Analyse des requêtes http pour la détection d'intrusion web

par

Othmane LAGRINI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE
LA MAÎTRISE AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M. Sc. A

MONTRÉAL, LE 28 OCTOBRE 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Chamseddine Talhi, directeur de mémoire
Département Génie logiciel et des TI à l'École de technologie supérieure

M. Ouni Ali, président du jury
Département Génie logiciel et des TI à l'École de technologie supérieure

M. Kaiwen Zhang, membre du jury
Département Génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 27 SEPTEMBER 2021

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEU

REMERCIEMENTS

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je voudrais tout d'abord adresser toute ma reconnaissance à mon directeur de ce mémoire, Monsieur Chamseddine Talhi, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je désire aussi remercier les professeurs de l'Ets, qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

Je tiens à remercier spécialement à Monsieur Oussama Boudar, qui fut la première personne à me faire découvrir le sujet qui a guidé mon mémoire.

Je voudrais exprimer ma reconnaissance envers mes parents, les amis et collègues qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche.

Un grand merci à Jamila El Youssefi pour ses conseils concernant mon style d'écriture, ils ont grandement facilité mon travail.

Enfin, je tiens à témoigner toute ma gratitude à ma femme Fatim Zahra Moufakkir et mes frères pour leur confiance et leur soutien inestimable

Analyse des requêtes http pour la détection d'intrusion web

Othmane LAGRINI

RÉSUMÉ

De nombreux problèmes de sécurité des applications Web liés aux intrusions sont dus au développement rapide des applications Web. Pour réduire le risque de problèmes d'application Web, les développeurs d'applications Web doivent prendre des mesures pour écrire des applications sécurisées afin d'éviter les attaques connues. Lorsque de telles mesures échouent, il est important de détecter ces attaques et de trouver la source des attaques pour réduire les risques estimés. La détection d'intrusion est l'une des techniques puissantes conçues pour identifier et prévenir les dommages au système.

La plupart des techniques défensives des systèmes d'intrusion Web ne sont pas capables de faire face à la complexité des cyberattaques dans les applications Web. Cependant, les approches d'apprentissage automatique pourraient aider à détecter les attaques d'applications Web connues et inconnues. Dans ce mémoire, nous présentons des techniques d'apprentissage automatique pour classer les requêtes HTTP dans le jeu de données bien connu CSIC 2010 HTTP comme trafic normal ou anormal. Ces expériences produisent des résultats pour des ensembles de techniques d'apprentissage automatique qui se chevauchent et différents ensembles de fonctionnalités, ce qui nous permet de comparer la qualité des différents ensembles de fonctionnalités pour les différentes techniques d'apprentissage automatique, au moins sur cet ensemble de données.

Mots-clés : HTTP, CSIC 2010, application Web, intrusions Web, apprentissage automatique

Analyzing http requests for web intrusion detection

Othmane LAGRINI

ABSTRACT

Several security issues aligned with web application are mostly relative to intrusions. These latter are due to the trend and fast development of Web Applications (Web-App). To reduce the risks associated with Web-App problems, the developers must take into consideration numerous measures by elaborating highly secured applications in order to avoid the known attacks. In case such measures fail to serve their purpose, it is imperative to detect and identify the source of these attacks to minimize the estimated risks. The Intrusion detection is one of most effective techniques that enables developers to identify and prevent system damage. Most of defensive techniques of Web intrusion detection are not capable of recognizing and managing the complexity of Web-App attacks. Nevertheless, the principle of Machine Learning (ML) approach allows efficiently to detect the known and the unknown attacks. In this thesis, we are studying and discussing ML techniques to classify HTTP requests in the dataset known as CSIC 2010 HTTP of which consists of normal or abnormal traffic. These experiences generate results for several ML techniques that overlaps and different set of functionalities, which allow to compare the quality of different sets of functionalities between different ML techniques, at least, under the dataset mentioned previously.

Keywords: HTTP, web application, CSIC 2010, intrusion, ML

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CART	Classification And Regression Tree
CFS	Correlation Feature Selection
CR	Carriage Return
CSIC	Consejo Superior de Investigaciones Científicas
CSV	Comma-Separated Values
CTF	Capture The Flag
DT	Decision Tree
ECML/PKDD	European Conference on Machine Learning and Principles/Practice of Knowledge Discovery in Databases
GeFS	Generic Feature Selection
FN	False Negative
FP	False Positive
FPR	False Positive Rate
JWT	JSON Web Token
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System IP Internet Protocol
LDAP	Lightweight Directory Access Protocol
ML	Machine Learning
mRMR	Minimal Redundancy Maximal Relevance
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
ROC	Receiver Operating Characteristic

SIEM	Security Information and Event Management
SOM	Self-Organizing Maps
SQL	Structured Query Language
SSI	Server-Side Include
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
WAF	Web Application Firewall

INTRODUCTION

Au cours des dernières années, il y a eu une augmentation significative de l'utilisation des réseaux de communication et des systèmes de traitement de données. L'émergence de nouveaux paradigmes et technologies, tels que Big data, l'internet des objets (IoT) et le cloud computing, ainsi que l'augmentation naturelle de la dépendance des personnes, des gouvernements et des entreprises vis-à-vis des systèmes informatiques favorisent de nouvelles menaces et tirent parti des impacts et des probabilités des failles de sécurité de l'information dans ce contexte nouveau et complexe. On constate une augmentation de la quantité de données transférées et traitées, de la diversité des nouveaux protocoles, de l'utilisation de données cryptées et du nombre d'activités malveillantes sur le réseau mondial. Ces questions posent de nouveaux défis pour les systèmes de détection d'intrusion, qui doivent s'adapter au nouveau scénario.

HyperText Transfer Protocol « http » est un protocole sans état qui utilise un modèle basé sur les messages. En gros, un client envoie un message de demande et le serveur renvoie un message de réponse. RFC 2616 définit de nombreux entêtes pour le message de demande et de réponse. Quand on attaque une application Web, le payload est envoyé via un message request. Il existe différente possibilité pour le faire ; utiliser des méthodes HTTP dangereuses, modifier les paramètres de la requête ou envoyer un autre trafic malveillant.

Dans ces systèmes, on distingue généralement deux étapes : le prétraitement et le traitement. Le prétraitement comprend les tâches effectuées avant le traitement formel des données, telles que la création de jeux de données, l'extraction de caractéristiques et la sélection de caractéristiques.

Ce travail est organisé comme suit : dans le chapitre 1, nous allons expliquer plus en détail les vulnérabilités Web. Ensuite, au chapitre 2, nous présentons plus en détail comment nous avons utilisé les caractéristiques des messages HTTP dans nos processus d'extraction de caractéristiques. Le chapitre 3 contient le processus de préparation des données. Ensuite, dans le chapitre 4, nous décrivons les tests réalisés avec notre mise en œuvre et nous présentons les résultats que nous avons obtenus. Nous terminons le présent travail avec nos conclusions au

chapitre 5, en ajoutant également quelques idées qui peuvent être le point de départ de travaux futurs.

CHAPITRE 1

INTRODUCTION AUX IDS ET VULNERABILITES WEB

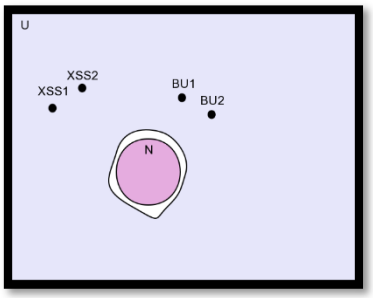
Il est important et difficile de sécuriser les systèmes informatiques. L'importance vient à la fois des pertes potentielles de vie privée ou d'argent d'un système compromis ainsi que de la responsabilité civile et pénale pour défaut de sécurisation des systèmes informatiques spécifiés dans des lois telles que Sarbanes- Oxley et la Health Insurance Portability and Accountability Act de 1996. La difficulté de sécuriser les systèmes existants ressort clairement de la liste toujours croissante des vulnérabilités de sécurité telles que les Common Vulnerabilities et Exposures (CVE) ou les bogues discutés sur des forums tels que Bugtraq. Comme une mesure du taux de découverte de problèmes de sécurité graves, depuis plusieurs mois Mitre a ajouté 500-700 nouvelles vulnérabilités candidats chaque mois à CVE.

Idéalement, les systèmes seraient conçus, construits et gérés d'une manière qui empêche les vulnérabilités de sécurité d'exister. Malheureusement, l'histoire des dernières années montre que les systèmes de production ne répondent pas à cette norme. Les clients, les serveurs et les systèmes embarqués ont tous des vulnérabilités en matière de sécurité.

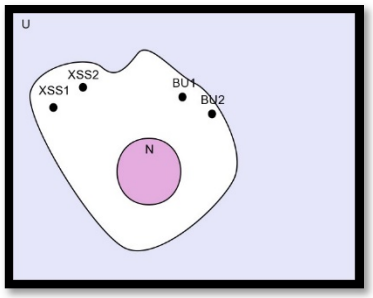
Étant donné que la sécurité est une exigence et que nous n'avons pas encore développé et déployé des systèmes sans vulnérabilités graves en matière de sécurité, nous devons prendre des mesures supplémentaires pour protéger le système contre la prochaine attaque. Une approche pour protéger les systèmes informatiques consiste à concevoir des défenses spécifiques pour chaque problème observé, soit sous forme de correctifs de code ou de signatures d'attaque. Les deux stratégies, cependant, exigent un humain pour analyser chaque problème et développer une solution. Cela limite le temps de réponse possible à un délai d’heures ou jours, mais les attaques par des programmes auto-répliquant peuvent se propager en quelques secondes. Par conséquent, des mécanismes automatisés permettant de détecter les menaces et d'y répondre en temps réel sont nécessaires. Des systèmes de détection des anomalies ont été proposés pour répondre à cette exigence parce qu'ils peuvent potentiellement détecter de nouvelles attaques sans intervention humaine.

1.1 Systèmes de détection d'anomalies et généralisation

Un système de détection d'anomalies développe un modèle de comportement normal à partir d'observations empiriques (l'ensemble d'entrainement) ; comportement normal implique que les attaquants n'utilisent pas le système pour effectuer des tâches en dehors de l'ensemble de celles que les administrateurs ont l'intention d'effectuer. Les observations secondaires qui s'écartent du modèle sont des anomalies marquées. Un système de détection d'anomalies développant un modèle de comportement normal est une forme d'apprentissage automatique.
Si un système d'apprentissage doit faire plus que simplement mémoriser les données de formation, il doit généraliser, c'est-à-dire générer un ensemble qui représente un exemple. Lorsqu'un système de détection d'anomalies se généralise, il accepte des entrées semblables, mais pas nécessairement identiques, à des instances de l'ensemble de données de formation, c'est-à-dire que l'ensemble d'instances considérés comme normales (l'ensemble normal) est plus grand que l'ensemble d'instances des données de formation. Pour la plupart des systèmes de détection d'anomalies (par exemple ceux utilisés avec les serveurs Web), l'ensemble des entrées légales possibles est infini et l'ensemble complet des comportements normaux n'est pas connu et pourrait changer avec le temps. Dans ce cas, le système de détection des anomalies doit utiliser un ensemble incomplet de données de formation. Pour ces systèmes, lorsque le système accepte plus de cas que les données sur la formation, la généralisation est une exigence.

(a) 

(b)



(c)

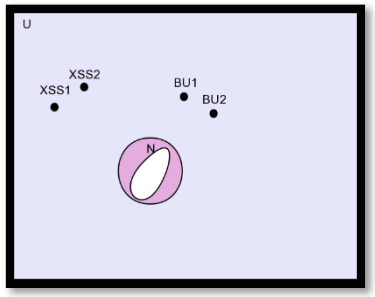


Figure 1.1 Diagrammes illustrant (a) la sous-généralisation souhaitée, (b) la sous-généralisation. Les points BU1, BU2, XSS1 et XSS2 représentent des attaques, N représente l'ensemble des requêtes normales, et la ligne entoure l'ensemble accepté par le système de détection d'anomalies. U est l'ensemble de toutes les entrées possibles du système (c) illustre la sous-généralisation.

Le but d'un système de détection des anomalies est un modèle qui décrit avec précision le comportement du mal, tel qu'illustré à la figure 1.1 (a). Si l'algorithme généralise trop (sur généralise), alors l'ensemble normal est trop grand. Dans ce cas, les attaques "proches" des données de formation pourraient être identifiées comme normales (un faux négatif), limitant l'utilité du système. La figure 1.1 (b) illustre un système de détection des anomalies sur généralisant.

D'autre part, un système qui mémorise simplement les données de formation aura besoin d'un stockage efficace pour l'ensemble complet de la normale. Évidemment, cela est impossible lorsque l'ensemble normal est inconnu ou infini. Un tel système sous-généraliserait, et signalerait par erreur les évènements normaux comme anormaux (faux positifs). Un système

sous-généralisant manque d'instances normales qui sont de légères variantes des données de formation. La figure 1.1 c) illustre la sous-généralisation.

La facilité d'identifier le bon niveau de généralisation dépend du flux de données et de la représentation utilisée pour celui-ci. La plupart des flux de données ne sont pas une collection de bits aléatoires ; il s'agit plutôt de lieux où des valeurs importantes pour la signification des données sont placées, c'est-à-dire qu'elles contiennent une structure. Un algorithme précis de détection d'anomalies aura une représentation correspondant au flux de données de manière à faciliter la détection d'anomalies. Par exemple, considérez un timbre d'heure et de date. Dans le flux de données, il peut être donné en nombre de secondes (ou minutes, heures, jours, etc.) à partir d'un temps de départ donné ; par exemple, la plupart des systèmes Unix gardent le temps comme le nombre de secondes depuis minuit le 1er janvier 1970. Le timbre horaire pourrait plutôt être présenté dans un format plus compréhensible pour l'homme, tel HH:MM:SS MM/DD/AA. Cependant, l'ordre de ces champs est différent selon les cultures, les mois et les jours peuvent être par le nom ou le nombre, le temps peut être indiqué comme B.C. ou A.D. (sans année 0, donc l'arithmétique simple sur les années échoue sans traitement spécial). Par exemple, dans les données envoyées aux serveurs web, trois formats de timbres horaires sont spécifiés comme acceptables par la norme, et encore plus sont utilisés régulièrement. À l'intérieur du système de détection d'intrusion (IDS), les données peuvent être représentées comme une chaine de caractères ou un flux de jetons (p. ex., heure, minute, deuxième, jour, mois, année). Les jetons peuvent contenir ou non la valeur associée. L'utilisation de jetons permet à l'algorithme d'apprendre la structure, ce qui rapproche le modèle de la compréhension de la signification de ce qu'il modélise.

En considérant la généralisation pour une date, si elle ne peut prendre que des valeurs d'un ensemble fini suffisamment petit pour être maintenu en mémoire, il suffit simplement de mémoriser la chaine, et aucune généralisation n'est nécessaire. Lorsqu'une date est légale, le nombre de valeurs juridiques est potentiellement infini (bien qu'en pratique il soit limité). Dans ce cas, les jetons seraient un meilleur choix, et il se pourrait que les valeurs associées aux jetons ne soient pas importantes pour identifier un comportement normal. La représentation concerne également les anomalies attendues. Si un format de date européen est considéré comme une

anomalie (par exemple, pour un système américain), la représentation doit être capable d'exprimer cette différence.

Cet exemple de date illustre les extrêmes entre la variabilité faible (un petit ensemble fini) et la variabilité élevée (toute date légale) d'un flux de données. Le problème est pire pour les données effectivement aléatoires (comme les hashs cryptographiques ou les données cryptées). Lorsqu'un système de détection d'anomalies sous-généralisant rencontre des données à grande variabilité, il doit tenter de mémoriser les données. Dans le pire des cas de données aléatoires (p. ex. hashes cryptographiques), cette approche est condamnée à utiliser une mémoire excessive et ne fournit toujours pas un système précis. Plus les régions avec une grande variabilité, les combinaisons doivent être reprécisées dans le modèle de comportement normal du détecteur d'anomalies. Le résultat sera un modèle trop grand pour être pratique, et/ou un modèle qui produit trop de faux positifs. Si, d'autre part, l'algorithme de détection d'anomalie généralise suffisamment pour éviter le faux problème positif, les parties moins variables des données vont probablement souffrir d'une surgénérisation et le système sera sujet à des attaques manquantes. Les régions à forte variabilité nécessitent plus de généralisation que les régions à faible variabilité. Le problème, alors, est comment identifier la quantité correcte de généralisation, et où l'appliquer.

Afin de trouver la tache douce, des généralisations supplémentaires seront nécessaires pour un système sous-généralisant alors que celui qui surgénéralise aura besoin de réductions dans la généralisation. Si les données ont des régions de variabilité différentes, ces modifications doivent être localisées là où le système sur- ou sous-généralise. De plus, pour cibler ces emplacements, la représentation des données doit être suffisamment fine pour faire la distinction entre les parties des données nécessitant plus et celles nécessitant moins de généralisation.

En résumé, une généralisation correcte est une condition nécessaire pour une détection précise des anomalies ; pour qu'un système de détection des anomalies ait une précision suffisante pour être déployé, il ne doit ni être préventif ni surgénéralisé. Si la généralisation n'est pas correctement contrôlée, alors la détection d'anomalie ne sera pas précise. Le modèle utilisé par

le système de détection des anomalies doit représenter les données d'une manière permettant de discriminer la normale des cas de données anormales.

