La SEE contribue à la reconnaissance du niveau scientifique et technique de ses membres et notamment des plus jeunes, étudiants d'universités et des grandes écoles.

Les prix « Jeunes » André Blanc-Lapierre, institués en mémoire de ce grand scientifique, sont déclinés en Prix régionaux, puis Prix National, Ce dernier récompense le meilleur rapport de stage reçu au niveau master.

© VideoFlow / Shutterstock

Etude et intégration de pare-feu applicatifs

Constance Chou

Prix national André Blanc-Lapierre 2021
Ingénieure en intégration de solutions de cybersécurité – Thales SIX GTS France

Introduction

Du fait de la croissance exponentielle de la quantité de contenu en ligne et de la rentabilité des cyberattaques, la sécurité web est un domaine au cœur des préoccupations des entreprises et entités gouvernementales depuis plus d'une décennie. Pourtant, malgré les efforts de leur part pour prioriser la cybersécurité face aux enjeux budgétaires et de productivité, celles-ci se heurtent à certaines limites. En effet, les approches de type DevSecOps et de multiples phases de tests de sécurité (SAST, DAST) avant le passage en production peuvent permettre d'intégrer la sécurité au cœur des nouvelles applications conçues en interne. Dans certains cas, il peut cependant être impossible de sécuriser l'application elle-même ou d'appliquer des correctifs dans des délais convenables. Cela peut advenir pour une ancienne application de production qui ne peut pas être entièrement refondue, pour des applications trop critiques pour être indisponibles le temps d'une opération de maintenance sans long préavis, ou encore pour des applications fournies par des acteurs externes qu'il nous est impossible de contrôler.

Ainsi, implémenter la sécurité dans l'application elle-même et la rectifier à la source si une vulnérabilité est si-

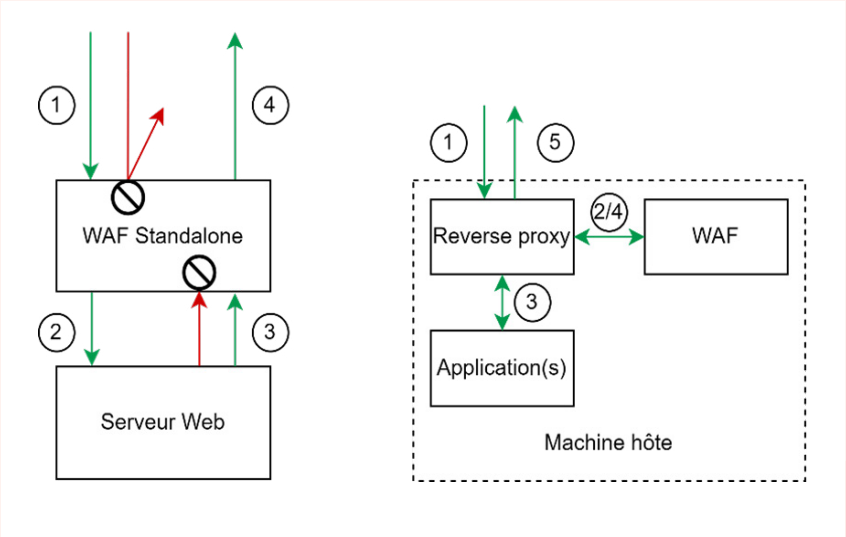

Figure 1 : Exemples d'architecture WAF.

gnalée n'est pas toujours possible ou acceptable pour une entreprise. C'est pour répondre à ce besoin que les *Web Application Firewalls* (WAF), ou pare-feu applicatifs, ont été développés. Au moyen d'une politique de sécurité modifiable à la volée (*virtual patching*), ils permettent d'analyser et filtrer le trafic web en amont de l'application, qui peut ainsi rester inchangée en attendant un correctif.

La technologie des pare-feu applicatifs

Fonctionnement

Un WAF permet de monitorer, filtrer et journaliser les flux HTTP/HTTPS entrants et sortants du site web. Il se positionne donc sur la couche applicative (L7) du modèle OSI [1]. Il dispose d'une politique de sécurité qui s'applique à ces flux afin de déterminer quelle(s) action(s) leur appliquer : accepter, journaliser, lever une alerte, bloquer le flux, nettoyer la requête, etc.

