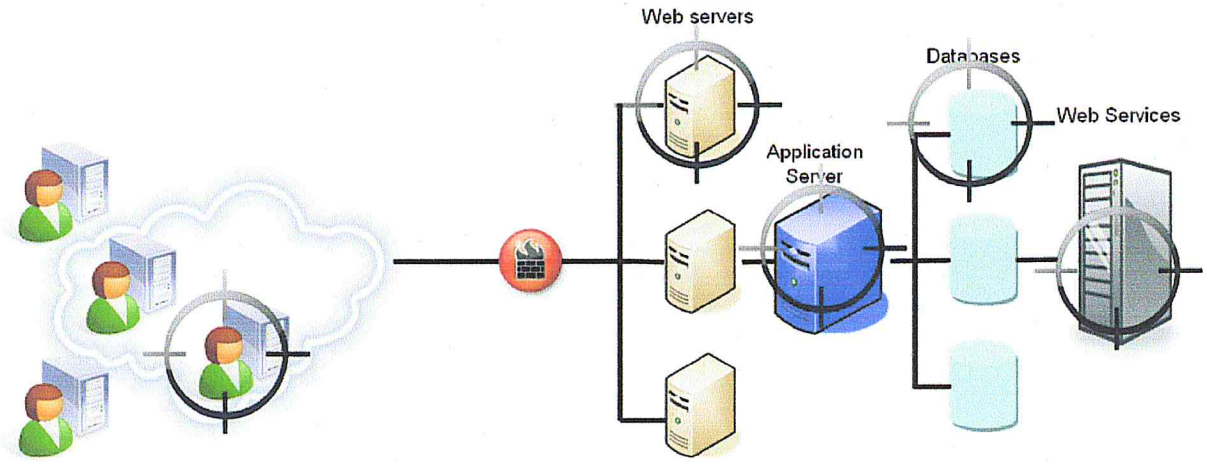


Illustration 4: Cibles des attaques Web (tous les maillons de la chaîne sont ciblés - Johanne Ulloa)

I.2.4 Évaluation des vulnérabilités

Il est évident que l'évaluation des vulnérabilités est une étape primordiale dans la validation d'un système, dans ce contexte, on trouve :

- Les **audits de sécurité** sont une évaluation du niveau de sécurité qui s'appuie sur un tiers de confiance comme les sociétés spécialisées en sécurité informatique afin de valider les moyens de protection mis en œuvre, au regard de la politique de sécurité [12].
- Les **tests d'intrusion** ou *pentest* sont une pratique d'audit technique. On distingue principalement trois catégories :
  - La méthode dite « boîte noire » (en anglais « *black box* ») consiste à essayer d'infiltrer le système de l'extérieur, autrement dit, sans avoir de connaissances préalables de celui-ci, afin de réaliser un test en situation réelle ;
  - La méthode dite « boîte blanche » (en anglais « *white box* ») consistant à tenter de s'introduire dans un système « transparent », c.-à-d., en ayant connaissance de l'ensemble du système, afin d'éprouver au maximum sa sécurité ;
  - Une troisième approche, dite « boite grise » où le testeur peut n'avoir qu'un nombre limité d'informations.

I.3 Solutions de sécurité Web

La sécurité dans le Web est un vaste domaine qui suit une multitude d'approches et peut être atteint de différentes manières. Toutefois, il est préférable de diversifier les défenses en mixant plusieurs solutions, méthodes ou approches pour augmenter le niveau de la sécurité, on appelle ça :

la sécurité multicouche, dite « défense en profondeur ».

La protection contre de nombreuses attaques comme les injections et les cross-site scripting est faisable depuis plusieurs niveaux ou « couches », on présentera les principales couches ainsi que les avantages et les inconvénients de chacune.

I.3.1 Codage sécurisé

Le meilleur endroit pour lutter contre les attaques d'applications Web est dans le code source lui-même. Les développeurs peuvent surpasser la plupart des types d'attaques en suivant les bonnes normes de codage telles que la manipulation des erreurs et la validation des entrées/sorties des données échangées avec les clients [13].

Donc, il s'agit de prendre la sécurité en tête dès le démarrage du projet, ce que l'on désigne par : « une stratégie de programmation défensive ». On peut énumérer une liste de mesures communes qui peuvent être employées par les développeurs pour améliorer la sécurité de leurs applications :

- **Code Source :** il est préférable que le code source soit clairement lisible, il faut vérifier que les commentaires sont inclus dans les délimiteurs du langage pour qu'elles n'apparaissent pas dans le source HTML reçu par le navigateur.
- **Authentification :** il s'agit de durcir les informations d'accès en les rendant plus aléatoires, elles doivent être sauvegardées sur la base de données dans un format bien crypté. Les actions sensibles des utilisateurs doivent être faites après une réauthentification. Il est également préférable que l'application informe l'utilisateur des tentatives d'accès échouées.
- **Manipulation de sessions :** le jeton de la session doit, autant que possible, être aléatoire, il est nécessaire d'en changer périodiquement pour minimiser les attaques de spoofing. L'application doit poursuivre ou bloquer les accès concurrents, cela peut bloquer les attaques de session hijacking. L'application doit terminer les sessions inactives automatiquement et achever totalement celles des utilisateurs déconnectés.
- **Manipulation des erreurs :** les erreurs doivent être contrôlées le maximum possible, les pages d'erreurs ne doivent pas fournir des informations sur l'état de l'application comme les noms de variables, les noms de fichiers ou les requêtes de base de données. Au lieu de ça, l'application devra écrire dans un fichier « error_log » spécial, ce fichier fournira les informations nécessaires pour le débogage.

- **Manipulation de la base de données :** les informations de connexion à la BDD doivent être stockées dans un endroit sécurisé. Si le nom de l'utilisateur et son mot de passe sont stockés en clair dans un fichier, il est nécessaire de vérifier que ce fichier ne peut être lu qu'à partir de l'application « permissions système - ACL ». Les requêtes SQL doivent être faites via un utilisateur qui a le minimum possible de privilèges sur la base de données. Il faut forcer les types de chacune des variables utilisées dans la requête (ex. : le champ numérique doit utiliser une variable de type entier).

- **Manipulation des fichiers :** les références vers les fichiers des « variables manipulées par les utilisateurs » doivent être nettoyées des caractères de parcours (ex. ../). Les fichiers doivent être récupérés à partir du même répertoire et ce dernier ne doit pas contenir le code source de l'application. Le répertoire qui contient les fichiers téléversés ne doit pas autoriser l'exécution.

- **Validation des entrées :** avant de filtrer les entrées, il faut les normaliser vers l'encodage standard (ex. Décodage d'URL), tous les caractères doivent être placés dans leurs représentations attendues. Les filtres basés sur les expressions régulières doivent être appliqués sur l'ensemble des entrées, cela implique que les opérateurs '^' et '$' doivent y être présents. Les données doivent être fortement typées dans leurs variables. L'application doit vérifier la longueur des données et les données supplémentaires doivent être tronquées et ignorées par l'application. Le contenu invalide des données est contrôlé, l'application vérifie d'une manière proactive les « mauvais » caractères (ex. ' ; < > et les parenthèses). L'application doit valider les données restreintes et spécifiques (ex. Les Wilayas de l'Algérie sont des chaînes de caractères, mais l'entrée est seulement une des 48 possibilités, l'entrée « AdsAE » doit être rejetée par l'application).

I.3.2 Frameworks :

Afin de rendre la sécurité moins complexe pour les développeurs, l'idée de mettre en œuvre des frameworks spécialisés dans la fourniture de routines pour la sécurité est surgie. Généralement, ces fonctions simplifient et en même temps, renforce l'authentification, l'autorisation, la cryptographie et la gestion des sessions.

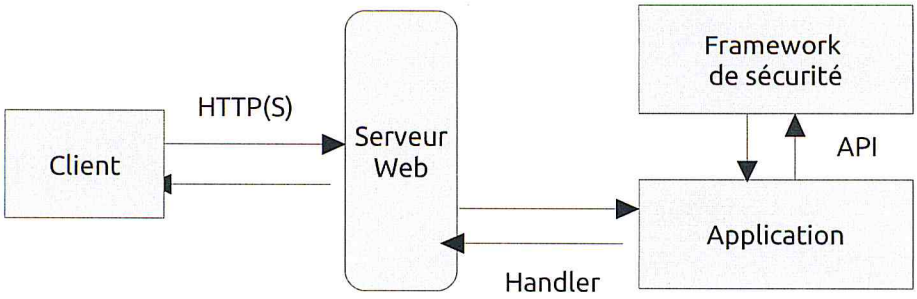


Illustration 5: Diagramme de communication général pour un framework de sécurité

Parmi les frameworks open-sources spécialisés dans la sécurité, on en trouve par exemple pour les applications écrites en Java :

- **Apache Shiro** conçu pour être un framework intuitif et facile à utiliser tout en offrant des fonctions de sécurité robustes.
- **Spring Security** est un framework Java/Java EE qui est largement utilisé pour la sécurité des applications basées sur le framework Spring.
- **jGuard** est basé sur JAAS (Java Authentication and Authorization Security) et est écrit pour les applications web ainsi que les applications autonomes, pour résoudre plus simplement, les problèmes liés au contrôle d'accès.

Aujourd'hui la majorité des frameworks de développement intègrent à leurs fonctionnalités la prise en charge de la sécurité, mais nécessitent d'avoir une maîtrise de l'API utilisée. C'est très utile pour construire des applications sur une stratégie de programmation défensive, mais que faire pour les applications déjà construites et désirant assurer plus leur sécurité. Les frameworks seuls sont insuffisants et inefficaces quand il s'agit d'attaques par déni de service distribué.

I.3.3 Intrusion Prevention System (IPS) :

Un IPS se place généralement dans les réseaux pour surveiller le trafic et le flux de paquets qui passent. Il agit de manière similaire à un système de détection d'intrusion (IDS) en essayant de faire correspondre les données dans les paquets grâce à une base de données de signatures ou en détectant les différentes anomalies de ce qui est prédéfini comme le trafic « normal ». En plus de ses fonctionnalités IDS, un IPS peut faire plus de logs et d'alertes. Il peut être programmé pour réagir à ce qu'il détecte. La capacité de réagir face aux détections est ce qui rend les IPS plus souhaitables que les IDS [14].

Les IPS peuvent être classifiés dans quatre (04) différents types selon l'objectif et la cible à sécuriser, il existe Network-based, Wireless et Host-based IPS, ou encore les NBA (Network Behavior Analysis), ce dernier examine le trafic réseau pour identifier les menaces qui génèrent des flux de trafic inhabituels, comme le déni de service distribué (DDoS) et certaines formes de logiciels malveillants.

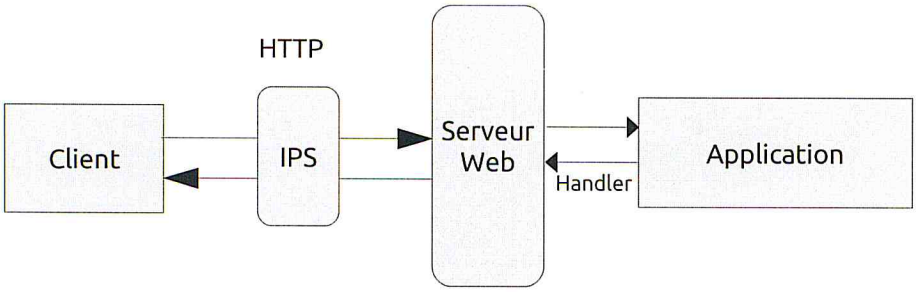


Illustration 6: L'un des déploiements possibles d'un IPS

Les IPS peuvent aussi être classifiés selon les méthodes de détection en trois catégories. Il y a la détection à base de signatures, la détection à base d'anomalies statistiques et la détection par l'analyse à états de protocole.

Il y a tout de même quelques inconvénients chez les IPS. Ils sont conçus pour bloquer certains types de trafic qui peuvent être identifiés comme potentiellement mauvais. Les IPS ne peuvent pas comprendre la logique des protocoles d'application Web. Ainsi, un IPS ne peut pas distinguer pleinement si une demande est normale ou malformée à la couche application (couche 7 – OSI).

I.3.4 Les pare-feux

En se référant à la définition de Cheswick et Bellovin [15]-[16], un pare-feu est élément ou un ensemble de composants placé entre deux réseaux ayant les propriétés suivantes :

- Tout le trafic transitant entre les deux réseaux passe nécessairement par le pare-feu ;
- Seul le trafic explicitement autorisé par la politique de sécurité appliquée localement est autorisé à passer au travers du pare-feu ;
- Le pare-feu est immunisé contre toute intrusion.

Le pare-feu est l'un des premiers équipements de sécurité réseau et en tant que tels, ils ont été soumis à de nombreuses évolutions. Suivant la génération ou leur rôle, on peut les classer de différentes manières [16]-[17] :

- **Stateless packet filtering** analyse le paquet indépendamment des autres et le compare à une

liste de règles définies, fonctionne principalement sur les trois premières couches OSI avec un bref aperçu de la quatrième couche pour avoir une idée de la source et du port destinataire, l'exemple des ACL Cisco sans état.

- **Stateful packet filtering** reprend le fonctionnement de son prédécesseur, mais opère jusqu'à la couche 4 et assure un suivi des échanges en tenant compte de l'état des connexions précédentes.

- **Application layery** appelées passerelles applicatives (application gateway). Filtrer la couche application exige de « comprendre » certains protocoles et applications (ex. : FTP, DNS, HTTP...), ceci est utile, car il sera capable de détecter un usage indésirable sur un port ou un service permis.

I.3.5 Pare-feux d'Applications Web « WAF »

Un pare-feu applicatif web (en anglais, Web Application Firewall ou WAF) est une partie logicielle ou un équipement matériel qui permet principalement de protéger les applications Web des attaques applicatives (SQL injections, Cross Site Scripting, injection de code…), en appliquant un ensemble de règles et de filtres sur les requêtes HTTP.

En personnalisant les règles à la demande, de nombreuses attaques peuvent être identifiées et bloquées. L'effort pour effectuer cette personnalisation peut être important et doit être maintenu, tant que l'application est modifiée [18].

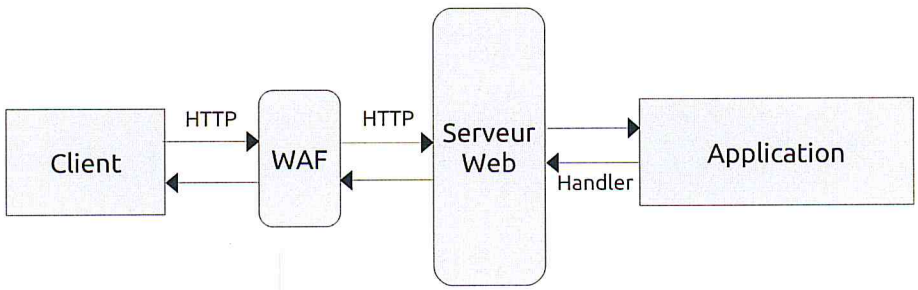


Illustration 7: L'un des déploiements répandu du WAF (Reverse Proxy)

Le niveau d'utilité et d'efficacité des WAF dépend de certains facteurs, comme le statut de l'infrastructure, l'application, son code source ou son développement.

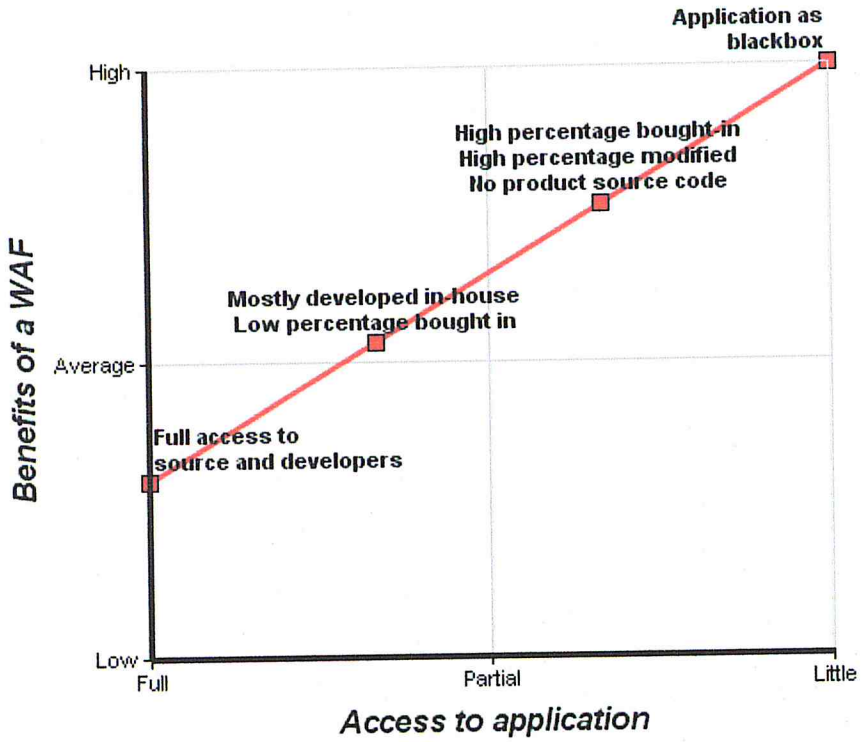


Illustration 8: Niveau d'utilité des WAF (OWASP)

I.3.5.1 Modèles de sécurité :

Un WAF suit généralement soit un modèle de sécurité positive ou négative. Un modèle de sécurité positive ne permet que le passage du trafic qui est connu pour être bon, tout autre trafic est bloqué. Un modèle de sécurité négative autorise tout le trafic et bloque les tentatives malveillantes. Certaines implémentations WAF tentent d'utiliser les deux modèles, mais en général, les produits utilisent l'un ou l'autre. « A WAF using a positive security model typically requires more configuration and tuning, while a WAF with a negative security model will rely more on behavioral learning capabilities [19]. »

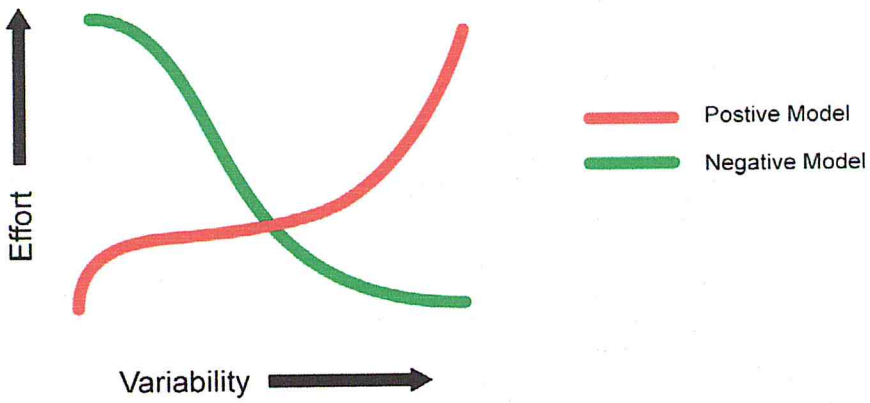


Illustration 9: L'effort nécessaire par les deux modèles positif et négatif en fonction de la variabilité des applications à sécuriser (Alan Murphy et Ken Salchow, F5 WhitePaper)

Le modèle positif est plus efficace en termes de taux de blocage des requêtes malveillantes, toutefois il nécessite plus d'efforts pour la génération des règles de la liste blanche à chaque variabilité au niveau de l'application.

I.3.5.2 Modes d'opération :

Il existe plusieurs modes d'opération pour les WAF, chaque mode a des avantages et des inconvénients qui nécessitent une étude pour évaluer celui approprié.

- **Reverse proxy :** le mode reverse proxy est le mode le plus commun du déploiement des WAF. Il consiste à mettre le WAF en frontal du serveur Web. 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. 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.
- **Transparent Proxy :** 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.
- **Host Based :** les WAF de ce type sont des applications installées sur le serveur lui-même comme plugin ou module en général.

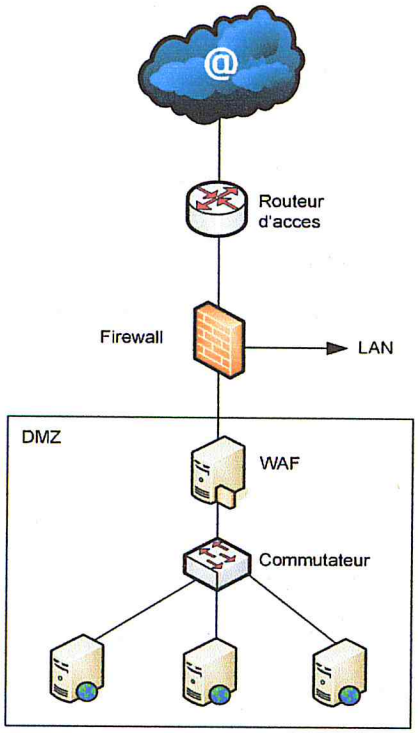


Illustration 10: Mode transparent d'un WAF. (CLUSIF)

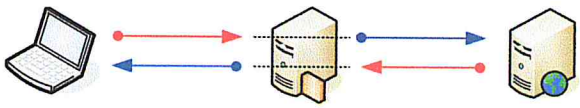


Illustration 11: Mode reverse proxy d'un WAF. (CLUSIF)

- **Monitoring Mode :** le pare-feu n'est pas aligné avec le sens de la communication, mais le surveille via un port, idéal pour les tests, il peut intervenir pour interrompre le trafic indésirable en envoyant au dispositif auquel il est lié (via TCP). Il n'inflige pas de latence supplémentaire, mais il n'est pas complètement efficace du fait qu'il n'est pas un intermédiaire direct.

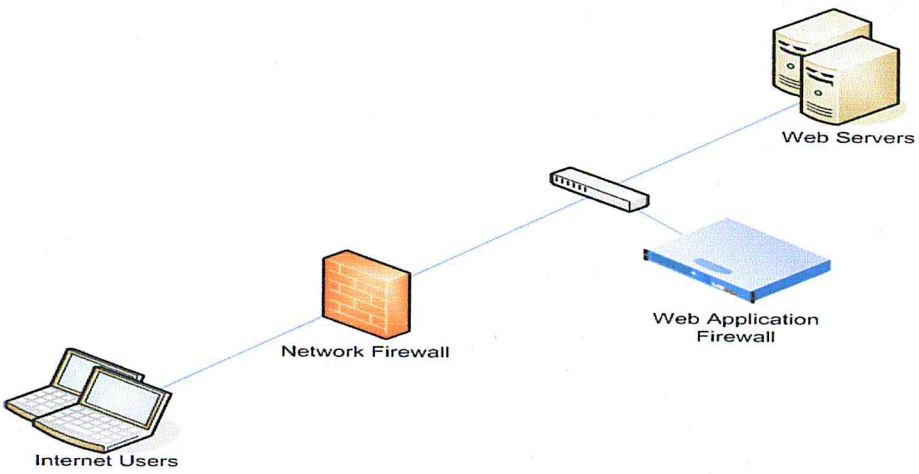


Illustration 12: WAF en mode monitoring. (Imperva)

I.3.5.3 Projets WAF non commerciaux et open sources :

Il existe des dizaines de projets WAF, mais la plupart sont conçus pour des tâches spécifiques ou pour des raisons de recherche, comme les projets de classe business, pour notre part, nous parlerons principalement de deux WAF très répandus.

I.3.5.3.1 ModSecurity :

Le WAF le plus utilisé est le pare-feu open source[a] ModSecurity [20]. ModSecurity est actuellement maintenu par Security Breach, une société qui vend également des appareils commerciaux qui contiennent cette solution, « ModSecurity is a web application firewall that can work either embedded or as a reverse proxy. It provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring, logging and real-time analysis [21]. ». Ce WAF fonctionne comme module sur de nombreux serveurs web dont Apache, MS IIS et NginX. Il utilise généralement un modèle de sécurité négative et a également plusieurs projets connexes qui contribuent à améliorer sa solution. Les règles de base de ModSecurity sont un ensemble de règles qui détectent les attaques Web les plus courantes.

ModSecurity a plusieurs fonctionnalités liées au domaine de la sécurité web, tels que :

- Le support des modèles positifs et négatifs.
- Règles basées sur les expressions régulières.
- Possibilité de filtrer le corps que ce soit pour la requête ou pour la réponse.
- Structuration des données sous des variables prédéfinies pour chaque requête.
- Possibilité de vérifier les fichiers téléversés.
- Injection de contenu dans la session HTTP.
- Blocage d'attaques à base de signatures.

I.3.5.3.2 Naxsi :

Naxsi est un module WAF pour NginX le serveur web et le fameux reverse-proxy, Naxsi est open source, hautement performant et à entretien réduit des règles [22]. Son but est d'aider à la sécurisation des applications Web contre les attaques de type SQL Injection, Cross Site Scripting, Cross Site Request Forgery, les inclusions de fichiers locaux et distants.

La différence principale entre Naxsi et d'autres WAF est qu'il ne bloque pas les attaques à base

a Sous « Apache License v2.0 »

de leurs signatures, mais il utilise un modèle plus simple où, au lieu d'essayer de détecter les attaques « connues », il détecte les éléments inattendus dans les requêtes HTTP.

Chaque type d'élément inhabituel va augmenter le score de la demande. Si la requête atteint un score considéré comme « trop élevé », la requête sera refusée et l'utilisateur sera redirigé vers une page spéciale. Il fonctionne un peu comme un système antispam.

I.3.5.4 Les limites des WAF :

Malgré que les WAF classiques comblent et complètent les limites des IDS/IPS dans ce qui concerne le Web, ils présentent souvent des problèmes de :

- Évolutivité : les WAF standards avec leur format brut ne supportent pas la scalabilité sur plusieurs machines, cela impose de nombreuses limites liées principalement aux performances et l'incapacité de résister aux attaques distribuées comme les attaques DDoS.

- Efficacité, facilité d'utilisation et de maintenance : les solutions actuelles, avec leur orientation à bloquer les attaques à base de signatures écrites en expressions régulières, l'installation depuis le code source comme module sur le serveur web et la configuration à base de fichiers concernent directement les experts en administration des systèmes et en sécurité Web.

I.3.5.5 Cloud-based WAF et les WAF distribués :

Dans le souci principal de supporter d'énormes trafics et de se défendre contre les attaques qui se basent sur les ressources (CPU, réseaux...), comme les attaques de DDoS, Flooding et Spamming, la nécessité de WAF scalables sur plusieurs machines s'est imposée.

L'architecture dWAF (distributed WAF) est réalisée sous la forme de composants séparés, pouvant exister physiquement dans différentes zones du réseau plutôt que d'en dépendre d'un seul dispositif. Ses ressources sont donc réparties, ce principe offre de la flexibilité et de la souplesse à la structure du fait qu'on peut rajouter ou soustraire des composants indépendamment les uns des autres. Cette approche est idéale pour les grandes infrastructures distribuées et virtualisées[23]-[24].

I.3.5.6 Le Cloud Compuing

L'une des tendances aujourd'hui pour mettre en œuvre la notion de service et de distributivité est le concept du **Cloud Computing** ou informatique en nuage. Devenu ces dernières années le « buzz » des services informatiques. Pour Wikipedia : « le cloud computing est un concept qui consiste à déporter sur des serveurs distants des stockages et des traitements informatiques

traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur [25]. ».

I.3.5.6.1 Définition

Le Cloud Computing permet la mise à disposition de ressources informatiques, délocalisées sur le réseau, sous la forme de services. Le *National Institute of Standards and Technology* en a donné une définition générale qui reprend ces principes de base : « L'informatique en nuage est un modèle permettant d'établir un accès par le réseau à la fois commode et à la demande, à un bassin partagé de ressources informatiques configurables (par exemple : réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement mobilisées et mises à disposition avec un minimum d'effort de gestion ou d'interaction avec le fournisseur de services [26]. ».

I.3.5.6.2 Caractéristiques :

Le NIST [26] définit dans son article de recommandations « The NIST definition of cloud computing » « cinq caractéristiques essentielles » dans un nuage :

1. **Le libre-service à la demande (On-demand self-service)** : les clients peuvent mettre à leur disposition les capacités de calcul et les ressources telles que l'espace de stockage ou la bande passante suivant leurs propres besoins, de façon automatisée, sans nécessiter d'interaction humaine avec le fournisseur.

2. **Un large accès au réseau (Broad network access) :** les ressources sont disponibles sur le réseau et accessibles via les mécanismes et les dispositifs standards tels que les PDA, tablettes ou ordinateurs.

3. **La mutualisation des ressources (Resource pooling)** : les ressources informatiques (physiques et virtuelles) du fournisseur sont mises en commun pour servir plusieurs clients à l'aide d'un modèle multi-locataire, elles sont affectées dynamiquement et réaffectées en fonction des besoins des clients.

4. **Une élasticité vivace (Rapid elasticity)** : c'est ce qu'on appelle une adaptation rapide aux variations des besoins du Cloud. Les ressources et les capacités peuvent être rapidement et automatiquement déployées et mises à l'échelle à tout moment et en fonction des besoins. Il est donc possible de disposer de plus de ressources très rapidement pour soutenir une forte demande par exemple ou d'en diminuer si celles-ci sont supérieures à ce qui est réellement nécessaire.