1.2 Les serveurs Web sont un bon essai pour la recherche de détection d'anomalies

Ma recherche utilise des serveurs Web pour étudier la détection d'anomalies. Ils constituent un bon environnement de test pour plusieurs raisons. Premièrement, les serveurs Web sont essentiels pour de nombreuses organisations. Ce sont des voies vitales pour la communication, le commerce et les services. Reflétant leur grande valeur, les attaques contre les serveurs web sont devenues banales et couteuses. Par exemple, le Code que le ver Rouge a couté 2,6 milliards de dollars en juillet et l'août de 2001 et Nimda1 est prévenu des ouvriers d'hôpital dans Västra Götaland, la Suède d'accéder à réservations et à dossiers médicaux informatiques le 23 septembre 2001. Bien qu'aucune vie n'ait été perdue dans cette attaque, elle a montré que les couts pouvaient être plus que monétaires. Robertson et coll. indiquent que les entrées CVE de 1999 à 2005 montrent que les vulnérabilités sur le Web représentent plus de 25 % du nombre total de failles de sécurité, et ils notent que ce nombre est un minimum puisqu'il ne compte pas les vulnérabilités dans les applications Web personnalisées. Pour tenter de se défendre contre des attaques constantes, les administrateurs du système déploient des défenses telles que le réseau Snort IDS. Snort utilise des modèles d'attaques, appelées signatures, pour identifier les attaques précédemment vues et analysées. Chaque signature représente une attaque ; les signatures dans les ensembles récents montrent la fraction des attaques visant les serveurs web et leurs applications : 27 % de la série de signatures Sourcefire Vulnerability Research Team du 25 octobre 2005 et 50 % de la série de signatures communautaires du 2 novembre 2005.
Une des raisons du grand nombre de signatures d'attaques d'applications Web est la variété d'applications Web, allant de simples scripts d'interface perl common-gateway (CGI) à des sites de commerce électronique complexes pilotés par des bases de données. Étant donné que les outils de création d'appareils Web sont faciles à utiliser, bon nombre des personnes qui les écrivent et les déploient ont peu de compétences en matière de sécurité (p. ex., l'édition, la conception graphique ou les informaticiens non expérimentés en programmation sécurisée). Par conséquent, ces applications web sont souvent précaires. La prévalence des applications

web personnalisées et leurs problèmes de sécurité fréquents amènent à exiger que des solutions telles que IDS soient appliquées en tant que couche supplémentaire de protection.

Ajout au problème est que HTTP est devenu le protocole de transport universel. Auparavant, pour chaque nouveau service, un développeur de logiciel développait un protocole de communication et utilisait un port attribué pour les communications. En raison de problèmes de sécurité associés à de nombreux nouveaux protocoles, les administrateurs réseau ont bloqué l'accès avec des pare-feu. HTTP traverse la plupart des pares-feux, souvent avec peu d'interférences. Cela a conduit les développeurs à utiliser HTTP comme protocole de transport pour leur nouveaux logiciels. Les exemples incluent SOAP3 (appels de procédure à distance utilisant le langage de balisage extensible (XML) pour les paramètres), les connexions SSH (tunneling Secure Shell), QuickTime multimédia d'Apple, Microsoft RPC pour accéder à Exchange (email), un protocole de négociation des paiements et un accès distant au système de fichiers. Ces nouveaux services via HTTP présentent des opportunités supplémentaires pour les attaquants.

Les approches IDS actuelles pour les serveurs Web sont insuffisantes. Un IDS de serveur Web doit :

1. être préci : identifiez la plupart des attaques, idéalement toutes, et identifiez rarement le trafic non attaqué. Il doit maintenir cette précision à mesure que le site Web évolue au fil du temps.
2. Détecter de nouvelles attaques.
3. Ne pas imposer une charge excessive à l'administrateur ou à la machine qu'il protège.

Comme le montrent le chapitre 2 et les résultats expérimentaux de ce mémoire, les approches IDS actuelles pour les serveurs Web ne répondent pas à ces critères. Une des raisons possibles est que, comme le montre la section 2.4, HTTP est un protocole difficile pour la détection des anomalies.

1.3 Intrusion détection systèmes test

D'autres chercheurs ont proposé de détecter les anomalies pour HTTP IDS. Malheureusement, les comparaisons d'IDS utilisant les mêmes données expérimentales sont rares. Dans certains cas, le ou les chercheurs ont utilisé un ensemble de données non disponible pour d'autres chercheurs (un ensemble de données fermé) et ont montré que l'IDS peut détecter des attaques. Dans d'autres cas, le ou les chercheurs ont utilisé des ensembles de données accessibles à tous les chercheurs (ensembles de données ouvertes), comme les données des laboratoires Lincoln du MIT. Les documents décrivant ces IDS ne font état que des résultats sur l'ensemble de données, et les comparaisons sont généralement laissées au lecteur pour effectuer. Par conséquent, comment pouvons-nous vraiment savoir à quel point les IDS sont exactes ?
Une base théorique pour la détection des intrusions rendrait les essais moins importants, nous n'avons pas de telle théorie. Par conséquent, j'ai utilisé une approche expérimentale. Dans le cadre de la recherche présentée dans ce mémoire, j'ai utilisé un cadre pour tester troi algorithmes afin d'évaluer leur performance dans des circonstances réalistes. Ce cadre, les algorithmes IDS et l'analyseur HTTP sont open-source pour encourager d'autres chercheurs à développer mes tests.

1.4 Les vulnérabilités des protocoles HTTP

1.4.1 Injection

Des failles d'injection, telles que l'injection SQL, OS et LDAP, se produisent lorsque des données non approuvées sont envoyées à un interpréteur dans le cadre d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent inciter l'interpréteur à exécuter des commandes involontaires ou à accéder aux données sans autorisation appropriée.

1.4.2 Authentification et gestion de session vulnérables

Les informations d'identification du compte et les jetons de session ne sont souvent pas correctement protégés. Les attaquants compromettent les mots de passe, les clés ou les jetons d'authentification pour assumer l'identité des autres utilisateurs.

1.4.3 Exposition de données sensibles

Cette vulnérabilité permet à un attaquant d'accéder à des données sensibles telles que des cartes de crédit, des identifiants fiscaux, des identifiants d'authentification, etc. pour commettre une fraude sur une carte de crédit, un vol d'identité ou tout autre crime. La perte de telles données peut avoir un impact important sur l'entreprise et nuire à sa réputation. Les données sensibles méritent une protection supplémentaire, telles que le cryptage à l'arrêt ou en transit, ainsi que des précautions spéciales en cas d'échange avec le navigateur.

1.4.4 Entités externes XML (XXE)

Entité externe XML (XXE) fait référence à un type spécifique de Contrefaçon de demande côté serveur (SSRF) attaque, par laquelle un attaquant peut provoquer un déni de service et accéder à des fichiers et services locaux ou distants en abusant d'une fonctionnalité largement disponible et rarement utilisée dans les analyseurs syntaxiques XML.
XML est un format de données très utilisé que l'on trouve dans tout, des services Web (XML-RPC, SOAP, REST, etc.) aux documents (XML, HTML, DOCX) et aux fichiers d'image (SVG, données EXIF, etc.) qui utilisent XML. Naturellement, là où il y a XML, il y a un analyseur XML – tenez compte de cette pensée, nous y reviendrons bientôt

1.4.5 Cross site Scripting (XSS)

Des failles XSS se produisent chaque fois qu'une application prend des données non fiables et les envoie à un navigateur Web sans validation ou échappement approprié. XSS permet aux attaquants d'exécuter des scripts dans le navigateur de la victime qui peuvent détourner les sessions des utilisateurs, dégrader des sites Web ou rediriger l'utilisateur vers des sites malveillants.

1.4.6 Référence d'objet directe non sécurisée

Une référence d'objet directe se produit lorsqu'un développeur expose une référence à un objet d'implémentation interne, tel qu'un fichier, un répertoire ou une clé de base de données. Sans vérification de contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées.

1.4.7 Contrôle d'accès cassé

Le contrôle d'accès applique une stratégie empêchant les utilisateurs d'agir en dehors des autorisations prévues. Les défaillances conduisent généralement à la divulgation non autorisée d'informations, à la modification ou à la destruction de toutes les données, ou à l'exécution d'une fonction commerciale en dehors des limites de l'utilisateur. Les vulnérabilités courantes du contrôle d'accès incluent :

- Contournement des contrôles de contrôle d'accès en modifiant l'URL, l'état de l'application interne ou la page HTML, ou simplement en utilisant un outil d'attaque d'API personnalisé permettant de changer la clé primaire en un autre enregistrement d'utilisateur, permettant d'afficher ou de modifier le compte de quelqu'un d'autre.
- Élévation de privilège-Agir en tant qu'utilisateur sans être connecté ou en tant qu'administrateur lorsqu'il est connecté en tant qu'utilisateur.
- Manipulation de métadonnées, telle que la relecture ou l'altération d'un jeton de contrôle d'accès JSON Web Token (JWT) ou d'un cookie ou d'un champ masqué manipulé pour élever des privilèges ou abuser de l'invalidation de JWT.
- Une mauvaise configuration de CORS autorise un accès non autorisé à l'API.
- Forcer l'accès aux pages authentifiées en tant qu'utilisateur non authentifié ou aux pages privilégiées en tant qu'utilisateur standard. Accéder à l'API avec des contrôles d'accès manquants pour POST, PUT et DELETE.

1.4.8 Mauvaises configurations de sécurité

Une mauvaise configuration survient lorsque les paramètres de sécurité sont définis, mis en œuvre et conservés par défaut. Une bonne sécurité nécessite une configuration sécurisée définie et déployée pour l'application, le serveur Web, le serveur de base de données et la plateforme. Il est également important que le logiciel soit à jour.

Mauvaise configuration du serveur ou de l'application Web entrainant diverses failles :

- Débogage activé.
- Autorisations de dossier incorrectes.
- Utilisation de comptes ou de mots de passe par défaut.
- Pages de configuration / configuration activées.

Toutes les données pourraient être volées ou modifiées lentement au fil du temps.

Les architectures de sécurité des applications actuelles ne suivent pas la sécurité par défaut. Au contraire, les programmeurs doivent appliquer des mesures de sécurité pour éviter l'accès à des ressources privées ou confidentielles.

1.4.9 Désérialisation non sécurisée

La plupart des risques critiques pour la sécurité des applications Web sont **Désérialisation non sécurisée**. Cette vulnérabilité se produit lorsque des données non fiables sont utilisées pour abuser de la logique d'une application ou d'une interface de programme d'application (API).

Par exemple, un attaquant peut rechercher un objet ou une structure de données dans l'intention de le manipuler à des fins malveillantes. OWASP a répertorié les types d'attaques principaux en tant qu'attaques par déni de service (DoS), contournements de l'authentification et attaques d'exécution de code / commande à distance, dans le cadre desquels les attaquants manipulent du code arbitraire lors de sa désérialisation.

Pour bien comprendre la désérialisation non sécurisée, nous devons comprendre ce que sont en premier lieu la sérialisation et la désérialisation. Ce blog expliquera ce qu'il en est en détail, ce que signifie une désérialisation non sécurisée, son impact sur les applications et les

meilleures pratiques pour l'empêcher. Nous aborderons ensuite quelques solutions pour prévenir la désérialisation non sécurisée.

1.4.10 Utilisation de composants avec des vulnérabilités connues

De nos jours, même les sites Web simples tels que les blogs personnels ont beaucoup de dépendances.

Nous pouvons tous convenir que le fait de ne pas mettre à jour tous les logiciels du backend et du front-end d'un site Web entraînera sans aucun doute de lourdes menaces pour la sécurité, le plus tôt possible.

Par exemple, le rapport de sites Web piraté pour 2017 a une section dédiée autour des CMS obsolètes. Ce rapport montre qu'au moment de l'infection :

- 3% des sites Web WordPress étaient obsolètes ;
- 8% de Joomla les sites Web étaient obsolètes;
- 3% des sites Web Drupal étaient obsolètes ;
- 3% des sites Web Magento étaient obsolètes.

La question est de savoir pourquoi nous ne mettons pas à jour notre logiciel à temps. Pourquoi est-ce encore un si gros problème aujourd'hui ?

Il existe certaines possibilités, telles que :

- Les webmasters / développeurs ne peuvent pas suivre le rythme des mises à jour ; après tout, mettre à jour correctement prend du temps.
- Un code hérité ne sera pas travaillé sur les nouvelles versions de ses dépendances.

Cela peut paraitre un peu trop dramatique, mais chaque fois que vous ignorez un avertissement de mise à jour, vous permettez peut-être à une vulnérabilité désormais connue de survivre dans votre système. Les cybercriminels sont prompts à enquêter sur les logiciels et à mettre à jour les changes logs.

Quelle que soit la raison pour laquelle vous exécutez un logiciel obsolète sur votre application Web, vous ne pouvez pas le laisser sans protection. Sucuri et OWASP recommandent des patches virtuels dans les cas où l'application de correctifs n'est pas possible.

Les correctifs virtuels permettent de protéger les sites Web obsolètes (ou ceux qui présentent des vulnérabilités connues) des attaques en empêchant l'exploitation à la volée de ces vulnérabilités. Cela se fait généralement par un pare-feu et un système de détection d'intrusion.

1.4.11 Insuffisance de journalisation et de surveillance

L'enregistrement et la surveillance vont de pair. Il est peu utile de disposer de journaux adéquats s'ils ne font pas l'objet d'une surveillance adéquate.

Le problème de la journalisation et des surveillances insuffisantes concerne l'ensemble de l'infrastructure informatique et pas seulement l'application Web avec accès à Internet, comme le fait la solution. Pour cette raison, nous ne limiterons pas cette discussion à la journalisation et à la surveillance des applications Web.

L'un des principaux problèmes est qu'il y a tellement de journaux – presque tous les systèmes contemporains génèrent leurs journaux. La gestion des journaux devient alors un problème majeur. Au moment où tous les différents journaux sont rassemblés et de préférence assemblés, la taille même de l'ensemble de données devient trop volumineuse pour permettre une surveillance manuelle efficace.

La solution réside dans une automatisation accrue du processus. Par exemple, certains systèmes de contrôle d'accès peuvent avoir leurs propres règles de surveillance. Les règles de connexion peuvent être définies pour autoriser un nombre prédéfini de tentatives de connexion par session. Le système enregistre les tentatives, puis bloque les accès à partir de cette adresse IP, pour une période prédéfinie ou indéfiniment. De tels systèmes alerteront probablement aussi l'équipe de sécurité que quelque chose ne va pas.

1.5 Système de détection d'intrusion (IDS)

Un système de détection d'intrusion (IDS) est un système qui surveille le trafic réseau pour détecter toute activité suspecte et alerte lorsqu'une telle activité est découverte. Bien que la détection et le rapport d'anomalies soient les principales fonctions, certains systèmes de

détection d'intrusion sont capables de prendre des mesures lorsqu'une activité malveillante ou un trafic anormal est détecté, y compris le blocage du trafic envoyé à partir d'adresses IP (Internet Protocol) suspectes.
Les catégories de système les plus connues sont les détections par signature, les détections par anomalie et les détections hybride

1.6 Apprentissage automatique ou Machine Learning

L'apprentissage automatique est un sous-domaine de l'informatique qui a évolué à partir de l'étude de la reconnaissance de formes et de la théorie de l'apprentissage informatique en intelligence artificielle. En 1959, Arthur Samuel définissait l'apprentissage automatique comme un « domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés ». L'apprentissage automatique explore l'étude et la construction d'algorithmes qui peuvent apprendre et faire des prédictions sur les données. Ces algorithmes fonctionnent en construisant un modèle à partir d'un exemple d'ensemble d'apprentissage d'observations d'entrée afin de faire des prédictions ou des décisions basées sur des données exprimées en tant que sorties, plutôt que de suivre des instructions de programme strictement statiques.
Les sections suivantes répertorient et décrivent les différentes techniques d'apprentissage automatique, qui sont utilisées pour les problèmes de classification, et seront comparées dans cet article.

1.6.1 Random Forest (RDF)

Breiman (2001) a proposé des forêts aléatoires, qui ajoutent une couche supplémentaire de caractère aléatoire à l'ensachage. En plus de construire chaque arbre à l'aide d'un échantillon bootstrap différent des données, les forêts aléatoires modifient la façon dont les arbres de classification ou de régression sont construits. Dans l’arborescence standard, chaque nœud est divisé en utilisant la meilleure répartition parmi toutes les variables. Dans une forêt aléatoire, chaque nœud est divisé en utilisant le meilleur parmi un sous-ensemble de prédicteurs choisis au hasard à ce nœud. Cette stratégie quelque peu contre-intuitive s'avère très performante par

rapport à de nombreux autres classificateurs, y compris l'analyse discriminante, les machines vectorielles de support et les réseaux neuronaux, et est robuste contre le surajustement. De plus, il est très convivial dans le sens où il n'a que deux paramètres (le nombre de variables dans le sous-ensemble aléatoire à chaque nœud et le nombre d'arbres dans la forêt), et n'est généralement pas très sensible à leurs valeurs.

1.6.2 Régression Logistiques (LRg)

La régression logistique a une longue tradition avec des applications très variées telles que la modélisation des données dose-réponse et des données de choix d'achat. Malheureusement, de nombreux cours d'introduction aux statistiques ne couvrent pas cette méthode assez simple. De nombreux textes de statistiques catégorielles le couvrent, en plus des textes sur la régression logistique. Certains analystes utilisent la méthode avec une fonction de distribution différente, la normale. Dans ce cas, on parle d'analysepar probit. Certains analystes utilisent l'analyse discriminante au lieu de la régression logistique, car ils préfèrent considérer les variables continues comme des Y et les catégories comme des X et travailler à rebours. Cependant, l'analyse discriminante suppose que les données continues sont des réponses aléatoires normalement distribuées, plutôt que des répresseurs fixes.

La régression logistique nominale estime la probabilité de choisir l'un des niveaux de réponse en tant que fonction lisse du facteur x. Les probabilités ajustées doivent être comprises entre 0 et 1 et la somme de 1 pour tous les niveaux de réponse pour une valeur de facteur donnée.

Dans un diagramme de probabilité logistique, l'axe des y représente la probabilité. Pour k niveaux de réponse, k - 1 courbes lisses répartissent la probabilité totale (qui est égale à 1) parmi les niveaux de réponse. Le principe d'ajustement pour une régression logistique minimise la somme des logarithmes naturels négatifs des probabilités ajustées aux évènements de réponse qui se produisent (c'est-à-dire le maximum de vraisemblance).

Lorsqu'Y est ordinal, une version modifiée de la régression logistique est utilisée pour l'ajustement. La probabilité cumulative d'être égal ou inférieur à chaque niveau de réponse est

modélisée par une courbe. Les courbes sont les mêmes pour chaque niveau mais elles sont décalées vers la droite ou la gauche.
Le modèle logistique ordinal correspond à une intersection différente, mais avec la même pente, pour chacune des r − 1 comparaisons logistiques cumulatives, où r est le nombre de niveaux de réponse. Chaque estimation de paramètre peut être examinée et testée individuellement, bien que cela soit rarement d'un grand intérêt.
Le modèle ordinal est préféré au modèle nominal lorsqu'il est approprié car il a moins de paramètres à estimer. En fait, il est pratique d'ajuster les réponses ordinales avec des centaines de niveaux de réponse.