Un WAF peut venir sous la forme d'une *security appliance* (boîtier de sécurité) autonome, ou d'un module de *reverse proxy* [2] (figure 1). Il se positionne entre l'utilisateur et le serveur web afin d'analyser le trafic et de repérer des motifs non autorisés, généralement à l'aide d'un ensemble de règles, ou signatures, nommé *ruleset*. Il est ainsi à même de protéger un ou plusieurs sites web d'une multitude de vulnérabilités : injections SQL, exploitations de *Cross Site Scripting* (XSS), *Cross-Site Request Forgery* (CSRF), injections de commandes de toutes sortes, empoisonnement de session, *Path Traversal*, *Local/Remote File Inclusion* (LFI/RFI), etc. Il peut également vérifier les flux sortants afin d'empêcher les fuites de données.

Ainsi, un WAF, bien implémenté, a un rayon de protection très large, et cette technologie autrefois considérée trop contraignante devient un élément de plus en plus incontournable au sein des architectures web.

Filtrage et modèles de sécurité

Le processus de décision sur l'action à effectuer varie selon les solutions WAF. Dans de nombreux cas, l'action est décidée par un *score d'anomalie*. Un score indépendant existe pour de nombreuses sous-catégories d'attaque, et à chaque signature déclenchée, le score d'anomalie de la requête est incrémenté d'une valeur donnée, dépendante de la signature. Lorsque l'un des scores de la requête atteint un certain seuil, configurable par l'utilisateur, l'action définie est réalisée : lever une alerte et le cas échéant bloquer et journaliser la requête.

Afin d'opérer le filtrage, les solutions WAF peuvent employer deux modèles de sécurité (figure 2), ou une combinaison de ceux-ci.

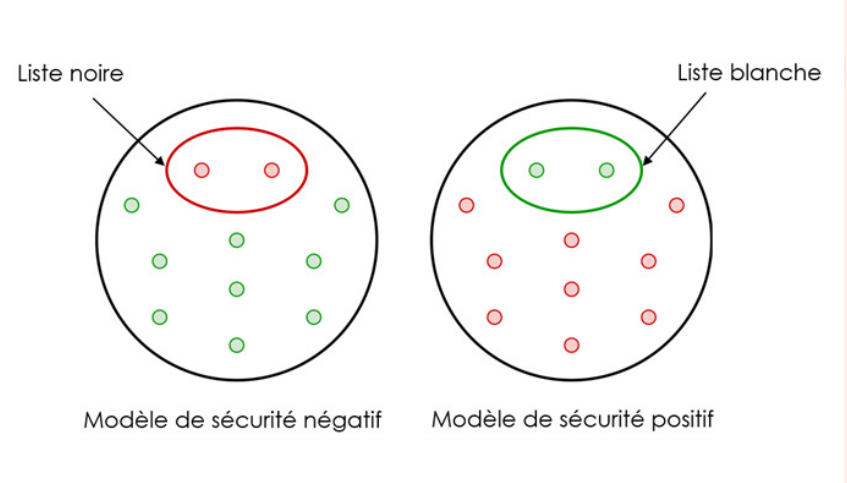

Figure 2 : Représentation des modèles de sécurité applicables au filtrage.

1 Norme ISO décrivant un modèle de communication réseau par couches.

2 Serveur mandataire inverse : Type de serveur usuellement placé en frontal d'un ou plusieurs serveurs web. Il permet à des utilisateurs externes de contacter des applications internes, et offre des services de sécurité, répartition de charge, etc.