5. **Un service mesuré et facturé (Measured service) :** le système contrôle et optimise l'utilisation des ressources en s'appuyant sur une capacité de mesure à un certain niveau

d'abstraction approprié pour le type de service (par exemple, le stockage, le traitement, la bande passante ou les comptes d'utilisateurs actifs). L'utilisation des ressources et des services peut être surveillée, contrôlée et rapportée, en assurant la transparence à la fois pour le fournisseur et le consommateur. L'utilisateur ne paye alors que ce qu'il consomme.

I.3.5.6.3 Virtualisation :

Niveau implémentation, l'architecture Cloud bénéficie des technologies de virtualisation, un véritable atout qui permet de faire fonctionner plusieurs entités comme les machines virtuelles (ex. : systèmes d'exploitation) sur une seule machine physique, ce qui nous rapproche vers une utilisation optimale des ressources, grâce au partage de celles-ci et à la répartition des machines virtuelles en fonction des charges. Aussi, la virtualisation simplifie la création, la sauvegarde, le déploiement et la migration des machines virtuelles entre celles physiques.

Différentes techniques de virtualisation existent, parmi eux [27]-[28] :

- **L'isolation :** est une technique permettant d'isoler l'exécution des applications dans des contextes ou zones d'exécution. Les applications peuvent alors tourner dans un mode multi-instance même si elles ne sont pas conçues pour ça, par exemple plusieurs serveurs Apaches ou Nginx qui écoutent sur le même port 80. L'avantage de cette solution est qu'elle est très performante, car elle ne consiste pas réellement à virtualiser les systèmes d'exploitation, mais plutôt d'isoler les applications ou les ressources (système de fichiers, espaces mémoire, etc.). Elle est uniquement liée aux systèmes Unix et il n'est pas possible d'utiliser des noyaux différents. Un exemple des plus basiques est la commande Unix chroot pour isoler le répertoire racine d'un processus,

- **Les machines virtuelles complètes ou partielles :** les machines virtuelles complètes ont une simulation presque complète du matériel réel, ainsi le système invité fonctionnera nativement sans nécessité de modifications, cette approche est plutôt gourmande en ressources et est donc moins performante. Une autre approche pour les machines virtuelles est la simulation partielle de l'environnement, les systèmes invités peuvent avoir besoin de modifications.

- **Hyperviseur (moniteur de machines virtuelles)** : face aux inconvénients rencontrés par les deux premières solutions, une approche intermédiaire s'est installée qui consiste à dédier un système hôte qu'on appelle moniteur et qui s'intercale entre le matériel et les OS invités. Les deux solutions les plus connues sont le projet open source Xen[b] et ESX Server de Vmware. Si les OS invités fonctionnent en ayant « conscience » d'être virtualisés c.-à-d. qu'ils peuvent

b Sous le contrôle de la fondation Linux, depuis le 15 avril 2013

interagir avec le système hôte et sont optimisés pour ce fait, on parle alors de para-virtualisation.

D'autres techniques existent comme la virtualisation matérielle qui peut être, par exemple, intégrée au processeur. Ce qui simplifie la virtualisation et réduit la dégradation de performances.

I.3.5.6.4 Origine

D'un aspect purement technique, le Cloud Computing est loin d'être en soi une technologie nouvelle, le Cloud Computing provient de l'aboutissement de plusieurs technologies existantes antérieurement : internet et la virtualisation le tout appuyé sur un réseau fiable et à haut débit [29].

Ces premières traces remontent aux années soixante avec les *mainframes* où les utilisateurs accédaient depuis leurs terminaux à des applications fonctionnant sur des systèmes centraux, à cette époque l'expression Cloud n'était pas encore née, c'est avec l'apparition d'Internet que les architectes de réseaux la schématisaient par un nuage dans leurs croquis, on parlait alors de « the cloud » [30].

Par la suite, au début des années 2000, sont apparus des hébergeurs et des fournisseurs d'applications en ligne ASP (Application Service Providers) qui s'appuyaient sur une forme d'externalisation « Software as a Service » [30]-[31]. Et ce n'est que peu après, qu'Amazon Web Services « AWS », orienté vers les entreprises, et Google, orienté vers le grand public, font émerger le marché du cloud computing, suivis après par les autres éditeurs comme Microsoft et Oracle.

Pour l'aspect service, le Cloud Computing reprend des notions de produit qui ne sont pas nouvelles et s'appuie sur un puissant e-marketing et des modèles de consommation [32] bien connus comme le « pay per use service » (services payés à l'utilisation) et le « pay as you grow service » (évolution en fonction de la demande). Économiquement tentant, du fait que son concept de centralisation et de mutualisation réduit considérablement les coûts.

Note

Il se peut que l'on aille du mal à différencier entre Cluster, Grid et leur successeur le Cloud :

Le Cluster est un groupe d'ordinateurs généralement reliés par un réseau local (LAN), tandis que les deux autres sont plus échelonnables et peuvent être distribués géographiquement.

Quant au Grid Computing, il est plutôt utilisé en recherche, une sorte de superordinateur distribué sur plusieurs machines il est plus orienté « resource on demand ». La différence avec le cloud computing vient en réalité de son utilisation.

Cloud Computing Competence Center for Security [33] donne un tableau comparatif qui se base sur les caractéristiques définies par le NIST :

	Cluster	Grid	Cloud
On-demand self-service	No	No	Yes
Broad network access	Yes	Yes	Yes
Resource pooling	Yes	Yes	Yes
Rapid elasticity	No	No	Yes
Measured service	No	Yes	Yes

Illustration 13: Comparaison entre Cluster, Grid et Cloud Computing

I.3.5.6.5 Modèles de service

Pour l'ENISA : « *Le cloud computing est une nouvelle façon de délivrer les ressources informatiques et non une nouvelle technologie* [34]. ». C'est donc une nouvelle manière de fournir et d'utiliser les aptitudes des systèmes informatiques. Le NIST [26] distingue trois niveaux de service :

- **Le logiciel en tant que service (Software as a Service – SaaS)** : est comme son nom l'indique, un modèle de fourniture de logiciels hébergés à distance. L'utilisateur ne gère ni l'infrastructure du cloud ni la plate-forme où l'application s'exécute. Plus besoin d'installer l'application sur ses propres ordinateurs, le client y accède via sa connexion internet et n'a donc pas à mettre à jour ou à gérer le fonctionnement et la sécurité du logiciel, toutes ces tâches sont effectuées par l'éditeur. Ce qui simplifie la maintenance et le support. Ces applications sont accessibles à partir de différents périphériques clients par le biais d'une interface client légère, comme un navigateur Web (par exemple: le courrier électronique basé sur le Web), ou une interface spéciale. Parmi les exemples les plus connus, on retrouve : Google Apps, Microsoft Office 365 et OnLive.

- **PaaS (Platform as a Service)**, les PaaS sont des services Cloud destinés aux développeurs d'applications qui leur facilitent le déploiement de leurs applications dans le cloud à l'aide d'outils (langages de programmation, bibliothèques, services...) pris en charge généralement par le fournisseur. Les développeurs n'ont donc pas accès à l'infrastructure, mais ont le contrôle sur les paramètres de configuration de leur environnement d'hébergement (serveur, base de données...), leur permettant ainsi de se concentrer uniquement sur le développement de leurs applications et de ne pas perdre de temps sur leur déploiement. (exemples de PaaS: Google App engine ou AppFog).

- **IaaS (Infrastructure as a Service),** c'est la couche la plus basse des niveaux de services Cloud. Sur une IaaS l'utilisateur gère librement son infrastructure et peut définir et contrôler précisément les serveurs qu'il utilise, le système d'exploitation, le stockage, etc. (exemple d'IaaS: Amazon et son EC2).

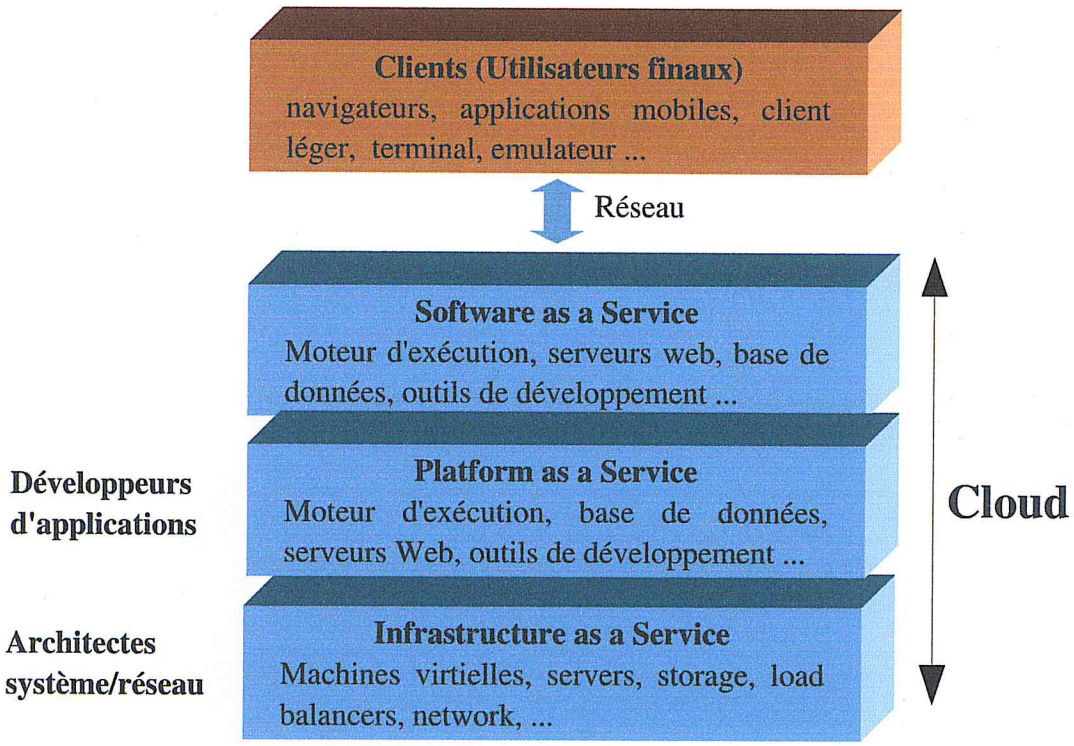


Illustration 14: Principaux modèles de service

À côté de ces modèles de services, on trouve une multitude de modèles de service construits sur la même expression « as a service ». En voici quelques-uns :

- **Desktop as a Service :** est la mise à disposition de bureaux virtuels hébergés à distance et accessibles via les terminaux clients.
- **Security as a Service (SecaaS) :** consiste à offrir un service de sécurité à la demande selon un modèle d'affaires précis. Différents services sont regroupés dans cette catégorie comme : Anti-virus as a service, WAF as a service, Backup as a service, Identity as a service ...
- **STorage as a Service (StaaS) :** correspond au stockage de fichiers chez des prestataires externes, qui les hébergent pour le compte de leurs clients, voici quelques exemples : Amazon Simple Storage Service, Dropbox, Google Drive, iCloud, SkyDrive, Ubuntu One, Windows Live Mesh, Wuala

I.3.5.6.6 Modèles de déploiement

Ces modèles de déploiement, tels que définis par le NIST :

- **Public Cloud,** l'infrastructure du Cloud public est accessible et partagée par plusieurs entités l'exemple du cloud d'Amazon qui est accessible à tous.
- **Community cloud**, le Cloud communautaire partage l'infrastructure entre plusieurs entités ou organisations soutenant une communauté spécifique dont les préoccupations et les intérêts sont communs (objectifs, sécurité, politique, compétence, etc.). Il peut être géré par la communauté ou par un tiers.
- **Hybrid cloud,** l'infrastructure est composée de deux nuages distincts ou plus (privé, communautaire ou public), qui demeurent des entités uniques, mais sont liés entre eux par une technologie standardisée ou exclusive qui permet la portabilité des données et des applications. L'intérêt est par exemple de stocker les données sensibles sur son cloud privé et de se procurer de la puissance de calcul sur un cloud public. Ou encore de faire appel au cloud public lors d'un pic de charge.

- **Private Cloud,** l'infrastructure du cloud privé est réservée et exploitée par une seule entité (entreprise, organisation, business unit, etc.). Cette infrastructure cloud peut appartenir et être contrôlée par l'entité elle-même ou par un tiers spécialisé et peut être hébergé sur le site ou à l'extérieur.

I.3.5.6.7 Critiques

Avec tout ce que le Cloud offre comme avantages, il présente des inconvénients liés à sa nature :

- L'utilisation des réseaux publics, entraîne des risques supplémentaires de cyberattaques liés à la sécurité du cloud. (par ex. : le *storage)*.

- La sécurité du Cloud devient cruciale, vu la sensibilité de certains services. Le service en lui-même peut devenir une vulnérabilité et une source d'attaque. En 2009, par exemple, un cheval de Troie a utilisé illégalement un service du cloud public d'Amazon pour infecter des ordinateurs [35].

- Le client d'un service devient très dépendant de la qualité du réseau et à la disponibilité du fournisseur, l'exemple des défaillances subies sur les services Google (Gmail, Document...) et Cloud d'Amazon EC2 [36].

- Juridiquement parlant, le cloud n'a pas de statut clair notamment dû à l'absence de

localisation précise des données. La question de la confidentialité est toujours posée, aux États-Unis par exemple les serveurs du cloud sont soumis au Patriot Act [37], qui autorise des perquisitions aux données sans la requête d'un juge.

Parmi les nombreuses critiques qui furent publiées : Larry Ellison, le PDG d'Oracle [38] voit le Cloud comme un phénomène de mode dans l'industrie de l'informatique pour regrouper tout ce qui se fait déjà. Rejoint par Richard Stallman [39] fondateur du projet GNU[c] qui le décrit comme une stupidité où les utilisateurs perdent le contrôle de leurs applications. Il le considère comme un concept marketing.

Steve Wozniak, cofondateur d'Apple, prévoit des « problèmes horribles » à venir avec le développement croissant du cloud computing et l'externalisation des données. Déclarant [40] qu'avec le nuage, rien ne nous appartient et que plus on transfère dans le nuage, plus on perd le contrôle.

Enfin, il est vrai que plusieurs critiques sont objectives, depuis l'apparition du Cloud, mais il faut reconnaître que le concept du Cloud a réussi à commercialiser l'informatique d'une façon innovante. On ne peut fuir cette réalité ni nier le fait que ça soit révolutionnaire, en se propageant rapidement et largement, conquérant de nombreux domaines et prévoyant un large horizon de l'avenir.

I.3.5.7 Les Pare-feux applicatifs basés sur le cloud

C'est la solution sur laquelle notre projet puise sa recherche, un pare-feu applicatif Web distribué en nuage (en anglais : *Cloud-based Web Application Firewall* ou *CWAF)* est un membre de la famille des technologies de sécurité Web (Saas-based Web application Security) et un service Cloud désigné souvent sous le nom Security as Service (Secaas) ou Firewall as a Service.

Cette technologie est un peu spéciale en raison du fait qu'elle peut être une plateforme agnostique et ne nécessite aucun changement matériel ou logiciel sur l'hôte client, mais on exige généralement un changement DNS avec lequel tout le trafic Web est routé via le WAF où il est inspecté et les menaces sont déjouées.

Le cloud-based WAF a l'avantage d'être centralisé, ce qui signifie que les informations de détection de menace sont partagées entre tous les locataires du service. Il en résulte de cette collaboration des taux de détection améliorés et des faux positifs réduits.

Comme d'autres solutions basées sur le cloud, cette technologie est élastique, évolutive et est généralement offerte comme un « pay as you grow service ». Cette approche est idéale pour les

c GNU's Not UNIX, pour un système d'exploitation complètement libre

applications Web hébergées sur le cloud ou les sites des petites ou moyennes entreprises qui nécessitent de sécuriser leurs applications web, mais ne disposent pas ou ne sont pas en mesure de faire des changements logiciels ou matériels à leurs systèmes.

I.3.5.8 Architecture

On reconnaît généralement deux architectures, la première est moins utilisée que la seconde :

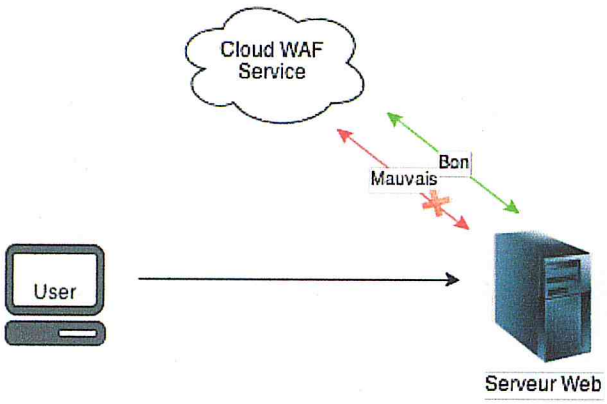


Illustration 16: Architecture CWAF basée sur l'interrogation

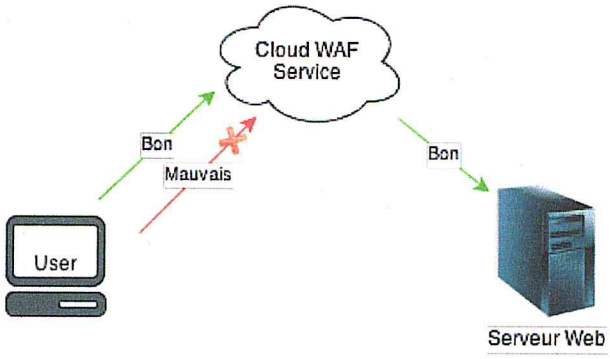


Illustration 15: Architecture CWAF comme intermédiaire

Par interrogation

Consiste à interroger le CWAF pour ensuite prendre la décision, une sorte d'API distante. Un exemple d'utilisation est l'ajout de code au sein de l'application web pour faire valider les entrées/sorties par le pare-feu.

Inconvénients :

- Nécessite une maîtrise de l'API ou de l'implémentation.
- Rend plus complexe le processus de durcissement du serveur web (connexions externes, vulnérabilité de l'infrastructure face aux attaques DDoS) [41].

Mandataire

Son principe de fonctionnement est similaire à un reverse proxy et peut être fait avec la modification de l'entrée DNS du serveur web pour passer par le Cloud.

Inconvénients :

- Dans le cas DNS, le serveur est toujours exposé sur internet, ce qui rend le WAF contournable, si l'adresse IP réelle du serveur web venait à être découverte et à être utilisée directement pour accéder à l'application web.

=> Peut être renforcé avec une liaison stricte au CWAF depuis l'application ou son infrastructure.

Pour la méthode basée sur l'interrogation, on voit que c'est un peu double travail spécialement pour une petite infrastructure, un petit serveur qui doit recevoir les requêtes client puis les renvoyer

au CWAF et attendre sa réponse. Mais on est moins dépendant de la disponibilité du fournisseur de services, s'il venait à tomber en panne, vu que l'infrastructure du client est indépendante de celle du CWAF, contrairement, si ça venait d'arriver pour les CWAF intermédiaires où le client s'étaient automatiquement avec l'infrastructure de son fournisseur. Bien qu'un service Cloud est qualifié pour être hautement disponible et que ces cas sont étudiés et gérés (ex. : failovers).

I.3.5.9 Les avantages :

Les avantages de CWAF sont nombreux et surpassent la majorité des limites qu'imposait une architecture WAF standard. Peut-être le plus important, est la centralisation des ressources, ce dernier facteur joue le rôle clé dans la défense contre les attaques distribuées telles que DDoS, car si l'on considère qu'il y a un trafic malveillant sur Internet et qu'il est estimé par 20Gbps, et qu'il y a X sites Web qui veulent se défendre contre ce trafic. Il faudra que chaque site possède une bande passante supérieure à ce trafic malveillant d'un côté et de posséder les ressources qui peuvent exploiter cette bande passante d'un autre côté, en tout, on a besoin de « *X*20Gbps+(le trafic légitime de chaque site)* ». Toutefois, cela peut être réduit favorablement en centralisant les ressources par l'utilisation de l'approche CWAF, il ne sera dans le besoin que d'une bande passante de 20Gbps+Trafic légitime pour sécuriser l'ensemble des X sites. Autres avantages des CWAF :

- Partage d'informations entre les sites sécurisés.
- Résoudre le maximum de problèmes d'attaques en masse (Flood, DoS, DDoS).
- Simplicité de déploiement du côté webmaster (ex. : pour se soumettre à la PCI Compliance).
- Configuration simple grâce à une plate-forme IHM ergonomique (rapports...).
- Gestion centrale, application rapide des patchs de sécurité.
- Un modèle d'affaires « Business Model » pour les WAF.

I.3.5.10 Implémentations actuelles :

Il y a quelques efforts d'open source pour réaliser des WAF distribués comme openWAF de la société « Art of defence », annoncé comme mort après deux ans de son démarrage. Il y a aussi IronBee de la société Qualys, le premier développeur de ModSecurity, ce projet est en phase de développement, il a été remis à zéro en décembre 2012 pour changer totalement sa structure.

À part ces deux-là, toutes les solutions Cloud WAF existantes aujourd'hui sont commerciales :

- XyberShield est probablement le seul CWAF public qui opère autrement qu'en reverse proxy. Il ne nécessite pas de changement DNS, mais à la place il s'appuie sur un script local

de 20k et une communication constante à sa plateforme de service composé de 20 datacenters dans le monde [42].

- CloudFlare, le fameux CDN, est à l'origine un pare-feu sur le nuage. L'infrastructure du service s'appuie sur une version modifiée de Nginx et intègre la technologie SPDY développée par Google [43]. En mars 2013, il compte 23 datacenters [44].
- En 2010, Imperva lance Incapsula, une start-up israélienne, qui fournit un cloud-based WAF pour les petites et moyennes entreprises [45].
- Depuis 2011, United Security Providers fournit un Secure Entry Server en tant que pare-feu d'applications Web pour le Cloud Amazon EC2 [46].
- Akamai Technologies propose un cloud WAF qui incorpore des fonctionnalités avancées telles que le contrôle du débit et des règles personnalisées lui permettant de traiter à la fois la couche 7 et les attaques DDoS. « Les paramètres de sécurité du noyau WAF reposent sur le jeu de règles normalisé et hautement sécurisé ModSecurity [47] ».
- Août 2011, BinarySEC lance son pare-feu nuageux « EasyWAF » qui se base le changement DNS et qui propose des services de CDN et d'accélération du contenu.
- Depuis 2012, Penta Security Systems, Inc offre un CWAF public nommé WAPPLES V-Series [48] avec le partenariat stratégique des FAI tels que KT en Corée.
- Nexusguard propose sa solution ClearWeb comme Cloud-based WAF avec trois modes de déploiement : tunnel GRE, circuits directs et Network Proxy (changement DNS) [49].

Généralement, ces pare-feux d'applications Web se doivent de fournir une protection contre le Top 10 des attaques OWASP. Ils essayent d'être conformes aux standards de sécurité et aux recommandations PCI (Payment Card Industry) et offrent un support pour le chiffrement SSL.

Les CWAF reverse proxy, se basant sur le changement DNS, sont souvent accompagnés par des services CDN pour optimiser la transmission et la livraison du contenu notamment grâce aux systèmes de cache et réservent généralement une protection contre les attaques DDoS dans leurs « Packs Pros ».

Les inconvénients qu'on rencontre en général sur ces pare-feux sont relatifs à l'adaptation de la sécurité, par exemple les règles de filtrage sont souvent non personnalisables et communes à tous les clients, et le mode d'apprentissage généralement non disponible ou inefficace.

Le tableau ci-dessous résume un comparatif de fonctionnalités entre quelques CWAF

commerciaux connus :

CWAF	Imperva Incapsula	CloudFlare	XyberShield	WAPPLES V-Series
Type	Commerciale	Commerciale	Commerciale	Commerciale
Disponibilité d'informations techniques	Non	Très limitée	Très limitée	Non
Traitement	Cloud	Cloud	Client+Cloud	Installable sur IaaS
Redirection Trafic	DNS	DNS	Module ajouté sur l'App	Gérer par Client
Scanne la réponse	Oui	Non	Non	Non
Personnalisation	Par défaut	Par défaut	Selon la catégorie	Avancée
Facturation	Par niveaux de sécurité	Par niveaux de sécurité	Par requêtes	Par produit Forfait de support par an
Utilisation	Interface Web	Interface Web	Logiciel agent (API)	Installation dans une infra. privée / Gestion avancée
SPAM/BOTS	Par liste noire	Par liste noire	Par liste noire	-
Distribution Internationale	Oui	Oui	Oui	-

En fin de mars 2013, lors de la journée de la sécurité des systèmes d'information (JSSI) organisée annuellement par l'Observatoire de la sécurité des systèmes d'information et des réseaux (OSSIR). La société Synacktiv [41] spécialisée dans la sécurité des SI publie ses tests « WAF contest », qui traitent le sujet des cloud-based WAF afin de constater si ces solutions améliorent-elles le niveau de sécurité et si le service est à la hauteur du tarif et de la brochure marketing proposés.

Pour ça trois pare-feux sont choisis et mis à l'épreuve : XyberShield, CloudFlare, Incapsula. Prenant le premier choix **XyberShield**, la grande fierté de la société Marsys : « *XyberShieldis is the most effective web application firewall (WAF) solution on the market today.* ».

XyberShield ne supporte actuellement que les environnements : .NET, Java (bibliothèques compilées) et PHP (plus facile à analyser).

Le filtrage n'est pas trop dur à contourner, par exemple les techniques d'obfuscation classiques ne sont pas détectées, exemple de SQL Injection (OWASP Top 10 2013-A1) :

- Sera bloquée : ...?user=admin%27%20**or**%201%3D%271
- Ne sera pas bloquée : ...?user=admin%27/***!or***/1%3D%271

Les erreurs sont divulguées et leur algorithme fait maison (à base de XOR) pour l'échange sécurisé avec le WAF n'est pas dur à casser (La clef utilisée est commune à toutes les installations Xybershield et correspond à une clé de tutoriel très répandue sur Internet), de plus l'interface d'administration se révèle vulnérable et peut causer des fuites de données des autres clients...

CloudFlare est une solution d'accélération HTTP intégrant un WAF. 20$/mois + 5$ pour chaque nouveau site web, le WAF ne réagit pas … et 200$/mois pour l'édition des règles de filtrages ModSecurity, autant installer mod_security sur le serveur web et le configurer...

Incapsula fonctionne par redirection DNS, à l'aide d'un CNAME vers la zone x.incapdns.net. Son filtrage est plus avancé que celui de XyberShield avec une liste noire qui est relativement bien construite même si des éléments sont manquants.

Les techniques d'obfuscation sont bien repérées, le blocage est paramétrable et peut aller jusqu'au bannissement de l'IP.

Pour contrer les XSS (OWASP Top 10 2013-A3), un mécanisme de rating est utilisé : si une requête contenant plusieurs mots-clefs marqués comme risqués la requête est rejetée, toutefois, il ne prend pas en compte de très nombreux mots-clés, pourtant importants (self, parent, this...).

Le filtre reste tout de même contournable, par exemple pour l'injection SQL : « UNION SELECT » sera bloquée, mais avec un mot-clé « UNION DISTINCT SELECT », elle ne le sera pas.

Les entêtes HTTP ne sont pas contrôlés et un conflit peut être provoqué en injectant des entêtes.

Il est naturel que l'interface d'administration offre des fonctionnalités sensibles, comme l'activation ou la désactivation du filtrage, la définition de l'adresse IP vers laquelle renvoyer les requêtes validées, la gestion des listes blanches, etc. Un accès illégitime à l'interface d'un client est donc un risque majeur. Alors que cette interface est affectée par de multiples failles XSS et toutes les fonctionnalités sont vulnérables aux « Insecure Direct Object Reference » (OWASP Top 10 2013-A4).

Synacktiv finit par conclure que les WAF en mode SaaS mis sur le marché sont encore relativement immatures. Leur facilité de déploiement doit être mise en parallèle avec la protection limitée qu'ils offrent. En outre, le niveau de qualité global de ces produits est faible :

- Méconnaissance des techniques de contournement triviales.
- Paramétrisation exposée aux risques.
- Utilisation de chiffrements faibles.
- Sociétés n'appliquant même pas leurs propres recommandations pour leurs serveurs web (filtrage des accès directs chez Incapsula).
- Erreurs de programmation triviales dans les interfaces d'administration.

I.4 Notre solution GhaimNet

La solution que l'on propose « GhaimNet » est aujourd'hui la seule solution libre déployée et maintenue, mais pourquoi tenons-nous à l'Open Source ?

Premièrement, ce travail est censé être une recherche, dont le but est d'étudier les solutions existantes et d'en proposer d'autres. C'est une bonne chose de présenter un aspect théorique abstrait, mais dans un monde tel que celui de l'informatique, tenir compte de l'implémentation, c'est se donner une propulsion vers l'avant pour aboutir à de futurs travaux ou à des collaborations. Si l'on doit comparer ou tester des solutions, des concepts ou des principes, on n'a pas à recommencer du néant.