1.6.3 Arbres de décisions (DTr)

Un arbre de décision est un classificateur exprimé sous la forme d'une partition récursive de l'espace d'instance. L'arbre de décision se compose de nœuds qui forment un arbre enraciné, ce qui signifie qu'il s'agit d'un arbre dirigé avec un nœud appelé « racine » qui n'a pas d'arêtes entrantes. Tous les autres nœuds ont exactement un bord entrant. Un nœud avec des arêtes sortantes est appelé un nœud interne ou de test. Tous les autres nœuds sont appelés feuilles (également appelés nœuds terminaux ou de décision). Dans un arbre de décision, chaque nœud interne divise l'espace d'instance en deux ou plusieurs sous-espaces selon une certaine fonction discrète des valeurs d'attributs d'entrée. Dans le cas le plus simple et le plus fréquent, chaque test considère un seul attribut, de sorte que l'espace d'instance est partitionné en fonction de la valeur de l'attribut. Dans le cas des attributs numériques, la condition fait référence à une plage. Chaque feuille est affectée à une classe représentant la valeur cible la plus appropriée. En variante, la feuille peut contenir un vecteur de probabilité indiquant la probabilité que l'attribut cible ait une certaine valeur. Les instances sont classées en les naviguant de la racine de l'arbre à une feuille, en fonction du résultat des tests le long du chemin. Les nœuds internes sont représentés par des cercles, tandis que les feuilles sont désignées par des triangles. Notez que cet arbre de décision intègre à la fois des attributs nominaux et numériques. Compte tenu de ce classificateur, l'analyste peut prédire la réponse d'un client potentiel (en la triant dans l'arborescence) et comprendre les caractéristiques comportementales de l'ensemble de la

population de clients potentiels concernant le publipostage direct. Chaque nœud est étiqueté avec l'attribut qu'il teste, et ses branches sont étiquetées avec ses valeurs correspondantes.

1.7 Résumé et aperçu du mémoire

Les serveurs Web sont importants à protéger parce qu'ils sont omniprésents et critiques pour de nombreuses organisations, mais ils sont actuellement mal défendus. Les serveurs Web sont un bon banc d'essai pour la recherche de détection d'anomalies en raison de l'impact qu'une bonne solution peut avoir sur les systèmes de production. La plupart des défenses déployées sont basées sur la signature, et ces systèmes exigent des humains pour analyser les attaques et ils ne peuvent pas détecter de nouvelles attaques.

Les techniques de détection des anomalies peuvent détecter de nouvelles attaques, et sont donc une solution prometteuse, et quelques chercheurs ont essayé. Les systèmes de détection d'anomalies utilisent la généralisation chaque fois que le comportement normal est infini, inconnu ou non stationnaire. La généralisation doit être soigneusement contrôlée pour une précision acceptable.

À ce jour, les tests IDS ont été faibles, mais des tests rigoureux sont nécessaires pour savoir si le système est précis et dans quelles circonstances il fonctionne bien. Les tests devraient inclure des attaques inoffensives, car elles font normalement partie du trafic Internet.

Dans ce mémoire on essaye aussi d’analyser le contenu d’un paquet (payload) afin d’améliorer la détection, réduire les faux positif et détecter de nouvelles attaques.

CHAPITRE 2

DÉTECTION D'ATTAQUES WEB PAR APPRENTISSAGE AUTOMATIQUE

Dans le cadre de la protection des serveurs Web, un administrateur système a besoin de savoir quand son système est attaqué et a été (ou risque d'être) compromis : il s'agit de la détection d'intrusion. Le chapitre 1 a introduit ce problème ; nous considérons ici l'état actuel de la technique. Ce chapitre passe en revue la littérature décrivant les résultats de recherches antérieures, en commençant par les systèmes de détection d'intrusion dans la section 2.1. Le chapitre 1 a fait valoir qu'un IDS protégeant les serveurs Web doit détecter de nouvelles attaques sans intervention humaine, et que les systèmes de détection d'anomalies sont une solution qui répond à cette exigence. Tous les systèmes de détection d'anomalies partagent un trait commun d'apprentissage d'un modèle de comportement normal. Au fil des ans, les chercheurs ont essayé de nombreux algorithmes différents pour apprendre ce modèle. Certains de ces algorithmes sont prometteurs pour http ; d'autres ont des limites qui les empêchent de travailler dans ce domaine.

Chaque fois que l'ensemble de données d'apprentissage représentant le comportement normal du système de détection d'anomalies est incomplet ou que l'ensemble réel est infini, le système de détection d'anomalies doit effectuer une généralisation. Bien que les chercheurs aient réalisé l'utilité de la généralisation, la mesure dans laquelle ils l'ont étudiée est limitée. La section 2.2 passe en revue les travaux antérieurs dans ce domaine.

La grande variété d'algorithmes de détection d'anomalies est un symptôme du fait que nous manquons d'une théorie complète de la détection d'intrusion et de détection d'anomalies pour fournir des conseils.

Sur des approches plus prometteuses, la section 2.3 montre la plupart des théories qui ont été développées dans la littérature sans être exhaustive. Elle montre que des tests rigoureux et une comparaison des algorithmes sont nécessaires pour déterminer quels systèmes fonctionnent bien, quand et pourquoi. L'examination des travaux antérieurs conclut que la qualité des tests et des données de test varie et que les résultats comparables et reproductibles pour HTTP sont rares.

La section 2.3 fournit aussi des informations générales sur le protocole HTTP, la structure des requêtes HTTP qui pourraient être utiles pour un système de détection d'anomalies, les sources de diversité entre les sites Web qu'un IDS pourrait utiliser pour forcer un attaquant à personnaliser une attaque pour un site Web donné, et des exemples d'attaques contre des serveurs Web. La section se termine par des informations sur les difficultés que pose HTTP pour les IDS

2.1 Détection d'Intrusion

La détection d'intrusion informatique a commencé en 1972 avec un article d'Anderson identifiant le besoin de ce qui évoluerait dans les systèmes de détection d'intrusion actuels. Les premiers chercheurs de l'IDS se sont concentrés sur les statistiques du comportement du système et de l'utilisateur sur une machine donnée (un IDS basé sur l'hôte) pour traiter les menaces. En pratique, ces premiers systèmes présentaient des taux de faux positifs élevés.
À mesure que les réseaux branchaient des machines ensemble, les serveurs réseau devenaient communs et les attaquants pouvaient être physiquement retirés de leurs victimes. Cela a conduit à la nécessité de détecter les intrusions en réseau, et les chercheurs ont essayé de détecter et de prévenir les attaques dans de nombreuses couches du modèle ISO de réseautage. Les restrictions d'accès physique (serrures traditionnelles et autres mesures de sécurité physique) fonctionnent bien pour prévenir les attaques au niveau de la couche phyique1. Les approches comprenaient l'identification du trafic inhabituel en fonction de ses adresses IP source et destination, de son protocole (TCP ou UDP) et de ses ports. Malheureusement, rien n'est inhabituel ou particulièrement dangereux à propos d'une nouvelle machine se connectant à un serveur web, donc une autre approche est nécessaire pour protéger les serveurs web.

2.1.1 Architecture

Comme les chercheurs ont travaillé sur le problème de la détection d'intrusion, ils ont développé trois architectures IDS : détection de signature, spécification et détection d'anomalies. Quelques chercheurs ont essayé des approches hybrides, où les forces d'une architecture sont exploitées pour couvrir les faiblesses d'une autre.

Détection par signature (aussi appelée détection d'abus, ou détection par l'apparence) : Un humain étudie une attaque et identifie les caractéristiques (p. ex. comportement et/ou contenu) qui la distinguent des données normales ou du trafic. La combinaison de ces caractéristiques est connue sous le nom de signature, et elle fait partie d'une base de données de signatures d'attaque. Lorsque l'IDS rencontre des données correspondant à la signature, il déclenche une alarme. Les systèmes de signature représentent la grande majorité d'IDSs installé ; elles sont importantes, bien que limitées, comme le montrent les paragraphes suivants. Tous les produits antivirus commerciaux utilisent la détection de signature, tout comme le réseau IDS snort.

Les systèmes basés sur la signature sont généralement rapides, et ils peuvent souvent détecter avec précision les pots pour lesquels ils ont des signatures. Cependant, comme un humain doit analyser chaque attaque pour développer la signature, le temps de réponse possible à de nouvelles attaques est limité à un laps de temps d'heures ou de jours. Les attaques par des programmes d'autoréplication (virus ou vers) peuvent apparaitre et se propager en quelques secondes. Lorsque de nouvelles attaques apparaissent, les systèmes protégés par signature IDS sont vulnérables jusqu'à ce qu'un ensemble de signature mis à jour soit disponible et installé.

Tous les IDS basés sur la signature utilisent une forme de correspondance de motifs pour identifier les chaines qui pourraient être associées à une intrusion, que ce soit un virus dans un fichier ou une attaque comme Code Rouge contre un serveur web. Dans de nombreux systèmes de correspondance de motifs, un motif est simplement une chaine (idéalement) qui ne se trouve dans aucune donnée normale. Certains systèmes de correspondance de motifs comprennent d'autres informations, comme où chercher la chaine (soit par emplacement absolu, soit par emplacement par rapport au protocole réseau ou au format de fichier). Les approches d'appariement de motifs comprennent des filets pétris colorés et des langages de spécification d'attaque.

Les systèmes basés sur la signature souffrent de faux positifs, surtout si la configuration du système ou l'environnement change. Patton et coll. décrit ce phénomène, qu'ils appellent le « squelling », et montre comment les intrus peuvent nier l'avantage d'un IDS basé sur la

signature avec des faux positifs soigneusement conçus. IDS avec ce problème ne répond pas au critère 3 de la section 1.2.

Les systèmes basés sur la signature peuvent également souffrir de faux négatifs. Pour vérifier l'efficacité des deux "meilleurs" systèmes basés sur la signature (snort et ISS RealSe- cure), Vigna et al. a constaté que les mutants d'attaques connues ne seraient pas détectés par l'IDS. IDS avec ce problème ne répond pas aux critères 1 de la section 1.2.

La plupart des systèmes de signature ne font que peu, voire pas du tout, généralisation. L'une d'elles est rapportée par Ning et al. ; ils ont recueilli des signatures pour aider à détecter les attaques connexes. Un test intéressant, apparemment jamais réalisé, serait d'appliquer le travail de Vigna et al. générant des mutants d'attaques aux Ning et al. Travailler là où les signatures est généralisées.

Spécification : Un expert étudie le programme à protéger et produit une spécification qui identifie les mesures autorisées (c.-à-d. celles prévues par l'auteur du programme) qu'un programme peut prendre. Lors du fonctionnement du système, le comportement est comparé à la spécification, et des écarts sont notés et/ou évités. Le complément de cette approche est de décrire le comportement indicatif des attaques ; parfois, décrire ce qui n'est pas permis est plus facile que décrire ce qui n'est pas permis.

Un problème avec cette approche est que le niveau d'expertise nécessaire pour élaborer une spécification appropriée est considérablement plus élevé que celui nécessaire pour mettre en oeuvre un programme. En règle générale, chaque programme a besoin d'une spécification distincte, et chaque fois que le programme change, la spécification doit être examinée par un expert. Ces exigences augmentent le cout et ont donc réduit son utilisation dans les systèmes de production.

Les systèmes basés sur les spécifications sont également confrontés à des problèmes de bonne utilisation. Par exemple, Hogland et McGraw notent que les listes de contrôle d'accès, l'une des mesures de sécurité fondées sur des spécificités plus simples, sont si compliquées qu'elles échouent normalement dans la pratique.

En supposant une spécification appropriée, un tel SDI répond aux critères de la section 1.2.

La détection d'anomalie (aussi appelé la détection par le comportement) : IDSs basé sur l'Anomalie le suppose les tentatives d'intrusion sont rares et présentent des caractéristiques

différentes du comportement normal. L'IDS induit un modèle de normal à partir d'un corpus de données d'entrainement. Lorsqu'une instance qui ne correspond pas au modèle tiré des données de formation apparait, le système déclenche une alarme. Trouver la bonne représentation des données et identifier un algorithme d'apprentissage à utiliser avec les données peut être difficile pour les systèmes de détection d'anomalies.

Systèmes hybrides : Un IDS utilisant cette architecture utilise des parties de deux ou plusieurs de ces architectures pour profiter des forces d'une architecture pour couvrir la faiblesse d'une autre. À titre d'exemple, Sekar et al. Statistiques combinées pour la détection d'anomalies avec une machine d'État conçue par l'homme spécifiant la communication de réseau autorisée.

Un seul système hybride a été signalé pour HTTP. Tombini et coll. La détection combinée des abus et des anomalies pour trouver des attaques dans les requêtes HTTP enregistrées, et ils ont analysé comment résoudre les conflits entre les deux architectures pour fournir la meilleure précision. Ce système présentait plusieurs inconvénients qui entraveraient son déplacement dans un environnement de production. Premièrement, ils ont rapporté que sur 56 attaques, le détecteur d'abus pouvait détecter 30 - un peu plus de la moitié. Ils n'ont pas publié de résultats qui ont testé comment le système combiné effectuait les attaques. En outre, ils ont supprimé les données du compte d'utilisateur dynamique de leur suite de tests (données que j'ai conservées dans mon jeu de test), n'utilisant que le contenu dynamique sur le site web officiel. Ils ont également ajusté manuellement le modèle induit par leur détecteur d'anomalie. Ils ont également utilisé des méthodes manuelles pour identifier les demandes Web sécuritaires ou dangereuses, signalant qu'une personne peut effectuer cette tâche en quelques jours. Cette forte dépendance envers les humains limite l'utilité de leur système.

Bien que ce mémoire n'étudie pas l'hybride IDS, si un système de détection ou de spécification d'anomalie est l'une des architectures de l'hybride, le SDI résultant peut satisfaire à tous les critères spécifiés à la section 1.2.

Détection d'anomalies (également appelée détection par comportement) : Les IDS basées sur des anomalies supposent que les tentatives d'intrusion sont rares et qu'elles ont des caractéristiques différentes du comportement normal. L'IDS induit un modèle de normal à

partir d'un corpus de données d'entrainement. Lorsqu'une instance qui ne correspond pas au modèle tiré des données de formation apparait, le système déclenche une alarme. Trouver la bonne représentation des données et identifier un algorithme d'apprentissage à utiliser avec les données peut être difficile pour les systèmes de détection d'anomalies. Un système de détection des anomalies bien conçu peut satisfaire à tous les critères spécifiés à la section 1.2. Elles constituent l'un des principaux axes de ce mémoire et sont donc traitées plus en détail à la section suivante.

2.2 Détection d'anomalies

Parce que la détection d'anomalies a le potentiel de détecter de nouvelles attaques, de nombreux modèles et algorithmes ont été essayés. Un système de détection d'anomalies utilise un ensemble de données d'entrainement pour induire un modèle de normal, puis il signale comme anormal toutes les instances de données non incluses dans ce modèle.

Les premiers systèmes de détection des anomalies étaient fondés sur des mesures statistiques de la normale, les dossiers de vérification des hôtes et les données de réseau de niveau inférieur servant de source de données. Forrest et coll. a introduit l'idée de modéliser le comportement des programmes en enregistrant de courtes séquences d'appels système. Ce travail a suscité un intérêt important et de nombreuses propositions pour des modèles alternatifs de comportement des appels système, y compris des séquences de longueur variable et l'extraction de données fondées sur des règles. Malheureusement, le moniteur d'appel système ne semble pas être une approche prometteuse pour détecter les attaques contre les serveurs web. Un travail antérieur avec pH, un IDS basé sur Linux qui surveille les séquences d'appels système, suggère que les grands programmes de serveur tels que les serveurs Web sont difficiles à surveiller en pratique, parfois en prenant des mois pour atteindre un modèle stable de comportement normal (bien que voir pour une idée sur la façon de résoudre ce problème).

Un autre problème est que de nombreuses attaques contre les serveurs web sont causées par des erreurs de configuration ou des erreurs dans le code d'application web. Les exploits basés sur ces vulnérabilités peuvent ne pas générer des modèles d'appels système inhabituels parce qu'ils ne provoquent pas d'exécution inhabituelle de code de serveur web. Le serveur Web

interprète simplement le script vulnérable, et la trace d'appel système générée par le processus d'interprétation est similaire, que le script contienne ou non une vulnérabilité.
Parce que tous les comportements d'E/S passent généralement par l'interface d'appel système, pratiquement toutes les attaques sur les serveurs Web devraient, en principe, être détectables en observant des arguments aux appels système. Cependant, les tentatives antérieures de modéliser des arguments d'appel système ont généré des faux positifs élevés, malgré l'utilisation d'une stratégie de modélisation beaucoup plus sophistiquée que Forrest et al. à l'origine proposé. Cet écart semble être une conséquence naturelle de la grande variabilité des arguments d'appel système par rapport aux séquences d'appels système.

2.3 HTTP

Mes recherches utilisent HTTP pour étudier les algorithmes de détection d'anomalies. Les informations de la section 2.3.1 décrivent et fournissent des exemples du protocole. Le protocole contient une structure ; des exemples de structure et des illustrations des différents niveaux de variabilité de la structure illustrent cette caractéristique des données de la section 2.3.2. La section 2.3.3 présente les sources de diversité entre les sites Web ; un IDS utilisant cette diversité pourrait être en mesure de forcer un attaquant à personnaliser une attaque pour chaque site Web. De nombreuses attaques contre HTTP existent ; La section 2.3.4 en montre plusieurs et illustre la diversité entre les attaques. Plusieurs fonctionnalités de HTTP en font un protocole difficile à utiliser avec succès pour un IDS ; certaines de ces caractéristiques éliminent les algorithmes de la considération.

2.3.1 Protocole HTTP

HTTP est un protocole sans état décrit par la norme Internet RFC 2616 avec des extensions décrites par d'autres documents standards ou éditeurs de logiciels. Le protocole fonctionne en mode client-serveur ; les clients sont soit des agents utilisateurs (navigateurs), soit des robots Web (par exemple, le robot d'indexation de Google). Lorsqu'un client envoie une requête au

serveur, la nature sans état de HTTP implique que la requête ne sera pas nécessairement liée à des requêtes antérieures ou futures. Le serveur attend une requête complète, puis envoie une réponse complète, exemple :

```
GET /home/daily/20030715/thumbnails/009.jpg HTTP/1.1
Accept: */*
Referer:
http://www.home.org/home2003/daily/20030715/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5;
Windows NT 5.0; (R1 1.3))
Host: www.aya.org Connection: Keep-Alive
Cookie: PHPSESSID=8ce5ac388fc6c1f07d213df818f5a2e9
Extension: Security/Remote-Passphrase
```

G

Figure 2.1 Un exemple de requête HTTP d'un agent utilisateur, Internet Explorer version 5.5 de Microsoft. Une ligne a été interrompue pour lui permettre de tenir sur la page.

```
GET /~agar/album/fort/petroglyphs_02/page1.html HTTP/1.0
Host: www.fb.unm.edu
User-Agent: WebVac (fes@pita.stanford.edu WebVac.org)
From: fghw@pita.stanford.edu
```

Figure 2.2 Exemple de requête HTTP d'un agent de robot Web, WebVac de l'Université de Stanford.