Modèle négatif

Le modèle négatif, ou par liste noire, est le mode de fonctionnement traditionnel d'un WAF : tout ce qui n'est pas interdit explicitement est autorisé. Un jeu de règles très détaillé est donc utilisé afin de recenser les motifs interdits dans une requête. Des *ruleset* clé en main existent, en particulier le *Core Rule Set* de l'OWASP [3], auquel quelques signatures supplémentaires spécifiques aux besoins du projet peuvent être ajoutées. Il faut ensuite configurer des exceptions, dépendantes de l'application à protéger, afin d'éviter les faux positifs et donc le blocage de requêtes légitimes. Ce modèle est le plus simple à configurer initialement, en revanche, il ne protège pas des attaques non répertoriées dans le *ruleset*, en particulier les *Zero Day* (attaques encore inconnues).

Modèle positif

Plus récemment, certains WAF ont adopté un comportement opposé avec un modèle positif, ou par liste blanche : tout ce qui n'est pas autorisé explicitement est interdit. Il faut donc lister l'ensemble des requêtes et paramètres acceptables, selon l'application et nos cas d'usage. Ce travail de configuration est bien plus complexe que dans le cas d'un modèle négatif ; cependant il peut être grandement simplifié par des outils d'apprentissage du trafic légitime. Le modèle par liste blanche est généralement préférable car il protège également des vulnérabilités encore inconnues ou des attaques contre lesquelles nous n'aurions pas pensé à nous prémunir, sans nécessiter des mises à jour trop fréquentes du *ruleset*. Cependant, il requiert un important travail de configuration du trafic autorisé afin de ne pas impacter l'expérience utilisateur, et une configuration trop laxiste des listes blanches peut fortement abaisser le niveau de sécurité offert.

3 Open Web Application Security Project : communauté ouverte créée en 2001, qui publie des recommandations relatives à la sécurité des applications et propose des outils et méthodes pour implémenter et tester la sécurité web.

Modèles hybrides

Un exemple de modèle hybride est celui de la solution NAXSI : un *ruleset* très réduit permet de détecter dans le trafic HTTP/HTTPS des « primitives d'attaque », des motifs courts très génériques nécessaires à un grand nombre d'attaques. Cependant, il y a évidemment de nombreux cas dans lesquels une primitive d'attaque pourrait être légitime. Afin d'éviter autant que possible les faux positifs, il faut donc ajouter à ce *ruleset* un grand nombre de règles de *whitelisting*. Celles-ci permettent de spécifier des exceptions, en indiquant tous les cas de présence légitime de primitives d'attaque dans le contexte de notre application. Ce mode de fonctionnement offre des avantages similaires à ceux du modèle positif et minimise le nombre de signatures nécessaires, améliorant ainsi la maintenabilité ainsi que les performances en termes de latence.

Éventuellement, ces modèles de filtrage peuvent ensuite être complétés par une analyse et une corrélation par des algorithmes d'intelligence artificielle.

Place dans l'écosystème des solutions de sécurité

Les solutions de *Web Application Firewall* se sont multipliées au cours des dernières années, et peuvent être très diverses. Les fonctionnalités, performances et méthodes d'implémentation varient, en particulier entre solutions libres et propriétaires. Parmi ces fonctionnalités, nous pouvons citer l'identification de motifs suspects pouvant indiquer une tentative d'exploitation de vulnérabilité web, l'apprentissage du trafic légitime, l'analyse comportementale, le filtrage des adresses IP, le repérage et blocage de fuites d'information, la protection contre les attaques par déni de service ou l'attaque par force brute d'identifiants, etc.

Ces fonctionnalités demeurent majoritairement limitées à la couche applicative, et ne remplacent en rien l'implémentation d'un filtrage réseau en amont des serveurs web. Il peut donc être intéressant de coupler l'utilisation d'un WAF avec l'emploi d'une solution de détection d'intrusion réseau (*Network Intrusion Detection/Prevention System* - NIDS). Il s'agit d'une sonde réseau, pouvant permettre de se protéger d'attaques exploitant des protocoles de couches plus basses dans le modèle OSI. Par exemple, un NIDS serait à même de détecter une déformation malveillante d'un segment TCP (couche Transport) ou d'un paquet IP (couche Réseau). Notons que certains NIDS implémentent une gestion de la couche applicative, mais leur compréhension de ce type de trafic demeure bien plus superficielle que celle d'un outil dédié tel que le WAF.