Dans une interview avec Richard Stallman il disait que : « À une époque où beaucoup de ce que nous faisons, nous le faisons avec des ordinateurs, si on n'a pas de la liberté dans nos ordinateurs, ça sera difficile pour nous de défendre ou de combattre pour la liberté dans d'autres domaines [50]. ».

L'expression « logiciel libre » veut dire que le logiciel respecte la liberté de l'utilisateur et de la communauté. En gros, les utilisateurs ont la liberté d'exécuter, de copier, de distribuer, d'étudier, de modifier et d'améliorer le logiciel. Avec ces libertés, les utilisateurs (à la fois individuellement et collectivement) contrôlent le programme et ce qu'il fait pour eux. Quand les utilisateurs ne contrôlent pas le programme, c'est le programme qui les contrôle. Le développeur contrôle le programme et par ce biais contrôle les utilisateurs [51].

Le fait de dépendre d'une industrie fermée où l'on devient des consommateurs aveuglés est critique, imaginez un Cloud auquel nous confions aveuglément nos données sans connaître les stratégies pratiquées ou la confidentialité garantie. C'est à peu près la même chose de ce que nous

avons vécu d'une certaine manière avec les systèmes d'exploitation. L'open source est une sorte de garantie pour continuer à progresser sur l'échelle technologique et civilisationnelle.

Notre but est donc de créer une solide communauté pour un développement puissant et durable. Cette solution étant open source, plusieurs possibilités et adaptations s'offrent à nous, par exemple : elle peut être déployée en SaaS hébergement pour sécuriser les applications ou utiliser comme un WAF privé.

En second, nous cherchons à améliorer le rendu de cette solution par rapport aux autres, pour offrir une meilleure expérience avec ces pare-feux qui, grâce à leurs centralisations et à leurs partages d'informations induisent à un apprentissage collectif supervisé hormis celui individuel pour chaque application.

Le noyau du pare-feu qu'on a développé est un fork basé sur ModSecurity, ce qui nous permet de profiter de sa puissance et d'aller au-devant pour combler et corriger ses limites.

Zero Sciene Lab[d] rédige un rapport dernièrement [52], comparant le WAF de ModSecurity à deux autres CWAF commerciaux : CloudFlare et Incapsul.

La pénétration via des tests « boîte noire » a été menée contre les trois services, en appliquant des techniques d'évasion de filtres connus, pour contourner leur solution de pare-feu d'applications Web, en se basant sur des scénarios réels et une variété de vecteurs d'attaques.

CloudFlare et Incapsula offrent tous les deux des plans pour des comptes gratuits et payants. Les fonctionnalités de sécurité avancées WAF ne sont incluses que dans les plans payés.

Les tests ont été faits par rapport au plan de Bussiness **Incapsula** :

Illustration 17: Les plans d'Incapsula

d Macedonian Information Security Research and Development laboratory www.zeroscience.mk

Et le plan Bussiness **CloudFlare** :

Illustration 18: Les plans de CloudFlare

ModSecurity, comme indiqué précédemment, est une solution gratuite et open source. Il a été alimenté avec l'ensemble de règles de base OWASP « OWASP Core Rule Set ».

Le tableau ci-dessous résulte de ces tests :

	CloudFlare $200/month	ModSecurity Free	Incapsula $59/month
Total SQL Injection Tests	54	54	54
SQL Injection Bypassed	54	0	1
SQL Injection Blocked	0	54	53
Total XSS Tests	46	46	46
XSS Bypassed	46	0	3
XSS Blocked	0	46	43
Total LFI/RFI Tests	23	23	23
LFI/RFI Bypassed	23	2	4
LFI/RFI Blocked	0	21	19

Illustration 19: Pentest - Zero Sciene Lab

On voit que ModSecurity a le taux le plus élevé de blocage et de détection de vulnérabilités et d'attaques. CloudFlare, même étant une solution commerciale, elle n'a bloqué aucune attaque. Il faut dire que le résultat de CloudFlare est vraiment décevant et ce n'est pas la seule étude [53], cela dit, CloudFlare se base sur une heuristique d'apprentissage plutôt que des règles statiques [54]. Un

apprentissage est un bon réflexe pour reconnaître de nouvelles attaques et s'adapter à son environnement, mais il ne faut pas laisser un nouveau-né dans la jungle sans protection ou une configuration basique. Incapsula est plus sophistiqué dans le contrôle de la sécurisation des sites, il y en a eu zéro faux positif, contrairement à ModSecurity qui lui a montré un comportement plus agressif en présentant un nombre élevé de faux positifs, en raison de sa conception et aux règles strictes utilisées.

La différence avec les solutions commerciales est qu'elles sont fermées, elles ne sont pas mises à la critique ou à la correction par la communauté et généralement on ne peut faire que des tests du genre boîte noire, à la différence de ModSecurity qui est maintenu et soutenu par une large communauté de plus de dix ans d'expérience.

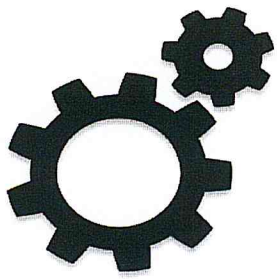
Voilà donc, pourquoi l'on doit adapter ModSecurity aux sites des clients via des fonctionnalités ou des apprentissages automatiques. Il doit aussi être adapté à la distributivité, dans ses interactions, sa lecture des configurations personnalisées des clients, ses sauvegardes de logs, le moyen de factorisation pour le service, etc. Ces points seront traités dans la partie architecture du système.

I.4.1 Conclusion

Bien que la sécurité se tienne de différentes manières, on a pu énumérer quelques solutions principales comme : les framworks applicatifs, les équipements IDS/IPS, les pare-feux, mais également leurs limites. On a étudié en bref la nouvelle tendance aujourd'hui à l'externalisation de la sécurité des applications Web qui est très prometteuse et tend à combler les limites de ses prédécesseurs.

CHAPITRE II: ARCHITECTURE DU SYSTÈME

Après un état de l'art sur le concept de la sécurité Web et des solutions existantes et plus précisément les pare-feux d'applications web et l'appui de l'informatique en nuage, nous nous attacherons à expliquer notre conception pour la solution que nous proposons.



II.1 Introduction

Pour construire un service basé sur la philosophie du Cloud Computing, il est indispensable de concevoir une architecture spéciale en termes de ses caractéristiques :

- Haute disponibilité (Tolérance aux pannes/redondance optimale).
- Évolutivité horizontale (performance).
- Sécurité (accès restreints).

II.2 Généralités

II.2.1 Principe de fonctionnement

Afin de traiter les requêtes envoyées vers les sites web des abonnés, le système doit se positionner au milieu. Il opère comme un reverse proxy. Il traite les requêtes et s'il les juge comme faisant partie du trafic légitime, il les renvoie vers le serveur d'origine et redirige ensuite la réponse vers le client.

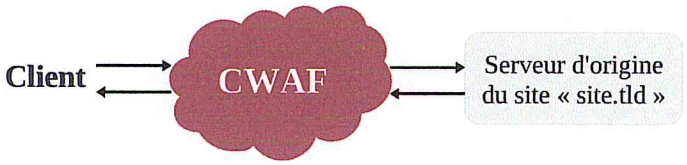


Illustration 20: Interaction du système avec une requête légitime

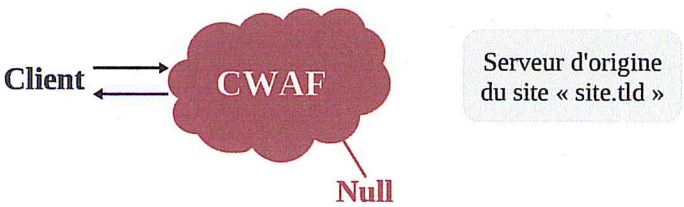


Illustration 21: Interaction du système avec une requête d'attaque

II.2.2 Comportement général

Le système réagit face au risque, non seulement, s'il se situait dans la requête, mais également le contre, s'il se trouvait dans la réponse elle-même. Cela nous permet de défendre les applications contre les attaques détectables au retour des réponses comme les attaques de type : Information disclosure, Directory indexation, Invalid redirections et bien d'autres.

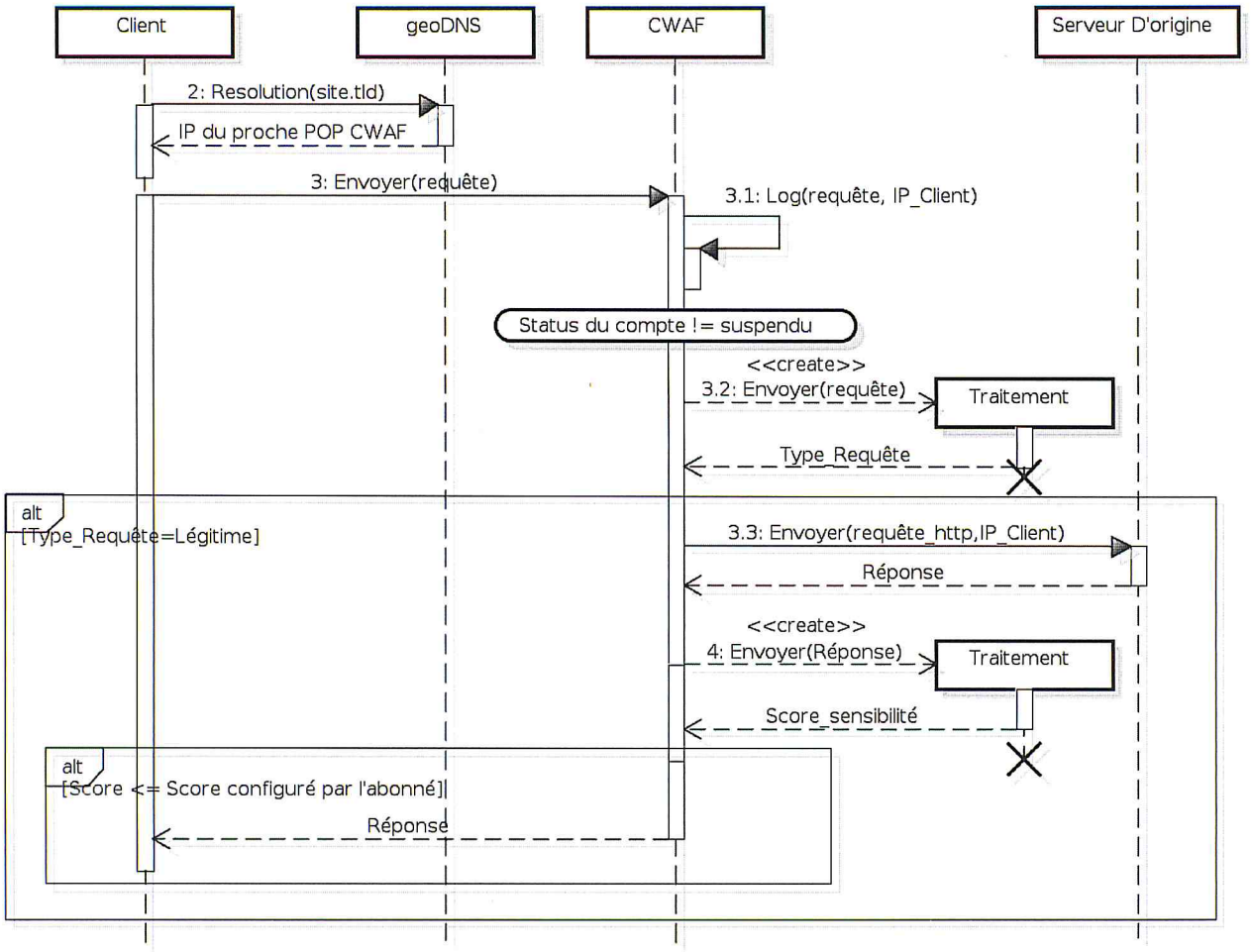


Illustration 22 : Diagramme de séquence représentant le comportement général du système avec les requêtes

II.2.3 Composition générale

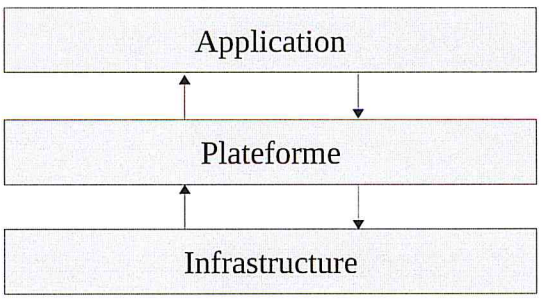


Illustration 23: Pile système

D'un point de vue « pile », le système est composé de trois briques :

- L'infrastructure représente la couche physique et les ressources basiques ;
- La plateforme, composée de services et de pièces logicielles qui gèrent l'infrastructure et fournissent les fonctions de base du système ;
- L'application est le résultat du service, configurable et consultable via l'API ou l'IHM.

II.3 Cas d'utilisation

Ce diagramme détermine les principales fonctions qu'attribue le système aux utilisateurs finaux. Nous entendons par les acteurs :

- Webmaster intéressé : la personne désirant profiter du service.
- Webmaster abonné : est un client du service.
- Le support : le service clientèle et l'équipe technique chargée d'apporter son soutien aux clients.

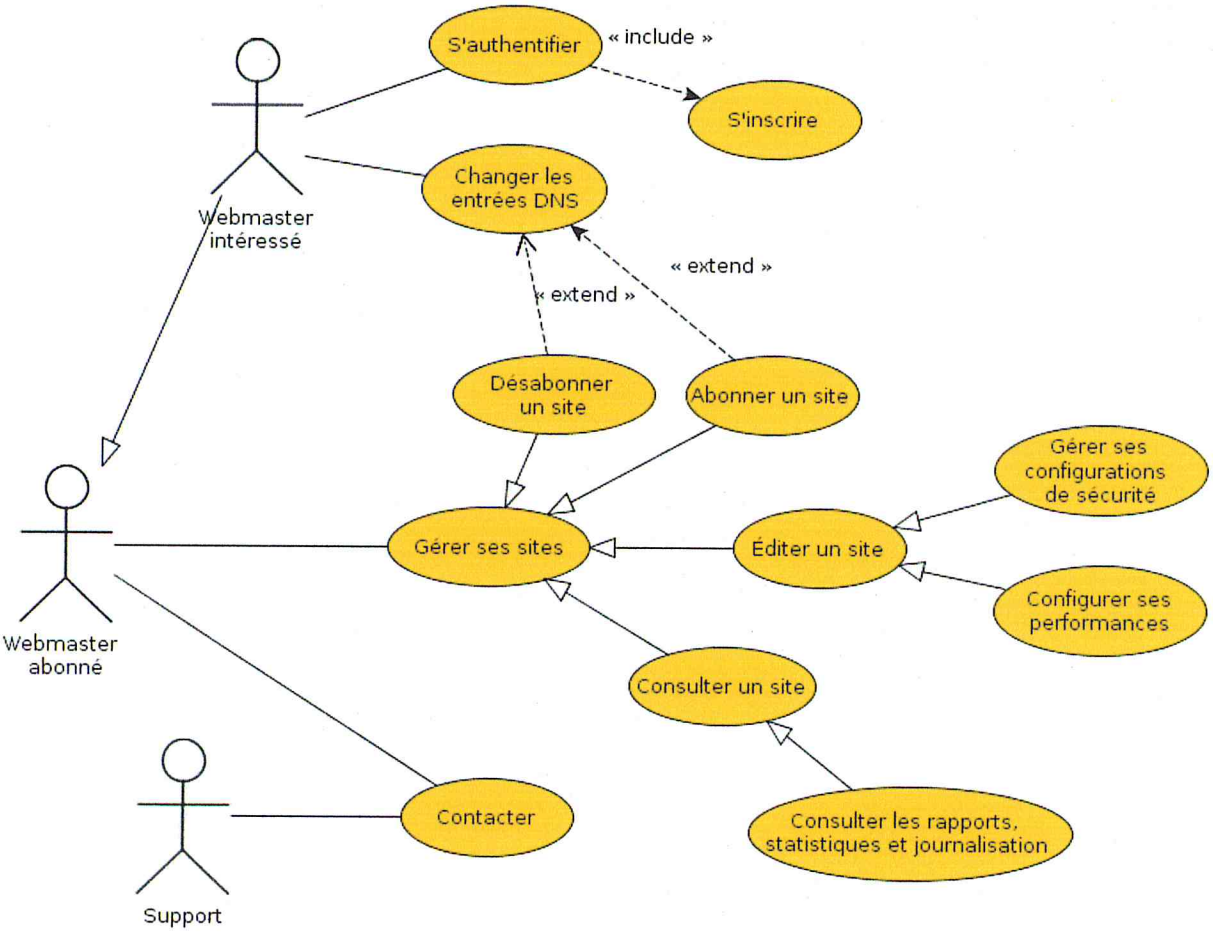


Illustration 24: Diagramme de cas d'utilisation des abonnés

II.4 Infrastructure et plateforme

Vu la forte interdépendance de l'infrastructure et de la plateforme, nous les étudierons ensemble en formant ce que l'on appelle l'infrastructure du système.

L'architecture de l'infrastructure se compose de trois parties :

- Les nœuds de traitement.
- Les nœuds de données et de partage.
- Les nœuds d'administration.

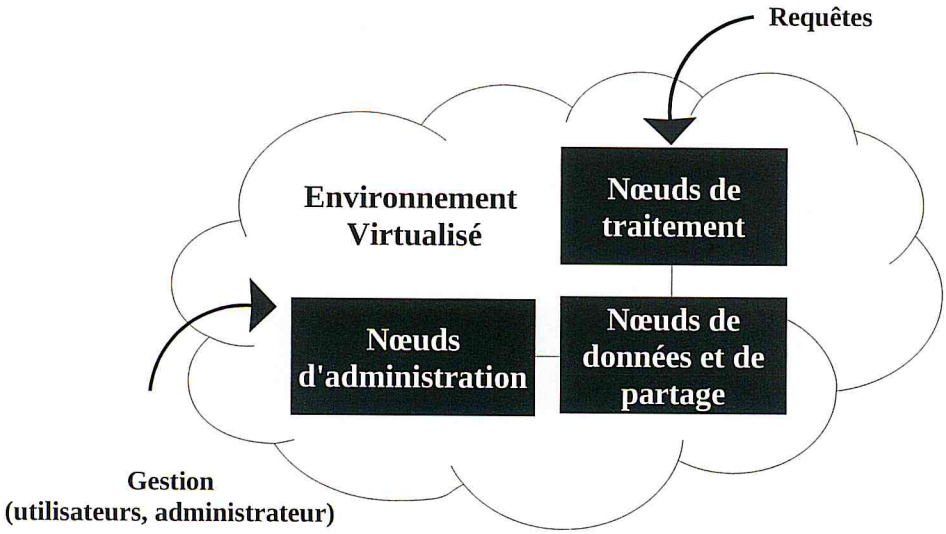


Illustration 25: Composition globale de l'infrastructure du système

II.4.1 Composants

II.4.1.1 Les nœuds de traitement

Ces nœuds seront chargés du traitement des requêtes accueillies. Il y a plusieurs types de nœuds dans cette catégorie tels que les nœuds de filtrage qui traitent les requêtes et décident de leur légitimité, ou les nœuds répartiteurs qui font le balancement de charge entre les nœuds de filtrage.

II.4.1.2 Les nœuds de données et de partage

Spécialisés dans le stockage et l'alimentation des nœuds, ils peuvent être des dépôts de bases de données ou des entrepôts contenant les informations des clients, de leurs sites web à sécuriser et de leurs configurations, etc. ; et peuvent être aussi des nœuds qui partagent les configurations entre les nœuds de filtrage en tant que système de fichiers distribué (NFS).

II.4.1.3 Les nœuds d'administration

Les nœuds d'administration ont l'accès sécurisé via une authentification asymétrique à l'intégralité du système et fournissent une API de gestion accessible depuis les nœuds de gestion (ex. : ceux hébergeant l'IHM) qu'on peut aussi considérer comme nœuds d'administration.

II.4.2 Architecture

L'architecture à concevoir doit être hautement disponible, ça veut dire qu'elle doit être tolérante aux pannes et cela peut être atteint par la redondance et le basculement à chaud entre les nœuds et leurs ombres (esclaves).

II.4.2.1 En un seul POP

Dans un premier temps, nous devons concevoir une architecture évolutive et redondante à deux échelles (DNS round-robin et répartition de charge HTTP) représentant le système en un seul point de présence.

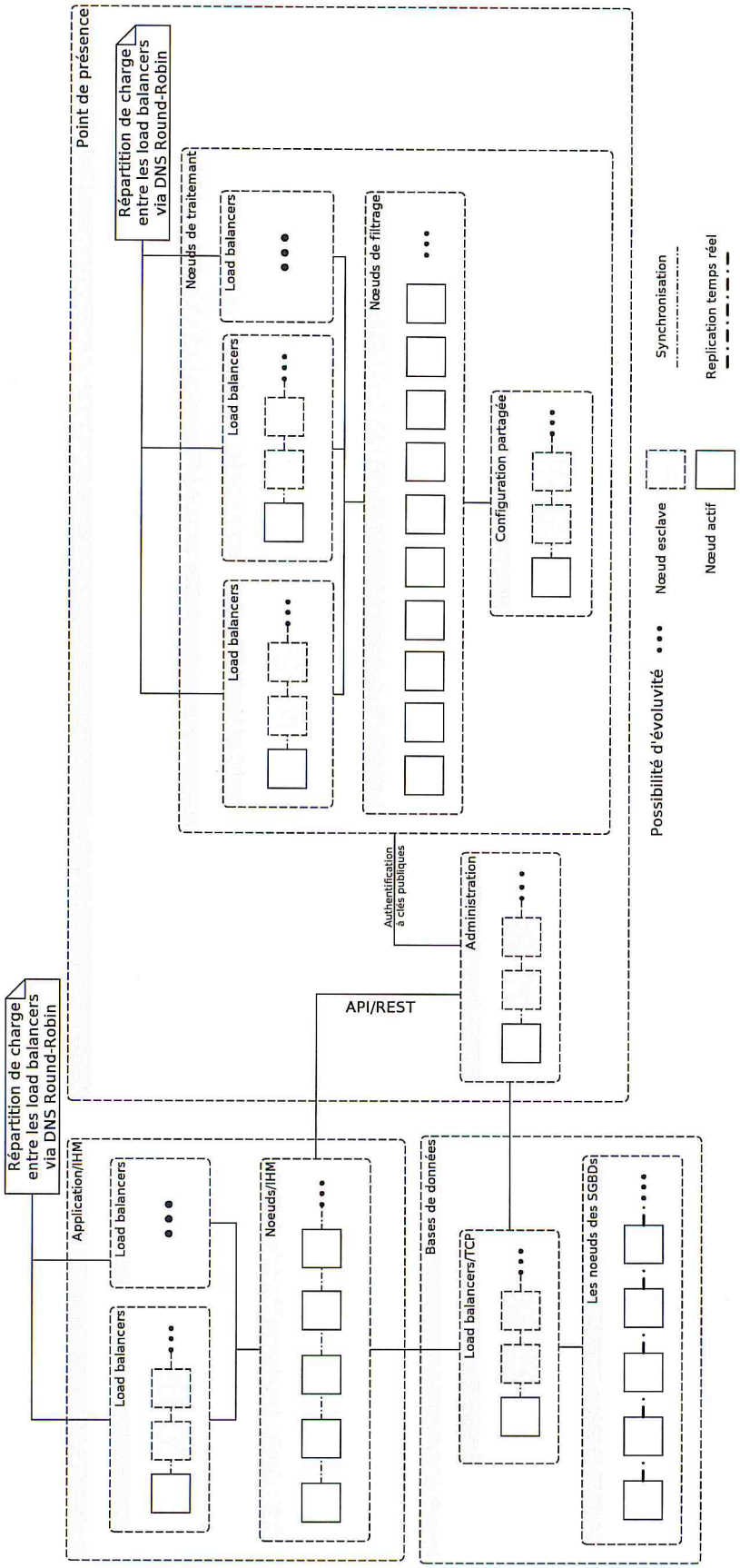


Illustration 26: Schéma représentant l'architecture en haute disponibilité et en un seul point de présence infrastructurel

II.4.2.2 En multi POP

Afin de minimiser les latences liées aux réseaux et de distribuer la charge géographiquement, et d'ajouter une couche d'évolutivité et de redondance principalement géographique, on a conçu une architecture distribuée en multipoints de présence.

Il s'agit de placer des POP contenant les nœuds d'administration et de traitement dans plusieurs emplacements géographiques en fonction de la charge, par exemple, si la charge devient plus lourde sur le POP de l'Asie, on peut grâce à l'architecture extensible, ajouter un autre POP sur ce continent pour se charger de l'Ouest par exemple et on ne chargera l'ancien que de l'Est.

Pour atteindre cet objectif, on doit ajouter à l'architecture une autre couche d'administration qui s'occupe d'orchestrer les nœuds d'administration locaux (à chaque POP), et ajouter une nouvelle couche de répartition de charge géographique, et cela peut être fait par le geoDNS (couche DNS).

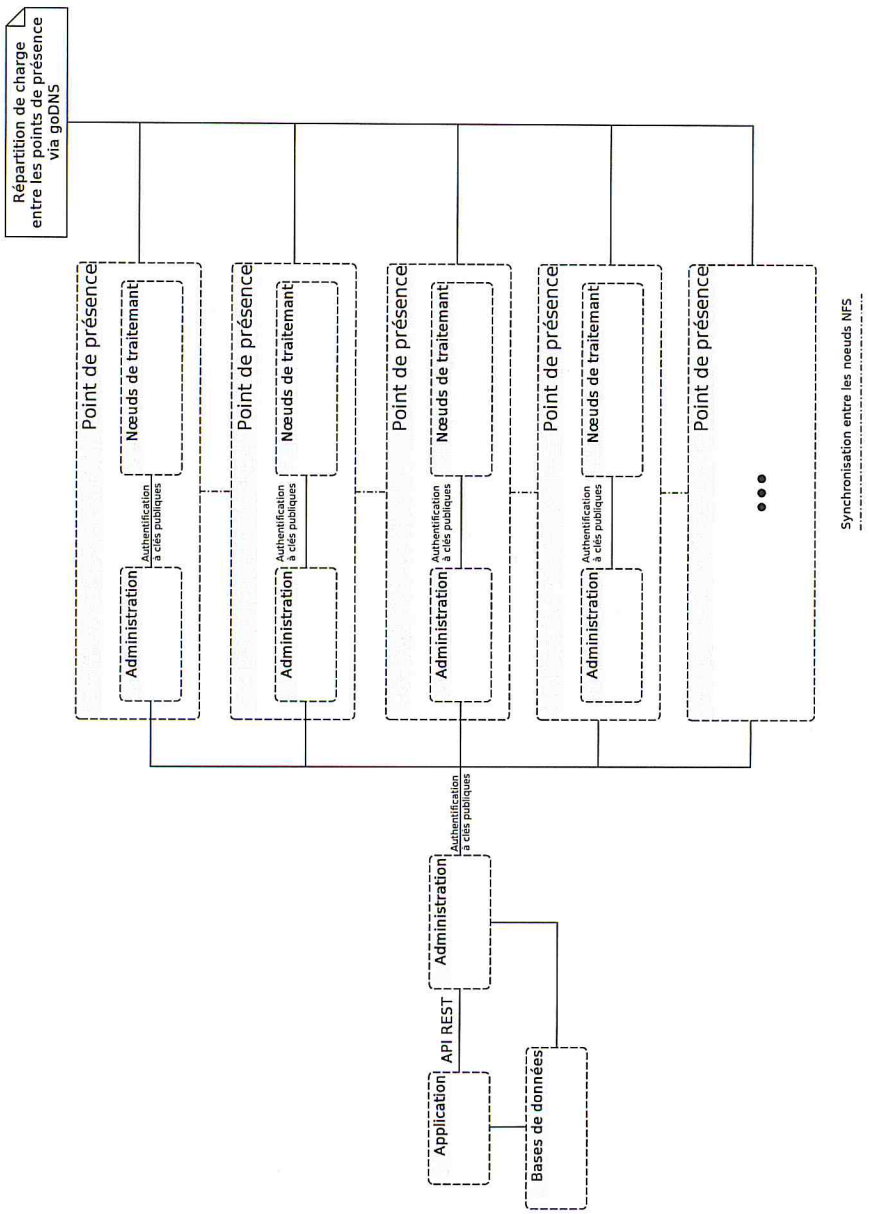


Illustration 27: Schéma représentant l'architecture en multipoints de présence

II.4.2.3 Défis

II.4.2.3.1 Synchronisation pseudo temps réel

Afin de garder l'ensemble du système à jour (prise en charge des modifications les plus récentes), lors du basculement entre les nœuds (vers les nœuds esclaves), en cas de pannes, nous avons à concevoir une méthode de synchronisation bien fiable (de fichiers entre nœuds maîtres et nœuds esclaves).