La figure 2.1 montre un exemple de requête d'un agent utilisateur et la figure 2.2 en montre une d'un robot Web.

Le chemin de ressource demandé peut être un nom de fichier, mais l'interprétation est jusqu'au serveur web. Certains chemins sont destinés à être interprétés par des programmes, et RFC 3875 décrit l'utilisation courante de scripts d'interface de passerelle commune (CGI). Un des

principaux objectifs des scripts est de fournir une expérience plus dynamique pour l'utilisateur ; une méthode de passage des paramètres au script est donc prévue. Normalement, le chemin vers le script fait partie du chemin de ressource demandé, les paramètres suivent dans le chemin, et sont de la forme `key = valeur`. La figure 2.3 montre un exemple.

Afin de permettre à un utilisateur de personnaliser sa vue d'un site Web, plusieurs rubriques HTTP existent ; des exemples de ces lignes figurent aux figures 2.1, 2.2 et 2.3. Par exemple, la ligne d'entête `Accept-Language` permet à l'utilisateur de spécifier sa (ses) langue (s) préférée (s). Certains serveurs Web utiliseront cette information pour présenter une version linguistique du site. Ces entêtes sont généralement optionnels, et un serveur est libre de les ignorer.

```
GET/~midhun/gallery/view.php?gid=2&phid=64 HTTP/1.1 Host:
cs.unm.edu
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1;
rv:1.7.3) Gecko/20040913 Firefox/0.10
Accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-
Alive: 300
Connection: keep-alive
Referer: http://cs.unm.edu/~midhun/gallery/view.php?
gid=2&phid=63
Cookie: PHPSESSID=383ca0237f3fbeecc253a988d99d8f70
```

Figure 2.3 Un exemple de requête HTTP appelant un script CGI et passant des paramètres. Trois lignes ont été interrompues pour leur permettre de tenir sur la page.

2.3.2 Structure d’une requête HTTP

Bien qu'une requête HTTP soit une chaine, elle est également structurée. Certaines parties de la structure varient plus que d'autres. Par exemple, le procédé de la requête est le premier mot d'une requête. La grande majorité des demandes utilisent `GET`, mais d'autres méthodes, comme `HEAD` et `POST` existent, et les extensions à la norme HTTP en définissent plus. Par exemple, WEBDAV) définit `PROPFIND`. Seul un serveur web supportant cette extension ne verrait jamais cette méthode dans le trafic non-attaque ; d'autres serveurs Web génèreront une erreur de méthode invalide. Comme deuxième exemple, le chemin de ressource représente habituellement un fichier dans le système de fichiers. Comme tous les systèmes d'exploitation répandus utilisent un système de fichiers structuré par arbre, un arbre est susceptible d'être une bonne représentation pour cette partie de la requête.

D'autre part, certains domaines de la demande varient davantage. Par exemple, l'entête `Referer` fournit à l'URI de la page Web un lien vers la page demandée. Lorsqu'un utilisateur recherche une page, les paramètres de recherche sont inclus dans l'URI du moteur de recherche dans le champ `Referer`. La diversité des référentiels des moteurs de recherche n'est donc limitée que par les différentes commandes des différents termes de recherche utilisables pour trouver la page sur les différents moteurs de recherche. En général, les valeurs de `Referer` ne sont limitées que par les pages sur le web avec un lien vers la page demandée ; rappelez-vous que les administrateurs Web à distance contrôlent leur contenu, donc en fait la diversité est limitée à l'imagination de tous les développeurs de contenu Web qui pourraient vouloir lier à la page demandée. À titre d'autre exemple, une partie du protocole HTTP a été conçue pour permettre à un proxy Web d'interroger le statut d'une ressource avant de la demander. Pour illustrer cela, la figure 2.4 contient un exemple de demande qui a été transmis par procuration. Notez la date sur la ligne `If-Modified-Since` et l'en-tête `If-None-Match` contenant une valeur de hachage. Ces hashes sont conçus pour éviter les collisions, et devraient donc être uniques.

```
GET /~girar/.../new1mexico/BuddhtSprings_3.jpg
HTTP/1.1 Accept: */*
-------: ----:----------------------------------------
-----
----------------------
---------------------
Accept-Language: en-us
---------------: ----- -------
If-Modified-Since: Sun, 23 Mar 2003 07:37:28 GMT If-
None-Match: "3faffe-a52f-3e7d6438"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host:www.fb.unm.eu
Connection: Keep-Alive
```

Figure 2.4 Un exemple de requête HTTP qui passait par un proxy en route du client vers le serveur. Le proxy a remplacé certaines des données par -s. Deux lignes ont été brisées pour lui permettre de s'adapter à la page, et... dans le chemin a remplacé deux noms de répertoire pour la même raison.

2.3.3 Sources potentielles de diversité entre les sites Web

Détecteur formé pour un site ne fonctionne pas bien pour les autres. Si l'IDS peut distinguer les deux sites Web, un attaquant pourrait être forcé de concevoir une attaque distincte pour chaque site Web, ralentissant le taux d'attaque à un plus gérable par les humains.

Les premiers exemples de diversité sont les chemins de ressources choisis par le ou les developer (s) du site Web. Ils reflètent la diversité des stratégies organisationnelles des développeurs pour un site Web. Une requête pour le fichier /foo/bar/blah.jpg n'a aucun sens sur un site Web qui n'a même pas de répertoire /foo.

HTTP étant apatride, il existe diverses méthodes de fourniture d'un (pseudo) état. L'une de ces méthodes est l'entête Cookie. Un cookie est une valeur que le serveur donne au client. Lorsque

le client demande une nouvelle ressource, elle inclut les valeurs des cookies pertinents. Entre autres utilisations, les cookies peuvent personnaliser un site Web 6. Si le cookie utilise correctement la cryptographie, il peut être utilisé pour valider un état revendiqué et pour s'assurer que l'utilisateur suit un chemin prescrit à travers un site Web (p. ex., le processus de caisse sur un site Web de commerce électronique). Les valeurs des cookies varient avec :

- Le logiciel exécutant les applications sur le site web. Par exemple, Java et PHP ont tous deux des fonctions de bibliothèque pour la gestion de session, mais ils utilisent des cookies différents pour cette tâche.
- Différentes fonctions de bibliothèque peuvent définir différents cookies (pour suivre différentes parties de l'état de l'utilisateur).
- Le développeur du site Web peut choisir les noms de cookies pour les parties personnalisées du site Web.

Certains cookies contiennent des hashs (p. ex., le `PHPSESSID` de la figure 2.3).

Alors que quelques navigateurs web dominent le marché, la liste des clients web est vaste. Wikipedia répertorie 138 navigateurs différents, sans compter les micro-entreprises utilisées sur les téléphones et autres appareils informatiques mobiles. Ajoutez à cela la diversité des robots web (au moins un par moteur de recherche ainsi que ceux qui soutiennent la recherche ou le balayage des vulnérabilités), puis multipliez par le nombre de versions de chacun d'entre eux et la liste des valeurs potentielles pour l'entête User-Agent devient substantielle. Cependant, de nombreux sites Web répondent à un ensemble ciblé de personnes, et ces personnes et les robots Web qui visitent le site pourraient donc représenter un petit sous-ensemble de cette diversité potentielle.

Au-delà de la diversité des `User-agent`, les utilisateurs dans différentes parties du monde ont des langues différentes préférées. Par exemple, un site Web en chinois est plus susceptible de recevoir des demandes de personnes dont la langue préférée est le chinois qu'un serveur Web hébergeant un site Web uniquement en swahili.

Une autre source de diversité potentielle entre les sites Web est le type de fichier préféré de l'utilisateur. Par exemple, les utilisateurs visitant un site Web fournissant des informations pour les personnes aveugles auront probablement leur navigateur configuré pour demander des versions audios lorsque le serveur les a disponibles.

2.3.4 Attaques HTTP

Les attaques ont une grande diversité dans l'emplacement de l'attaque et l'apparence, et diffèrent des demandes normales. À ce stade, la plupart des attaques se sont concentrées sur le chemin des ressources. Quelques-uns ont ciblé d'autres parties de la requête, et il serait insensé de suggérer que le code qui traite d'autres parties de la requête HTTP est invulnérable. La diversité des attaques contre HTTP est élevée, représentant la diversité des bugs que les programmeurs introduisent dans le code. Cette diversité implique que la connaissance d'une attaque ne fournit aucune indication sur la structure de la prochaine attaque. Cette section illustre une partie de cette diversité en présentant des exemples d'attaques.

Une attaque célèbre dans le chemin des ressources est Nimda. Cette attaque visait une collection de bugs dans les systèmes Microsoft, et utilisait le fait que la configuration par défaut permettait à l'attaquant d'exploiter la vulnérabilité. La figure 2.5 montre un exemple (sur 16 dans ma base de données d'attaque). Une autre attaque dans le chemin des ressources est l'attaque beck, illustrée à la figure 2.6. Cette attaque est contenue dans les données du MIT Lincoln Labs.

```
GET /scripts/root.exe?/c+dir HTTP/1.0
Host: www
Connnection: close
```

Figure 2.5 Example d’attaque Nimda

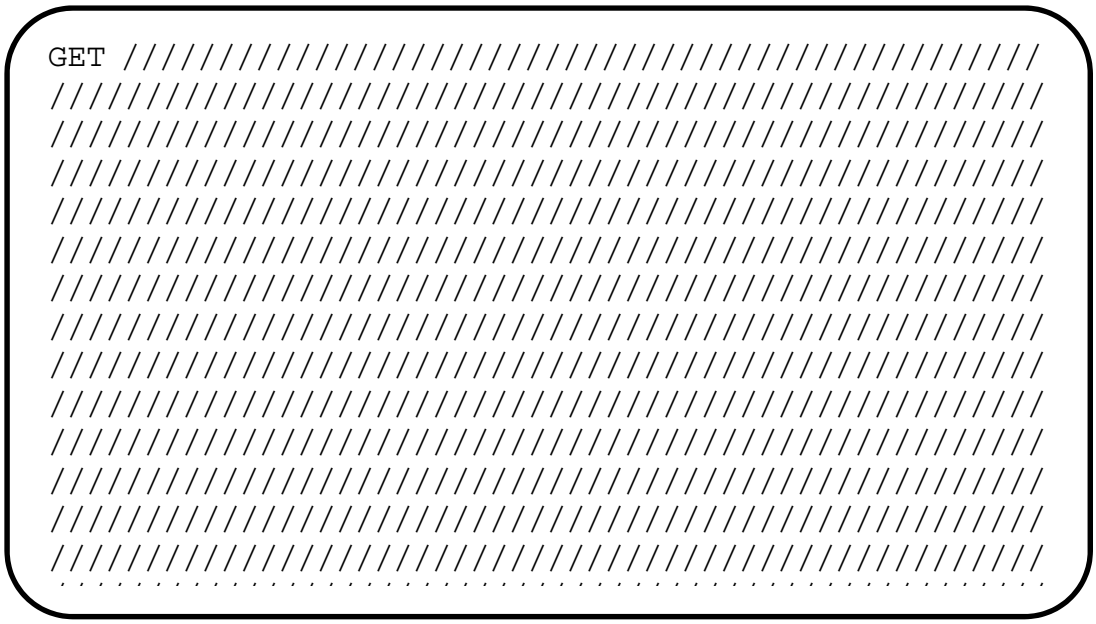


Figure 2.6 L'attaque de Beck. L'original avait 790 / s; les extra / s ont été supprimés pour économiser de l'espace.

Un autre exemple intéressant est illustré par une vulnérabilité dans Apache qui a permis la divulgation accidentelle d'informations ; un exemple est donné à la figure 2.7. Lorsque vous incluez le ? M = D dans la requête, l'attaquant contournerait la page d'index fournie par le développeur Web, en listant tous les fichiers, y compris ceux destinés à être plus privés. Search en- gines comme Google a utilisé ce bug pour indexer des pages qui n'avaient pas de liens externes accessibles à eux.

Parmi les attaques qui ciblent une partie de la requête non dans le chemin des ressources, la figure 2.8 montre Apache Sioux. Cette attaque répète une ligne d'entête non standard pour provoquer un dénial de service. Un autre exemple est l'erreur de transfert coupé Apache (figure 2.9). Cette attaque a exploité le fait qu'Apache avait une erreur lorsque le client a demandé des transferts d'encodage découpés. Qu'un serveur Web utilise ou non ce type de transfert dépend des données qu'il a servies et des clients qu'il a servis.

Un type d'attaque qui augmente à mesure que le nombre d'applications web personnalisées croît est contre les scripts CGI. Un exemple d'une de ces attaques est illustré à la figure 2.10 - l'attaque Traversal du répertoire du serveur d'index NT. Cet exemple a été généré en tapant l'URL dans un navigateur, et semble donc similaire aux requêtes normales de ce même type de navigateur.

2.3.5 Les difficultés de la détection des anomalies au niveau de HTTP

Les solutions IDS existantes pour les serveurs Web sont faibles (au mieux). L'une des raisons est que HTTP est un protocole difficile à gérer pour un IDS. Les défis incluent la nature sans état de HTTP, la nature non stationnaire des serveurs Web, le fait que les requêtes HTTP sont de longueur variable, les données d'entraînement pour la détection des anomalies sont déséquilibrées et les données d'entraînement avec les attaques supprimées sont difficiles à obtenir. De plus, comme les attaques inoffensives font partie du trafic Internet normal, un IDS devrait être capable de les tolérer sans produire d'alarmes, tout en détectant de nouvelles attaques. Ces problèmes éliminent de nombreux algorithmes qui ont réussi ailleurs.

Puisque HTTP est sans état, l'IDS ne peut pas s'appuyer sur des relations entre les requêtes, basées sur une séquence ou autre ; en effet, la plupart des attaques existantes ne durent qu'une seule requête. Lors d'attaques contre d'autres services ou contre d'autres couches du modèle standard de l'Organisation internationale de normalisation (ISO) pour la mise en réseau, un attaquant peut vérifier si une vulnérabilité est susceptible d'exister avant de tenter de l'exploiter. Mes ensembles de données montrent que les sondages de vulnérabilités des serveurs Web précèdent rarement les attaques réelles. Avec l'utilisation de botnets pour les attaques, l'attaquant et l'utilisateur légitime peuvent utiliser le même ordinateur. Ainsi, une méthode de détection d'anomalies de serveur web doit être capable d'analyser des requêtes de transaction isolées et de déterminer si elles représentent ou non une attaque sur le serveur.

```
GET / HTTP/1.1
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
User-Agent: simon
```

Figure 2.7 L'attaque Apache Sioux. L'original avait 1000 User-Agent : lignes; les extra ont été supprimés dans la version présentée ici. Cette attaque a été extraite des données du MIT Lincoln Labs.

```
GET /x.html HTTP/1.1
Host: 192.168.x.x Transfer-Encoding:
chunked

80000000
Rapid 7
0
```

Figure 2.8 L'une des variantes d'attaque par erreur de transfert par blocs Apache

```
GET /iissamples/issamples/oop/qfullhit.htw?CiWebHitsFile=
/../../winnt/system32/logfiles/w3svc1/ex000121.log&
CiRestriction=none&CiHiliteType=Full HTTP/1.1
Host: www.i-pi.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.4.1) Gecko/20031114
Accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,
image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-
Alive: 300
Connection: keep-alive
```

Figure 2.9 Un exemple NT Index Server Directory Traversal attaque. Trois lignes ont été interrompues pour lui permettre de tenir sur la page

Au-delà de la nature sans état de HTTP, le contenu du site Web change rapidement. Un site e-commerce ajoute de nouveaux produits et supprime ceux qui sont obsolètes. Le site Web d'un département universitaire change à mesure que les semestres passent, les cours vont et viennent, de nouveaux horaires sont publiés, etc. Les blogs sont mis à jour aussi souvent que le blogueur fournit de nouveaux essais, souvent quotidiennement. Les changements de contenu impliquent des changements dans les requêtes HTTP du client au serveur, entraînant des

changements dans le comportement du serveur Web. De plus, de nouvelles versions de navigateurs Web existants et de nouveaux navigateurs Web et robots sont fréquemment introduits. Pour réussir, tout IDS doit être capable de faire face à cet environnement (non stationnaire) en constante évolution.

Les défis décrits ci-dessus limitent les algorithmes qui peuvent être utilisés. L'IDS doit pouvoir s'adapter au fur et à mesure que le contenu du serveur Web ou les logiciels évoluent, et cette adaptation ne doit pas nécessiter de recyclage chronophage. Un serveur Web doit être étroitement surveillé pendant la période de formation pour s'assurer qu'aucune attaque réussie ne se produit, sinon le système de détection d'anomalies apprend un comportement nuisible comme d'habitude. Ce coût humain plus élevé affecte davantage les systèmes qui nécessitent un recyclage périodique que les systèmes qui peuvent s'adapter à mesure que le site Web change.

De nombreux algorithmes (y compris certains qui ont été utilisés pour la détection d'anomalies ailleurs) nécessitent des quantités similaires de données normales (sans attaque avec des attaques inoffensives, ou du moins sans attaque) et anormales (attaque). Une collection de requêtes HTTP normales peut être obtenue en interceptant les requêtes reçues par le serveur. Un ensemble de données de taille similaire pour les attaques n'existe pas.

```
GET /error/HTTP_NOT_FOUND.html.var
HTTP/1.1 Host: www.i-pi.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.6) Gecko/20040114
Accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,
image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-
Alive: 300
Connection: keep-alive
```

Figure 2.10 Un exemple d'attaque exploitant le bogue de divulgation du chemin Web du fichier .var Apache Win32. Deux lignes ont été interrompues pour leur permettre de tenir sur la page. Cette attaque a été générée à l'aide du navigateur Web Firefox, elle est donc similaire à d'autres requêtes générées par le navigateur

2.3.6 Généralisation et http

La quantité correcte de généralisation est nécessaire pour qu'un système de détection d'anomalies soit précis. Comme exemple de sur-généralisation avec HTTP, considérez une mesure utilisant la longueur d'une requête HTTP. Des attaques telles que celle illustrée à la figure 2.9 peuvent facilement se situer dans la normale; celui-ci est de 432 caractères, ce qui le place dans la moyenne ± σ pour les données d'entraînement filtrées cs.unm.edu 2004-11-12. Pour cet ensemble de données, la longueur moyenne est de 343 caractères et le sigma est de 126, ce qui conduit à une plage de déviation standard de 217 à 469.