Un pare-feu applicatif gagne également à être combiné avec une solution RASP (*Runtime Application Self-Protection*). Tout comme les WAF, cette technologie est dédiée à l'analyse de données applicatives, mais utilise un paradigme radicalement différent, et très complémentaire. Elle se greffe directement au code d'une application et permet de surveiller l'exécution de cette dernière et ses appels systèmes. Le RASP permet une meilleure compréhension du fonctionnement interne de l'application, menant à une très forte réduction des faux positifs, et rendant inopérantes les tentatives d'offuscation des attaques. En effet, la donnée est analysée non pas telle qu'elle est réceptionnée, mais telle qu'elle est interprétée par l'application.

Cependant, l'analyse par le RASP est tardive et très localisée. Il est difficile d'assurer ainsi une protection exhaustive de l'ensemble des briques logicielles. Le WAF apporte quant à lui une meilleure visibilité sur l'ensemble du système et du trafic, et donc plus de contexte dans l'analyse. Il permet également le *monitoring* en amont des tentatives d'attaques bruyantes et pas seulement des attaques fonctionnelles.

Les deux solutions implémentées ensemble offrent une défense en profondeur de qualité.

Objectifs

L'objectif du stage était d'étudier le fonctionnement, les avantages et limitations de la technologie des pare-feu applicatifs et d'évaluer diverses solutions WAF, afin d'en sélectionner une à intégrer pour répondre aux besoins de projets qui ne seront pas détaillés ici. Pour justifier auprès de clients la légitimité de ce choix, il a fallu mettre au point une méthodologie d'évaluation des performances des pare-feu applicatifs.

Ensuite, la solution retenue a été implémentée dans plusieurs environnements, ce qui a permis d'aborder divers enjeux de configuration et de sécurité qui seront brièvement présentés ci-dessous. Enfin, il a fallu simplifier au maximum le déploiement et la configuration du WAF pour un système donné. Dans le cadre de cette démarche, les possibilités d'automatisation de l'intégration d'un WAF et les limites de cette configuration automatisée ont été étudiées.

Méthodologie d'évaluation

Critères de sélection

Dans le cadre de cette étude, il a fallu déterminer des critères d'évaluation des différentes solutions WAF. Certaines considérations étant spécifiques aux besoins du projet, ne sont présentés ci-après que les critères génériques, pondérés par leur importance estimée :

• la stabilité de la solution, évaluée sur la base de l'état du projet et de ses dépendances : différences introduites entre deux versions majeures, durée du cycle de vie d'une version, etc. (valeur : 6) ;

• la proportion des types de vulnérabilités gérés, c'est-à-dire contre quelles familles de vulnérabilités protège le WAF (valeur : 4) ;

• la couverture offerte pour un type de vulnérabilité donné : typiquement, est-ce qu'un outil gérant les injections SQL parviendra à bloquer plutôt 70 ou 90 % d'entre elles (valeur : 4) ;

• la performance, en termes de latence et de ressources physiques utilisées (valeur : 3) ;

• la flexibilité de la solution : à quel point celle-ci est-elle configurable et va pouvoir s'adapter aux divers projets et applications ? (valeur : 3) ;

• la simplicité d'installation et de configuration, évaluée sur la durée d'intégration de la solution par une personne expérimentée, ainsi que la spécificité et le nombre des règles à définir (valeur : 2) ;

• la simplicité de prise en main : à quel point la technologie nécessite-t-elle une connaissance précise de son fonctionnement, à quel point la documentation est-elle détaillée ou la communauté est-elle active ? (valeur : 2) ;

• les besoins en termes de maintien en condition opérationnelle : fréquence des mises à jour du jeu de règles, de l'ajout d'exceptions, sensibilité aux évolutions de l'application protégée, etc. (valeur : 2).

Méthode et environnement

Les critères 1, 2 et 8 ont pu être calculés sur la base d'une étude bibliographique. Les critères 5, 6 et 7 ont été évalués lors de l'implémentation de chacune des solutions WAF à analyser.