Dans ce but, on a conçu une routine de synchronisation pseudo temps réel qui se base sur un modèle « Event-driven[e] » combiné à un modèle « Time-driven[f] » au lieu de se baser totalement sur la seconde (malgré que ça soit le modèle le plus répandu du fait de sa simplicité). Ce modèle hybride est basé sur les fonctions d'observation des systèmes de fichiers fournies par les noyaux récents des systèmes d'exploitation (ex. : POSIX – Unix-like). Ces fonctions sont largement connues sous le nom « Inotify ».

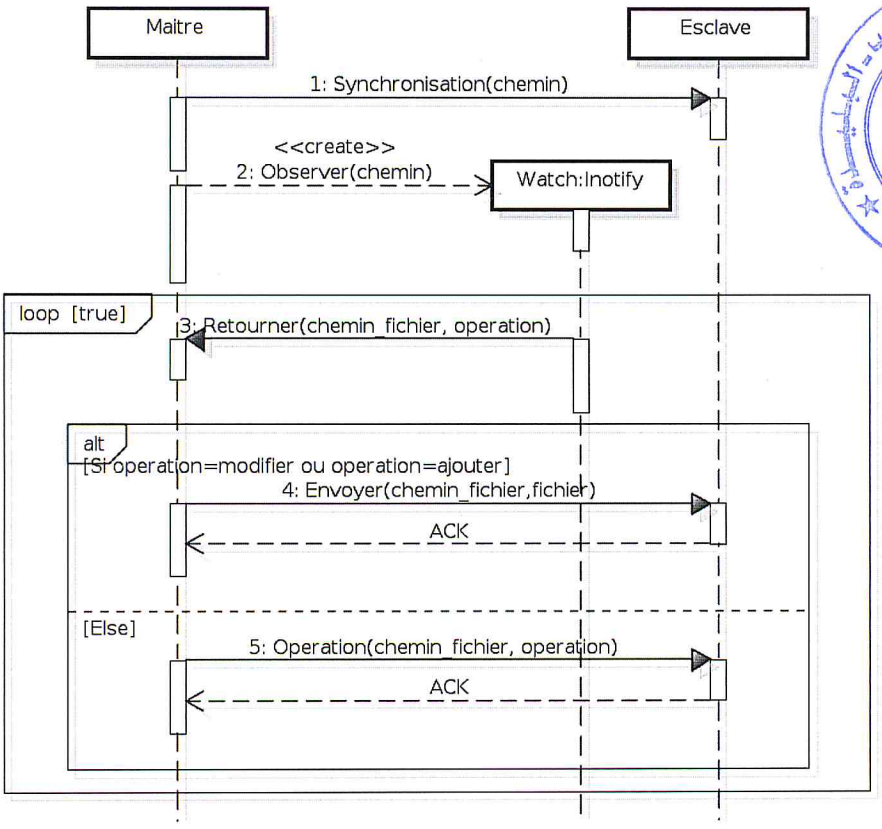


Illustration 28: Synchronisation basée sur un modèle hybride de Event/Time driven

e Synchronisation basée sur les événements (ajout, suppression, déplacement, accès ou modification de fichiers).
f Synchronisation à chaque intervalle du temps.

II.4.2.3.2 Basculement de nœuds

Pour basculer entre les nœuds (maître – plusieurs esclaves), on a choisi le basculement basé sur les IP virtuelles – couche connectivité, l'adresse IP virtuelle est l'adresse utilisée dans les communications, si le maître tombe en panne l'esclave le plus prioritaire prend la relève en se liant à cette adresse IP virtuelle. Alors, chacun des nœuds sera lié à un réseau utilisé pour surveiller l'état des nœuds d'un côté et d'un autre sera prêt à ajouter l'IP virtuelle sur l'interface utilisée en cas de problème avec le maître.

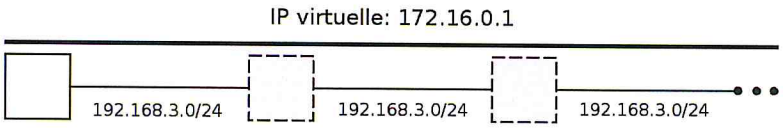


Illustration 29: Réseau de surveillance entre les nœuds, et une IP virtuelle qui se bascule entre les nœuds en cas de problème.

Pour surveiller l'état des nœuds, nous devons mettre en place sur chacun des nœuds un service qui travaille comme le cœur de ce nœud, il envoie le message d'existence (une pulsation) à chaque petit intervalle de temps suivant notre algorithme parallèle ci-dessous qui traite le basculement en multi-esclaves – multi-priorités :

```
maIP ← <<l'IP de machine>>
Machines ← <<ensemble d'adresses IP des autres machines clones>>
PrioritéConst ← <<un nombre représentant la priorité de cette machine – plus il est grand, plus
la machine est prioritaire>>
ipVirtuelle ← <<l'IP virtuelle>>
cd ← <<durée en secondes entre chaque pulsation>>
db ← <<durée à attendre pour valider qu'une machine est morte>>
port ← <<le port utilisé pour la surveillance>>
états[] ← <<table de valeurs "machine => priorité, secondes">>
maPriorité:= PrioritéConst;
Maitre ← false;

// la fonction qui se charge d'envoyer les pulsations de cette machine
fonction pulsation:
    début
        tantque true :
            pour chaque machine de Machines :
                Envoyer(machine:port, maIP, maPriorité, Maitre);
            fpour
            sleep(cooldown);
        ftanque
    fin

// la fonction chargée d'incrémenter les secondes du tableau d'états
```

```
fonction incrémentation_temps:
    début
        tantque true :
            pour chaque machine de Machines:
                état[machine][secondes]++;
            fpour
            sleep(1);
        ftanque
    fin

// la fonction chargée d'écouter les messages extérieurs et d'initialiser le tableau d'états
fonction surveillance:
    début
        tantque true :
            message := écouter(port);
            état[message[IP]][secondes] := 0;
        ftanque
    fin

// la fonction chargée de vérifier le tableau d'états et de gérer l'IP virtuelle.
fonction bascule:
    début
        tantque true :
            basculement := true;
            pour chaque état de états:
                si (état[priorité] > maPriorité et état[seconde] < cd+db) ou
état[Maitre]==true:
                    basculement := false;
                    Maitre:= false;
                    maPriprité:= PrioritéConst;
                fsi
            fpour
            si basculement = true:
                maPriprité :=infini;
                Maitre:= true;
                prend_en_charge(ipVirtuelle);
            fsi
        ftanque
    fin

// On lance les fonctions pour travailler en parallèle - multi-threading
thread_create(pulsation);
thread_create(incrémentation_temps);
thread_create(surveillance);
thread_create(bascule);
```

II.4.2.3.3 Répartition de charge

La répartition de charge (Load balancing) est vitale pour notre système et pour les systèmes évolutifs sur les réseaux, son rôle principal est de répartir les requêtes qui passent par elle entre

plusieurs nœuds/backends avec différents protocoles et sous différentes couches (couche transport et couche applications), et suivant une variété d'algorithmes de répartitions.

Les requêtes des clients passent par une répartition de charge multicouche, une répartition entre les points de présence géographiquement distribués, une répartition entre les répartiteurs de charge et enfin le balancement entre les nœuds de filtrage.

Entre les nœuds de filtrage

Les nœuds de filtrage travaillent principalement sur le protocole HTTP/HTTPS, pour que le répartiteur soit performant, il doit « comprendre » ce protocole.

Les répartiteurs de charge entre les nœuds de filtrage doivent avoir quelques aptitudes nécessaires pour le système, comme :

- L'équilibrage entre des nœuds de différentes capacités, les répartiteurs doivent se baser sur des algorithmes qui supportent les poids. Parmi les algorithmes de répartition répandus (aléatoire, par charge, Round-Robin, par nombre de connexions actives …), on a choisi Weighted Round-Robin, c'est l'algorithme round-robin avec un critère de poids.
- Compression à la volée des données à envoyer aux clients spécialement celles qui sont textuelles (js, css, html, txt, etc.).
- Injection d'entêtes HTTP (ex. : pour passer les IP réelles des clients aux nœuds de filtrage).
- SYN Cookies et delayed-binding[g], pour lutter contre les attaques SYN Flood.

Entre les répartiteurs de charge – DNS Round-Robin

Afin de rendre la couche des répartiteurs de charge évolutive, on a rajouté une couche de répartition de charge basée sur DNS et qui se positionne au-dessus des répartiteurs de charge HTTP (vu que le DNS est sollicité avant le HTTP dans la séquence de navigation). Simplement dit, le serveur DNS envoie une liste d'adresses IP des répartiteurs de charge en rotation circulaire à chaque résolution du nom de domaine.

g Il s'agit de n'ouvrir la connexion entre le LB et le Backend qu'après terminer le TCP handshake avec le client.

Exemple :

1re :	2de :
$ nslookup secured.ghaim.net server4-r.dzsecurity.net	$ nslookup secured.ghaim.net server4-r.dzsecurity.net
Server: server4-r.dzsecurity.net	Server: server4-r.dzsecurity.net
Address: 94.23.22.105#53	Address: 94.23.22.105#53
Name: secured.ghaim.net	Name: secured.ghaim.net
Address: 46.105.45.239	Address: 46.105.45.230
Name: secured.ghaim.net	Name: secured.ghaim.net
Address: 46.105.45.245	Address: 46.105.45.239
Name: secured.ghaim.net	Name: secured.ghaim.net
Address: 46.105.45.230	Address: 46.105.45.245

Entre les points de présence – geoDNS

Pour un maximum d'évolutivité du système, on a introduit une approche géographique basée sur geoDNS, il s'agit de retourner les IP du plus proche point de présence du système par rapport à la position géographique de l'utilisateur. Les nœuds de DNS peuvent être eux-mêmes sur une sous-infrastructure AnyCast pour être résistante aux fortes demandes ainsi qu'aux attaques de flood.

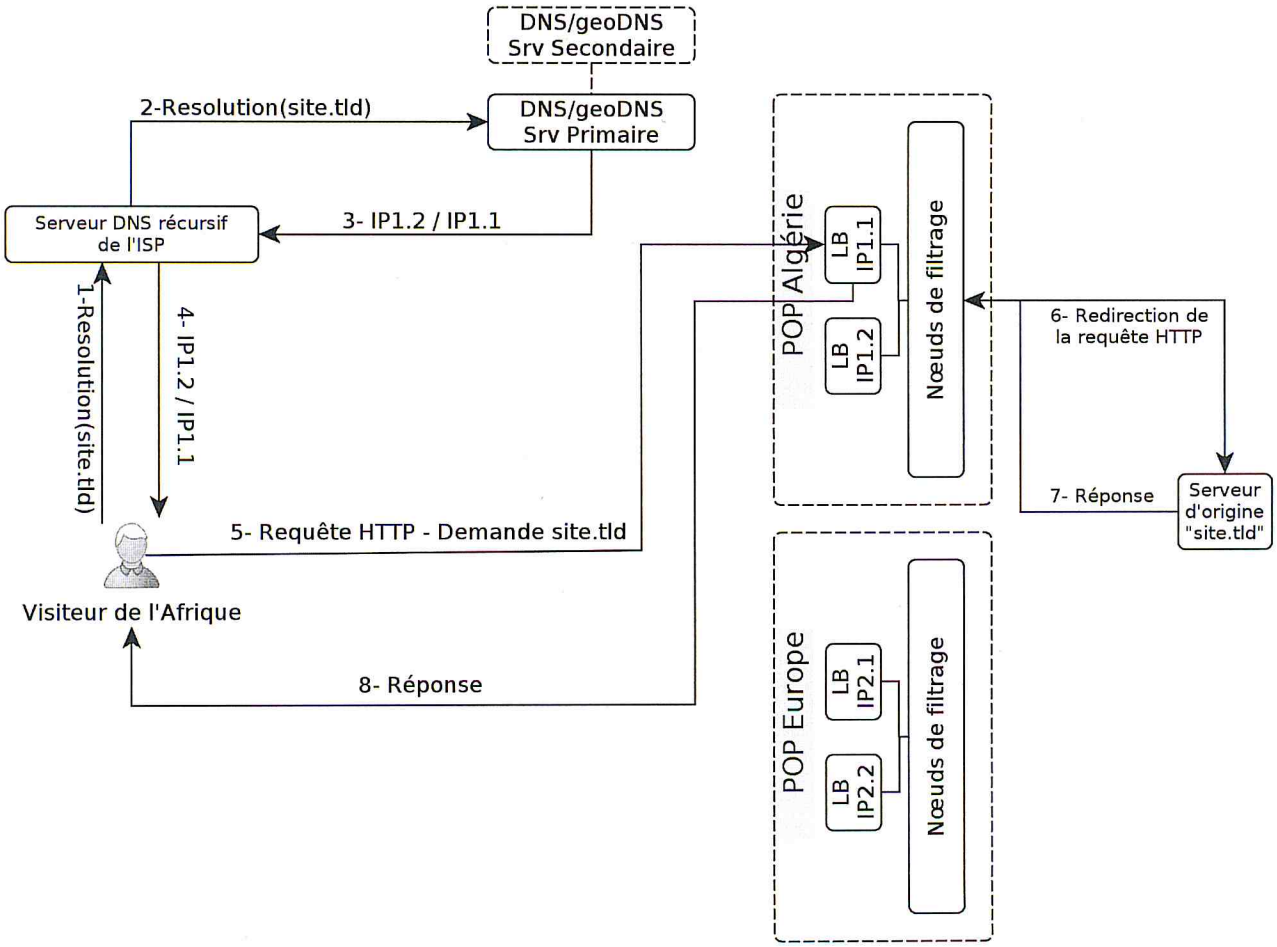


Illustration 30: Déroulement d'une requête légitime - le rôle de la répartition de charge basée sur DNS/geoDNS

II.4.2.4 L'infrastructure et la virtualisation

La virtualisation des ressources nous permet de bénéficier d'une optimisation de valeur, et du fait que nous n'utilisons qu'un seul type de système d'exploitation sur nos nœuds, on a choisi l'approche la moins coûteuse en termes de surcharge sur les machines hôtes, qui est l'approche d'isolation[voir p.27].

La distribution optimale des nœuds virtualisés sur les machines hôtes permet le déploiement du système en toutes ses capacités de redondance en utilisant un minimum de machines physiques, mais il faut d'abord mettre une stratégie bien définie de la distribution de nœuds virtuels sur les machines physiques tout en gardant un taux abordable entre la redondance virtuelle/physique.

Il y a plusieurs stratégies, l'une des plus simples et la plus utilisée consiste à placer l'ensemble du système sur une machine hôte et les nœuds esclaves sur d'autres machines, malgré sa simplicité, cette stratégie ne nous fournit que la tolérance aux pannes (Failover) et freine les aspects les plus importants sur lesquels notre architecture repose : l'évolutivité horizontale et la performance.

Donc, on est obligé de concevoir d'autres techniques de distribution de nœuds virtuels sur les machines hôtes, pour maintenir toutes les aptitudes nécessaires à notre architecture :

- Optimisation de ressources.
- Évolutivité horizontale.
- Redondance maximale sur les machines hôtes.
- Répartition efficace de charge entre les machines hôtes.

II.4.2.4.1 Stratégie de distribution

Notre stratégie de distribution a deux dimensions : une distribution verticale et une autre horizontale, la verticale est la redondance, l'horizontale est la performance, c'est un compromis entre les deux.

On a conçu un algorithme qui nous donne un plan de distribution de machines virtuelles sur les machines hôtes tout en gardant un maximum de compromis entre la redondance virtualisée et celle physique, cet algorithme nous donne le résultat en fonction de deux entrées qui sont le nombre de machines physiques disponibles ainsi que leurs capacités, et le taux de redondance souhaité (un nombre élevé réfère à plus de redondance).

```
Algorithme Distribution {
 // Entrées
 Entier Redondance ← « Nombre positif représente le taux de redondance désiré>> » ;
 Vecteur Machines ← « Vecteur dont la taille est le nombre de machines physiques que l'on a, et
la valeur de chaque indice est le nombre de machines virtuelles que cette machine peut
supporter » ;
 Vecteur d'ensembles Nœuds ← « Vecteur des ensembles de sorties » ;

 si (taille(Machines)>Redondance){
  MachinesTmp=Machines;
  Tantque (! estVide(MachinesTmp)) {
   Indice = laPlusGrandeMachine(MachinesTmp);
   // Ajouter l'ensemble dans "Nœuds", sa taille est Machines[Indice] (capacité de la machine), et
son nom Indice.
    Ajouter_ensemble(Noeuds, MachinesTmp[Indice], Indice);
    Supprimer_par_indice(MachinesTmp, Indice);
  }

  // Marquage de Load balancers
  Pour (c=1; c <= Redondance; c++)
       Append(Nœuds[c], 'RM' . c);

  Pour (c=Redondance+1; c <= Redondance^2; c++) {
   si(c <= Taille(Machines)){
    Append(Nœuds[c], 'RS' . (Redondance  - c%Redondance));
   }sinon{
       Indice=getMachineHasNot('RM' . (Redondance  - c%Redondance));
       Append(Nœuds[Indice], 'RS' . (Redondance  - c%Redondance));
   }
  }

  // Les nœuds de partage
  Entier i = (Redondance^2 < Taille(Machines)) ? Redondance^2 + 1 : Redondance;
  Pour (c=1; c <= Redondance; c++){
   si(!plein(Nœuds[i])){
    si (c==1) Append(Nœuds[i], 'PM' ); sinon Append(Noeuds[i], 'PS' );
   }
   i = (i == Taille(Machines)) ? 1 : i+1; // Circulaire
  }

  // les nœuds d'administration
  Pour (c=1; c <= Redondance; c++){
   si(!plein(Nœuds[i])){
    si (c==1) Append(Nœuds[i], 'AM' ); sinon Append(Nœuds[i], 'AS' );
   }
   i = (i == Taille(Machines)) ? 1 : i+1;
  }

  // Tout le reste sera des nœuds de filtrage
  Pour chaque Indice de Nœuds{
   Entier j=capacité(Nœuds[Indice]);
   Entier Vides=nbrVide(Nœuds[Indice]);
```

```
    Tantque(j > Vides){
      Append(Nœuds[j], 'F');
      j-=1;
    }
   }
  }
  return Nœuds ;
}
```

La sortie « Nœuds » est un vecteur d'ensembles, où chaque ensemble représente une machine physique et ses valeurs représentent les machines virtuelles qu'elle doit contenir. Les différents types de valeurs sont :

- RMx: Répartiteur de charge maître X.
- RSx: Répartiteur de charge esclave X.
- F: Nœud de filtrage.
- PM: Nœud de partage maître.
- PS: Nœud de partage esclave.
- AM: Nœud maître d'administration.
- AS: Nœud esclave d'administration.

Exemple : Six machines ont des capacités différentes : Machines={4,3,4,6,6,5}, la « Redondance » est initialisée à 3, le résultat obtenu par l'algorithme est :

Nœuds={5{RM1, RS2, AS, F, F, F}, 4{RM2, RS1, AS, F, F, F}, 6{RM3, PM, F, F, F}, 3{RS2, RS3, PS, F}, 1{RS1, PS, F, F}, 2{RS3, AM, F}};

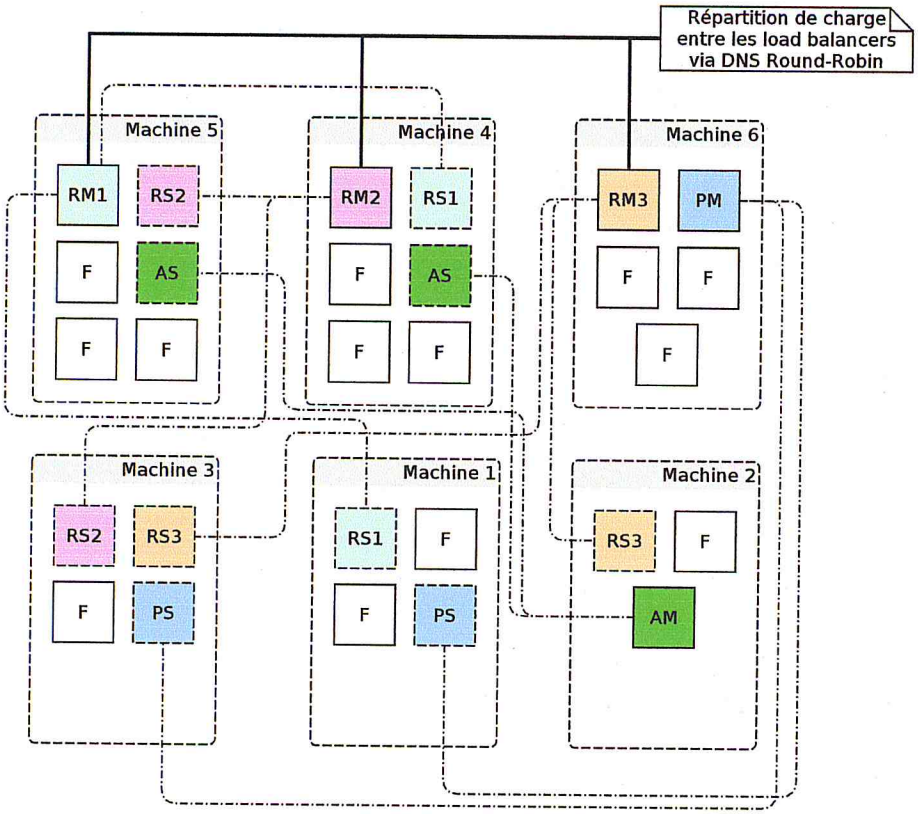


Illustration 31: Schéma représentant le résultat de l'algorithme de distribution des VMs sur la couche physique

II.4.2.5 Interaction unitaire

Dans cette partie, nous parlerons de l'interaction des nœuds vis-à-vis de leurs semblables et du reste de l'infrastructure, selon leur rôle dans l'architecture générale du système.

II.4.2.5.1 Nœud DNS

Les nœuds de DNS, sont les premiers nœuds sollicités lors d'une requête vers l'un des sites sécurisés par le cloud. Les nœuds DNS gèrent deux couches de répartition de charge, geoDNS et DNS Round-Robin, tous les nœuds de ce type sont déclarés au niveau supérieur de la hiérarchie DNS, chacun des nœuds contient et gère toutes les zones de tout site accueilli par notre système.

Quand une requête de résolution d'un nom de domaine pris en charge par le système arrive, le nœud fait une correspondance de l'IP du client avec les plages d'adresses IP dans sa base de données, afin de lui renvoyer les adresses IP des répartiteurs de charge du POP le plus proche (ou le plus éligible en termes d'élaboration de critères de répartition de charge géographique) tout en respectant la rotation circulaire de la liste, à chaque demande de résolution.

Comme on l'a cité auparavant les nœuds de DNS peuvent être déployés sur une infrastructure géodistribuée grâce à la technologie IP AnyCast, et ce n'est pas seulement pour des raisons de performance, mais aussi pour être plus résistant aux attaques (D)DoS qui visent à neutraliser la couche DNS.

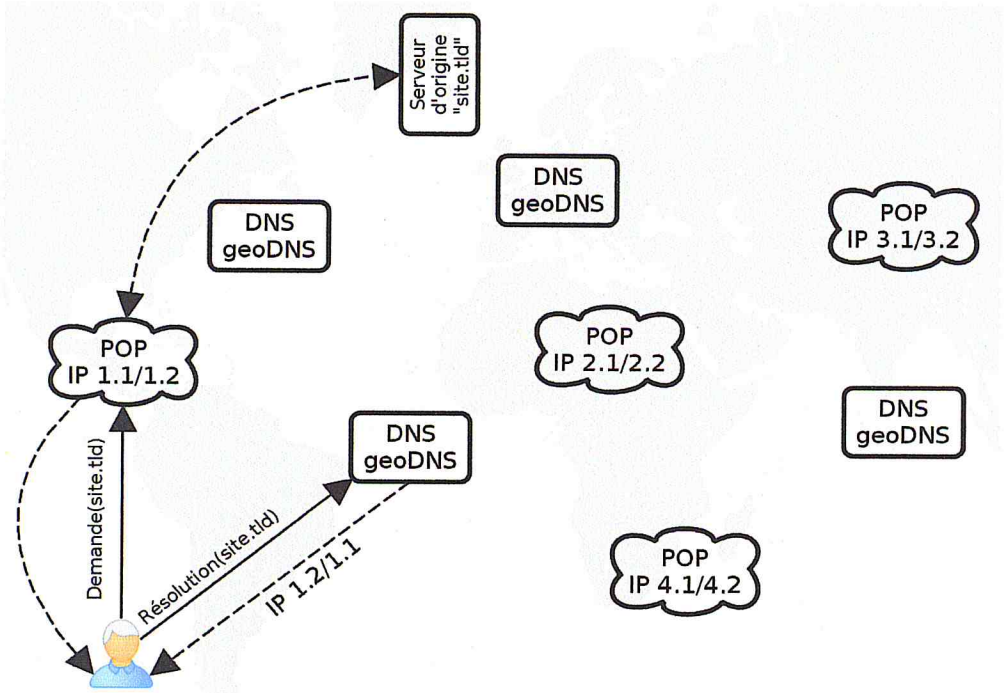


Illustration 32: Tous les nœuds DNS ont une IP AnyCast commune routée dynamiquement selon la localisation géographique du demandeur

.

II.4.2.5.2 Nœud de répartition de charge (LB)

Du point de vue, requêtes HTTP(S), les nœuds de répartition de charge sont les opérateurs frontaux du système, ils représentent notre première ligne de défense spécialement contre les attaques qui visent à saturer les ressources sur les couches OSI, comme la saturation des files des serveurs web (ex. SlowLoris) ou la saturation des limites de connexions exercées par le système d'exploitation (ex. TCP SYN Flood). Donc, les répartiteurs de charge doivent avoir bien plus de fonctionnalités que le balancement de charge :

- TCP SYN Cookies (limiter les attaques TCP SYN Flood)
- Limitation du temps de l'envoi de la requête HTTP et « Delayed-Bind » afin de limiter les attaques slowloris.
- Si l'on doit supporter du SSL sur notre système, l'emplacement le plus logique est de le déployer sur les répartiteurs de charge.

- Traffic shaping et la limitation de nombre de connexions par source pour limiter l'effet des abuseurs.
- Compression à la volée du HTTP pour les clients qui supportent gzip/deflate.
- Cache du contenu statique sans solliciter les serveurs d'origine à chaque requête.
- Injection d'entêtes HTTP dans les requêtes.

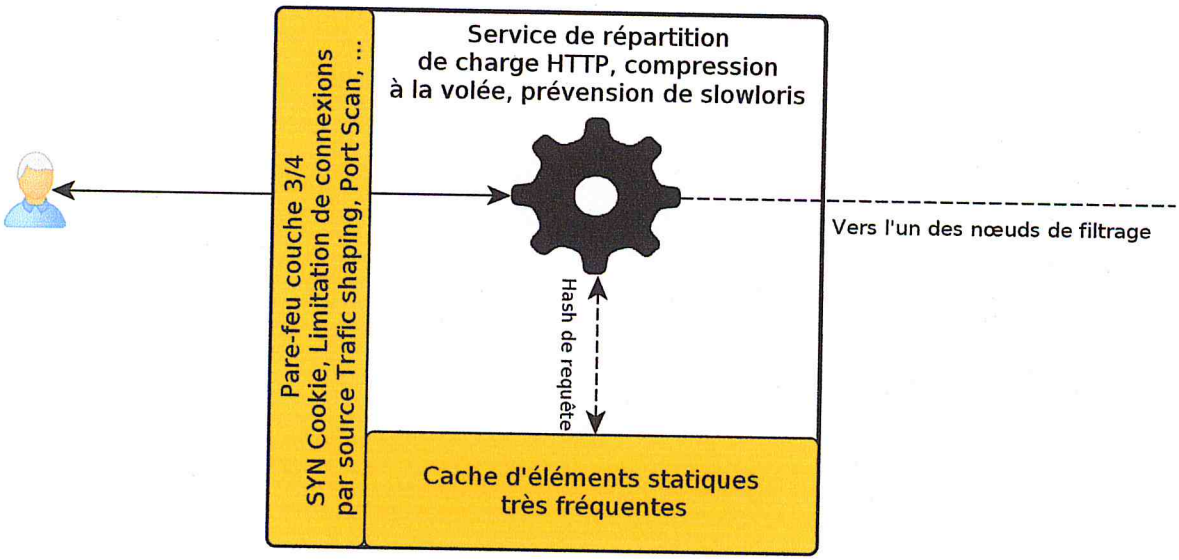


Illustration 33: Un nœud de répartition de charge HTTP et ses fonctions supplémentaires.

II.4.2.5.3 Nœud de filtrage

Les nœuds de filtrage sont les nœuds chargés en général de la détection et la prévention des attaques web actives ainsi que l'application des règles mises en place par les abonnés pour leurs sites. La diversité des attaques et le contraste des manières de détection font de la présence d'une plateforme intelligente et bien équipée une obligation afin d'exploiter et d'appliquer efficacement les configurations de sécurité.