Comme exemple de sous-généralisation, considérons un simple détecteur d'anomalie qui mémorise ce qu'il voit et marque tout ce qui ne figure pas dans cet ensemble comme anormal. Cet algorithme serait sous-généralisé chaque fois que les données d'apprentissage sont un sous-ensemble approprié des données normales, par exemple lorsque l'ensemble des normales est inconnu ou infini. À titre d'exemple concret, cet algorithme serait sous-généralisé sur les requêtes HTTP, car certains proxys interrogent pour voir si une page a été modifiée avant de la demander (pour un exemple, voir la figure 2.4). Ces requêtes contiennent des dates et au fil du temps, les dates changeront. Par conséquent, ce simple détecteur d'anomalie marquerait à tort les demandes normales comme anormales en raison d'une sous-généralisation.

Un problème particulier pour les systèmes de détection d'anomalies est le mélange de données à variabilité élevée et faible. Si l'algorithme traite chaque champ de manière uniforme, il risque de sous-généraliser les données à haute variabilité, de sur-généraliser les données à faible variabilité, ou les deux. Le problème avec de nombreux systèmes réels, tels que HTTP, est que différentes parties des données ont différents niveaux de variabilité. Par exemple, les touches de ligne d'en-tête telles que `Host` et ses valeurs changent peu - la clé introduit les noms d'hôte qui sont utilisés pour décrire le serveur Web, vraisemblablement un petit ensemble fini. De même, `Connection` : a deux valeurs normales `keep-alive` et `close`. Pour ces parties de la requête HTTP, la mémorisation des valeurs régulièrement vues est susceptible d'être précise sans consommer une grande quantité de mémoire.

D'autre part, une partie de la requête HTTP est plus variable. Les champs tels que `If-None-Match` : incluent les hachages, `If-Modified-Since` : contient des dates, et les cookies contenant des chaînes d'identification de session utilisées par des langages tels que PHP, Java et ASP.NET sont des hachages cryptographiques.

Pour un système de détection d'anomalies fonctionnant avec HTTP, le modèle doit avoir la résolution pour pouvoir distinguer les différentes parties des données. L'algorithme doit être capable d'appliquer différents niveaux de généralisation à différentes parties des données.

CHAPITRE 3

LE PROCESSUS DE PREPARATION DES DONNEES

3.1 Introduction

Des ensembles de données adéquats sont d'une importance vitale pour la formation et les tests WAF. Le présent chapitre étudie les caractéristiques qu'un ensemble de données devrait présenter pour être jugé approprié à cette fin. Malheureusement, trouver des ensembles de données qui répondent à ces caractéristiques n'est pas une tâche facile. L'étude des ensembles de données les plus utilisés dans la détection d'intrusion révèle que la plupart d'entre eux présentent un certain nombre d'inconvénients à appliquer à l'évaluation de WAF. Afin de résoudre cette situation, ce mémoire fournit un nouvel ensemble de données appelé CSIC qui satisfait aux conditions établies pour la bonne évaluation de WAF. L'ensemble de données est utilisé par la communauté scientifique. Les caractéristiques de l'ensemble de données ainsi que le processus suivi pour le créer sont expliqués dans ce chapitre. De plus, les applications actuelles de l'ensemble de données dans la communauté scientifique sont présentées. Enfin, des conclusions sont présentées.

3.2 Problèmes d'acquisition de données pour la détection d'intrusions

Il est essentiel de compter sur des ensembles de données appropriés pour former et tester les WAF. La qualité de ces ensembles de données influence directement l'évaluation de ces systèmes. Cependant, dans la communauté de détection d'intrusion Web, il y a une rareté des ensembles de données standard et communs pour évaluer ces systèmes. En relation avec ce problème, plusieurs auteurs ont affirmé que « le défi le plus important auquel une évaluation est confrontée est le manque d'ensembles de données publiques appropriés pour évaluer les systèmes de détection des anomalies » [Sommer et Paxson, 2010], [Tavallaee et al., 2010]. En raison de cette rareté, de nombreux chercheurs ont choisi de créer leurs propres ensembles de données, dont beaucoup sont à usage privé. Le problème de cette situation est qu'elle empêche la comparaison entre différents systèmes. En outre, de nombreux ensembles de données

existants présentent une série d'inconvénients qui rendent leur utilisation difficile dans les systèmes de détection Web. Cela fait remarquer que la nécessité de compter sur des ensembles de données étiquetés et adéquats pour former, tester et comparer les WAF n'est pas couverte dans la détection d'intrusion Web. Combler cette lacune est la motivation de ce chapitre.

3.2.1 Exigences relatives aux ensembles de données adéquats

Afin de configurer et d'évaluer correctement les systèmes de détection d'intrusion Web, il est nécessaire que l'ensemble de données utilisé réponde à une série d'exigences :

- Il est pratique qu'il soit accessible au public. Cela permet à d'autres chercheurs de l'utiliser et de comparer leurs systèmes.
- Il doit contenir du trafic HTTP, car c'est le type de trafic analysé par les WAF.
- L'ensemble de données doit être étiqueté. Sinon, il n'est pas possible d'évaluer les performances des WAF.
- Il doit contenir au moins deux classes : normale et attaque. Des anomalies pourraient également être incluses. Le trafic anormal est expliqué plus en détail dans la Sec. 3.4.
- L'ensemble de données doit contenir une variété d'attaques modernes et leurs variations. Selon Symantec, les attaques Web ont considérablement évolué ces dernières années [Wood, 2014]. Inclure les attaques modernes permet de vérifier si le système est capable de détecter les attaques Web actuelles.
- De plus, il est souhaitable que le trafic contienne des valeurs réalistes et qu'il ne soit pas anonymisé, dans le but de ne pas perdre de réalisme.

3.2.2 Évaluation des ensembles de données existants

Malheureusement, les difficultés associées à l'obtention d'ensembles de données adéquats pour évaluer les WAF sont multiples. Comme mentionné précédemment, il existe une rareté des ensembles de données étiquetés et appropriés pour évaluer les systèmes de détection d'intrusion Web. En outre, de nombreux ensembles de données existants sont confrontés à un certain nombre de problèmes. Plusieurs ensembles de données existants ont été analysés pour vérifier s'ils remplissent les conditions présentées dans la section précédente pour considérer un

ensemble de données comme adéquat. À cet égard, un certain nombre de problèmes ont été constatés :

- **Dataset sans Label**. Un problème courant est que les demandes n'ont pas d'étiquette indiquant la classe à laquelle elles appartiennent. L'ensemble de données est simplement du trafic capturé. Cependant, cela ne suffit pas pour évaluer les WAF. Des étiquettes sont nécessaires pour mesurer la capacité du système à classer les instances.
- **De nombreux ensembles de données ne contiennent pas de trafic HTTP**. Par exemple, l'ensemble de données LBNL contient du trafic réseau. Cet ensemble de données contient des traces avec des informations de réseau d'en-tête complètes, mais sans charge utile [LBNL et ICSI, 2005]. La payload contient les informations appartenant à la couche application.
- **De nombreux ensembles de données ne sont pas accessibles au public ou peuvent être difficiles à obtenir**. Lorsque les ensembles de données sont privés, ils ne sont pas utilisables par la communauté scientifique. Cela ne rend pas possible la comparaison des systèmes entre eux. Dans certains cas, la raison de l'obscurité est la confidentialité des données, ce qui empêche le partage du trafic.

  Dans d'autres cas, les ensembles de données sont partiellement disponibles, c'est-à-dire qu'ils ne sont accessibles qu'aux chercheurs sélectionnés. C'est le cas de l'ensemble de données d'évaluation de la détection d'intrusion ISCX de l'UNB [Shiravi et al., 2012]. Il est basé sur le concept de profils, qui contiennent une représentation abstraite des événements et des comportements observés sur le réseau. Il existe deux types de profils : le premier décrit un scénario d'attaque et le second extrait des distributions mathématiques ou des comportements d'applications, de protocoles ou d'entités réseau de bas niveau. Les agents sont utilisés pour générer du trafic HTTP, entre autres protocoles, à partir des profils créés. Cet ensemble de données n'est disponible que pour les chercheurs sélectionnés et il est nécessaire d'en faire la demande. Le jeu de données prend plusieurs semaines pour être obtenu et dans notre cas, il n'a pas encore été possible d'y accéder.

L'ensemble de données CELV / PKDD, qui a été généré pour le défi de la Conférence européenne sur l'apprentissage automatique et les principes / pratiques de la découverte des connaissances dans les bases de données (ECML / PKDD) en 2007, est également partiellement disponible [Raïssi et al., 2007]. Dans ce cas, il a été possible de l'obtenir. Cet ensemble de données est étiqueté et contient exclusivement du trafic HTTP.

- **L'ensemble de données n'est pas mis à jour**. Étant donné que de nouvelles attaques Web apparaissent constamment, il est important que l'ensemble de données soit à jour. Cela implique de contenir les attaques modernes afin de tester adéquatement l'efficacité des WAF dans les environnements contemporains.

  L'ensemble de données DARPA [Lippmann et al., 2000a], [Lippmann et al., 2000b] a été présenté en 1998 et 1999 par le MIT. Il contient le trafic réseau, y compris le trafic HTTP. Cet ensemble de données est l'un des plus utilisés pour évaluer les systèmes de détection d'intrusion. Cependant, le jeu de données DARPA a été critiqué par la communauté des systèmes de détection d'intrusion [McHugh, 2000], [Brown et al., 2009]. L'une des raisons est qu'il est obsolète et qu'il n'inclut pas la plupart des attaques modernes, ce qui le rend insuffisant pour évaluer les WAF actuels. En fait, certains chercheurs ([Estévez-Tapiador et al., 2004], [Hosseinkhani et al., 2011]) ont utilisé cet ensemble de données en conjonction avec d'autres, ou avec leurs propres attaques créées, pour surmonter cet inconvénient.
- **Le trafic est anonymisé**. Les problèmes de confidentialité sont souvent une cause d'anonymisation des données. Le processus de prétraitement qui implique l'anonymisation de l'ensemble de données peut conduire à une perte de réalisme et peut également affecter négativement la qualité des résultats de détection. L'ensemble de données ECML / PKDD mentionné précédemment est un exemple de trafic anonyme
  À l'exception de la partie attaque, toutes les parties de ses requêtes sont anonymisées. Par conséquent, il n'y a pas deux demandes adressées à la même application Web. Cette caractéristique le rend inutilisable pour un certain nombre de systèmes, par exemple [Hosseinkhani et al., 2011].

Ce problème affecte également l'ensemble de données LBNL. Le trafic de cet ensemble de données a été anonymisé pour supprimer les informations susceptibles d'identifier une adresse IP individuelle (protocole Internet).

- **Quantité non équilibrée de trafic normal et d'attaques**. Ce problème est notable, par exemple, dans le trafic accumulé lors des compétitions de guerre. Les jeux de données créés dans DEFCON Capture the Flag (CTF) [The Shmoo Group, 2011] en sont un exemple. Puisqu'il a été généré pendant la compétition, ce trafic est principalement constitué de trafic intrusif [Kruegel et al., 2005a]. Dans des scénarios différents des environnements conflictuels, cette caractéristique peut rendre le trafic irréaliste, car la proportion d'attaques peut en résulter disproportionnée par rapport à la quantité de trafic normal. En relation avec ce sujet et en essayant de surmonter certains des inconvénients actuels, Sangaster et al. ont étudié comment générer des ensembles de données utiles en collectant le trafic des compétitions de guerre [Sangster et al., 2009].

3.3 Caractéristiques de l'ensemble de données de la CSIC

L'ensemble de données de la CSIC a été en 2010. En tant que contribution de ce travail, il a été conçu dans le but de surmonter les inconvénients décrits des ensembles de données existants. Un jeu de données public, utilisable par l'ensemble de la communauté scientifique, permet de comparer différents systèmes de détection.

Au total, l'ensemble de données du CSIC contient 36000 demandes normales et plus de 25000 demandes anormales. Les requêtes sont étiquetées comme normales ou anormales. En ce qui concerne la génération d'attaques, des attaques statiques et dynamiques ont été générées, y compris des attaques Web modernes telles que l'injection SQL, Buffer Overflow, la collecte d'informations, l'injection CRLF, Cross-site Scripting, le côté serveur et la manipulation de paramètres.

L'ensemble de données de la CSIC est accessible au public à l'adresse http://www.isi.csic.es/dataset. Là, trois fichiers peuvent être trouvés : un fichier pour la

formation et deux fichiers pour les tests. Le fichier de formation contient 20 Mo de trafic normal uniquement. Concernant les jeux de données de test, l'un d'entre eux contient 20 Mo de trafic normal et l'autre 15,7 Mo de données anormales.

3.4 Processus de génération

Étant donné que lors du processus de génération de l'ensemble de données, il était nécessaire de générer des attaques, cibler les applications Web publiées sur Internet n'était pas une option réalisable. Par conséquent, un environnement ad hoc a été créé pour nos besoins. Une application web a été spécialement créée pour cet objectif. De plus, un serveur Web et un WAF pour le protéger ont été déployés dans un environnement virtualisé. Les environnements virtualisés présentent plusieurs avantages, tels que l'économie de ressources, la préservation de la confidentialité des données, la sécurité et la flexibilité [Sahoo et al., 2010], [Li et al., 2015]. Bien que l'application Web cible n'ait pas été publiée sur Internet, elle a la même structure et les mêmes fonctionnalités que les applications réelles, afin de la rendre la plus réaliste possible. Il se compose d'une application web e-commerce, développée avec JSP et fonctionnant sous Apache Tomcat.
En outre, une version de cet ensemble de données au format CSV (valeurs séparées par des virgules) a été réalisée par Scully de l'Université d'Aberystwyth [Scully, 2015]. Ce format facilite l'utilisation de l'ensemble de données avec des outils tels que Weka [Hall et al., 2009], qui fournit une large gamme d'algorithmes d'apprentissage automatique.

3.5 Modèles de Détection

Des modèles de détection sont appliqués pour caractériser le comportement des valeurs d'argument et d'en-tête. Rappelons que les modèles sont appliqués à chaque argument ou en-tête présent dans la requête et que les modèles sont appliqués individuellement à chacun d'eux, ce qui donne plus de précision dans la détection.
Ces modèles de détection sont basés sur des intervalles de normalité qui définissent une plage dans l'argument ou les valeurs d'en-tête sont considérées comme normales. Les valeurs situées en dehors des intervalles sont considérées comme anormales. Les limites des intervalles sont

apprises pendant la phase d'entraînement. Si seul le trafic normal est utilisé pendant la phase d'apprentissage, les limites des intervalles établissent des seuils entre ce qui est considéré comme normal ou anormal.

Les deux systèmes utilisent deux modèles de détection. L'un des modèles analyse la longueur de l'argument/valeur header et l'autre calcule les propriétés systèmes de ses caractères. Les modèles de détection sont mis en œuvre différemment selon l'algorithme utilisé. Ensuite, des détails sur les modèles de détection statistiques et markoviens sont donnés.

3.5.1 Modèles de Détection Statique

Comme mentionné, l'approche statistique utilise deux modèles de détection : 1) la longueur et 2) la distribution des caractères des valeurs d'argument / header. Ces modèles sont expliqués ensuite.

Modèles de longueur : La longueur est un critère utile pour détecter les attaques en raison du fait que, d'une part, les valeurs des requêtes normales ne contiennent généralement pas beaucoup d'octets et, d'autre part, de nombreuses attaques Web utilise une quantité considérable de caractères d'entrée (tels qu'injection de code, XSS et buffer overflow). Ce modèle capture la longueur de l'argument / valeur header. Les limites de l'intervalle sont établies comme la longueur minimale et la longueur maximale des valeurs d'argument / d'en-tête vues dans la formation. Dans les expériences précédentes, nous avons fixé ces limites à des valeurs différentes, comme cela sera expliqué plus loin.

Modèles de distribution de caractères : plusieurs intervalles sont définis pour ce modèle. En utilisant nos connaissances d'expert sur les attaques Web, nous avons constaté que tous les personnages n'ont pas la même importance dans les attaques Web. Les caractères spéciaux sont particulièrement pertinents pour la détection de nombreuses attaques Web. De plus, Sriraghavan et Lucchese ont prouvé que considérer les lettres, les chiffres et les caractères non alphanumériques est plus efficace que de considérer la fréquence relative des 256 caractères ASCII [Sriraghavan et Lucchese, 2008]. Par conséquent, au lieu de considérer les 256

caractères ASCII individuellement, le WAF statistique modélise les caractères des valeurs d'argument / header en trois groupes :

- Lettres.
- Chiffres.
- Caractères non alphanumériques.

Outre l'amélioration de la détection, un avantage de considérer ces trois groupes est que cela accélère les processus d'apprentissage et de vérification des modèles.

Les pourcentages de la distribution des caractères, selon ces trois groupes, ont été utilisés comme caractéristiques pour construire le modèle. Il est composé de trois intervalles :

- Pourcentage de lettres.
- Pourcentage de chiffres.
- Pourcentage de caractères non alphanumériques inclus dans un ensemble de caractères non alphanumériques autorisé pour l'argument / header correspondant. Cet ensemble est formé lors de la formation du système. Autrement dit, les caractères non alphanumériques correspondant à la valeur d'argument / header sont inclus dans l'ensemble.

Comme pour le modèle de longueur, les limites des intervalles sont fixées avec les pourcentages minimum et maximum trouvés sur les valeurs d'argument / header d'entraînement. Ensuite, un exemple de la façon dont la limite inférieure de l'intervalle de lettres est calculée. Au début, sa valeur correspond au pourcentage de lettres correspondant au premier header / argument analysé. Si le pourcentage des valeurs suivantes analysées est inférieur à la limite établie, la limite de l'intervalle est remplacée par le nouveau pourcentage.