Pour cela, une architecture de test a été déployée (figure 3), avec une machine GNU/Linux dédiée par solution à évaluer, disposant si nécessaire d'un *reverse proxy*, et placée en protection de diverses applications web.

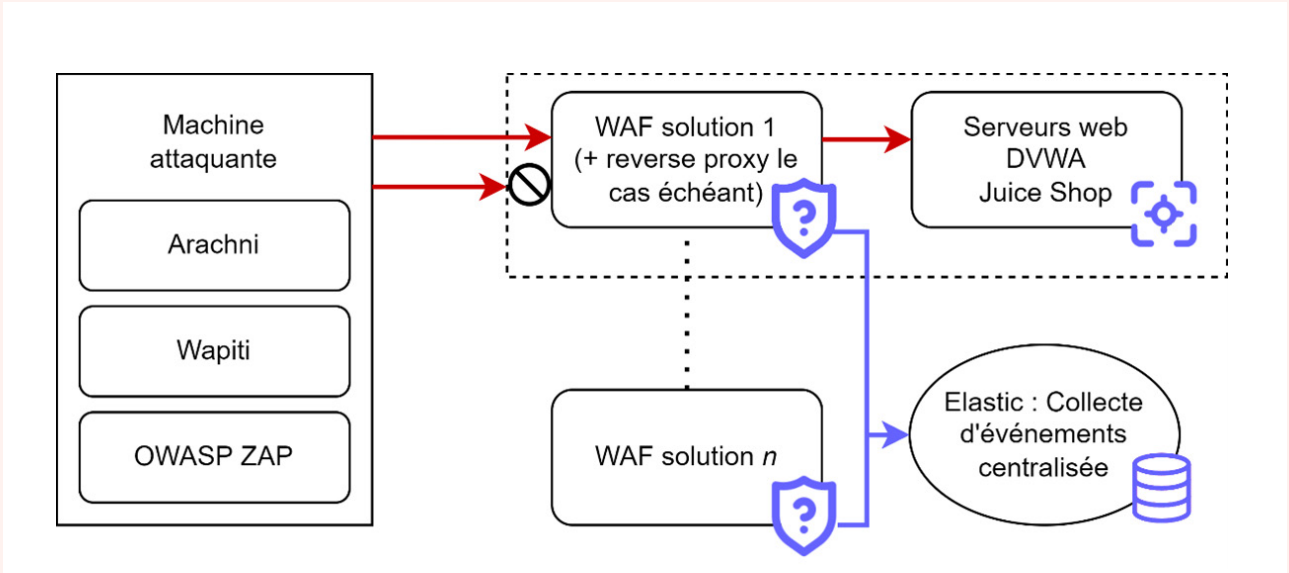

Figure 3 : Architecture de l'environnement de test.

Une solution de remontée des alertes de sécurité a également été déployée. Certaines solutions WAF, majoritairement des solutions propriétaires, disposent d'une interface graphique de *monitoring*. Dans d'autres cas, des outils de visualisation externes dédiés aux données issues de pare-feu applicatifs peuvent être ajoutés au besoin, comme par exemple l'interface WAF-FLE pour ModSecurity. Cependant, pour recueillir simultanément les retours de plusieurs WAF à des fins de comparaison, la solution de visualisation retenue a été un outil plus générique et puissant : la suite Elastic. Il s'agit d'un ensemble de logiciels permettant de recueillir, formater finement, stocker et exploiter des données.

Le critère 4 était non pertinent dans le cadre spécifique du projet et n'a pas été mesuré. Enfin, le critère 3 a nécessité des tests en situation réelle, en générant des attaques contre des applications web à protéger.

Test de couverture de protection

Le test d'évaluation de la couverture moyenne offerte pour un type de vulnérabilité donné a été réalisé au moyen d'attaques réelles contre diverses applications volontairement vulnérables protégées successivement par chaque solution WAF. Ce type d'application a pour objectif de sensibiliser ses utilisateurs à la sécurité web, ainsi que favoriser l'autoformation de *pentesters* ou de développeurs dans un cadre légal. Dans notre cas, cela permet d'avoir à disposition un grand nombre de vulnérabilités aisément exploitables, uniquement protégées par le pare-feu applicatif testé, et donc d'évaluer pour chacune d'entre elles si le WAF est à même de bloquer une tentative d'exploitation.