Du côté : communication et interaction avec le reste du système, les nœuds de filtrage reçoivent les requêtes depuis les répartiteurs de charge, puis procèdent à la détection de la présence d'un facteur suspect, sinon ils redirigent la requête vers le serveur d'origine. Les nœuds de filtrage communiquent aussi avec les nœuds de partage (NFS), mais juste dans le cas de mise à jour de l'arbre de configurations.

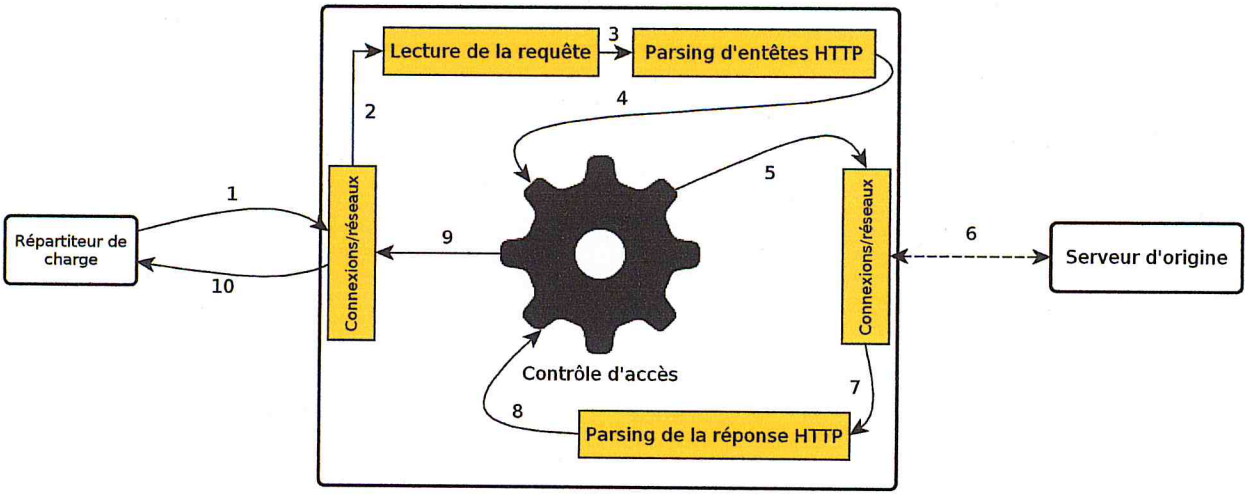


Illustration 34: L'interaction d'un nœud de filtrage avec les requêtes

Puisque la mission de ces nœuds est une mission clef dans notre système, vous trouvez plus d'informations sur le rôle de ces derniers sur le chapitre « Mitigation d'attaques ».

II.4.2.5.4 Nœud de partage de fichiers (NFS)

Ces nœuds gèrent principalement le partage des fichiers de configuration des autres nœuds afin de les référencer en cas de mises à jour. Chaque nœud NFS en réserve (esclave) doit contenir les mêmes informations que les autres, une synchronisation pseudo temps réel répond aux besoins [voir p.50].

II.4.2.5.5 Nœud de bases de données

Les bases de données seront répliquées sur plusieurs nœuds en maître-maître avec un répartiteur de charge spécial, qui assure que les requêtes d'édition de données (update, delete, alter …) seront exécutées sur tous les nœuds, et maintient l'équilibrage de charge dans le cas des autres requêtes d'interrogation (select, show, describe …). Cette architecture garantit la haute performance et l'intégrité des données entre les bases de données répliquées grâce à la réplication synchrone.

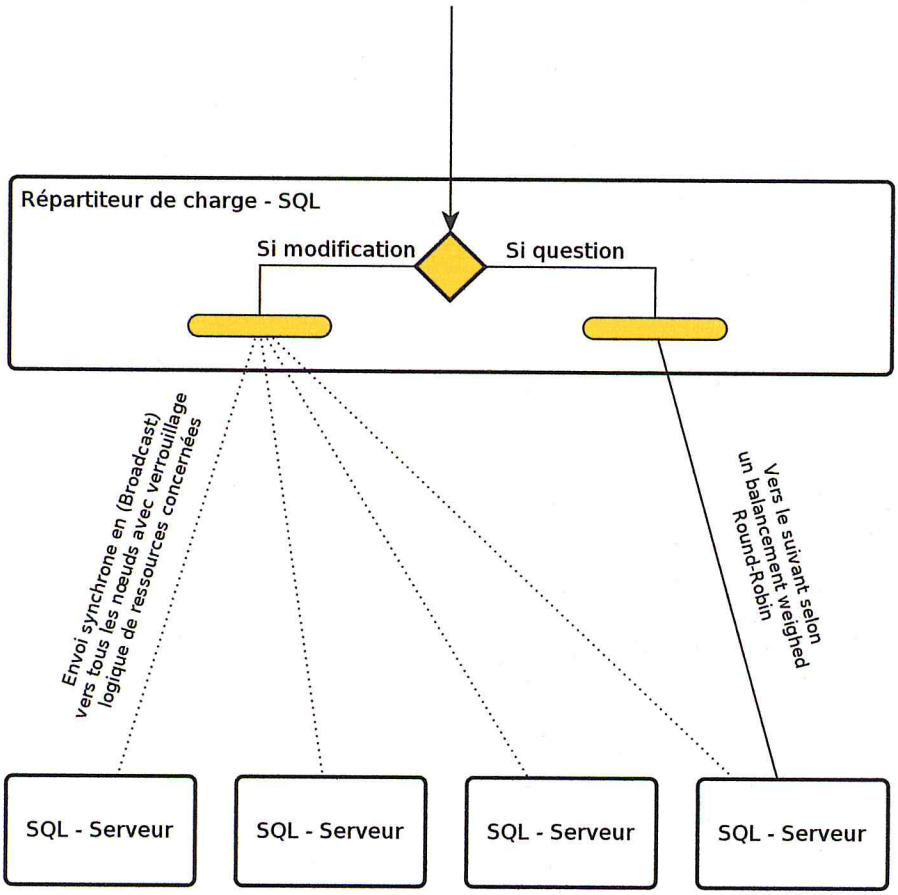


Illustration 35: Interaction du répartiteur de charge SQL avec les requêtes - architecture en haute performance

II.4.2.5.6 Nœud d'administration

Le système, grâce à l'architecture extensible à diverses échelles, peut être composé de plusieurs nœuds dont le nombre de ces derniers peut aller jusqu'à des dizaines de milliers de nœuds. Pour orchestrer ce nombre immense et exploiter les données distribuées sur le système, pour par exemple prendre des décisions contre des risques communs, mesurer la facturation du service, ou n'importe quelle autre chose nécessite une supervision et une collecte particulière des données. C'est pourquoi on a mis en place des nœuds spéciaux d'administration.

Les nœuds d'administration assurent aussi la communication entre l'application web et le reste du système avec une API RESTful.

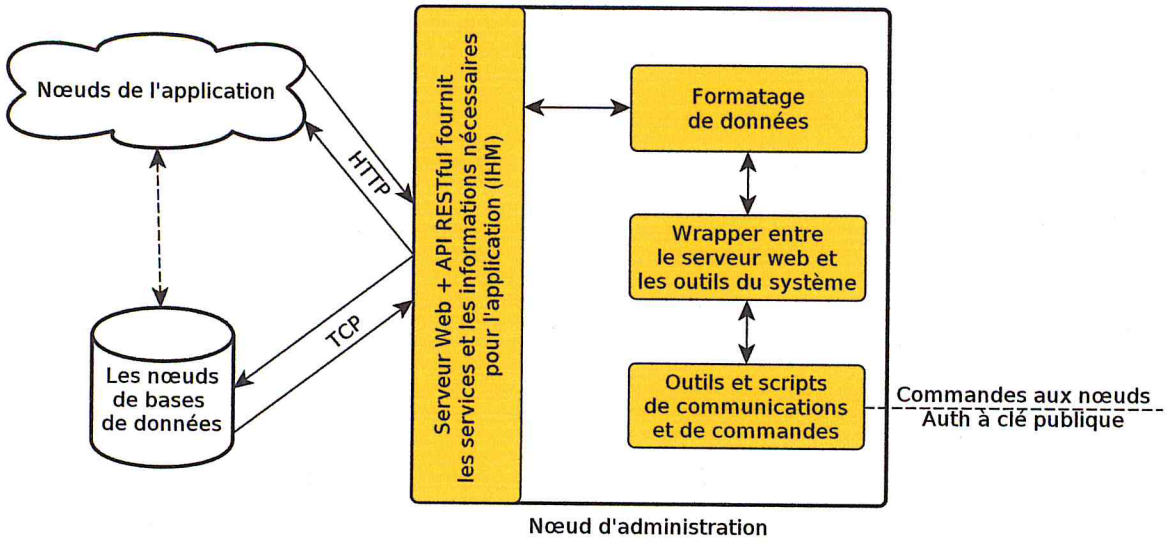


Illustration 36: Interaction de nœud d'administration avec les requêtes

Les nœuds d'administration ont un service (daemon) qui s'exécute à chaque intervalle de temps pour actionner les opérations nécessaires et collecter les informations considérées pour la routine de facturation, parmi ses fonctions :

- La collection des logs, formatage et stockage des données extraites des logs sur la base de données. Parmi ces données, on trouve :
  - Le trafic consommé de chaque site abonné ;
  - Le nombre de requêtes par site ;
  - Les profils des visiteurs regroupés pour chaque site ;
  - Les attaques détectées et contrées par site ;
- La vérification des quotas des clients, si un site a dépassé son quota, il sera suspendu par la réjection des requêtes qui le demandent ;
- La détection des abus dont les requêtes sont distribuées sur plusieurs répartiteurs de charge ;
- La détection des tentatives de bruteforce.

Pour le rôle important d'orchestre que jouent les nœuds d'administration, plus d'informations sont disponibles sur la partie « API interne ».

II.4.2.6 API interne

L'API interne (RESTful) du système est fournie par les nœuds d'administration, nous citerons

dans ce qui suit quelques fonctionnalités fournies par cette API.

II.4.2.6.1 Services fournis :

L'API interne doit supporter toute option fournie par l'application (IHM) du système et qui nécessite une intervention approfondie sur ce dernier, comme la modification des fichiers de configuration sur les autres nœuds.

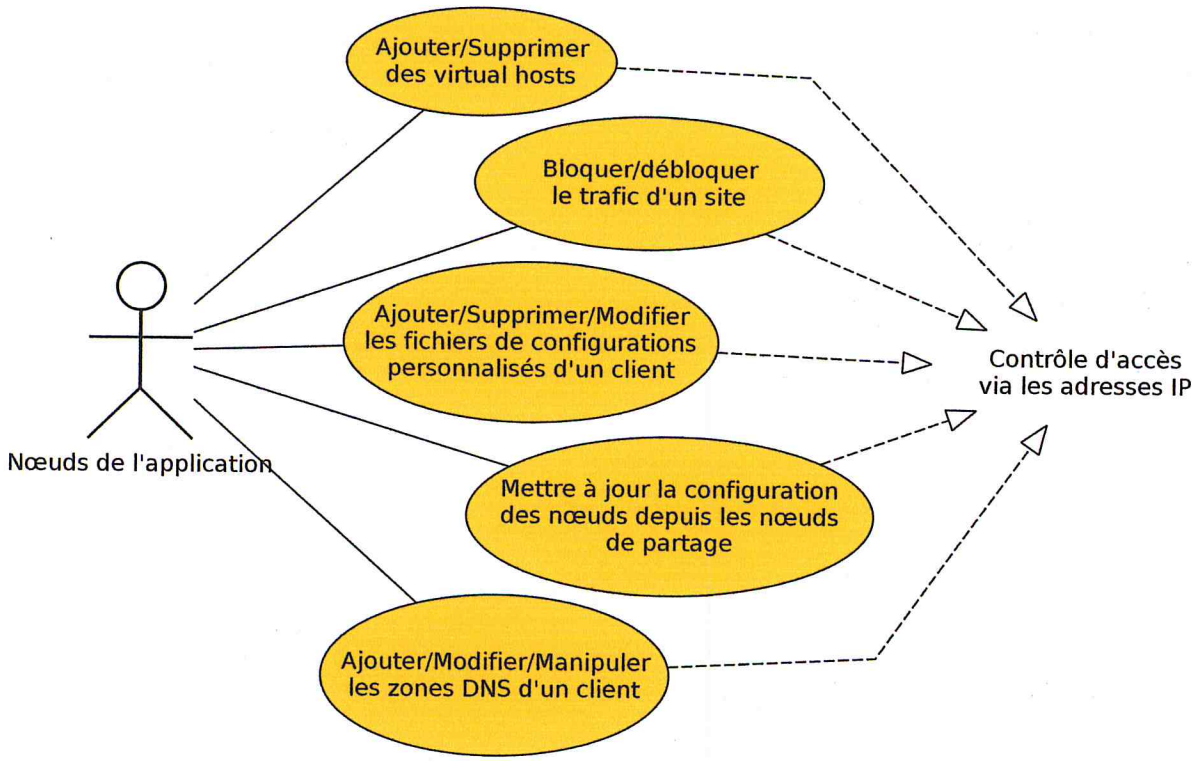


Illustration 37: Use case basique de l'API interne du système

II.4.2.6.2 Diagrammes de séquence

Gestion des configurations des nœuds de filtrage

Est l'un des services fournis par l'API interne du système, il nécessite une intervention sur les nœuds de configuration partagés ainsi que sur les nœuds de filtrage pour recharger la nouvelle configuration. Prenant l'exemple de l'ajout d'un nouveau site web à sécuriser :

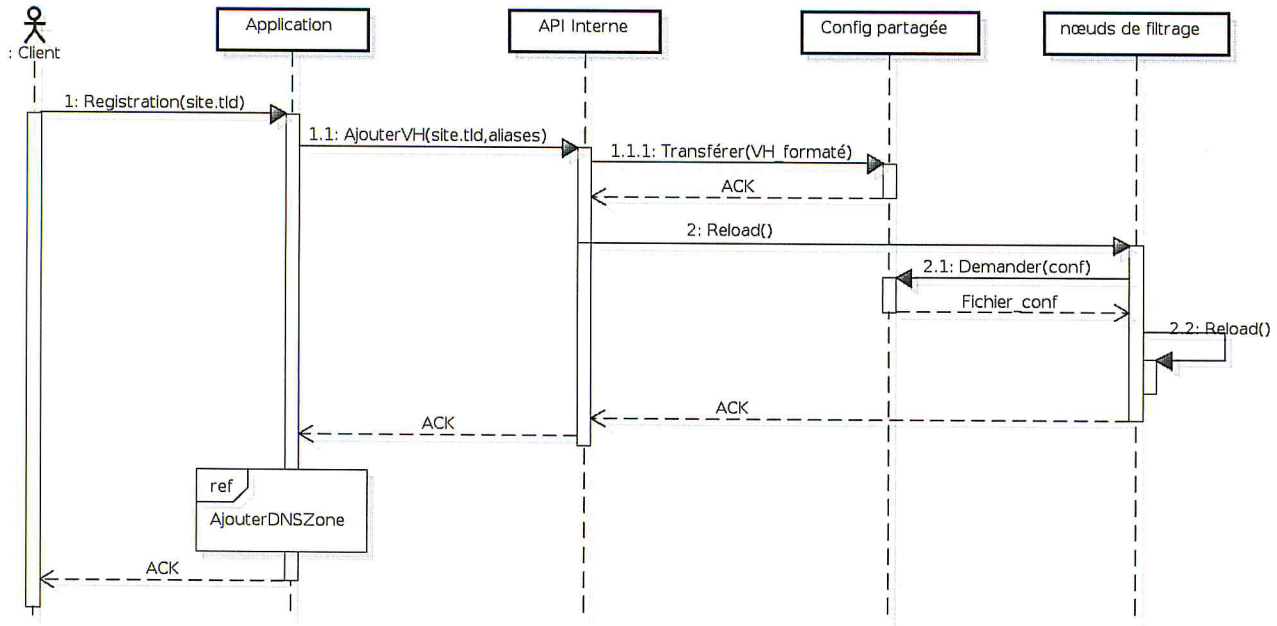


Illustration 38: Diagramme de séquence d'ajout d'un virtual host, le même principe pour la modification et la suppression

Avec le même principe (sauf quelques détails de moins, comme l'intervention au niveau des nœuds de DNS), le système ajoute les instructions de suspension des sites (Virtual Host) et met en place une configuration personnalisée par abonné.

Gestion de DNS

L'intervention au niveau des nœuds DNS se fait à chaque ajout d'un site web, modification des champs DNS (sauf les champs « A ») ou à chaque suppression de site.

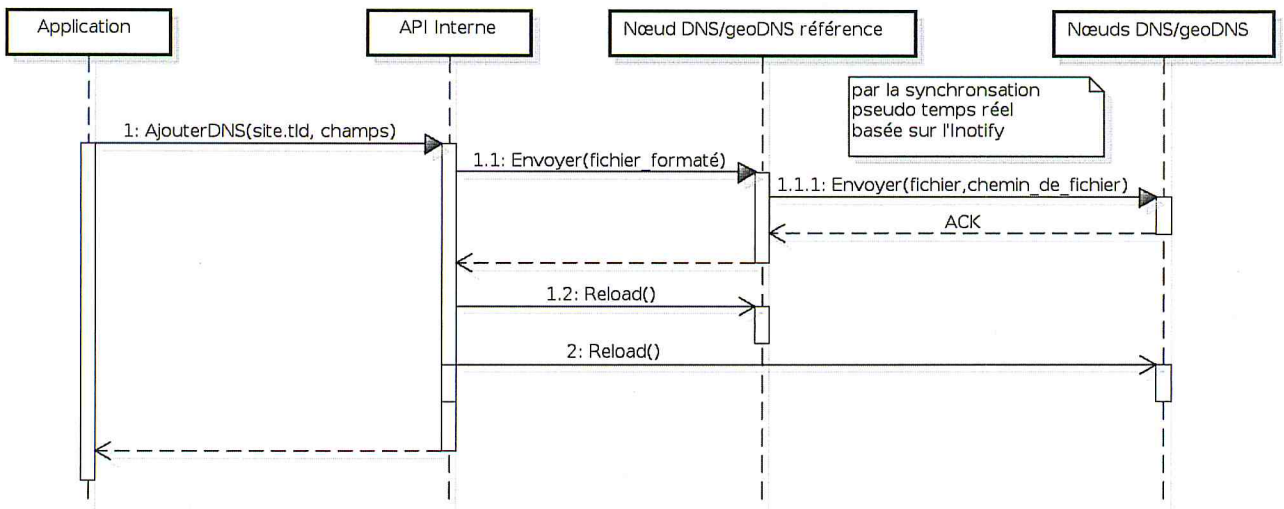


Illustration 39: Diagramme de séquence de l'ajout d'une zone DNS pour "site.tld", le même principe, pour la suppression ou l'ajout de nouveaux champs sur la zone

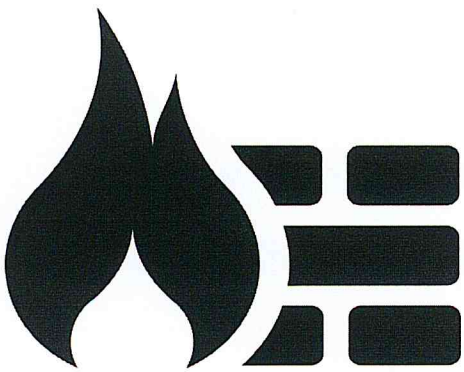
II.5 Conclusion

L'infrastructure que l'on a conçue est évolutive à trois échelles : filtrage, répartiteurs de charge et points de présence. Ceci est assuré grâce à la spécialisation des nœuds et à la répartition de charge des requêtes en trois couches geoDNS, DNS-RR et WRR dans la couche HTTP. L'infrastructure est conçue pour être extrêmement tolérante aux pannes. La probabilité d'avoir des pannes paralysantes est très réduite grâce à l'architecture hybride (Multi-maîtres, Maître–multi-esclaves) applicable sur toutes les parties de l'infrastructure.

Ainsi, on a tiré le meilleur profit des technologies de virtualisation et grâce à l'algorithme conçu, on peut distribuer les machines virtuelles sur celles physiques tout en gardant un maximum de souplesse et de redondance et un maximum d'équilibrage de charge entre les machines physiques, de même, on peut automatiser le rééquilibrage de la distribution des machines virtuelles à chaque modification au niveau des ressources physiques.

CHAPITRE III: MITIGATION D'ATTAQUES

Ce chapitre vise à expliquer succinctement la stratégie de notre service versus les menaces dont souffrent les applications Web. Nous abordons la nature de leurs faiblesses et les mécanismes de défense pour les contrer.



III.1 Introduction

Dans « l'art de la guerre », Sun Tzu rappelle que gagner ou perdre une guerre est une question de méthode et de stratégie, il dit : « Connais ton ennemi et connais-toi toi-même ; eussiez-vous cent guerres à soutenir, cent fois vous serez victorieux. Si tu ignores ton ennemi et que tu te connais toi-même, tes chances de perdre et de gagner seront égales. Si tu ignores à la fois ton ennemi et toi-même, tu ne compteras tes combats que par tes défaites [55]. ».

Il est donc primordial de connaître son ennemi pour le contrer. Le secret repose sur la préparation et la bonne connaissance du terrain.

III.2 Applications Web

Une application Web est un logiciel applicatif censé être hébergé sur un serveur distant. Elle interagit avec l'utilisateur (le client) grâce à un protocole de communication prédéfini, par exemple : via HTTP. Les sites Web sont la forme la plus répondue de ces applications auxquelles on accède à leurs pages depuis un navigateur.

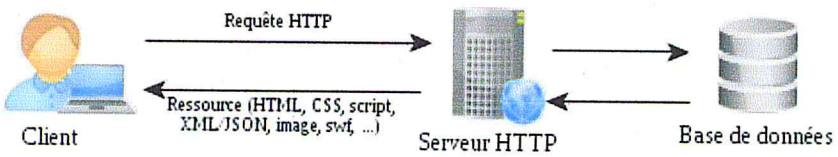


Illustration 40: Fonctionnement d'une application Web

La manière d'interpréter les ressources ou les requêtes que communique l'application et le client diffère selon le statut de ces deux-là, l'exemple des applications riches (Rich Internet Application) ou des services web qui échangent le contenu de différentes façons et sous différents formats.

III.3 Failles de sécurité

« Une vulnérabilité ou faille est une faiblesse dans un système informatique, permettant à un attaquant de porter atteinte à l'intégrité de ce système, c'est-à-dire à son fonctionnement normal, à la confidentialité et l'intégrité des données qu'il contient [56]. »

Ces vulnérabilités sont issues de défauts dans la conception, l'implémentation ou l'utilisation d'un composant matériel ou logiciel du système. Il arrive que la procédure d'exploitation de ces failles soit publiquement documentée et utilisable sous la forme d'une partie logicielle que l'on appelle « exploit », comme il est dans la communauté du site exploit-db.com successeur de Milw0rm.

Le « Hacking » est l'art d'exploiter ces failles, il porte derrière lui bien des philosophies et des approches. Les attaques menées peuvent cibler différentes parties avec lesquelles interagit l'application[voir p.14].

Notre étude se portera principalement sur les vulnérabilités les plus importantes proposées par la liste du projet Top 10 d'OWASP dans sa version 2013 (release candidate – RC1) qui ne changera probablement pas et n'a d'ailleurs que peu changé de celle de 2010. En se référant aussi à d'autres études comme celle de WASC, qui en 2004 décrit dans son rapport « WASC Threat Classification V1 »[3] 6 catégories [57] :

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

Le tableau ci-dessous sert à mettre une correspondance approximative entre les listes données par OWASP[58] et WASC[59] V2 :

Risques identifiés par l'OWASP en 2013	Attaques identifiées par le WASC en 2010	Catégories
A1 - Injection	SQL Injection (WASC-19) XML Injection (WASC-23) Null Byte Injection (WASC-28) LDAP Injection (WASC-29) Mail Command Injection (WASC-30) OS Commanding (WASC-31) XPath Injection (WASC-39) XQuery Injection (WASC-46)	Exécution de commandes
A2 - Violation de Gestion d'Authentification et de Session	Insufficient Authentication (WASC-01) Brute Force (WASC-11) Credential/Session Prediction (WASC-18) Session Fixation (WASC-37) Insufficient Session Expiration (WASC-47) Insufficient Password Recovery (WASC-49)	Autorisation, Authentification
A3 - Cross-Site Scripting (XSS)	Cross-Site Scripting (WASC-08)	Attaques côté client
A4 - Référence directe à un objet non sécurisée	Insufficient Authentication (WASC-01) Insufficient Authorization (WASC-02) Path Traversal (WASC-33) Predictable Location (WASC-34)	Autorisation, Authentification, Révélation d'informations
A5 - Mauvaise configuration de sécurité	Server Misconfiguration (WASC-14) Application Misconfiguration (WASC-15)	Révélation d'informations
A6 - Exposition des données sensibles	Insufficient Transport Layer Protection (WASC-04) Information Leakage (WASC-13) Insufficient Data Protection (WASC-50)	Révélation d'informations
A7 - Missing Function Level Access Control	Insufficient Authorization (WASC-02) Denial of Service (WASC-10) Brute Force (WASC-11) Insufficient Anti-automation (WASC-21) Predictable Location (WASC-34)	Autorisation, Authentification, Révélation d'informations, Attaques logiques
A8 - Falsification de requête intersite (CSRF)	Cross-Site Request Forgery (WASC-09)	Attaques logiques
A9 - Using Components with Known Vulnerabilities	any	any
A10-Redirections et renvois non validés	URL Redirector Abuse (WASC-38)	Attaques logiques

III.4 Contre-mesures

Évidemment, certaines de ces failles doivent être prises en compte au niveau de l'application[60], on n'ouvre pas la porte aux bandits puis on se demande pourquoi on se fait voler. Un WAF est censé remédier à ces vulnérabilités étant le pont par lequel tout passe.

Présenter toutes ces vulnérabilités n'est vraiment pas aisé dans notre contexte d'étude, car chacune des attaques nécessite en elle-même une recherche approfondie de ces usages théoriques et pratiques. C'est pourquoi nous expliquerons en bref les grandes lignes en tenant exemple de certaines de ces attaques et des approches de défense munies par le pare-feu. Les règles de politiques de nos configurations standards tirent profit de plusieurs signatures génériques comme OWASP ModSecurity Core Rule Set (CRS).

Un pare-feu applicatif Web a une profonde compréhension des protocoles HTTP et du HTML, repose sur un parsing robuste (form encoding, XML, etc.) et permet de contrôler et de manipuler les entrées, mais aussi les sorties.

III.4.1 Stratégies de mitigation

Premièrement, l'infrastructure de l'abonné est vulnérable du fait que l'attaquant peut contourner notre CWAF, s'il connaissait l'adresse IP du serveur, c'est pourquoi nous les conseillons de limiter la connexion à l'infrastructure du cloud.

Le pare-feu de son côté assurera une protection contre les attaques connues, telles que :

Les injections, qui sont définies comme le risque numéro « 1 » par l'OWASP. Les attaques par injection se produisent lorsqu'une donnée non contrôlée est envoyée à un interpréteur dans le cadre d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent leurrer l'interpréteur afin d'exécuter des commandes imprévues ou d'accéder à des données non autorisées.

Un exemple des plus basiques consiste à usurper une identité pour se connecter à une application Web sur des entrées client chargées d'exécuter une requête SQL :

```
String query = "SELECT * FROM accounts WHERE username='"+request.getParameter("login")
+"' AND password='"+request.getParameter("pass")+"'";
```

L'attaquant modifie les champs des inputs, par exemple login égalera « administrator' or 1=1-- » :

```
SELECT * FROM accounts WHERE username='administrator' or 1=1--'  AND password='123456'
```

Cela change le sens de la requête pour donner un accès, auquel l'utilisateur n'a pas droit et ce

n'est pas la seule exploitation il peut faire énormément de choses avec des requêtes imbriquées, unies … et toute la puissance de l'interpréteur (insert, drop, select …).

Bien qu'il existe d'autres formes d'exploitation, les différentes attaques par injection telles que XPath/XQuery, LDAP ou OS reposent principalement sur le même principe qui est l'utilisation de mots clefs et de caractères spécifiques qui permettent par exemple de mettre en commentaire des portions de code et d'insérer du code frauduleux. Il est cependant rare que l'application ait besoin d'accepter tous ses éléments, comme :

```
` ' " \ * / = + ! & < > ~ # { [ ( ) ] } | ^ $ @ . , : ;
```

Anticiper ces attaques consiste donc à bien s'attendre aux entrées et à empêcher que les requêtes porteuses d'éléments malveillants passent.

Les failles XSS (Cross-Site Scripting) se produisent chaque fois qu'une application prend des données non fiables et les envoie à un navigateur Web sans validation appropriée. XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites Web, ou rediriger l'utilisateur vers des sites malveillants.

Exemple : une application réutilise des données soumises dans la requête par l'utilisateur pour élaborer du contenu HTML, sans les valider ou les échapper au préalable.

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

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

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

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

Cette attaque a différentes formes et méthodes. Les contrer revient dans ce cas à vérifier si ces URL ou ces requêtes qu'envoient les utilisateurs contiennent un contenu susceptible d'être exécuté sur les clients tel que le JavaScript.