CHAPITRE 4

MACHINE LEARNING POUR LA DETECTION D'INTRUSIONS WEB

4.1 Introduction

Les capacités d'automatisation de l'apprentissage automatique ont été appliquées à de nombreux domaines informatiques, parmi lesquels la détection d'intrusions. Quelques exemples d'IDS basés sur le ML sont présentés dans [Chen et al., 2005], [Mulay et al., 2010] et [Sangkatsanee et al., 2011b]. Le chapitre précédent proposait des WAF qui appliquent des techniques stochastiques pour effectuer la détection. Ce chapitre présente également un WAF, mais dans ce cas, le système utilise des techniques ML. Comme dans les systèmes stochastiques, le WAF basé sur ML suit l'approche des anomalies. L'objectif est à nouveau de concevoir des systèmes de détection à haute vitesse et à haute vitesse qui ont une conception simple. La conception du système comprend une étape de prétraitement et de traitement. Lors de la phase de prétraitement, l'extraction et la sélection d'entités sont appliquées. En particulier, de nouvelles méthodes d'extraction de fonctionnalités sont proposées dans ce chapitre. Ces méthodes combinent des connaissances d'experts et des n-grammes, et elles se révèlent plus efficaces que ces deux techniques séparément. Au sein de la variété d'algorithmes ML, les arbres de décision sont choisis pour l'étape de classification en raison de leur application réussie dans la détection d'intrusion. Quatre algorithmes d'arbre de décision différents sont appliqués, à savoir C4.5, CART, Random Tree et Random Forest. De plus amples détails sur le système et les expériences réalisées sont expliqués dans ce chapitre.

4.2 Architecture Générale

L'architecture du système ML présente de nombreuses similitudes avec les systèmes stochastiques présentés dans le chapitre précédent. Le système fonctionne au niveau de la couche application en analysant l'ensemble de la charge utile des requêtes HTTP. Après avoir analysé les demandes entrantes, le système sort sa décision de classification concernant la

normalité ou l'anomalie de la demande. Le système détecte une variété d'attaques Web qui impliquent une seule requête.

Le WAF basé sur le ML analyse les charges utiles au niveau des jetons et des demandes. Cela rend l'approche plus complète du point de vue de la sécurité, car davantage d'attaques peuvent être détectées. L'inclusion du niveau de requête est utile dans les cas où tous les jetons satisfont aux exigences de détection, mais pas la requête entière. Par exemple, lorsque la longueur de tous les jetons est comprise dans les limites de longueur mais que l'ensemble de la demande ne l'est pas. Cela ne signifie pas que l'analyse au niveau du jeton n'est pas nécessaire. Comme il a été soutenu précédemment, il permet de détecter de nombreuses attaques Web qui, autrement, ne seraient pas détectées.

4.3 Conception

La conception du système comprend deux étapes : le prétraitement et le traitement. Un schéma de cette conception est illustré à la Fig 4.1

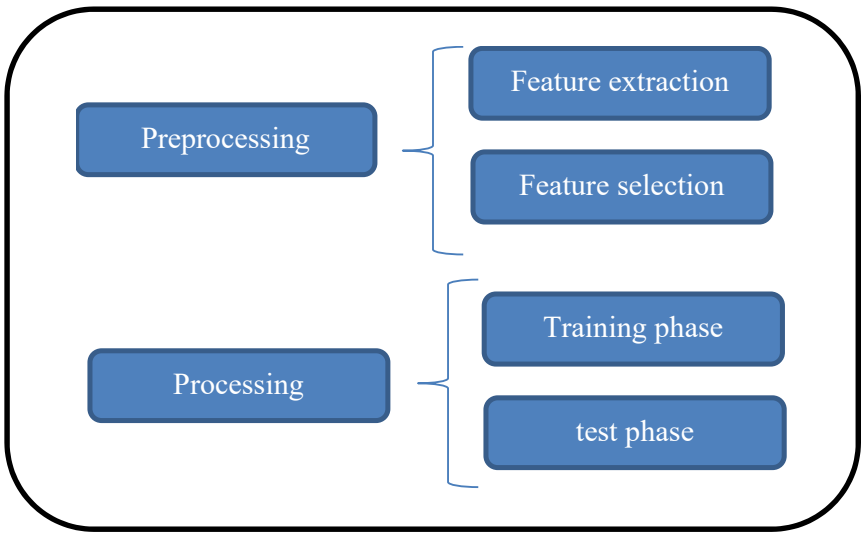


Figure 4.1 Structure de conception du WAF basé sur ML

Prétraitement. Cette étape consiste en **une extraction et une sélection de « features »**. En ce qui concerne l'extraction de caractéristiques, dans la détection d'intrusion, elle se fait généralement en appliquant des techniques manuelles ou automatiques. Ce mémoire propose de nouvelles méthodes d'extraction de fonctionnalités pour le ML en combinant les deux

techniques. Les systèmes stochastiques proposés précédemment utilisent des connaissances d'experts pour extraire des caractéristiques. De manière différente, le système ML combine des fonctionnalités de connaissances expertes avec des fonctionnalités n-gramme. L'objectif est que les méthodes proposées améliorent les résultats de détection des techniques distinctes en même temps qu'elles consomment peu de ressources.

Afin de garantir que cette combinaison consomme peu de ressources, la sélection de fonctionnalités est appliquée pour réduire le nombre de fonctionnalités non pertinentes et redondantes. Le modèle de filtre a été choisi, car il nécessite moins de ressources de calcul. En particulier, la mesure GeFS pour la détection d'intrusion est appliquée. Cette mesure a été testée avec succès avec le trafic réseau d'après [Nguyen et al., 2010], [Nguyen et al., 2011]. Cependant, il n'a pas été appliqué au trafic Web jusqu'à présent. Dans ce mémoire, son comportement dans le trafic HTTP est étudié.

Traitement. Cette étape comprend la phase de **classification**. ML comprend un ensemble d'algorithmes, tels que les réseaux bayésiens, les réseaux de neurones artificiels, les algorithmes génétiques, les machines vectorielles de support ou les arbres de décision. À partir de ces algorithmes, les arbres de décision ont été largement et avec succès appliqués à la détection d'intrusions. Pour cette raison, ils sont choisis pour le processus de classification du système ML. Ce choix est basé sur le fait que cette famille d'algorithmes est l'un des algorithmes d'apprentissage automatique les plus populaires [Wu et al., 2007]. De plus, il a été prouvé que les arbres de décision permettent d'obtenir des résultats expérimentaux réussis dans la détection d'intrusions. En fait, le gagnant du célèbre concours de détection d'intrusions DARPA [Lippmann et al., 2000a] était un algorithme basé sur des arbres de décision [Pfahringer, 2000].

Ces caractéristiques les rendent prometteurs pour être appliqués à la détection d'attaques Web. Les arbres de décision ont rarement été utilisés pour classer le trafic Web. L'un des objectifs de ce mémoire est d'étudier leurs performances dans la classification de ce type de trafic.

Selon le théorème du No-Free-Lunch, il n'y a pas de classificateurs universels pour chaque type de données [Wolpert, 1996], [Wolpert, 2001]. Puisqu'il n'y a pas d'algorithme de

classification standard pour les WAF, quatre algorithmes d'arbre de décision ont été appliqués : C4.5, arbre de classification et de régression, arbre aléatoire et forêt aléatoire. Ils ont une formation et une phase de test.

4.3.1 Extraction de caractéristiques (Features Extraction)

Ce mémoire propose de nouvelles méthodes d'extraction de caractéristiques. Ces méthodes combinent des connaissances d'experts et des fonctionnalités n-gramme. Trois méthodes d'extraction combinées sont proposées :

- **Combine-select** : Cette alternative mélange tout d'abord toutes les fonctionnalités extraites par des connaissances d'experts et des n-grammes. Comme le nombre de n-grammes conduit souvent à des problèmes de dimensionnalité, la sélection de caractéristiques est ensuite appliquée afin de réduire le nombre de caractéristiques. Dans les rares cas où la combinaison de fonctions automatiques et manuelles était appliquée dans la littérature, cette alternative a été utilisée. D'autres alternatives sont présentées ensuite.
- **Select-combine** : Cette alternative combine des fonctionnalités de connaissances expertes précédemment sélectionnées. Contrairement à l'option précédente, la sélection des fonctionnalités est effectuée en premier et les fonctionnalités résultantes sont ensuite mélangées
- **Select-n-gram-combine** : Cette alternative suit l'idée de Select-combine, cependant, elle ne prend que les valeurs sélectionnées de n-grammes. Les fonctionnalités de connaissances spécialisées ne sont pas sélectionnées car elles proviennent des connaissances d'experts en attaque Web. Par conséquent, le sous-ensemble dérivé de cette alternative est composé, d'une part, de fonctionnalités de connaissances expertes, d'autre part, de fonctionnalités sélectionnées à partir de n-grammes.

Un schéma de ces trois alternatives peut être vu sur la figure 4.2. Chacune des alternatives génère différents sous-ensembles de fonctionnalités, recevant le même nom que l'alternative correspondante. Autrement dit, les sous-ensembles correspondants sont appelés combine-select, Select-combine et select n-gram combine. Ces sous-ensembles sont appelés « cas de

combinaison ». En revanche, les connaissances d'experts et les sous-ensembles de n-grammes sont appelés « cas de base ».

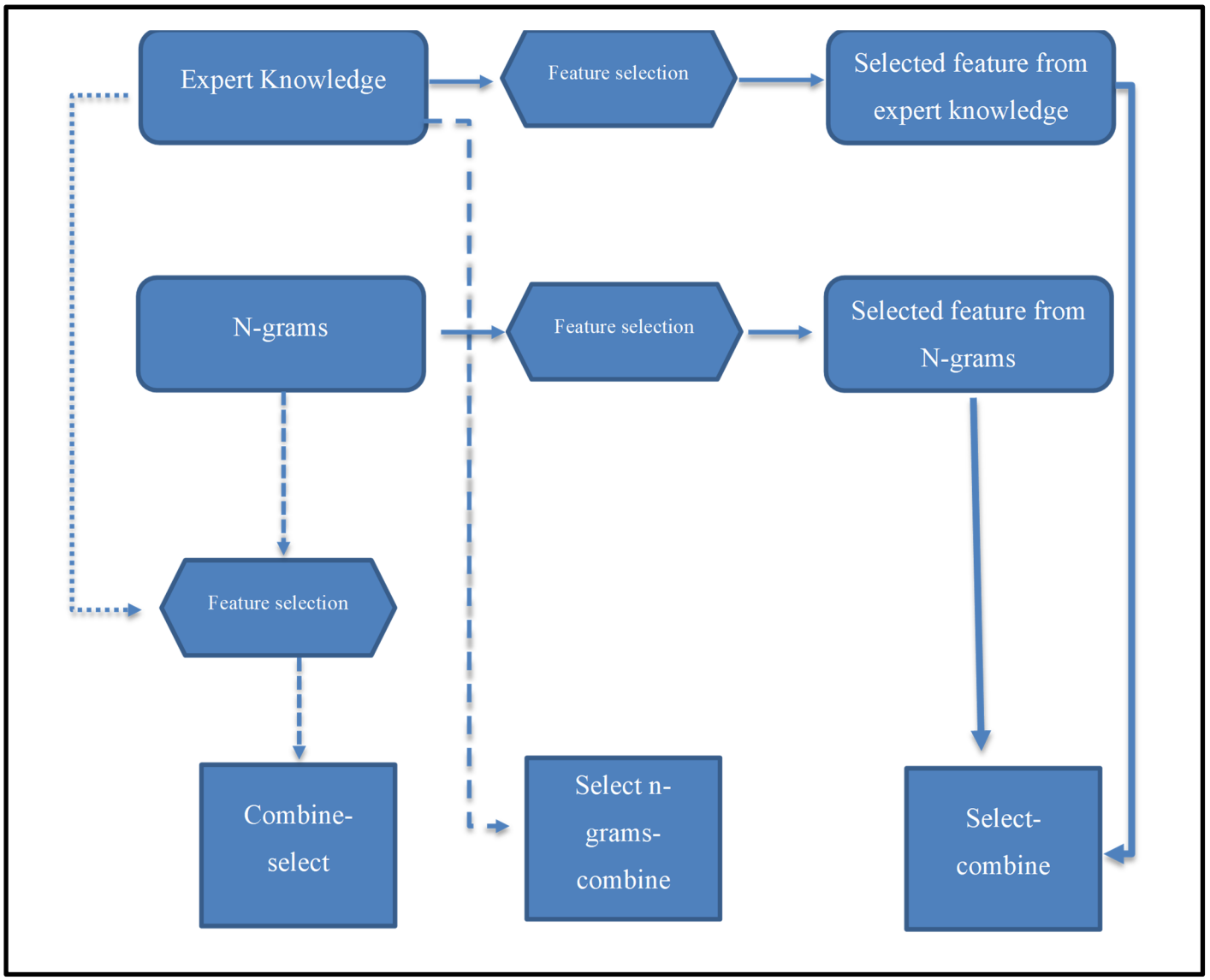


Figure 4.2 Structure des trois alternatives de combinaison proposées.

4.3.2 Sélection de caractéristiques (Feature selection)

La mesure GeFS, proposée par Nguyen et al., Est utilisée dans ce mémoire pour la sélection des features. Les principaux concepts de la mesure GeFS sont expliqués ci-après. Pour plus de détails, voir [Nguyen et al., 2010b].

Le problème de sélection des features peut être défini comme la recherche de x = (x1,..., Xn) ∈ {0, 1} n qui maximise la fonction GeF S (x):

$$max_{x\in\{0,1\}^n}\ GeFS(x) = \frac{a_0+\sum_{i=1}^{n} A_i(x)x_i}{b_0+\sum_{i=1}^{n} B_i(x)x_i} \tag{4.1}$$

où les valeurs binaires de la variable xi indiquent l'apparition (xi = 1) ou l'absence (xi = 0) le features fi; a0, b0 sont des constantes; Ai (x), Bi (x) sont des fonctions linéaires de variables x1,. . . , xn et n sont le nombre d'entités.

Il existe plusieurs mesures de sélection de fonctionnalités qui peuvent être représentées par ce formulaire, telles que les mesures de sélection de fonctionnalité de corrélation et de redondance minimale de pertinence maximale.

Mesure de sélection de fonction de corrélation : La mesure de sélection de fonction de corrélation évalue des sous-ensembles d'entités sur la base de l'hypothèse suivante : «Les bons sous-ensembles d'entités contiennent des caractéristiques fortement corrélées avec la classification, mais non corrélées les unes aux autres» [Hall, 1999].

Considérant le mérite d'un sous-ensemble de features S avec k features comme :

$$Merits_k = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}} \tag{4.2}$$

Le critère CFS tente de maximiser le mérite du sous-ensemble de features:

$$max_{s_k}\{\frac{r_{cf_1}+r_{cf_2}+\cdots+r_{cfk}}{\sqrt{k+2(r_{f_1f_2}+\cdots+r_{fifj}+\cdots+r_{fkf_1}}}\} \tag{4.3}$$

$\overline{r_{cf}}$ représente la valeur moyenne de toutes les corrélations de classification d'entités et $\overline{r_{ff}}$ est la valeur moyenne de toutes les corrélations d'entités-entités. En utilisant les valeurs binaires de la variable xi pour indiquer l'apparition (xi = 1) ou l'absence (xi = 0) de la fonction fi dans l'ensemble de fonctionnalités globalement optimal, l'expression (4.3) peut être réécrite comme un problème d'optimisation :

$$max_{x\in\{0,1\}^n}\{\frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n} x_i + \sum_{i\neq j} 2b_{ij}x_i x_j}\} \quad (4.4)$$

Mesure de Redondance minimale de la pertinence maximale : cette méthode considère simultanément les fonctionnalités pertinentes et les fonctionnalités redondantes. Pour l'ensemble de caractéristiques donné S et la classe c, D (S, c) représente la pertinence de S pour la classe c et R (S) la redondance de toutes les caractéristiques de l'ensemble S. Ensuite, la mesure mRMR est définie comme

$$max_S\ \{\mathrm{D(S,c)} - \mathrm{R(S)}\} \quad (4.5)$$

La pertinence est définie comme :

$$D(S,c) = \frac{1}{|S|}\sum_{f_i\in S} I(f_i;c) \quad (4.5)$$

et redondance

$$R(S) = \frac{1}{|S|^2}\sum_{f_i,f_j\in S} I(f_i;f_j) \quad (4.6)$$

Où I (fa, fb) est la fonction d'information mutuelle, avec fa et fb étant des feartures ou l'étiquette de classe

Dans le critère mRMR, les valeurs binaires de la variable xi sont utilisées afin d'indiquer l'apparition (xi = 1) ou l'absence (xi = 0) de la caractéristique fi dans l'ensemble de caractéristiques globalement optimal. Les valeurs d'information mutuelle I (fi; c), I (fi; fj) sont notées respectivement par les constantes ci, aij. Supposons qu'il existe n fonctionnalités complètes, alors le problème (4.5) peut être décrit comme un problème d'optimisation :

$$max_{x\in\{0,1\}^n}\ \{\frac{\sum_{i=1}^{n} c_i x_i}{\sum_{i=1}^{n} x_i} - \frac{\sum_{i,j=1}^{n} a_{ij}x_i x_j}{(\sum_{i=1}^{n} x_i)^2}\} \quad (4.7)$$

Pour résoudre le problème de sélection de caractéristiques 4.1, il est transformé en un problème de programmation linéaire mixte 0-1, qui est ensuite résolu en utilisant l'algorithme de branche et lié.

Ensuite, la stratégie de recherche pour obtenir les caractéristiques pertinentes et le critère pour choisir l'instance appropriée de la mesure GeFS sont détaillés :

- Étape 1 : Analyser les propriétés statistiques de l'ensemble de données donné afin de choisir l'instance appropriée (CFS ou mRMR) de la mesure GeFS. Le critère pour

choisir l'instance appropriée dans chaque cas est le suivant : la mesure CFS est choisie si le jeu de données a de nombreuses caractéristiques qui sont linéairement corrélées à l'étiquette de classe et entre elles. Sinon, la mesure mRMR est sélectionnée.