Notre choix s'est finalement porté sur deux solutions, *Damn Vulnerable Web Application* (DVWA) ainsi que le *Juice Shop* de l'OWASP. Il semblait pertinent de combiner ces applications, car elles utilisent des technologies radicalement différentes. D'une part, DVWA est une application relativement traditionnelle, en PHP et MySQL. Ces technologies représentent une forte proportion des sites web actuellement disponibles en ligne. De l'autre, le *Juice Shop* utilise des technologies plus modernes [4], qui ont récemment pris beaucoup d'ampleur. Le cumul de ces deux solutions permet donc d'obtenir des tests suffisamment représentatifs des applications et vulnérabilités que le WAF devra être capable de gérer.

Pour réaliser les tentatives d'exploitation, des scanners de vulnérabilités automatisés ont été employés. En effet, contrairement à des attaques manuelles, ceux-ci permettent d'effectuer des tests reproductibles et raisonnablement représentatifs, en un temps restreint, avec un résultat quantifiable. Pour plus d'exhaustivité, trois scanners différents ont été utilisés parmi les solutions les plus populaires : Arachni, l'OWASP ZAP et Wapiti. Les performances des solutions WAF ont été confrontées sur la base du nombre et du type de vulnérabilités trouvées par les scanners. Une méthode d'interprétation du retour de ces différents scanners, non détaillée ici, a été mise au point en sorte que leurs résultats soient comparables entre eux.

Intégration d'un WAF

Considérations de sécurisation

Une fois une solution WAF appropriée au projet sélectionnée, il faut l'intégrer au système existant. Rappelons tout d'abord les considérations de sécurité génériques. Les supports matériels (serveur, disque, *appliance* dédiée, etc.) doivent être dupliqués. S'il s'agit d'une solution logicielle, à installer sur une machine GNU/Linux, le système d'exploitation lui-même doit être durci, ainsi que la solution de *reverse proxy* employée le cas échéant. Pour cela, divers référentiels de sécurité peuvent être appliqués, comme les guides de l'Agence nationale de la sécurité des systèmes d'information (ANSSI) ou les directives du *Center for Internet Security* (CIS).

Dans le cas où la solution WAF viendrait sous la forme d'un ou plusieurs exécutable(s), ceux-ci doivent être compilés avec les diverses options de durcissement recommandées, telles que le PIE (*Position-Independent Executable*), l'utilisation de canaris, le *full* RELRO (*RELocation Read Only*), la pile non exécutable, etc. Aucun compilateur ne doit être installé sur les machines de production.

Enfin, certaines précautions plus spécifiques aux pare-feu applicatifs sont nécessaires. Par exemple, *logrotate* a été utilisé afin de s'assurer que les journaux ne remplissent pas l'ensemble de l'espace de stockage, à la fois pour des enjeux de *forensics*, mais aussi et surtout pour éviter un potentiel déni de service. Les permissions sur les fichiers et les droits des comptes services sur la machine doivent être correctement minimisés. En particulier, l'accès aux journaux d'alerte du WAF doit être restreint au maximum ; en effet, ceux-ci peuvent parfois contenir des informations sensibles, telles que des données personnelles ou des mots de passe en clair. Il faut bien prendre en compte le RGPD [5] dans le recueil, l'analyse et le stockage des données issues du WAF, qui peuvent être extrêmement détaillées.

Automatisation et configuration

Lors de l'automatisation, l'enjeu est de pouvoir installer et configurer la solu-

4 Il emploie du Node.js avec le *framework* Express et un *frontend* en Angular.

5 Règlement général sur la protection des données de l'Union européenne adopté en 2016.

"L'enjeu lors de la configuration d'exceptions est de minimiser la perte de sécurité, en créant des exceptions extrêmement précises."

tion de façon rapide et reproductible, en s'adaptant aisément à de nombreux environnements. L'installation sécurisée d'une solution WAF a pu être automatisée sans difficulté au moyen d'Ansible [6] ; en revanche, la configuration détaillée de sa politique de sécurité est plus complexe.