Cross-Site Request Forgery (CSRF), une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement incluse, à une application

Web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime.

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

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

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

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

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

Il est possible de déterminer la réputation du site depuis lequel vient la requête sinon un captcha ou un système de jeton sur mesure pourra remédier à ça [61]. Bien que la question qui se pose dans certaines attaques comme celle-ci c'est le niveau de performance que peut assurer le CWAF.

Ces failles et bien d'autres partagent l'aspect du filtrage à base de signatures, qui peuvent soit rejetées ou autorisées la requête avec une expression régulière générique, ce modèle est souvent utilisé avec un modèle de sécurité positive (listes blanches), soit utiliser une technique de scoring par exemple avec un modèle de sécurité négative (listes noires) pour décider de la requête. L'utilisation d'une technique spécifique dépend du contexte de l'attaque, ainsi, on peut assurer une meilleure détection et moins de faux positifs. La centralisation du WAF favorise l'apprentissage commun, les règles et les bases de signatures sont automatiquement mises à jour.

Les techniques d'évasion et d'obfuscation doivent être prises en charge par le WAF, ce genre de techniques consiste à rendre indétectable l'attaque, tout en la gardant pleinement efficace. C'est une sorte de dissimulation et de brouillage.

Pour **l'authentification**, le CWAF étant l'entre-deux, il peut fournir une interface de connexion intermédiaire pour l'accès à certaines zones. Il peut s'assurer des tentatives de connexion externes, évitant ainsi les attaques par brute force, identifiant les bots (ex. : captcha) et examinant la réputation des visiteurs, en ligne grâce à la centralisation du pare-feu et à ses tiers, il offre des listes de blocage propre au site à base de plages d'adresses IP ou de pays.

Ce firewall offre d'autres avantages en matière de sécurité, comme :

- Protection HTTP, en détectant les violations du protocole et de la politique d'utilisation définies ;
- Recherche sur les listes noires en temps réel et exploite en plus de sa base de connaissances des informations tierces pour la réputation des IP ;
- Détection des malwares Web ou du contenu malicieux (ex. : via l'API Google Safe Browsing) et peut être intégrée avec des antivirus pour le scannage des fichiers téléversés ;
- Détection des bots, crawlers, scanners et autres activités malicieuses ;
- Protection contre les chevaux de Troie par détection des accès (ex. : backdoor) ;
- Prévention des mauvaises configurations de sécurité **:** une bonne sécurité exige d'avoir une configuration sécurisée définie et déployée pour l'application, le serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus afin de ne pas comprendre de failles de sécurité. Ceci implique de maintenir tous les logiciels à jour, y compris toutes les bibliothèques de code employées par l'application. Dans le cas contraire, le WAF alertera l'utilisateur des problèmes de configuration signalés.
- Exposition des données sensibles : beaucoup d'applications web ne protègent pas correctement les données sensibles, telles que les cartes de crédit, les identifiants fiscaux et les informations d'authentification. Les attaquants peuvent voler ou modifier ces données faiblement protégées pour perpétrer un vol d'identité, la fraude par carte de crédit, ou d'autres crimes. Les données sensibles méritent une protection supplémentaire comme le cryptage au repos ou en transit, ainsi que des précautions particulières lors d'échanges avec le navigateur.
    - Le pare-feu garantit un suivi des fuites de données sensibles (ex. : cartes de crédit) ;
    - L'application Web peut bénéficier d'une transmission sécurisée et chiffrée en effet le CWAF peut assurer un chiffrement SSL et un certificat (ex. :Certificats X.509 multisites) entre l'utilisateur (ex : internaute) et le cloud sans avoir à toucher à l'infrastructure du serveur abonné et peut aussi intégrer une connexion sécurisée entre le cloud et le serveur, si ce dernier la supporte.

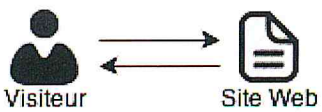


Illustration 42: Client-Serveur sans chiffrement SSL

Illustration 41: Client-Cloud-Serveur, avec chiffrement SSL

- Amélioration de l'opacité des applications en camouflant les messages d'erreurs envoyés par le serveur ;
- Protection DOS (HTTP flooding, ralentissement des attaques HTTP DOS) ;

Ces failles peuvent même être combinées et la plupart d'entre elles sont facilement exploitables, du fait qu'une grande partie des applications est issue de produits documentés ou open source ce qui rend la reconnaissance des technologies employées plus simple et la pénétration plus aisée avec la publication d'exploits et l'automatisation des tests grâce aux outils de scan et d'injection automatiques. C'est pourquoi nous pensons que personnaliser certaines règles selon le type et la technologie de l'application offrira plus de protection et sûrement moins de faux positifs. Par exemple, si l'on connaît le bon trafic circulant sur une solution CMS par défaut, lui offrir un modèle de sécurité propre au produit garantira encore plus de performance et si le produit venait à être personnalisé, les règles devraient s'adapter au nouveau comportement (ex. : via un apprentissage supervisé basé sur un modèle de sécurité négative ou en impliquant une partie logicielle sur l'application).

III.4.2 Dénis de service distribués

Une attaque par déni de service vise à empêcher le bon fonctionnement d'un service le rendant ainsi indisponible. Elle peut s'agir d'une saturation d'un composant de l'architecture (ex : base de données) ou d'une exploitation de vulnérabilité. Ces attaques ne sont pas spécifiques à la couche application, mais aussi aux autres couches telles que celle du transport ou du réseau. C'est pourquoi le cloud WAF doit pouvoir les gérer, mais le plus important est la possibilité de contrer les attaques DoS distribuées (DDoS), qui pour beaucoup deviennent impossibles à résister. Elles proviennent souvent d'hôtes dits zombies, contrôlés par l'attaquant. Ce qui rend la classification difficile.

Les attaques par déni de service non distribuées peuvent être contrées en identifiant l'adresse IP de la machine émettant les attaques et en la bannissant au niveau du pare-feu. Les paquets IP provenant de la machine hostile sont dès lors rejetés sans être traités empêchant que le service du serveur ne soit saturé et ne se retrouve donc hors ligne [62].

Les attaques par déni de service distribuées sont plus difficiles à contrer. Le principe même de l'attaque par déni de service distribuée est de diminuer les possibilités de stopper l'attaque. Celle-ci émanant de nombreuses machines hostiles aux adresses différentes, bloquer les adresses IP limite l'attaque, mais ne l'arrête pas, voilà pourquoi on parle de mitigation d'attaques [62].

On parle de classification intelligente en temps réel, en effet, c'est ce que résume Orange Labs

dans leur retour d'expérience dans la lutte contre les attaques en DDoS : « la difficulté inhérente aux attaques DDoS réside dans leur apparente légitimité, puisque les requêtes malveillantes utilisent des ports "autorisés" produisent des paquets a priori "bien formés". En ce sens, les pare-feux et autres Intrusion Prevention Systems (IPS) restent inefficaces face à la majorité des attaques DdoS [63]. »

« Le nombre fait la force » est le principe sur lequel elles se basent, généralement une attaque distribuant 50 Go/s est suffisante pour mettre hors-ligne une banque majeure ou une grande infrastructure. Ce mars 2013, des attaques à très grande ampleur, équivalentes à plus de 300 Go/s inonda Spamhaus [64], une organisation internationale sans but lucratif avec comme objectif de traquer les spammeurs et les activités relatives aux spams. Constituant ainsi la plus grande cyberattaque de tous les temps, un ralentissement de la connexion internet s'est fait ressentir au niveau du globe.

La stratégie que nous comptons déployer ressemble à un centre de renseignement qui travaille en temps réel. Dans notre architecture, les nœuds de filtrage sont distribués, ils ne décident du sort des requêtes que selon la configuration posée pour le client. Ils ne peuvent pas toujours prendre une décision décisive pour bloquer tel ou tel zombi. C'est là que nous consacrons des nœuds de traitement spécialisés dans l'analyse du trafic et la classification des logs susceptibles de contenir un trafic malveillant dans l'ensemble du cloud. Ces nœuds centralisés peuvent informer les nœuds d'administration pour que l'information soit propagée dans l'ensemble des nœuds de filtrage, ainsi les zombis seront automatiquement bloqués et neutralisés. Ce trafic pourra aussi être absorbé par des trous noirs dès les premières lignes de l'infrastructure cloud.

Nous avons développé un algorithme de classification propre à nos serveurs basé sur l'arbre de décision et sur des réseaux neurones, malheureusement, on n'a pas eu beaucoup de temps pour le tester réellement. Pour diminuer l'impact d'une attaque couche 7, le cloud WAF intervient en tant qu'une interface de vérification (ex : captcha, lien, mise en attente) et injecte des cookies sur les clients légitimes. Les données collectées lors de cette phase et par les tiers serviront à déterminer les machines hostiles.

Ce processus ne garantit pas que la sûreté globale des abonnés, mais celle du cloud aussi, il est vrai que le cloud est censé être un énorme réservoir de ressources à plusieurs POP, mais si on se laisse faire, c'est le service du cloud qui sera déni.

Il existe d'autres approches généralement dédiées à certaines offres d'abonnement qui consistent par exemple à basculer temporairement en cas d'attaques ou en permanence vers des nœuds de filtrage DDOS en temps réel (CleanPipe), une sorte de « cleaning center » qui peuvent être placés en monitoring ou peuvent être déployés en frontal. Ces nœuds permettent de retourner en sortie que

la partie purifiée et donc légitime vers le serveur final.

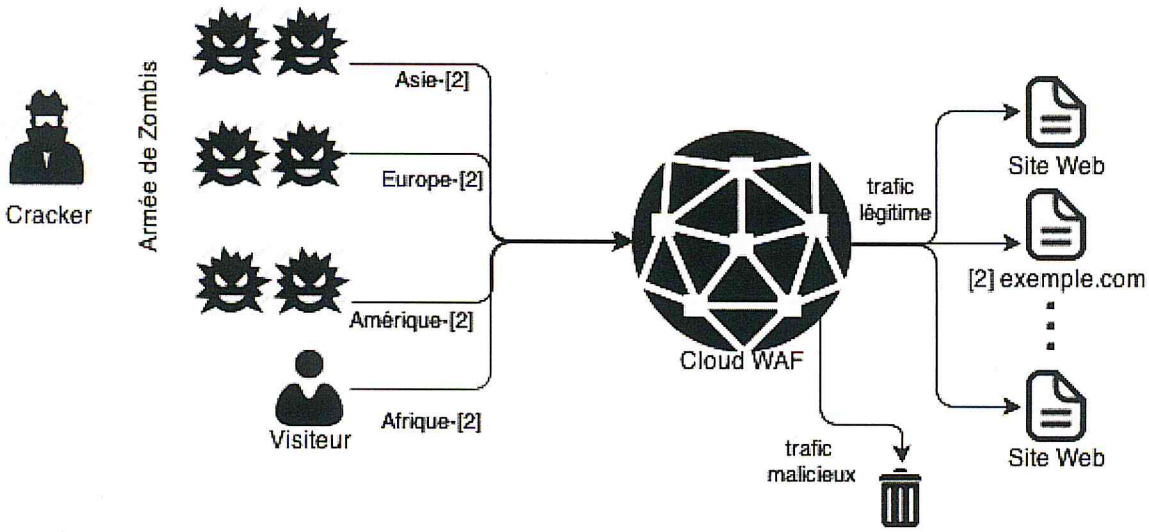


Illustration 43: Cloud WAF multi POP vs DDoS

La charge n'est pas directement dirigée vers le serveur final, mais passera par les POP du cloud distribués sur le globe.

III.4.3 Suivi et maintenance

Journalisation avancée : pour les rapports, nous avons des nœuds de données spécialisées dans l'organisation des logs et l'archivage ce qui servira comme matière primaire pour l'API utilisateur (ex. : IHM).

- Contrôler la sécurité et surveiller les alertes avec une plateforme ergonomique est important.
- Offrir une meilleure expérience à l'utilisateur le mettra plus à jour avec les besoins de son application (par exemple :pour prévoir des corrections).

Les audits semi-automatiques : sont une solution qui nous garantira un suivi de sécurité continuel et permettra la génération de configurations sur mesure et permet aussi l'application de patchs virtuels.

Honeypot : un aspect intéressant de sa centralisation, c'est qu'il agit comme un grand pot de miel comme le WASC Distributed Open Proxy Honeypot Project d'où il puise déjà ses informations avec d'autres bases de données publiques comme Open Source Vulnerability Database (OSVDB). Ce qui permettra d'observer les moyens de compromission des attaquants et de se prémunir contre de nouvelles attaques.

III.5 Conclusion

Depuis la version de 2004, le classement de l'OWASP a peu évolué, ce qui signifie que les attaques ne sont pas complètement nouvelles, mais prennent de nouvelles formes et gagnent en puissance d'où la nécessité d'un suivi professionnel.

Comme la technologie progresse, les attaques par déni de service ne cessent d'accroître en complexité et en ampleur. Les solutions DoS traditionnelles au niveau du site, ne peuvent tout simplement pas s'adapter à la large gamme des nouveaux vecteurs d'attaques et sont rendues totalement inefficaces contre des attaques dépassant la capacité du réseau.

Le réseau des Cloud WAF est conçu pour atténuer et suivre le rythme de l'évolution des menaces. Il est capable d'apprendre des attaques menées contre un client particulier pour protéger tous les clients sur le Cloud. Le fait de neutraliser le trafic malveillant économisera les ressources et la bande passante des clients.

CHAPITRE IV: RÉALISATION

Ce dernier chapitre est un rapport technique résumant le côté d'implémentation de l'étude. Il comportera les outils et les tests mis en œuvre :

- L'infrastructure et la plateforme ;
- L'application IHM : le tableau de bord ;
- Un test de performance et de pénétration.

Le projet a été intitulé « Ghaim », il est déployé aujourd'hui en version pre-alpha.

IV.1 Introduction

La réalisation est une tâche délicate, c'est pourquoi on a eu recours au déploiement d'une plateforme de gestion de projet et de suivi de bogues en ligne pour organiser les tâches et l'avancement du développement. Cette stratégie nous a assuré la synchronisation et un bon sens de communication. Elle peut servir au plus tard comme une base communautaire à ce projet.

Cette plateforme est basée sur Redmine, un outil à la fois simple et puissant. **Redmine**[h] est une application web libre de gestion presque complète de projet, développé en Rails[i]. Elle supporte la gestion multi-projets, la gestion et le suivi des versions comme SVN et fournit un wiki, forum, news et pleins de fonctionnalités comme GANTT/calendrier, notifications, etc. En lui intégrant Google Drive, la rédaction des rapports et du mémoire est prise en charge.

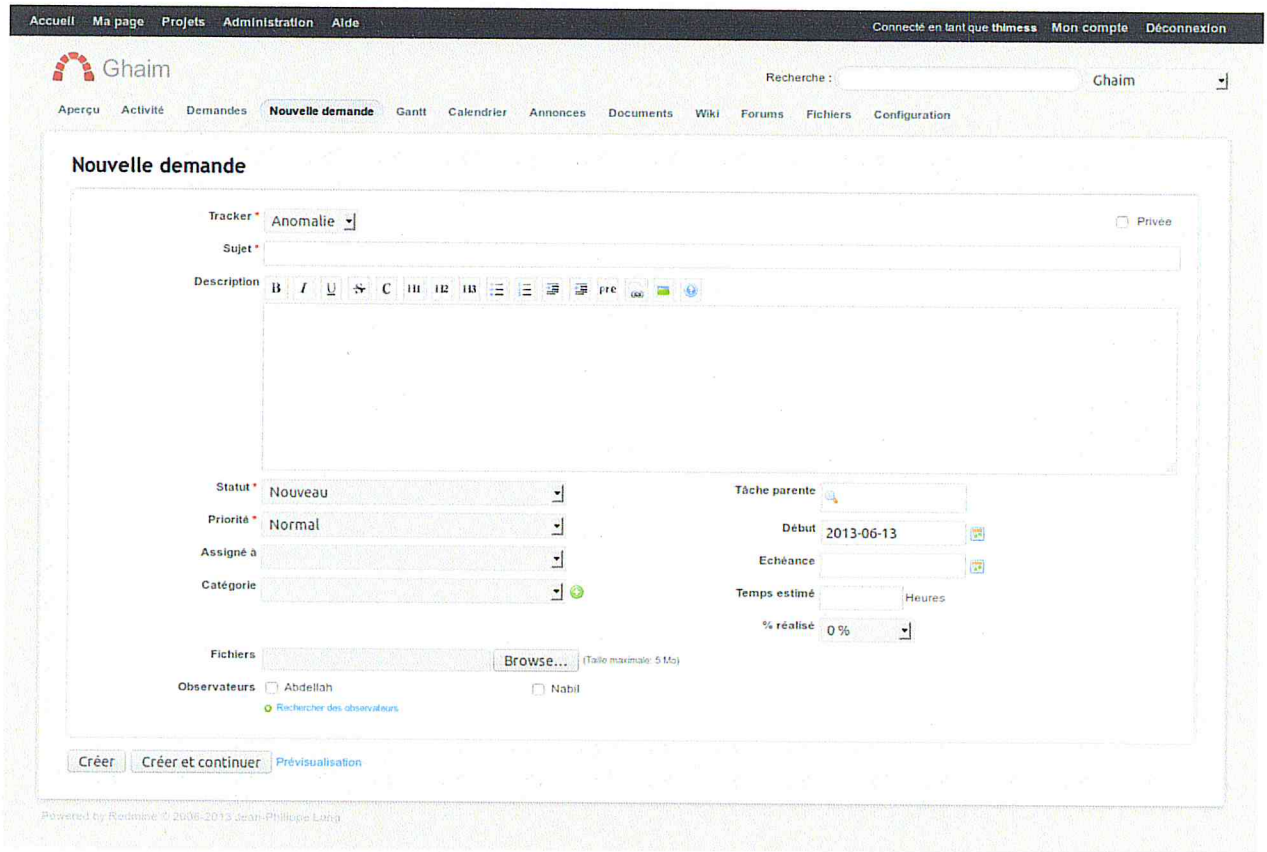


Illustration 44: Gestionnaire du projet http://track.ghaim.net

h Créé par Jean-Philippe Lang en 2006, d'autres développeurs de la communauté y contribuent depuis. http://www.redmine.org

i Ruby on Rails (RoR) est un framework web libre écrit en Ruby qui suit un pattern MVC et est connu pour favoriser les concepts de DRY (Don't Repeat Yourself) et la Convention over configuration.

IV.2 L'infrastructure

Nous avons déployé nos serveurs un peu partout dans le monde pour simuler le fonctionnement attendu de la distribution. Le réseau est censé être international grâce à une couverture mondiale geoDNS pour une répartition de charge au niveau de la résolution des noms de domaine et AnyCast pour le routage inférieur.

Voilà à titre d'exemple : un de nos serveurs qui est équipé avec SolusVM, pour donner une idée de ce que c'est l'administration des serveurs virtuels :

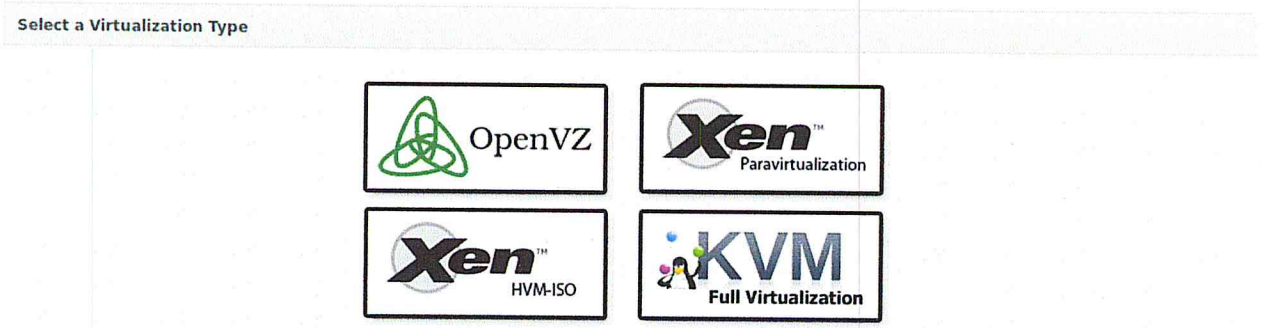


Illustration 45: Le choix de virtualisation - SolusVM

Actuellement, nous utilisons des conteneurs de virtualisation basés sur OpenVZ qui permettent d'isoler nos systèmes d'exploitation Linux. Ce qui favorise les performances, améliore la gestion dynamique des ressources et simplifie l'administration. Selon OpenVZ [65], cette méthode de virtualisation introduirait une très faible pénalité sur les performances : 1 à 3 % de pertes seulement par rapport à un serveur autonome.

Chaque classe de nœud est sauvegardée dans une image compressée (template) pour être facilement déployée et maintenue. Ces images sont équipées pour héberger nos besoins spécifiques pour chacune des plateformes (LB, filtrage, etc.).

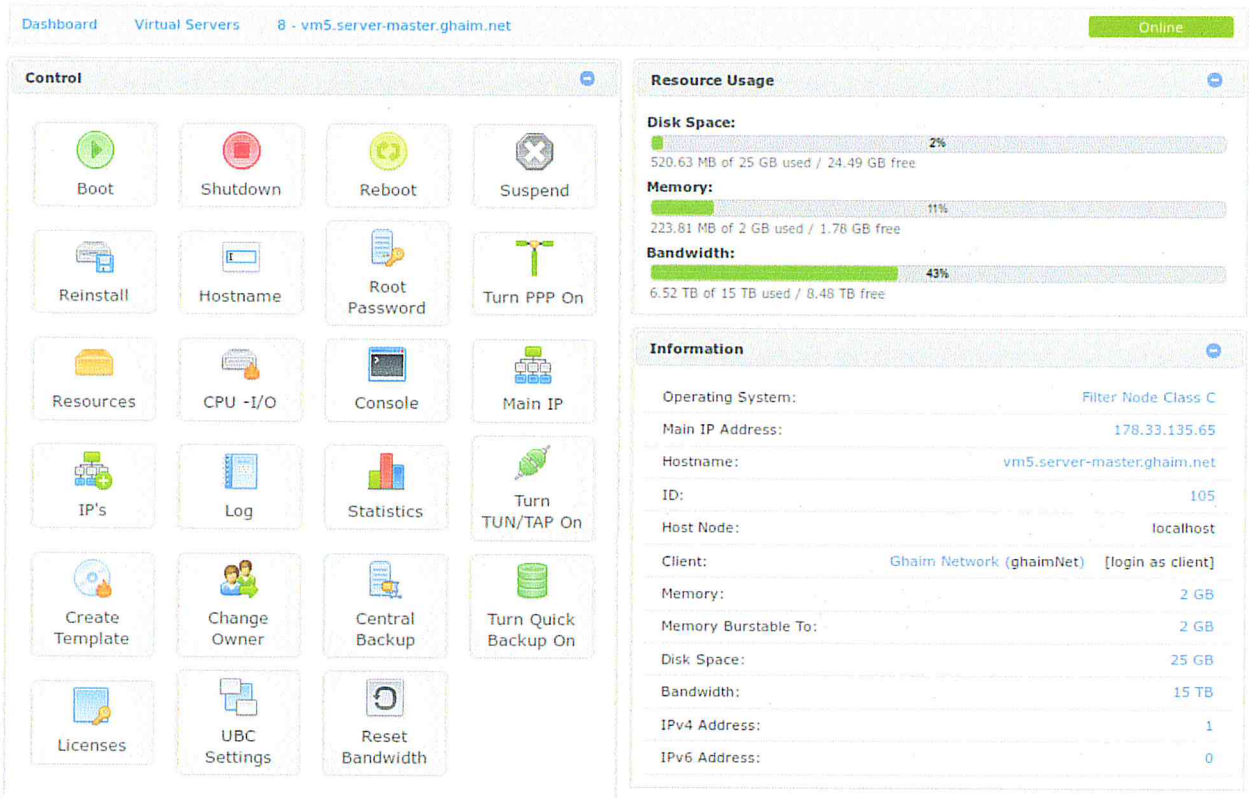


Illustration 46: Exemple d'un nœud de filtrage- SolusVM

OpenVZ est distribué sous la licence publique générale GNU. Il comprend un kernel Linux et un jeu de commandes « utilisateur », exploitable dans une API d'automatisation.

```
[root@server-master ~]# vzlist -a
      CTID      NPROC STATUS    IP_ADDR         HOSTNAME
       101        170 running   46.105.45.245   vm1.server-master.ghaim.net
       102         27 running   46.105.45.239   vm2.server-master.ghaim.net
       103         28 running   46.105.45.230   vm3.server-master.ghaim.net
       104         31 running   178.33.135.64   vm4.server-master.ghaim.net
       105         15 running   178.33.135.65   vm5.server-master.ghaim.net
```

Illustration 47: vzlist une portion des serveurs - OpenVZ

Ces informations sont exportables en Json (-j), pour vérifier l'état des serveurs et automatiser les rapports.

```
NAME
       vzctl - perform various operations on an OpenVZ container

SYNOPSIS
       vzctl [flags] create CTID --parameter value [...]
       vzctl [flags] start CTID [--wait] [--force]
       vzctl [flags] stop CTID [--fast] [--skip-umount]
       vzctl [flags] restart CTID [--wait] [--force] [--fast]
       vzctl [flags] suspend | resume CTID [--dumpfile name]
       vzctl [flags] snapshot CTID [--id uuid] [--name name]
             [--description desc] [--skip-suspend] [--skip-config]
       vzctl [flags] snapshot-switch | snapshot-delete CTID --id uuid
       vzctl [flags] snapshot-list CTID [-H] [-o field[,field...] [--id uuid]
       vzctl [flags] set CTID --parameter value [...]  [--save] [--force]
             [--reset_ub] [--setmode restart|ignore]
       vzctl [flags] destroy | delete | mount | umount | status | quotaon |
             quotaoff | quotainit CTID
       vzctl [flags] console CTID [ttynum]
       vzctl [flags] convert CTID [--layout ploop[:{expanded|plain|raw}]]
       vzctl [flags] compact CTID
       vzctl [flags] exec | exec2 CTID command [arg ...]
       vzctl [flags] enter CTID [--exec command [arg ...]]
       vzctl [flags] runscript CTID script
       vzctl --help | --version
```

Illustration 48: La commande vzctl - OpenVZ

IV.3 La plateforme

Nous entendons par plateforme la couche logicielle sur laquelle reposent nos services. Chaque nœud est équipé de manière à être optimisé à la tâche dont il est chargé, les outils sont donc principalement choisis pour répondre aux critères de stabilité, de performance et d'optimisation.

Le choix du serveur a été étudié pour répondre à nos besoins : **Nginx** est un serveur HTTP open source et un reverse proxy russe. C'est l'un des rares serveurs développés pour le problème des « C10K »[j]. Contrairement aux serveurs traditionnels tels qu'Apache, Nginx ne traite pas chaque requête par un processus dédié, mais utilise un système évolutif et une architecture asynchrone. Ce qui se traduit par de hautes performances, mais également par une charge et une consommation de mémoire particulièrement faibles. Utilisé aujourd'hui par plusieurs sites à fort trafic comme WordPress ou Github [66].

Voilà principalement la plateforme des nœuds architecturaux :

- Nœuds répartiteurs : Nginx+cache ;
- Nœuds de filtrage : Nginx + noyau de filtrage basé sur ModSecurity ;
- Nœuds de base de données : où nous utilisant principalement MySQL.

j C10K est le problème des dix mille connexions simultanées [kegel.com/c10k.html]

IV.4 Le service

IV.4.1 L'API

Afin que chaque client puisse interagir avec l'application, une API est mise à disposition des clients et est contrôlée à l'aide d'une clef unique propre à l'hôte. Cette API donne plus de flexibilité et offre un large domaine d'exploitation comme plugin intégré à l'interface de gestion du serveur (ex. : cpanel). Grâce à l'API d'administration, des configurations de sécurité peuvent être générées semi-automatiquement via des tests/audit.

IV.4.2 La plateforme IHM

La plateforme IHM ou le tableau de bord est une forme d'exploitation de l'API. Cette interface client a été implémentée dans un environnement Web 2.0 basé sur un framework PHP. Ce qui permet une meilleure standardisation du cycle de vie du logiciel (Spécification, développement, maintenance, évolution).

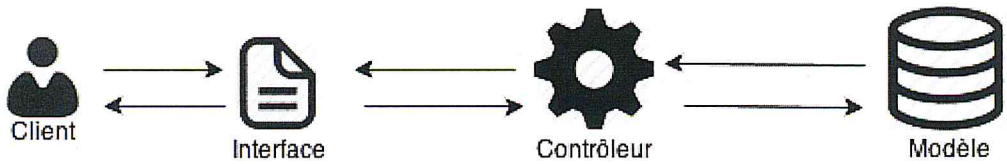


Illustration 49: Exploitation de l'API

IV.4.2.1 Yii

Yii[k] (acronyme de Yes, It Is) est un framework Web open source écrit en PHP, amorcé par Qiang Xue[l] en 2008. Axé sur la performance, il est basé sur une architecture MVC et suit le paradigme de programmation orienté objet/événementielle. La documentation est satisfaisante et la communauté autour du projet est très active. Ce framework favorise les conventions, mais ne les force pas, il offre une structure assez flexible.