Dans le cas du trafic web, la méthodologie appliquée pour mettre en œuvre cette étape est double :

1. Tout d'abord, le sous-ensemble d'entités correspondant est visualisé dans l'espace bidimensionnel pour obtenir une matrice de tracé. Dans la matrice, chaque élément représente la distribution des points de données en fonction, soit des valeurs d'une entité et de l'étiquette de classe, soit des valeurs de deux entités.
2. Dans le but de vérifier les observations à partir des graphiques, l'étape suivante consiste à calculer les coefficients de corrélation entre les entités. Pour cela, le coefficient de corrélation de Pearson couramment utilisé [Lutu, 2010] est utilisé. Ces coefficients prennent des valeurs comprises entre -1 et 1. Selon les critères d'interprétation des corrélations de Cohen, les valeurs inférieures à 0,1 n'ont aucune signification pratique [Lutu, 2010]. Ensuite, on considère qu'il existe des corrélations linéaires entre les caractéristiques lorsqu'au moins la moitié des coefficients sont supérieurs à 0,1 [Nguyen, 2012].

Cette méthodologie peut être vue graphiquement sur la figure 4.3.

- Étape 2 : Selon le choix de l'étape 1, construire le problème d'optimisation en 4.1 pour les mesures CFS ou mRMR. Dans cette étape, les connaissances d'experts peuvent être utilisées en attribuant la valeur 1 à la variable si la fonctionnalité est pertinente et la valeur 0 dans le cas contraire
- Étape 3 : Transformer le problème d'optimisation de la mesure GeFS en un problème de programmation linéaire mixte 0-1 (M01LP), qui doit être résolu au moyen d'un algorithme de branche et lié. Une valeur entière non nulle de xi à partir de la solution optimale x indique la pertinence de la caractéristique fi par rapport à la mesure GeFS

Ces étapes doivent être suivies pour chaque sous-ensemble dans lequel la sélection de fonctionnalités est appliquée.

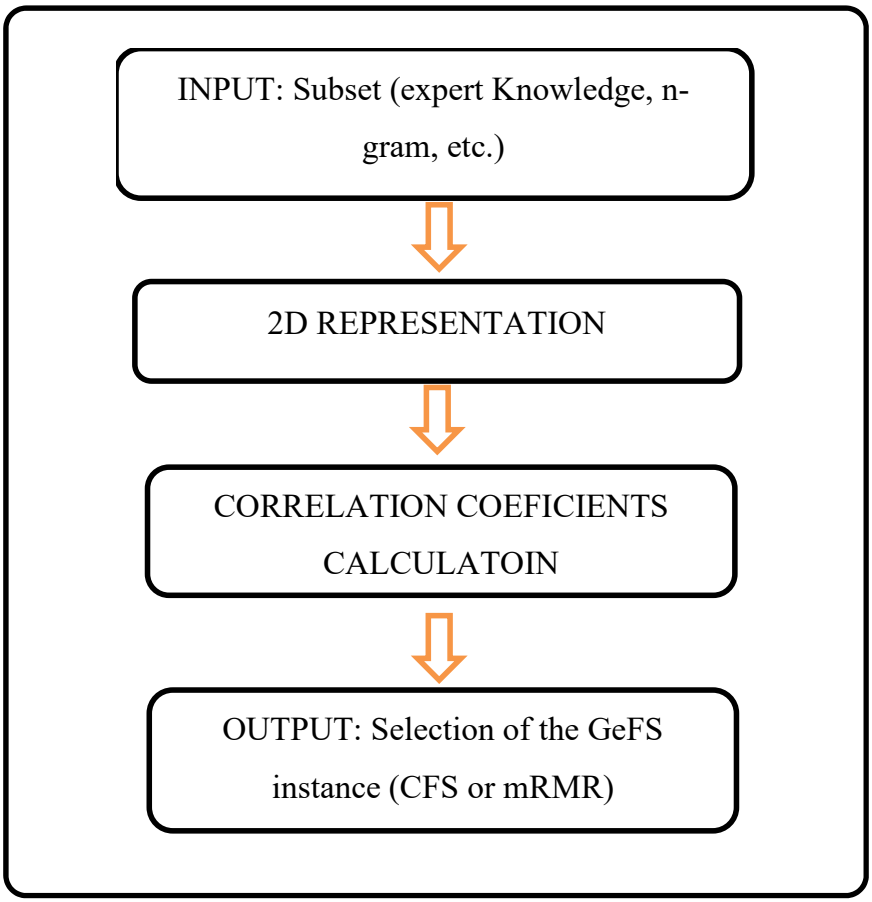


Figure 4.3 Schéma de sélection de l'instance appropriée du Mesure GeFS

4.3.3 Classification

Le problème de détection d'intrusion a été fréquemment formulé comme une tâche de reconnaissance de formes [Nguyen, 2012]. Ainsi, il peut être considéré comme un problème de classification, où les demandes doivent être classées comme normales ou anormales.

Les algorithmes d'arbre de décision sont des modèles prédictifs qui peuvent être utilisés comme classificateurs. Ils sont très utiles lorsqu'un volume massif de données doit être analysé. Dans ce travail, les algorithmes de classification sont chargés de décider s'il faut étiqueter la requête entrante comme normale ou anormale. Un avantage des arbres de décision est qu'ils construisent des règles de classification compréhensibles par l'homme, qui pourraient être

ultérieurement incluses dans un système de détection basé sur les signatures. En relation avec cela, des articles comme [Robertson et al., 2006] et [Bolzoni et Etalle, 2008] ont étudié comment dériver des règles à partir d'un système de détection d'anomalies.

Comme mentionné, quatre arbres de décision ont été appliqués comme algorithmes de classification dans le système ML. Ensuite, une brève explication de chaque algorithme est présentée. De plus amples détails sur les algorithmes peuvent être trouvés dans [Duda et al., 2001] et [Wu et al., 2007].

C4.5 a été introduit par Ross Quinlan. C'est un algorithme utilisé pour générer des arbres de décision qui sont construits à partir d'un ensemble de données d'apprentissage en utilisant le concept d'entropie d'information [Quinlan, 1993]. À chaque nœud de l'arbre, C4.5 choisit l'attribut qui divise le plus efficacement ses échantillons en classes différenciées. Pour cela, le critère est de choisir l'attribut qui donne le gain d'information normalisé le plus élevé. Ensuite, l'algorithme C4.5 se reproduit sur les petites sous-listes. L'arbre initial est élagué pour éviter le surajustement.

CART signifie Classification And Regression Tree. Il a été popularisé par Breiman et al. Il s'agit d'une méthode de partitionnement récursive qui construit des arbres pour prédire les variables dépendantes continues (régression) et les variables prédictives catégorielles (classification) [Breiman et al., 1984]. CART est un algorithme non paramétrique. Il génère un arbre de décision binaire qui est construit en divisant le nœud qui différencie le mieux la variable cible en deux nœuds enfants à plusieurs reprises. Il commence par le nœud racine, qui contient l'ensemble de l'échantillon d'apprentissage. Les aspects importants de CART sont de décider quand l'arborescence est complète et d'attribuer une classe à chaque nœud terminal.

Les arbres aléatoires incluent l'idée de sélectionner des entités au hasard. Cette idée a été introduite indépendamment par Ho [Ho, 1995], [Ho, 1998] et Amit et Geman [Amit et Geman, 1997]. L'implémentation utilisée dans ce mémoire construit un arbre qui considère K attributs choisis au hasard à chaque nœud. Il n'effectue pas d'élagage pour réduire la taille de l'arbre de décision. Selon Olaru et Wehenkel, l'objectif de la taille « est de fournir un bon compromis entre la simplicité d'un modèle et sa précision prédictive, en supprimant les parties non pertinentes du modèle » [Olaru et Wehenkel, 2003].

Random forest est un classificateur d'ensemble composé de nombreux arbres de décision. Sa classe de sortie est le mode de sortie de la classe par des arbres individuels. L'algorithme d'induction d'une forêt aléatoire a été développé par Breiman [Breiman, 2001], Cutler et Stevens [Cutler et Stevens, 2006].

La figure 4.4 montre un exemple d'arbre de décision extrait du logiciel Weka [Hall et al., 2009]. Il est construit par l'algorithme C4.5 sur le sous-ensemble select-n-gram-combine de l'ensemble de données ECML / PKDD lorsqu'il est formé avec 255 requêtes. Les valeurs dans les feuilles représentent la classe : normal est représenté par 0 et anormal par 1. Le nombre entre parenthèses signifie le nombre d'instances qui entrent dans cette catégorie. L'arbre peut être interprété comme suit : si la charge utile entrante contient ≤ 1 caractère '(' (attribut x85), alors la requête est classée comme normale. Sinon, si elle n'a aucun ou un caractère ')' (attribut x83), alors la demande est également considérée comme normale. Si ce n'est pas le cas, il est analysé si l'attribut x29 est inférieur ou égal à 7. L'attribut x29 représente le nombre de mots-clés dans le chemin. Dans ce cas, la demande est classée comme anormale. Sinon, la longueur de l'en-tête "User-Agent" (attribut x13) est vérifiée pour étiqueter la demande. Cette étiquette est anormale si x13 ≤ 66 et normale sinon. Comme on peut le voir, il est facile de dériver des règles à partir d'un arbre de décision.

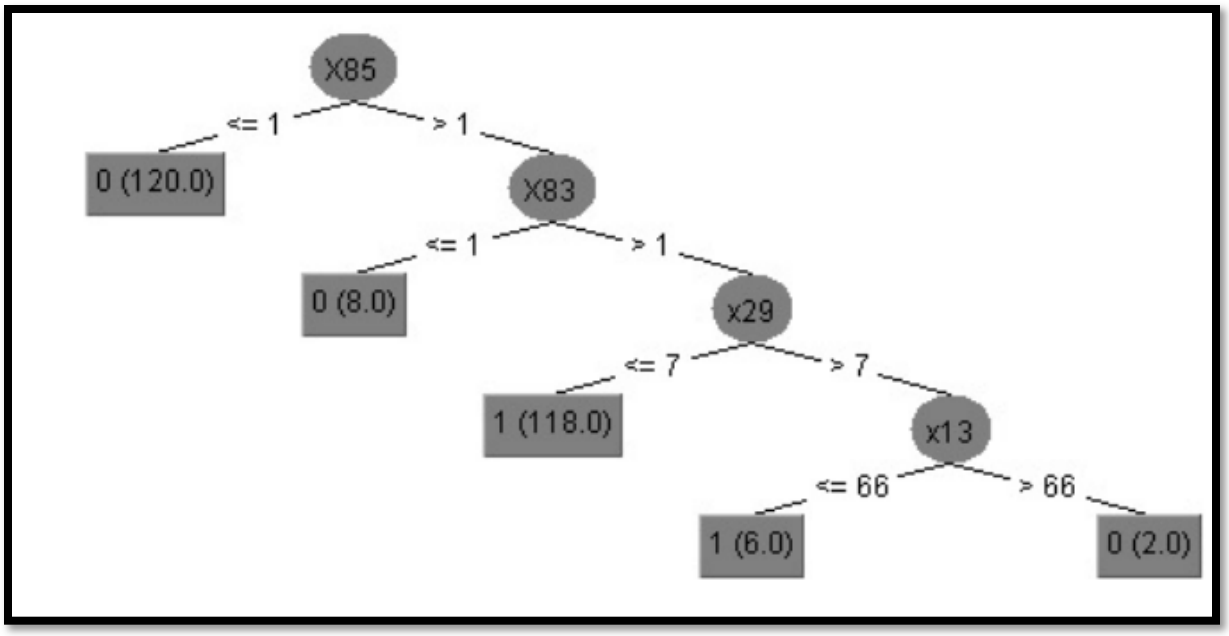


Figure 4.4 Exemple d'arbre de décision construit avec l'algorithme C4.5

4.4 Montage expérimental

Cette section explique la configuration expérimentale des expériences réalisées avec le WAF basé sur ML. Tout d'abord, les ensembles de données utilisés sont expliqués. Ensuite, les paramètres concernant l'étape de prétraitement sont présentés. Enfin, des paramètres pour l'étude de l'influence des demandes de formation sur les résultats de détection sont donnés.

4.4.1 Datasets

Dans ce chapitre, deux ensembles de données sont utilisés pour évaluer expérimentalement les algorithmes de détection : notre ensemble de données CSIC et l'ensemble de données ECML / PKDD.

L'ensemble de données de la CSIC a été présenté au chapitre 3. Il a été utilisé dans le chapitre précédent pour évaluer les algorithmes stochastiques. L'avantage d'utiliser cet ensemble de données pour les algorithmes stochastiques et ML est qu'il permet la comparaison des deux techniques.

En outre, l'ensemble de données public du CISC est utilisé. Cet ensemble de données a été généré pour le Défi Découverte 2007 du CISC [Raïssi et al., 2007]. Il est composé de 50 000 échantillons, dont 20% d'attaques. L'ensemble de données est divisé en ensembles de formation et de test. L'ensemble d'entraînement est utilisé pour les expériences. Les demandes sont étiquetées avec les spécifications du trafic normal ou les classes d'attaque suivantes : cross-site scripting, SQL injection, LDAP injection, XPATH Injection, path traversal, command execution et server-side include (SSI) attaque. Bien que les demandes de cet ensemble de données soient anonymisées, nous jugeons intéressant d'évaluer le système à l'aide d'un ensemble de données public.

Cet ensemble de données n'a pas pu être utilisé avec des systèmes stochastiques en raison des caractéristiques de ces systèmes et de l'anonymat de l'ensemble de données. Le fait que l'ensemble de données du CISC soit anonyme implique qu'il n'y a pas deux demandes adressant la même application Web. Cela rend l'ensemble de données inutilisable pour les algorithmes stochastiques conçus dans ce mémoire étant donné qu'ils utilisent la chaîne correspondant à la ressource dans le cadre du processus de détection. Le fichier XML généré serait énorme et il n'y aurait pas de ressources répétées, ce qui n'a pas de sens du point de vue statistique, puisque les algorithmes statistiques ont besoin d'un certain nombre de requêtes pour obtenir des résultats significatifs.

Dans le cas de ML, le système extrait les fonctionnalités des ressources, c'est-à-dire qu'elles utilisent les propriétés des ressources, comme leur longueur, au lieu de la chaîne du nom de la ressource elle-même. Ces caractéristiques rendent l'ensemble de données du CISC applicable dans le cas de ML.

4.4.2 Paramètres de prétraitement

Dans cette section, les paramètres des expériences concernant l'étape de prétraitement sont présentés.

La conception mentionnée dans la Sec. 4.3 est appliqué à tous les sous-ensembles de fonctionnalités : connaissance d'expert, n-gramme, combiner-sélectionner, sélectionner-

combiner et sélectionner-n-gram-combiner. Cette section explique, pour chaque sous-ensemble, les paramètres liés à l'extraction et à la sélection de caractéristiques. Concernant la première étape, il est expliqué comment chaque technique extrait les caractéristiques. Pour la sélection des caractéristiques, l'instance adéquate de la mesure GeFS est choisie pour chaque sous-ensemble, en suivant le critère qui a été donné dans la Sec. 4.3.2. De plus, il est montré comment le choix opposé de la mesure GeFS affecterait négativement les résultats.

4.4.2.1 Connaissance experte

Ensuite, les paramètres d'extraction et de sélection des fonctionnalités pour le sous-ensemble de connaissances expert sont présentés.

Feature sélection. Grâce à nos connaissances spécialisées et en nous inspirant [Rieck, 2009], diverses fonctionnalités considérées comme pertinentes pour la détection d'attaques Web a été extraites. Ces 29 features sont présentées dans le figure 4.5.

Comme on peut le voir dans le tableau, certaines des caractéristiques se réfèrent à la longueur des différentes parties de la requête, car la longueur est un aspect important à prendre en compte dans la détection d'attaques telles que le buffer-overflow ou XSS. Un autre groupe de fonctionnalités fait référence à l'apparence de certains types de caractères. En particulier, au nombre d'apparitions de lettres, de chiffres et de caractères non alphanumériques dans les valeurs de chemin et d'argument. Notez que, contrairement au cas stochastique, l'apparence de ces caractères n'est pas analysée dans les en-têtes. En approfondissant cette idée, l'extraction de fonctionnalités expertes dans ML distingue deux catégories de caractères non alphanumériques, selon qu'ils ont une signification particulière dans certains langages de programmation ou non.

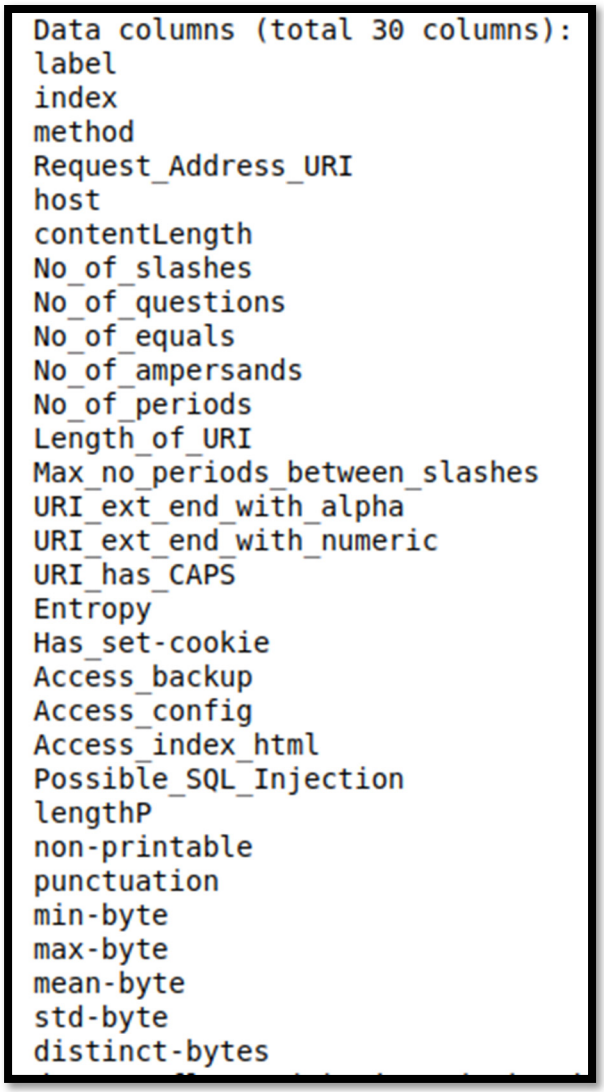


Figure 4.5 un ensemble de 29 fonctionnalités des features spécialisées considérées comme pertinentes pour la détection d'attaques Web pour l'ensemble de données de la CSIC

Par conséquent, quatre types de caractères sont considérés pour l'apprentissage automatique :

- Des lettres
- Chiffres

- Caractères non alphanumériques qui ont une signification particulière dans un ensemble de langages de programmation, SQL et Javascript dans notre cas. Ce type de caractères est désigné dans le Tableau 4.1 comme des caractères « spéciaux ».
- Autres caractères, c'est-à-dire les caractères non alphanumériques qui ne sont pas inclus dans la troisième catégorie

Une autre fonctionnalité est construite en étudiant l'entropie [Shannon, 2001] des octets composant les requêtes. Suivant l'idée de Rieck, un autre groupe de fonctionnalités est construit en comptant les apparitions de certains mots-clés dans différentes parties de la requête. Ces mots-clés ont été précédemment inclus dans une liste qui contient des termes avec une signification particulière dans certains langages de programmation qui sont souvent utilisés dans les attaques par injection, comme SQL et Javascript.