Dans le cas d'un WAF utilisant un modèle de sécurité négatif, nous avons pu obtenir un résultat assez satisfaisant en employant le *Core Rule Set* de l'OWASP et en ayant au préalable mis au point un ensemble d'exceptions, spécifique à l'application à protéger, mais généralisable à un grand nombre d'environnements. L'enjeu lors de la configuration d'exceptions est de minimiser la perte de sécurité, en créant des exceptions extrêmement précises. Il s'agit par exemple de désactiver une unique signature, pour un paramètre HTTP particulier, sur un URL donné. Cependant, il faut également rendre ces exceptions suffisamment générales pour s'assurer que toute modification mineure de l'application n'introduise pas immédiatement un grand nombre de faux positifs, ceci à l'aide d'expressions rationnelles par exemple.

En revanche, la configuration d'un WAF par liste blanche étant bien plus dépendante des cas d'usage, les résultats de l'automatisation sont très mitigés. Des outils de configuration automatique des listes blanches par apprentissage du trafic légitime existent, permettant de générer les règles nécessaires au non-blocage d'un ensemble de requêtes de référence. Cependant, l'obtention d'un ensemble représentatif desdites requêtes est complexe. Tout d'abord, cela implique une période d'apprentissage incompressible avant la mise en production, ce qui n'est pas compatible avec un déploiement rapide et automatisé tel que souhaité. De plus, cela nécessite d'exposer l'application à suffisamment d'utilisateurs pour générer un trafic représentatif. Or, durant cette phase, l'application est vulnérable, et toute tentative d'attaque contre celle-ci peut causer l'inclusion de vulnérabilités dans la configuration du WAF en mode apprentissage. Enfin, les règles ainsi obtenues ne sont pas toujours parfaitement optimisées.

Notons que des tests de vérification et validation du bon fonctionnement de l'application web et de ses cas d'usage sont nécessaires après l'implémentation d'un pare-feu applicatif, afin de s'assurer que celui-ci n'impacte pas l'expérience utilisateur et ne génère pas un trop grand nombre de faux positifs pouvant donner lieu à un déni de service.

Conclusion

La technologie des WAF est un élément incontournable de la sécurité applicative, qui offre entre autres une protection contre le Top 10 des vulnérabilités web recensées par l'OWASP, soit toutes les attaques courantes. Elle permet une remontée d'informations détaillées en temps réel et dispose de configurations très flexibles.

Cependant, elle a un lourd coût d'installation initiale et de maintien en condition opérationnelle ; elle nécessite d'être capable de formaliser les cas d'usage de l'application protégée et de les tester de façon exhaustive. Une partie importante du déploiement peut être automatisée, mais la phase de configuration initiale demeure relativement longue, en particulier dans le cas d'un WAF s'appuyant sur un modèle positif, bien que celui-ci soit plus intéressant en termes de sécurité. Le risque d'impact sur l'expérience utilisateur est également élevé.

Notons que si le WAF est combiné à d'autres solutions de protection, par exemple un RASP, il est possible de miser sur une politique de sécurité un peu moins stricte. Une politique plus équilibrée sera un peu plus vulnérable, mais causera nettement moins de faux positifs et d'impact sur le service. ■

6 Logiciel libre développé en Python, qui permet d'effectuer des configurations et déploiements automatisés depuis une machine distante, sans nécessiter l'installation d'un agent dédié sur la cible.

À propos

Constance Chou est diplômée de Télécom SudParis, en spécialité « Sécurité des Systèmes et des Réseaux ». Elle a effectué son stage chez Thales SIX dans le secteur Sécurité des Technologies de l'Information, au sein du service Intégration Validation Solutions.

Remerciements à Thales SIX GTS France et Télécom SudParis, et tout particulièrement aux encadrants de ce projet : M. Gregory Blanc en tant qu'enseignant référent et M. Rémi Di Valentin en tant que maître de stage.