Dans l'illustration 49, le modèle exploite des bases de données et des API des autres services.

IV.4.2.2 L'apparence

Vu que notre projet accorde de l'importance à l'ergonomie, à la convivialité et à une image moderne du Web 2.0, on a exploité des outils, comme :

k Site web : http://www.yiiframework.com/ - distribué sous licence BSD

l Il a également développé et maintenu le framework PHP Prado pendant 3 ans. Yii est d'ailleurs le successeur officiel de Prado.

Twitter Bootstrap[m] est un framework HTML/CSS développé par Twitter. Il réunit en un kit des composants HTML et CSS, qui sert à créer les fondations ainsi que les grandes lignes de l'apparence du site. Il offre des dizaines de composants (formulaires, grilles, boutons, menu de navigation, tooltip, etc.), des plugins JavaScript, des charts (diagramme…). Ce qui est très utile pour l'affichage de nos rapports.

Bootstrap est connu pour avoir une excellente compatibilité avec les navigateurs et pour être facilement adaptatif spécialement sur le plan d'intégration, ce qui est relativement économique en matière de temps/ressources. Étant modulaire, il est compatible avec **LESS** le fameux framework CSS pour gérer les feuilles de styles avec plus de souplesse et de rapidité.

JQuery est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant AJAX) et HTML, et a pour but de simplifier des commandes communes de JavaScript.

IV.4.2.3 Aperçu

Voilà, un prototype opérationnel et en ligne sur **ghaim.net**, conçu avec CodeIgniter. Évidemment, le client peut passer par une table de plan et éventuellement payer pour un service, afin d'assurer la survie du projet.

Standard Pack	Pro Pack	Business Pack
50 GO de trafic mensuelle	**1 TO** de trafic mensuelle	**10 TO** de trafic mensuelle
Sécurité basique	Support TLS/SSL	Niveau de sécurité très avancée
Statisiques quotidiennement	Sécurité avancée	Support dédié et téléphonique 24/7
Support communautaire	Statistiques en temps réel	Conformité PCI
CDN et optimisation du site web	Support dédié	Audit professionnel
-	**2000 DA/mois** pour chaque nouveau site	**10.000 DA/mois** pour chaque nouveau site
Gratuit	Essai 15 jours	Essai 15 jours

Illustration 50: Exemple des plans d'abonnement

m http://twitter.github.com/bootstrap/ publié sous licence libre Apache.

Exemple d'ajout d'un nouveau site :

Avoir la configuration des applications de nos clients nous permettra de mieux les connaître.

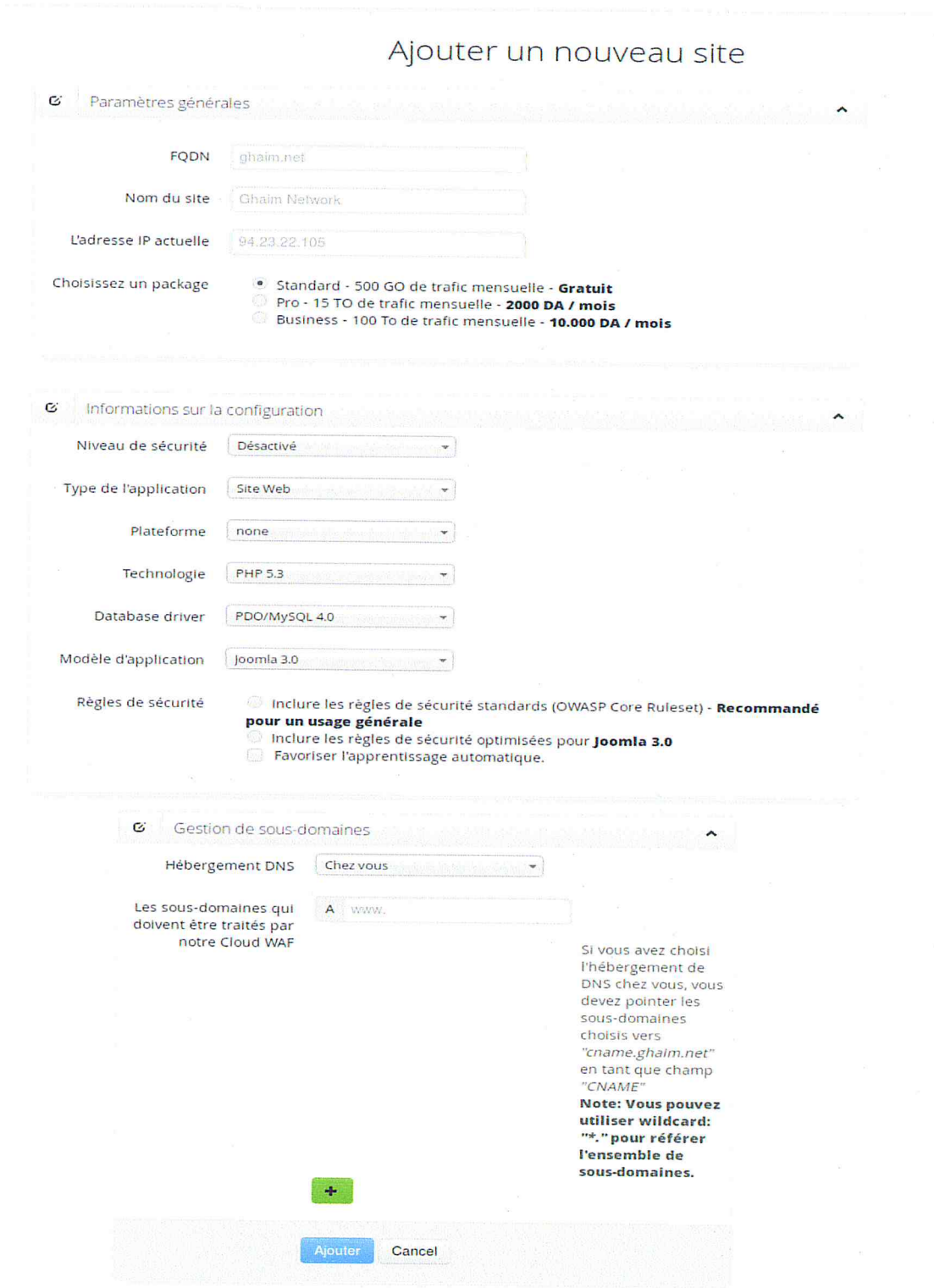


Illustration 51: Ajouter un nouveau site

Tableau de bord

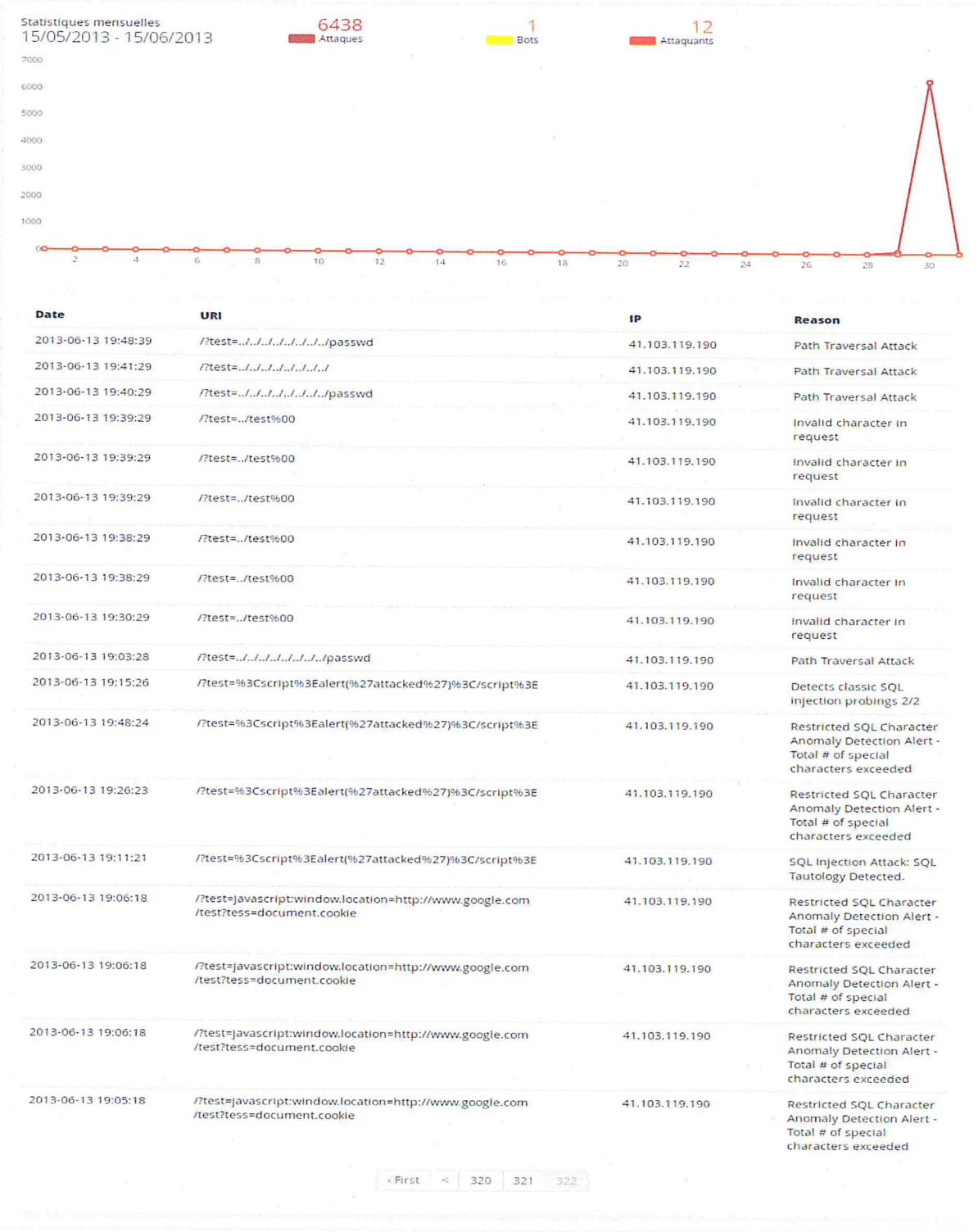
Sur le tableau de bord, on essaye de ne donner que les informations importantes que relèvent nos nœuds sinon ça deviendrait un service d'analyse d'audience qui peut être simplement intégré via une API telle que Google Analytics, même si la notion primaire du service diffère.

Le site offre plusieurs fonctionnalités et rapports comme la gestion de la sécurité, des performances pour le modèle CDN, une assistance et un soutien technique.

Illustration 52: Tableau de bord

Le site est bien sûr accessible depuis d'autres dispositifs : mobile, tablette… l'interface graphique s'adapte au profil des clients. L'interaction du système pourrait être améliorée en mettant à disposition une API de messages texto pour informer l'abonné d'un rapport ou d'un type d'attaque programmé ou depuis le support qui peut même appeler téléphoniquement en cas d'urgence.

Attaques bloquées



Date	URI	IP	Reason
2013-06-13 19:48:39	/?test=../../../../../../../passwd	41.103.119.190	Path Traversal Attack
2013-06-13 19:41:29	/?test=../../../../../../../../	41.103.119.190	Path Traversal Attack
2013-06-13 19:40:29	/?test=../../../../../../../../passwd	41.103.119.190	Path Traversal Attack
2013-06-13 19:39:29	/?test=../test%00	41.103.119.190	Invalid character in request
2013-06-13 19:39:29	/?test=../test%00	41.103.119.190	Invalid character in request
2013-06-13 19:39:29	/?test=../test%00	41.103.119.190	Invalid character in request
2013-06-13 19:38:29	/?test=../test%00	41.103.119.190	Invalid character in request
2013-06-13 19:38:29	/?test=../test%00	41.103.119.190	Invalid character in request
2013-06-13 19:30:29	/?test=../test%00	41.103.119.190	Invalid character in request
2013-06-13 19:03:28	/?test=../../../../../../../../passwd	41.103.119.190	Path Traversal Attack
2013-06-13 19:15:26	/?test=%3Cscript%3Ealert(%27attacked%27)%3C/script%3E	41.103.119.190	Detects classic SQL injection probings 2/2
2013-06-13 19:48:24	/?test=%3Cscript%3Ealert(%27attacked%27)%3C/script%3E	41.103.119.190	Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded
2013-06-13 19:26:23	/?test=%3Cscript%3Ealert(%27attacked%27)%3C/script%3E	41.103.119.190	Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded
2013-06-13 19:11:21	/?test=%3Cscript%3Ealert(%27attacked%27)%3C/script%3E	41.103.119.190	SQL Injection Attack: SQL Tautology Detected.
2013-06-13 19:06:18	/?test=javascript:window.location=http://www.google.com /test?tess=document.cookie	41.103.119.190	Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded
2013-06-13 19:06:18	/?test=javascript:window.location=http://www.google.com /test?tess=document.cookie	41.103.119.190	Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded
2013-06-13 19:06:18	/?test=javascript:window.location=http://www.google.com /test?tess=document.cookie	41.103.119.190	Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded
2013-06-13 19:05:18	/?test=javascript:window.location=http://www.google.com /test?tess=document.cookie	41.103.119.190	Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded

Illustration 53: Attaques bloquées

Règles supplémentaires

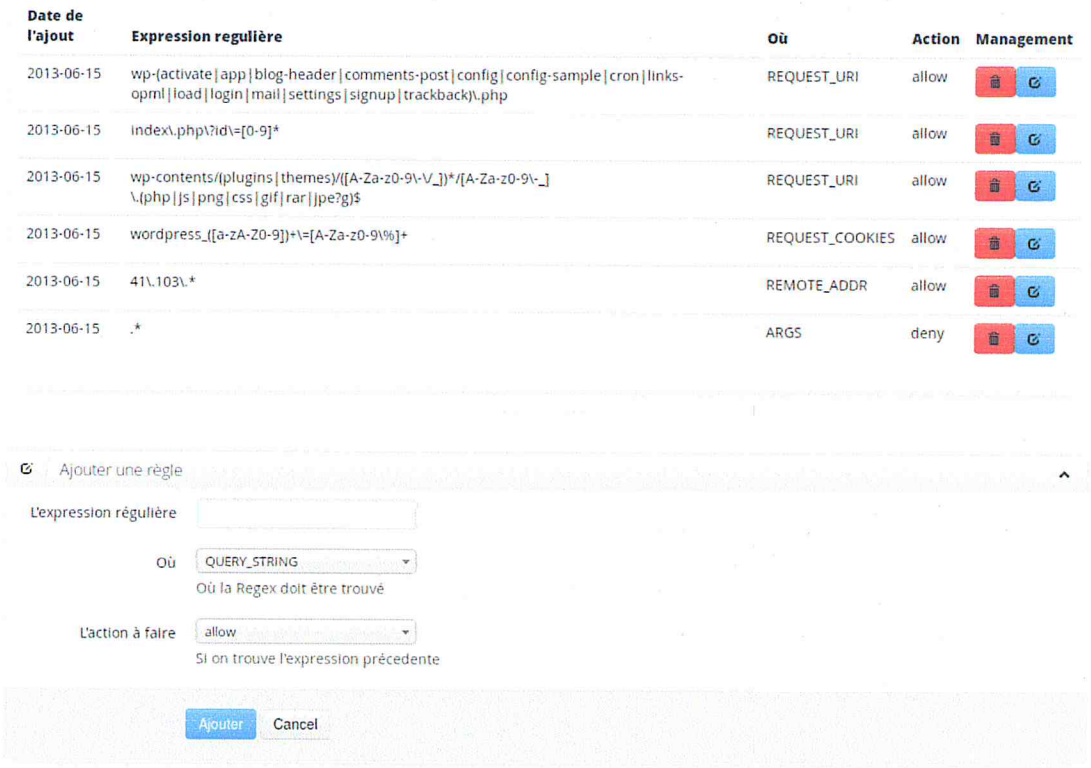


Illustration 54: Règles supplémentaires

Ces règles sont un atout simple pour les utilisateurs avancés.

Un service d'authentification : quand un client cherche à appliquer une politique d'authentification afin de protéger une zone spécifique tels un dossier ou un fichier (ex. : un document PDF). Il peut lui appliquer une règle qui est un peu inspirée du .htpasswd classique. Ce qui peut nous donner ça :

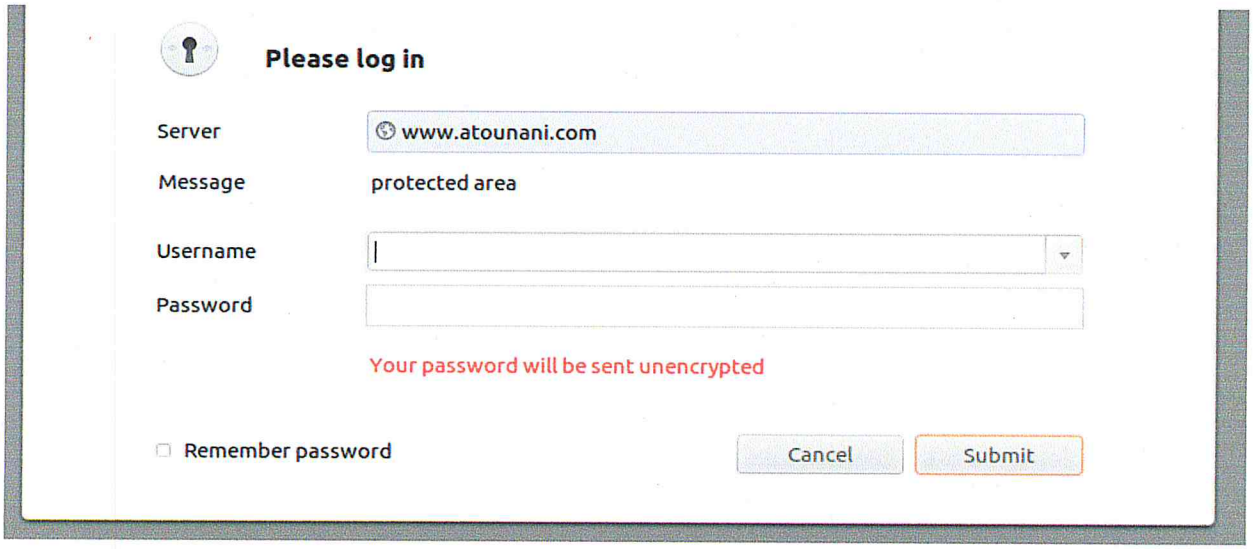


Illustration 55: Authentification

IV.5 Tests

Pour évaluer l'implémentation, nous avons réalisé deux tests principaux qui sont le pentest (test de pénétration) et un benchmark (test de performance) :

IV.5.1 Test de pénétration

Le test de pénétration a été mené principalement en boîte grise. Afin de simuler des cas réels, des applications Web vulnérables (CMS, plugins ...) ont été choisies et mises à l'épreuve. Parmi eux, une application PHP appelée « Mutillidae » qui a été conçue pour apprendre comment exploiter les failles des applications Web. Le test s'est fait manuellement et automatiquement à l'aide d'outils d'analyse tels que «WebCruiser». L'approche consistait à comparer principalement les résultats de deux tests, le premier étant avec une protection désactivée, le second, activée. Voilà l'exemple d'un test basique avec « Mutillidae » :

Version sans protection :

Après avoir lancé l'analyse automatique, les résultats obtenus sont composés de quatre vulnérabilités, deux sont de type SQL Injection et deux autres de type XSS.

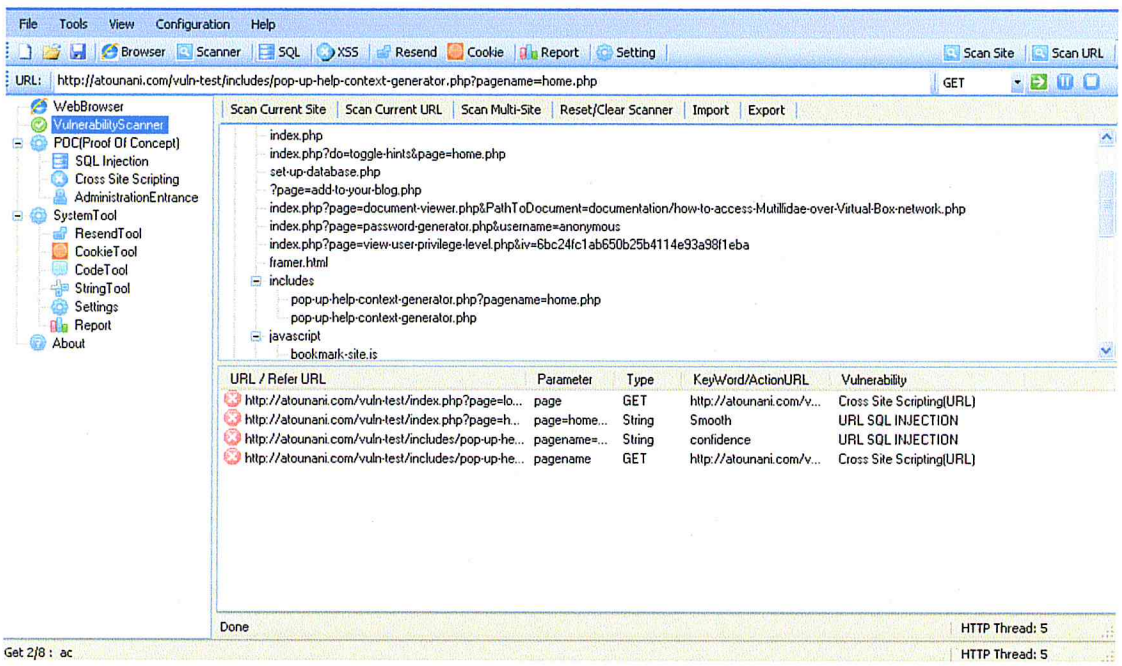


Illustration 56: L'interface de « WebCruiser » sur la version non filtrée du site

On a essayé d'exploiter l'une des vulnérabilités de SQL Injection, où l'on a pu extraire toutes les informations de la BDD :

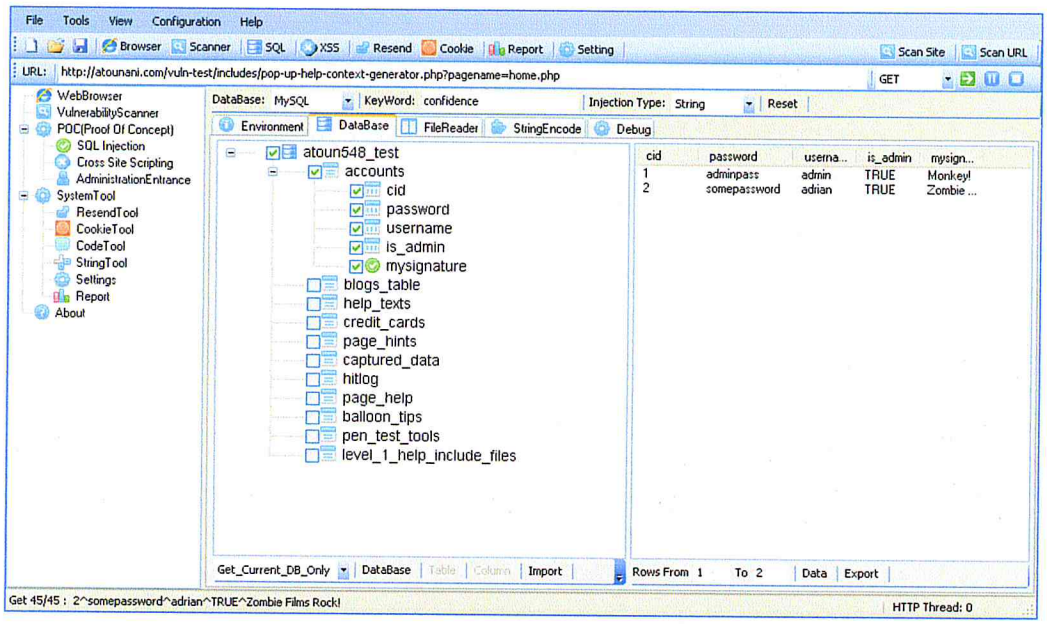


Illustration 57: L'interface de « WebCruiser » en exploitant l'une des failles SQL Injection.

Version avec protection :

Quand une attaque est bloquée, la réaction d'un pare-feu peut varier : comme passer inaperçu ou afficher la page d'avertissement, un exemple :

« /index.php ?' OR 1=1 -- » donnera ce résultat :

Illustration 58: Attaque détectée

On a relancé la même analyse que la première, mais cette fois-ci avec le filtrage activé, la première tentative d'analyse n'a retourné aucune information, l'outil n'a même pas été permis de se connecter au serveur d'origine, car il n'envoie pas de « Accept » dans ses entêtes HTTP, ce comportement est directement bloqué par le CWAF (Règles standards du projet OWASP core

ruleset) où l'expérience accumulée a prouvé que la majorité des requêtes envoyées sans cet entête sont de sources non légitimes.

Pour bien tester la situation, on a permis au client de se connecter à ce site sécurisé sans « Accept ». Le résultat est que l'outil n'a trouvé aucune faille :

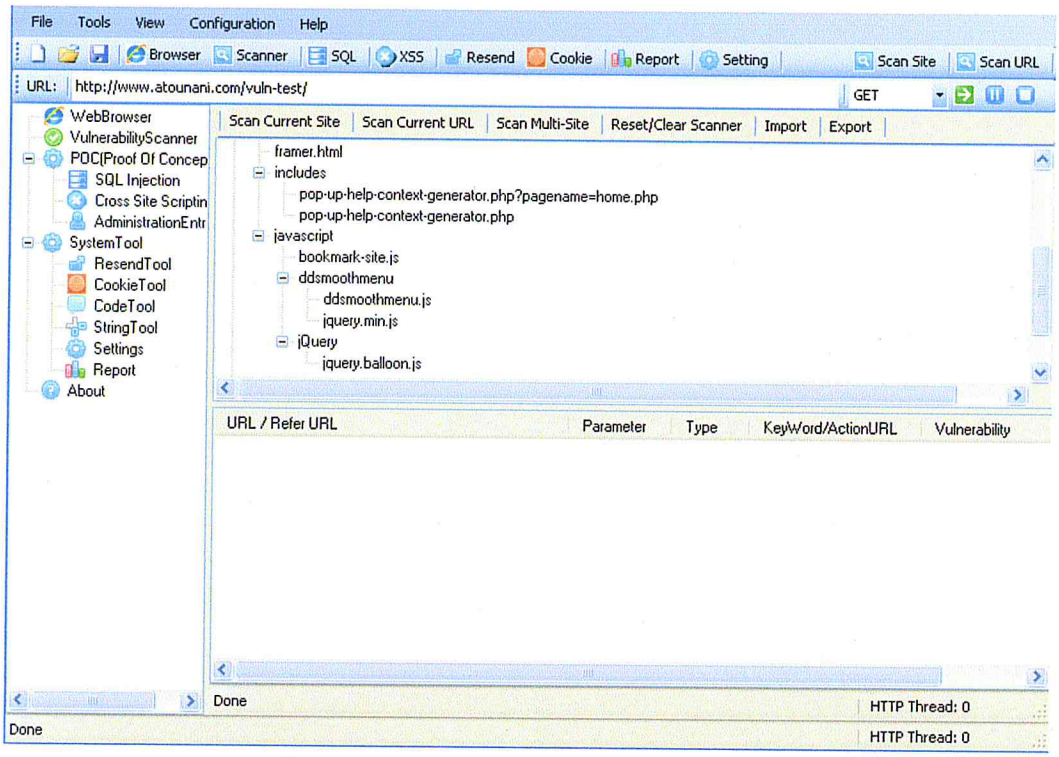


Illustration 59: L'interface de « WebCruiser » après lancement de l'analyse sur la version filtrée du site

Dans le panneau de contrôle du service Ghaim.net, on retrouve beaucoup de requêtes de tests de vulnérabilités qui ont été détectées et rejetées pour différentes raisons :

2013-06-15 18:44:52	/vuln-test/index.php?page=password-generator.php&username=99999999%20oR%207=7	41.103.95.202	SQL Injection Attack
2013-06-15 18:44:52	/vuln-test/index.php?page=password-generator.php&username=%27%20oR%20%27%25%27%3D%27	41.103.95.202	SQL Injection Attack: Common Injection Testing Detected
2013-06-15 18:44:52	/vuln-test/index.php?page=password-generator.php&username=!S!WCRTESTINPUT000001%3C%3E%3c%3e%253c%253e!E!	41.103.95.202	Multiple URL Encoding Detected

Illustration 60: Échantillon du tableau de prévention des attaques sur le service Ghaim.net

IV.5.2 Test de performance

Les tests de performance ont visé principalement les temps de réponse des deux versions des sites web, l'une qui passe par le CWAF et l'autre non.

Les tests sont faits avec l'outil open source « ApacheBench », qui donne la possibilité de lancer un nombre de requêtes défini et permet de manipuler le parallélisme de l'envoi de ces requêtes. On a utilisé aussi le logiciel « GnuPlot » pour visualiser les résultats graphiquement.

L'exemple suivant a été appliqué sur une application PHP dynamique qui a eu un temps d'exécution d'environ 1.213 sec pour la première exécution et 0.099 sec pour les suivantes. Ces tests sont faits depuis l'Algérie avec une liaison à Internet limitée à 1Mbps en Down et 256 Kbps en Up. Le site web qui héberge l'application testée est situé en Europe et la même chose pour le POP du CWAF chargé de nous répondre.

Applications dynamiques :

Le nombre de requêtes envoyées : 500

Nombre de connexions concurrentes : 5

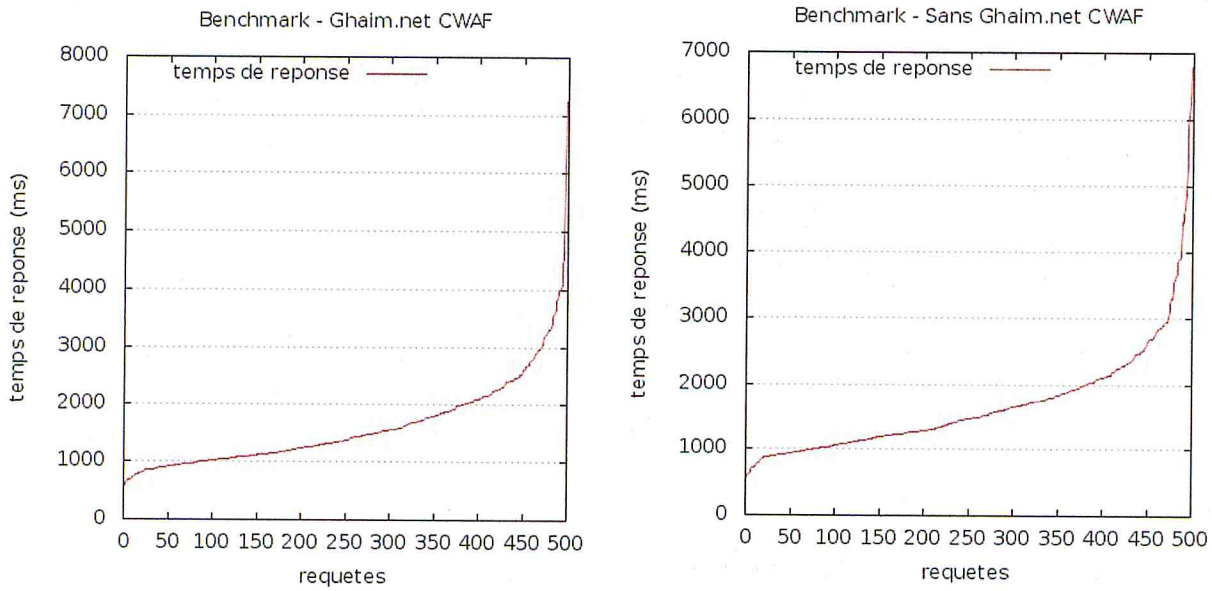


Illustration 61: L'exemple d'un benchmark sur une application dynamique

Fichiers statiques :

Le type de fichier utilisé dans ce test est .css ;

Le nombre de requêtes envoyées : 500

Nombre de connexions concurrentes : 5

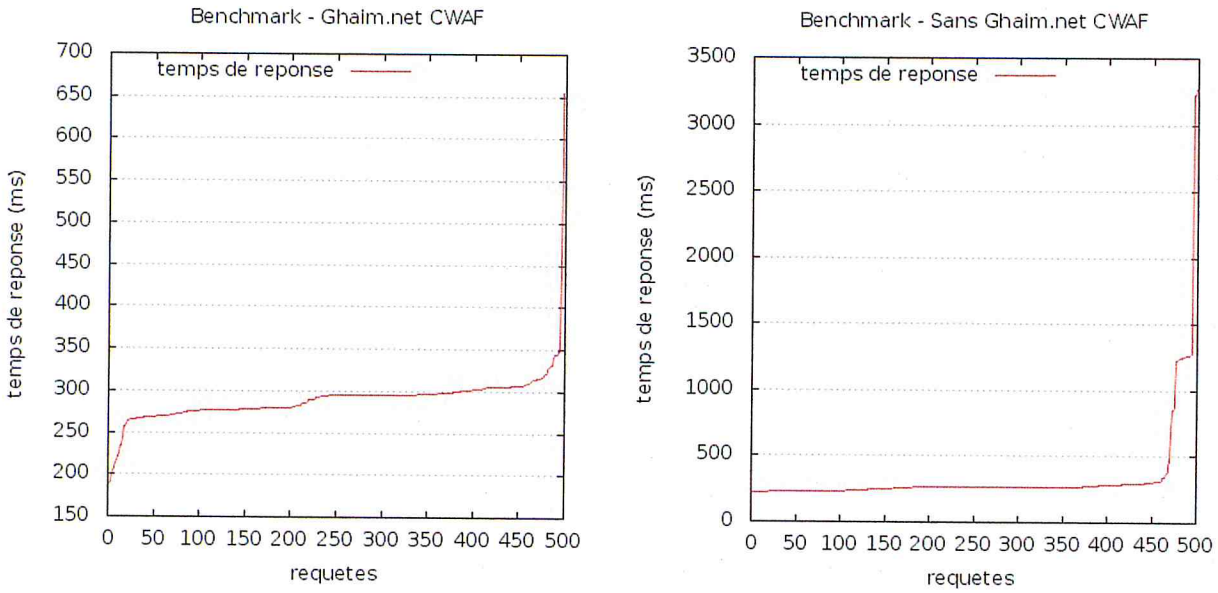


Illustration 62: L'exemple d'un benchmark sur un fichier statique

Pour résumer, dans le cas d'une application dynamique, la différence en général est en général négligeable, quelques millisecondes ajoutées par le CWAF par rapport au serveur d'origine. Pour ce qui concerne les fichiers statiques, ils traversent à peu de près le même trajet que les premières requêtes. Mais il est clair qu'une page qui nécessite beaucoup de ressources sera mieux servie avec le cache, ce qui allégera le serveur d'origine, du fait qu'elles seront distribuées sur l'ensemble des POP. Et avec une large couverture et un service d'accélération « smart cache » qui va par exemple regrouper les ressources, l'optimisation ne sera point discutable.

IV.6 Conclusion

Notre architecture et notre implémentation sont modulaires est tout élément ou composant peut être changé ou amélioré. Par exemple, le noyau des nœuds de filtrage peut évoluer séparément du reste du système. Tout en assurant la haute disponibilité et sans avoir à infliger des changements sur les clients. Les tests de pénétration et de performance sont encourageants, bien qu'ils nécessitent encore plus d'approfondissement dans une utilisation réelle afin d'approuver les résultats.

CONCLUSION GÉNÉRALE

Ces dernières années, en plus des exploits et des faiblesses applicatives Web, la surface des attaques par déni de services distribués s'est étendue, les contrer relève des grands moyens et de l'expertise. L'externalisation de la sécurité des applications Web est une approche qui cherche à remédier et à apporter des solutions à ces problèmes. C'est là que les pare-feux des applications Web distribués en nuage ont vu le jour. Tirant profit des notions du Cloud Computing, ils se présentent en tant que service à la demande chargé de maintenir la sécurité, de filtrer et de surveiller le trafic et de le livrer à l'application. Leur modèle de fonctionnement est simple et se base principalement sur un changement DNS qui au lieu de rediriger les requêtes vers le serveur de l'application d'origine, il les passe au nuage pour être filtrées selon la politique posée pour l'abonné. Ceci économisera probablement les ressources et la bande passante des clients au regard du trafic malveillant, mais engendrera un temps de latence supplémentaire, c'est pourquoi ces services sont appuyés par une stratégie de CDN afin d'optimiser la livraison et le rendu de la solution.

Toutes les solutions existantes, actuellement, sont commerciales, le but de cette étude était de présenter les aspects de ces produits, mais également de développer et de proposer une solution open source et d'innover en matière d'architecture et de stratégies de défense pour assurer une haute disponibilité et de hautes performances dans la sécurité, la livraison et la mitigation des attaques DDoS.

Ce travail, malgré sa valeur, et ses apports pour l'externalisation stratégique de la sécurité et pour les aspects conceptuels d'une architecture puissante, souple et extensible. Il n'est que le premier pas dans d'immenses domaines tels que la sécurité et le Cloud Computing. Cela dit, il est le résultat de ce que l'on a pu extraire pour la courte durée consacrée à la réalisation de ce travail.

Cette étude en est le commencement, mais surtout pas la fin. Dans un domaine aussi variant que la sécurité, ce type de solution favorise l'apprentissage collectif et supervisé de l'environnement, du fait de sa centralisation. Une intelligence poussée assura plus d'adaptabilité aux menaces. C'est aussi un aspect stratégique pour l'évaluation continuelle et le suivi professionnel de la sécurité, en effet, dédier des configurations spécifiques à certaines technologies et à certains produits ou en générer sur mesure au moyen d'un service d'analyse et de test avancé (audits automatiques ou semi-automatiques) optimisera plus les performances de traitement. Ce nuage étant l'entre-deux, le sujet peut être encore élargi davantage avec plus de fonctionnalités et plus de souplesse et de flexibilité

dans le maniement des règles et des politiques de sécurité. Une API avancée du pare-feu pourra interagir avec l'application à protéger pour en devenir le framework.

Parmi d'autres points, l'accélération et le stockage du cache peuvent également être améliorés pour assurer plus de performances, spécialement pour le stockage de masse où l'adoption d'un système de fichiers distribué et performant sera obligatoire, ce qui nous mène à des solutions existantes comme HDFS, CFS, ZFS ou autres, après une étude comparative basée surtout sur la fiabilité et la capacité de rendement du CDN.

L'architecture et l'implémentation sont conçues pour être parfaitement modulaires et extensibles et tout élément peut être amélioré indépendamment. Avec les dernières évolutions des pare-feux applicatifs ou ce que l'on tend à appeler la NGF ou « Next Generation Firewall », les noyaux de filtrage peuvent être améliorés pour supporter un contrôle avancé du réseau tel un IPS intégré.

En espérant que ce projet apportera un plus à la sécurité des applications Web, le chemin vers la sûreté parfaite reste loin dont nous-mêmes sommes la première faille. Et c'est ainsi que se poursuivra la fameuse traque entre gendarmes et voleurs.

RÉFÉRENCES

[1] Netcraft, « December 2012 Web Server Survey ». (04/12/2012). http://news.netcraft.com/archives/2012/12/04/december-2012-web-server-survey.html

[2] Nicolas Cavigneaux, « Les enjeux de la sécurité des applications Web ». (05/07/2011). http://www.synbioz.com/blog/les_enjeux_de_la_securite_des_applications_web

[3] WASC, « Web Application Security Consortium : Threat Classification V1 ». 2004. http://projects.webappsec.org/f/WASC-TC-v1_0.pdf

[4] Conseil des normes de sécurité PCI, « PCI DSS Quick Reference Guide Understanding the Payment Card Industry Data Security Standard version 2.0 ». (2010) https://www.pcisecuritystandards.org/documents/PCI%20SSC%20Quick%20Reference%20Guide.pdf

[5] Ciscomag, « La sécurité des applications Web ». (08/2008). http://www.cisco.com/web/FR/documents/pdfs/newsletter/ciscomag/2008/10/ciscomag_21__Dossier_La_securite_des_applications.pdf

[6] Robert S. Mueller, RSA Cyber Security Conference. (1/03/2012). http://www.fbi.gov/news/speeches/combating-threats-in-the-cyber-world-outsmarting-terrorists-hackers-and-spies

[7] Michel Hoffmann, « Sécurité informatique ». http://www.dicodunet.com/definitions/internet/securite-informatique.htm

[8] CommentCaMarche, « Introduction à la sécurité informatique », Dossier sécurité. http://www.commentcamarche.net/ (consulté le 15/06/2013)

[9] Julien BOURDIN, « Sécurité des applications Web », Klee Group. (18/01/2013) http://blog.kleegroup.com/teknics/?p=493

[10] Wikipédia, « Sécurité des systèmes d'information ». http://fr.wikipedia.org/wiki/Sécurité_des_systèmes_d'information (consulté le 15/06/2013)

[11] Johanne Ulloa, « Pourquoi un WAF », blog about Web Application Security. (09/01/2009). http://johanne.ulloa.org/pourquoi-un-waf.html

[12] CommentCaMarche, « Audits de sécurité », dossier sécurité. http://www.commentcamarche.net/ (consulté le 15/06/2013)

[13] MIKE SHEMA. « Hack Notes, Web Security Portable Reference ». 2e ed. (US, California - 2003) P.140

[14] Shashank Khandelwal et al., « Frontline Techniques to Prevent Web Application Vulnerability ». Volume 2. (IJARCSEE Lab. Inde, 2013) P.211

[15] William R. Cheswick, Steven M. Bellovin et Aviel D. Rubin – « Firewalls and Internet Security: Repelling the Wily Hacker », 2e édition (2003) P.13, Addison-Wesley, Reading, MA.

[16] Juliette André et Renaud Pelloux. « EXPOSE : Traitements Intermédiaires : NAT, Proxy, Firewall ». (10/2002). http://wapiti.telecom-

lille1.eu/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2002/andre-pelloux/resume.pdf

[17] Wikipédia, « Pare-feu ». http://fr.wikipedia.org/wiki/Pare-feu_(informatique) et la version anglaise : Firewall (computing) en.wikipedia.org/wiki/Firewall_(computing) (consulté le 15/06/2013)

[18] OWASP, « Web Application Firewall ».(2013) https://www.owasp.org/index.php/Web_Application_Firewall

[19] Young, G, « An Introduction to Web Application Firewalls. ». Gartner Research, (Mai 2008)

[20] Jim Beechey, « Web Application Firewalls ». SANS Technology Intitute, US 2009. http://www.sans.edu/student-files/projects/200904_01.doc

[21] ModSecurity. (2008). http://www.modsecurity.org

[22] OWASP, « OWASP NAXSI Project ». (08/2012). https://www.owasp.org/index.php/OWASP_NAXSI_Project

[23] Wikipédia, « Application firewall ». http://en.wikipedia.org/wiki/Application_firewall (consulté le 15/06/2013)

[24] Alexander Meisel, « Cloud-Based dWAF A Real World Deployment Case Study ». (05/04/2012). https://www.owasp.org/images/0/03/ASDC12-Cloudbased_dWAF_A_Real_World_Deployment_Case_Study.pdf

[25] Wikipédia, « Cloud computing ». dernière modification de la page comportant la définition le 5 février 2013 à 15:37. http://fr.wikipedia.org/wiki/Cloud_computing (archivée)

[26] National Institute of Standards and Technology, "The NIST Definition of Cloud Computing". (24/07/2011). http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[27] Wikipédia, « Virtualisation ». http://fr.wikipedia.org/wiki/Virtualisation et sa version anglaise : « Virtualization » http://en.wikipedia.org/wiki/Virtualization (consulté le 15/06/2013)

[28] Simia, « De l'état des lieux de la virtualisation libre ». http://www.simia.fr/ssll-presse-virtualisation-perspectives.html

[29] Eric Berthelotubo, « Le cloud computing quel impact organisationnel pour les équipes informatiques des systèmes d'information ? », mémoire de Master en management des systèmes d'information et de communication (2011).

[30] Wikipédia, « Cloud computing ». http://fr.wikipedia.org/wiki/Cloud_computing et la version anglaise : http://en.wikipedia.org/wiki/Cloud_computing (consulté le 15/06/2013)

[31] ADEN, Compubase et Orange Business Services, « Quelle est la place pour la distribution indirecte sur le marché du SaaS ? ». Livre blanc, p.24. (01/2011)

[32] Christophe Delsaux, « Cloud computing : le « Pay as you Grow » est-il une réponse adaptée ? », JDN. http://www.journaldunet.com/solutions/expert/53786/cloud-computing---le---pay-as-you-grow---est-il-une-reponse-adaptee.shtml/ (26/03/2013)

[33] Cloud Computing Competence Center for Security, « What are the difference between Cloud, Cluster and Grid Computing ? ». http://www.cloud-competence-center.com/understanding/difference-cloud-cluster-grid/ (12/2011)

[34] European Network and Information Security Agency, « Cloud computing: benefits, risks and

recommendations for information security », p. 4-5, (11/2009). www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport

[35] The Register, « PlayStation Network hack launched from Amazon EC2 ». (14/05/2011). http://www.theregister.co.uk/2011/05/14/playstation_network_attack_from_amazon/

[36] CNET, « Amazon cloud outage impacts Reddit, Airbnb, Flipboard ». (22/10/2012). http://news.cnet.com/8301-1023_3-57537499-93/amazon-cloud-outage-impacts-reddit-airbnb-flipboard/

[37] 107th Congress Public Law 56, « USA PATRIOT Act : Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism Act ». (10/2001) http://www.gpo.gov/fdsys/pkg/PLAW-107publ56/content-detail.html

[38] The Wall Street Journal, suite à l'Oracle OpenWorld 2008.

[39] Déclaré au quotidien britannique The Guardian.

[40] CNET, « Woz: The cloud is a nightmare », http://news.cnet.com/8301-17852_3-57486930-71/woz-the-cloud-is-a-nightmare/ (5/8/2012)

[41] Renaud Dubourguais et Renaud Feil, « WAF contest », Synacktiv, JSSI OSSIR. (19/03/2013). http://www.ossir.org/jssi/jssi2013/4B.pdf

[42] Xybershield, « Specifications ». https://www.xybershield.com/Overview/Specifications.aspx

[43] Matthew Prince, « Introducing SPDY », blog de cloudflare. (06/2012). http://blog.cloudflare.com/introducing-spdy

[44] Matthew Prince, « Today's Outage Post Mortem », blog de cloudflare. (03/2013) http://blog.cloudflare.com/todays-outage-post-mortem-82515

[45] Imperva Blogger, « Imperva Launches Spin-out Incapsula, Cloud-Based WAF Service », Imperva Data Security Blog. (12/2010). http://blog.imperva.com/2010/12/imperva-launches-spin-out-incapsula-cloud-waf-service.html

[46] United Security Providers, « Sicherheit für Webapplikationen und Webservices », https://www.united-security-providers.ch/de/loesungen/web-access-management/

[47] Akamai, Kona Security Solutions. http://www.akamai.fr/enfr/dl/brochures/Product_Brief_Kona_WAF.pdf

[48] Vmware, « WAPPLES V-Series ». https://solutionexchange.vmware.com/store/products/7755

[49] Easywaf, « Un Aperçu de EasyWAF ». https://www.easywaf.com/fr/features/overview/

[50] Issu d'une entrevue vidéo avec Russia Today (RT), «Stallman: Facebook IS Mass Surveillance ». (08/12/2011). http://rt.com/news/richard-stallman-free-software-875/

[51] GNU, « Qu'est-ce que le logiciel libre ? ». http://www.gnu.org/philosophy/free-sw.html

[52] Zero Sciene Lab, « CloudFlare vs Incapsula vs ModSecurity, Comparative penetration testing analysis report v2.0 ».(13/02/2013). http://zeroscience.mk/files/wafreport2013.pdf

[53] Perezbox, « Protecting Your Website – CloudFlare or Incapsula ». (13/11/2012). http://tonyonsecurity.com/2012/11/13/protecting-your-website-cloudflare-or-incapsula/

[54] Discussion avec Matthew Prince cofondateur de CloudFlare suite au rapport de Zero Sciene Lab sur leur twitter. (03/2013).https://twitter.com/eastdakota/status/307604650436210688

[55] Sun Tzu, l'art de la guerre, Article 4 (Début du Ve siècle av. J.-C.), Traduction de Joseph-Marie Amiot (1772)

[56] Glossaire de sécurité sur la toile (2001), repris par : Wikipédia, Vulnérabilité (informatique), http://fr.wikipedia.org/wiki/Vulnérabilité_(informatique)/ (consulté le 9/06/2013)

[57] Guillaume HARRY, « Failles de sécurité des applications Web ». (03/04/2012)

[58] OWASP, « OWASP Top 10 – 2013 RC1 ». http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013%20-%20RC1.pdf

[59] WASC « WASC Threat Classification V2 ». (2010). http://projects.webappsec.org/f/WASC-TC-v2_0.pdf

[60] OWASP, « Best Practices : Use of Web Application Firewalls » https://www.owasp.org/images/b/b0/Best_Practices_WAF_v105.en.pdf

[61] OWASP, « Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet ». https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

[62] Wikipédia, « Attaque par déni de service ». http://fr.wikipedia.org/wiki/Attaque_par_d%C3%A9ni_de_service (consulté le 15/06/2013)

[63] Jean-François Audenard, « Lutte contre les attaques en DDoS : retour d'expérience ». (19/02/2009). http://www.orange-business.com/fr/blogs/securite/nouvelles-technologies/lutte-contre-les-attaques-en-ddos-retour-dexperience-partie-1

[64] Dave Lee, « Global internet slows after 'biggest attack in history' », BBC News. (27/03/2013). http://www.bbc.co.uk/news/technology-21954636

[65] OpenVz, « Introduction to virtualization ». http://openvz.org/Introduction_to_virtualization et Wikipédia, http://fr.wikipedia.org/wiki/Openvz (consulté le 15/06/2013)

[66] Nginx Wiki, http://wiki.nginx.org/NginxFr

Outre les références citées, la documentation était plus étendue, principalement de sources comme :

- Michael Armbrust, Armando Fox, Rean Griffith and al., "Above The Clouds: A Berkeley View of Cloud Computing" (février 2009)
- Laurent Miltgen-Delinchamp, « Cloud Computing : Définitions et notions de base ». (2012/05/09). http://communication.sysdis.fr/2012/05/09/cloud-computing-dfinitions-et-notions-de-base/
- L'encyclopédie coopérative Wikipédia dans les deux langues française et anglaise, son aide était incontournable pour les aperçus ou les définitions profilées que nous nous affirmions.
- Clément Vouillo, « Qu'est Que Le Cloud Computing ? Tentative de Définition... » (2012) http://mag.welovesaas.com/index.php/2012/quest-que-le-cloud-computing-tentative-de-definition/
- CLUSIF, « Défense en profondeur des applications Web » (Décembre 2011) http://www.clusif.asso.fr/fr/production/ouvrages/pdf/CLUSIF-2011-Defense-en-profondeur-des-applications-Web.pdf
- Qualys, « GUIDE - Sécurité des Applications Web : Comment Minimiser les Risques d'Attaques les plus Courants » http://www.qualys.com/docs/qualys-was-guide-fr.pdf
- Les références techniques des WAF existants sur le marché comme : CloudFlare, Incapsula, Imperva, Akamai...
- Florian Malecki, « Pare-feu nouvelle génération : sécuriser sans compromettre les performances », JDN, http://www.journaldunet.com/solutions/expert/52829/pare-feu-nouvelle-generation---securiser-sans-compromettre-les-performances.shtml (23/11/2012)

GLOSSAIRE

ACL : « Access Control List » est une liste pour la politique de contrôle d'accès.

AnyCast : est une technique d'adressage et de routage permettant de rediriger les données vers le serveur le « plus proche » ou le « plus efficace » selon la politique de routage. (Wikipédia)

API : « Application Programming Interface » une interface de programmation applicative permet d'exploiter et de manipuler des routines ou des fonctionnalités pour un langage ou un service donné.

Bot (robot) : est un agent logiciel automatique ou semi-automatique qui interagit avec les services informatiques et peut simuler l'utilisateur humain, de ce fait, ils sont utilisés pour divers objectifs comme l'abus de fonctionnalités.

Captcha : « Completely Automated Public Turing test to tell Computers and Humans Apart » est un test dont le but est d'affirmer qu'un utilisateur est bien un humain, et non pas un bot.

CDN : Le « Content Delivery Network » est un système de livraison de données optimisé (routage réseau), généralement, accompagné de fonctionnalités telles que le caching.

CLUSIF : Club de la sécurité de l'information français, est une association française à but non lucratif. Elle est l'origine de la méthode de gestion des risques Mehari.

Cookie : ou témoin de connexion, est un fichier texte stocké du côté client, il est communiqué par le serveur en tant qu'entête HTTP et sert à marquer et à sauvegarder des informations propres à ce client et qui sont ensuite interroger pour savoir l'état de ce dernier.

CWAF : acronyme de « Cloud-based Web Application Firewall », est un pare-feu d'applications Web distribué dans l'informatique en nuage.

DDoS : ou « Distributed Denial of Service » déni de service distribué.[voir p.75]

DNS : *Domain Name System* ou système de noms de domaine est un service permettant de lier un identifiant unique à une entité dont l'infrastructure est reliée au réseau internet. On parle notamment de mettre en correspondance un nom de domaine à l'adresse IP du serveur.

DoS : acronyme de *Denial of Service,* une attaque par déni de service vise à rendre indisponible ce dernier.

DRDoS : ou « Distributed Reflection Denial of Service », il s'agit d'une attaque DDoS

réfléchie sur des serveurs DNS pour amplifier son ampleur.

dWAF : acronyme de « distributed Web Application Firewall » est un pare-feu d'applications Web qui peut être distribué dans le réseau.

GeoDNS : permet d'envoyer des réponses DNS différentes selon l'adresse IP à l'origine de la requête. Généralement, c'est la notion de proximité qui est recherchée.

HTTP/HTTPS: est un protocole de transfert hypertexte adapté à un modèle de communication client-serveur. Le HTTPS est la variante du HTTP sécurisée par l'usage des protocoles SSL/TLS.

IDS : *Intrusion Detection System est* un système de détection d'intrusion.[voir p.18]

IHM : est l'interface Homme-Machine qui définit les interactions entre l'utilisateur et le système.

IPS : *Intrusion Prevention System* est un système de prévention d'intrusion.

LB : *Load Balancer* ou répartiteur de charge.[voir p.52]

NFS : « Network File System » est un protocole de partage de fichiers en réseau.

NIST : Acronyme de « National Institute of Standards and Technology », un institut américain des standards et de la technologie.

OS (Operating System) : système d'exploitation.

OSI : est un standard et un modèle de communication abstrait en réseau « Open Systems Interconnection », composé de sept couches (1-Physique, 2-Liaison, 3-Réseau, 4-Transport, 5-Session, 6-Présentation, 7-Application).

OWASP : acronyme de « Open Web Application Security Project », est une communauté concentrée sur la sécurité des applications Web. Elle travaille sur de nombreux projets comme le Top 10 des risques de sécurité applicatifs Web les plus critiques, rédige aussi des articles et des guides pour la sécurité et organise régulièrement des meetings un peu partout dans le monde. Elle développe aussi des outils sur une philosophie est d'être à la fois libre et ouverte à tous.

Patch : un correctif logiciel.

PCI DSS : Payment Card Industry Data Security Standard est un standard de sécurité des données pour les industries de carte de paiement. Dans sa version actuelle (v2.0 – 2010), elle comporte 12 exigences.

POP : « Point Of Presence » ou points de présence représentent des accès au service répartis

géographiquement. Chaque POP est disposé pour répondre aux besoins des utilisateurs à proximité.

POSIX : acronyme de « Portable Operating System Interface » sont un ensemble de standards défini par IEEE et qui porte sur l'API des systèmes d'exploitation Unix-like.

REST : (REpresentational State Transfer) est un style de communication architectural pour les systèmes distribués qui indique une façon d'échanger et de manipuler les données en utilisant par exemple des verbes HTTP comme GET, POST, PUT et DELETE.

Reverse proxy : est habituellement un serveur positionné en frontal des serveurs web, et se charge d'être l'intermédiaire. C'est le même principe mandataire du client, mais pour serveur.[voir p.22]

Round-robin (RR) : est une répartition de charge équitable entre serveurs qui se base sur le principe de cet algorithme (un tourniquet).

SANS : est un institut et un organisme spécialisé dans la recherche et la sécurité des systèmes d'information.

SI : pour désigner un système d'informations.

SSL/TLS : sont des protocoles d'authentification, de chiffrement et de sécurisation des échanges sur réseau.

WAF : *Web Application Firewall* un pare-feu chargé de la sécurité des applications Web.

WASC : « The Web Application Security Consortium » est un consortium constitué d'experts internationaux, d'industriels et d'organisations, qui publie des recueils de bonnes pratiques de sécurité pour le Web ainsi que le rapport « WASC Threat Classification » qui décrit et classe les menaces de sécurité sur les applications Web [57].

WRR : Weighted Round-Robin répartit la charge selon le poids des serveurs qui réfère par exemple à leur capacité.

Zombi : une machine zombie est un ordinateur contrôlé à l'insu de son utilisateur par un hacker. Ce dernier l'utilise alors le plus souvent à des fins malveillantes, par exemple afin d'attaquer d'autres machines en dissimulant sa véritable identité. On parle de botnet ou de réseau de rebots.

TABLES DES ILLUSTRATIONS

TABLE DES MATIÈRES