4.4.2.2 N-grams

Les paramètres d'extraction et de sélection de caractéristiques pour les n-grams sont expliqués ci-après.

Extraction des caractéristiques (Feature extraction). Pour l'extraction d'entités avec n-grammes dans le système ML, les n-grammes sont mis à n = 1. Cette valeur a été choisie puisque n = 1 est le cas le plus simple. Il faut considérer que dans le trafic Web, il n'y a pas de longues chaînes car elles sont limitées par la longueur de la requête. Dans ces cas, de petites valeurs pour n sont généralement choisies. Bien que l'on puisse s'attendre à ce que les résultats s'améliorent à mesure que n augmente, il existe des articles dans la littérature utilisant des n-grammes dans le trafic HTTP qui montrent que ce n'est pas nécessairement le cas, comme [Song et al., 2009] et [Perdisci et al., 2009]. De plus, les cas avec n> 1 nécessitent un coût élevé en temps et en complexité de calcul, ce qui n'est pas approprié pour des algorithmes fonctionnant en environnement réel ou des scénarios avec des contraintes de ressources. Par conséquent, les cas avec n> 1 n'ont pas été considérés dans ce mémoire.

En suivant la formule donnée à la Sec. 2.3.1.1 pour calculer le nombre de n-grammes, $S = \{n\text{-}grams_i | i = 1...2^{8n}\}$, le nombre de tous les 1-grams possibles est de 256. Rappelez-vous que le système basé sur ML analyse l'ensemble de la demande. Lors de l'analyse des demandes des

ensembles de données étudiés, il a été révélé que tous les 1 grammes n'étaient pas présents. Le résultat obtenu est que seulement 96 entités (37,5% de 256) apparaissent au moins une fois dans l'ensemble de données du CISC et 114 (44,5%) dans le cas de l'ensemble de données de la CSIC.

En supposant que les charges utiles du trafic normal sont différentes de celles des attaques, la méthode automatique pour extraire les fonctionnalités des requêtes est la suivante : étant donné une requête HTTP req, un vecteur de fonctionnalité de req est construit comme $x_{req} = (x_1, x_2, \ldots, x_\lambda)$, où x_i est le nombre d'apparitions de n-gram$_i$ dans req et λ le nombre de 1-grams apparaissant au moins une fois dans l'ensemble de données correspondant. Le vecteur construit pour chaque requête représente le nombre d'apparitions du n-gram correspondant dans la requête HTTP. Dans notre cas, les 1 grams correspondent à des caractères individuels.

4.4.2.3 Algorithmes de classification

Comme mentionné, les expériences sont réalisées avec les quatre arbres de décision : C4.5, CART, Random Tree et Random Forest. Ces algorithmes sont décrits dans la Sec. 4.3.3. Ces algorithmes reçoivent en entrée les sous-ensembles de fonctionnalités. En particulier, ils sont appliqués pour analyser tous les sous-ensembles de caractéristiques expert knowledge, n-gram, combine-select, select-combine and select-n-gram-combine), à la fois avant et après la sélection de caractéristiques.

Les arbres de décision comportent deux étapes : la formation et le test. En phase d'apprentissage, les requêtes normales et anormales alimentent l'algorithme pour obtenir un modèle qui classe le trafic. Dans la phase de test, les connaissances acquises précédemment sont utilisées pour vérifier dans quelle mesure l'algorithme détecte, c'est-à-dire si les demandes sont correctement classées comme normales ou anormales.

Les implémentations des arbres de décision utilisés sont celles fournies par le logiciel Weka (University of Waikato, Hamilton, New Zealand) [Hall et al., 2009]. La validation croisée est une technique de validation pour évaluer comment les résultats de l'algorithme se

généraliseront à un autre ensemble de données. Dans la validation croisée de N fois, l'ensemble de données est divisé en N sous-ensembles de taille égale. Il est exécuté N fois. Dans chacun d'eux, l'un des sous-ensembles est utilisé pour les tests et les N - 1 restants sont utilisés pour l'apprentissage de l'algorithme. Les résultats sont moyennés et l'écart type est calculé [Kubat, 2015]. L'utilisation de la validation croisée permet d'obtenir des résultats plus fiables. Dans notre cas, les expériences sont menées avec une validation croisée de 10 fois. Selon Kohavi, la validation croisée stratifiée 10 fois est une méthode optimale pour les ensembles de données du monde réel [Kohavi, 1995]. Les valeurs de réglage restantes utilisées sont celles définies par défaut dans le logiciel Weka.

4.4.3 Paramètres pour l'étude de l'influence des demandes de formation sur les résultats de détection

Comme mentionné, les arbres de décision ont une formation et une phase de test. Pour ce type d'algorithmes, des requêtes normales et anormales sont utilisées pour entraîner le système. De cette manière, l'algorithme peut apprendre à classer les instances (détection à deux classes : normale et anormale). Les deux types de trafic sont également utilisés pendant la période de test.

Afin d'étudier comment le nombre de demandes d'entraînement influence les performances du système, une fois que tous les sous-ensembles précédents de fonctionnalités sont évalués, celui qui atteint les meilleurs résultats est choisi pour étudier comment le nombre de demandes influence les résultats de détection de l'algorithme.

Pour cela, un nombre d'expériences M = 15 est exécuté, avec un nombre croissant de requêtes par expérience. Comme dans le cas stochastique, le nombre de demandes de formation utilisées dans chaque expérience est donné par la formule $tr_i = 2^i - 1, \forall i \in [1, M]$. Dans chaque expérience, le système est d'abord formé avec trois requêtes choisies au hasard, puis il est testé avec un sous-ensemble fixe de requêtes. Ce sous-ensemble fixe est composé de la requête normale plus les requêtes anormales, avec te = 1000. Les M expériences sont exécutées H = 10 fois. A chaque fois, différentes possibilités de choix de tri requêtes sont analysées. L'exécution des expériences plusieurs fois permet de refléter réellement le comportement du WAF, indépendamment des demandes de formation choisies. L'aléatoire a été sélectionné pour

l'échantillonnage des requêtes car il n'est pas possible de savoir à l'avance quel type d'attaque le système va recevoir. La section suivante montre les résultats obtenus à partir des expériences.

4.5 Résultats

Cette section présente, d'une part, les résultats des différents sous-ensembles, afin de déterminer l'efficacité des méthodes combinées proposées. En revanche, les résultats de l'étude sur l'influence du nombre de demandes de formation sur la capacité de détection du WAF sont présentés.

4.5.1 Temps de traitement

Les expériences révèlent le temps de traitement suivant, en fonction de la technique de détection utilisée :

- L'algorithme statistique est capable de traiter les requêtes jusqu'à un taux de 0,59 ms / requête (c'est-à-dire que le traitement d'une seule requête prend 0,00059 s).
- Le temps de traitement est de 7,9 ms / requête pour les chaînes de Markov.
- Dans le cas du ML, une moyenne du temps de traitement des quatre arbres de décision est prise. Les expériences montrent que le ML peut atteindre un temps de traitement de 0,3 ms / requête pour l'ensemble de données CSIC. Notez que lorsque la moyenne des algorithmes est prise, l'utilisation d'un algorithme spécifique pourrait même améliorer les résultats. En particulier, Random Tree est l'arbre de décision le plus rapide (il peut être 7 fois plus rapide que C4.5)

4.5.2 Matériel utilisé

Toutes les mesures de temps de traitement ont été obtenues avec un processeur Intel core i7 à 2,40 GHz et 16 Go de RAM, SO linux Ubuntu 20,04 Lts, 64 bits.

4.5.3 Résultats des sous-ensembles

Dans le but de valider si les méthodes d'extraction des features combinées proposées sont plus efficaces que les techniques individuelles, les résultats correspondant aux deux cas de base sont d'abord présentés, puis ceux correspondant aux trois alternatives de combinaison. Les résultats de détection sont présentés en termes de DR et FPR. De plus, le nombre de fonctionnalités utilisées par chaque technique et son temps de traitement sont également inclus. Le nombre de fonctionnalités utilisées donne une mesure de la consommation des ressources. Les résultats obtenus sont mesurés au moyen du taux de détection et du taux de faux positifs.

4.5.3.1 Connaissance experte

En ce qui concerne les connaissances des experts, le tableau 4.1 montre les résultats de détection des quatre algorithmes de classification pour l'ensemble de données du CSIC. Les lignes affichent les résultats pour les différents arbres de décision utilisés, enfin, la valeur moyenne. L'ensemble complet de la colonne affiche les résultats avant la sélection des fonctionnalités et les colonnes CFS et mRMR correspondent aux résultats après la sélection des fonctionnalités. Dans le tableau, l'instance GeFS sélectionnée est mise en évidence par des lettres en gras (mRMR dans le cas de connaissances d'experts).

Tableau 4.1 Taux de détection et taux de faux positifs de quatre arbres décisionnels effectués sur le sous-ensemble de connaissances d'experts de l'ensemble de données de la CSIC

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	94.98	**94.07**	79.8	5.3	**6.8**	25.7
CART	94.31	**93.72**	79.86	7.0	**6.8**	25.3
Random Tree	92.76	**92.71**	71.36	7.7	**7.8**	30.6
Random Forest	93.93	**93.69**	71.70	6.0	**7.2**	30.5
Average	**93.99**	**93.55**	**75.5**	**6.5**	**7.15**	**28.02**

Si l'on considère également le nombre d'entités, il est à noter que la mesure GeFS réduit considérablement le nombre d'entités, tout en conservant presque les mêmes résultats de détection. Bien que CFS réduit le nombre de caractéristiques (de 30 à 2) plus que le mRMR (de 30 à 6), ses résultats de détection sont pires, avec un DR plus faible (89,1% contre 91,22%) et un FPR plus élevé (23,18% contre 15,3%). Ce fait montre que la méthode utilisée pour choisir l'instance appropriée du GeFS conduit à sélectionner l'instance qui atteint les meilleurs résultats, même si ce n'est peut-être pas celle qui sélectionne le moins de fonctionnalités. Notez que l'objectif de réduire le nombre de fonctionnalités et d'améliorer les résultats de détection est contradictoire, par conséquent, un compromis doit être fait. La sélection de fonctionnalités tente de réduire le nombre de fonctionnalités sans affecter négativement les résultats de détection.

4.5.3.2 Comparaison avec des ouvrages connexes

Dans cette section, nous voulons comparer notre implémentation et les résultats que nous avons obtenus avec certains travaux d'auteurs liés au domaine IDS avec détection d'anomalies. Nous nous concentrons sur les emplois qui utilisent les mêmes ensembles de données que nous. Le tableau 4.2 présente un résumé de la comparaison des résultats de notre mise en œuvre avec les travaux d'autres chercheurs.

Tableau 4.2 Comparaison avec les travaux connexes

Système de détection	Oobservations	TPR	FPR	F1-score
Dans ce travail	RF avec un nombre de features	0,93	0,06	-
ModSecurity (Gimenez, 2015)	Détecteur de signature d'attaque populaire	0,56	0,00	0,71
HTTP-WS-AD (Gimenez, 2015)	Modèles statistiques avec agrégation et cascade	0,99	0,02	0,99
OC-WAD (Parhizkar and Abadi, 2015)	Ensemble de SVM à une classe avec un nombre fixe de features	0,96	0,03	-

CONCLUSION

Ce chapitre présente d'abord un bref résumé du présent mémoire, puis les conclusions extraites de la recherche sont présentées, les contributions de la thèse ainsi que les publications qui en découlent. Enfin, de futures lignes de recherche sont tracées.

5.1 Résumé

Les applications Web participent de plus en plus à notre vie quotidienne. Ils deviennent de plus en plus populaires et complexes dans toutes sortes d'environnements, allant des applications de commerce électronique à la banque. Ce fait rend les applications Web très attrayantes pour les attaquants, qui ont l'intention d'exploiter des vulnérabilités Web. Les applications Web sont menacées, il est donc nécessaire de les protéger. Afin de détecter les attaques spécifiques au Web, des mécanismes de détection doivent être placés au niveau de la couche application. C'est l'objectif des WAF : analyser le trafic HTTP dans le but de détecter les intrusions web. Ces systèmes sont un cas particulier des IDS, avec la particularité d'être spécialisés sur l'analyse du trafic web. L'un des avantages des IDS et des WAF est qu'ils protègent une application Web cible sans qu'il soit nécessaire de modifier son code source.

L'objectif principal de ce mémoire est de développer des systèmes de détection d'intrusions capables de détecter avec précision les attaques Web avec une faible consommation de ressources, une vitesse élevée et une conception simple. Pour atteindre cet objectif, diverses techniques ont été utilisées pour la détection : techniques basées sur la stochastique et apprentissage automatique. Les systèmes proposés suivent une approche basée sur les anomalies. Concernant les techniques stochastiques, deux méthodes ont été appliquées : les algorithmes statistiques et les chaînes de Markov. Concernant l'apprentissage automatique, quatre arbres de décision ont été utilisés pour détecter les attaques web, à savoir C4.5, CART, Random Tree et Random Forest.

Des aspects supplémentaires de la détection des attaques Web ont été abordés dans ce mémoire. D'une part, il étudie l'influence que produit le nombre de requêtes utilisées dans la phase d'apprentissage sur la capacité de détection du système. D'autre part, il mène une étude sur les fonctionnalités les plus efficaces pour la détection d'intrusions Web. Pour cela, dans un premier

temps, trois méthodes d'extraction de caractéristiques ont été étudiées : la connaissance d'expert, les n-grams et une combinaison des deux options précédentes. Ils ont été analysés afin de déterminer lequel conduit aux meilleurs résultats de détection et à la plus faible consommation de ressources. Deuxièmement, les caractéristiques extraites ont été sélectionnées au moyen de la mesure GeFS.

Les systèmes de détection proposés ont été testés expérimentalement. Pour cela, un trafic HTTP est nécessaire. Il est essentiel de compter avec des ensembles de données appropriés pour la formation et le test des méthodes de détection. Cependant, dans le champ de détection d'intrusion Web, la collecte du trafic étiqueté et approprié se heurte à plusieurs difficultés. Le principal problème identifié est la rareté des jeux de données HTTP étiquetés [Sommer and Paxson, 2010], [Tavallaee et al., 2010]. Dans ce mémoire, l'ensemble de données du ECML/PKDD, généré pour le Défi de découverte du ECML/PKDD, a été utilisé, mais l'inconvénient de cet ensemble de données est que la plupart des parties des demandes sont anonymisées, ce qui complique l'évaluation approfondie des performances de l'algorithme de détection. Par conséquent, un ensemble de données commun et public serait nécessaire pour évaluer correctement les systèmes. Cela permettrait également de comparer différents schémas et techniques. Telle a été la motivation pour générer l'ensemble de données de la CSIC. Il s'agit d'un nouvel ensemble de données HTTP accessible au public qui est utilisé par la communauté de détection d'intrusion Web pour évaluer leurs systèmes de détection. Il se compose de requêtes étiquetées normales et anormales, y compris de multiples attaques modernes. De plus, il n'est pas anonymisé.

Après ce résumé, les conclusions obtenues lors de l'élaboration de ce travail sont données.

5.2 Conclusions

Les conclusions les plus pertinentes tirées de ce mémoire sont :

- **Les algorithmes stochastiques et d'apprentissage automatique peuvent être utilisés avec succès dans la détection d'attaques Web basée sur des anomalies.**
  Ils sont capables de faire la distinction entre le trafic normal et anormal avec une faible consommation de temps et de ressources. Les deux approches ont permis de détecter les attaques zero-day.

- **L'objectif de construction de WAF à haute détection a été atteint**

  Les résultats montrent que les meilleurs résultats de détection sont atteints par le système statistique.

  La performance des systèmes est mesurée comme la moyenne du taux de détection et du taux de faux positifs. Les valeurs obtenues à partir des expériences sont les suivantes:

  - Par rapport au système basé sur des statistiques, il est capable d'atteindre un taux de détection de 99,4% et un taux de faux positifs de 0,9%
  - Le système markovien atteint un taux de détection de 98,1% et un taux de faux positifs de 1%. Ces résultats ont été atteints avec les valeurs de paramètres suivantes pour la chaîne de Markov : $\tau = 50$, $\epsilon = 10^{-15}$ et $p = 0.99$.
  - En ce qui concerne les techniques de ML, les résultats ont été calculés comme la moyenne du taux de détection de quatre arbres de décision (C4.5, CART, Random Tree et Random Forest). Les résultats ont été montrés pour deux ensembles de données : dans le cas de l'ensemble de données de la CISC, le taux de détection augmente jusqu'à 95,1% lorsque le taux de faux positifs est de 4,9%. Pour l'ensemble de données ECML/PKDD, le DR atteint 97,8% avec 2,2% de FPR. Notez que les FPR supérieurs à 0,01 ne sont pas problématiques, étant donné que les systèmes SIEM analysent la sortie des IDS et corrélent les données, ce qui permet de réduire le nombre de faux positifs.

- **L'influence du nombre de demandes de formation sur les résultats de détection a été étudiée**

Pour cela, M = 15 expériences utilisant un nombre croissant de demandes de formation, de 1 à 32 767 ont été réalisées. Ils ont été exécutés H = 10 fois, en choisissant différents échantillons de demandes de formation.

Contrairement à ce à quoi on pourrait s'attendre, le comportement des systèmes n'est pas toujours linéaire, par conséquent, utiliser plus de demandes de formation n'implique pas toujours de meilleurs résultats

Des expériences ont révélé ce qui suit :

- Dans le cas du WAF statistique, en utilisant 16 383 demandes pour entraîner le système, un taux de faux positifs de 0,9% est atteint. Lorsque la formation se fait avec des quantités de demandes plus importantes, il est possible de réduire progressivement le taux de faux positifs, jusqu'à atteindre un FPR de 0,4%. Le taux de détection reste quasiment invariable : de 99,4% à 99,3%. Ceci est réalisé avec 32 767 demandes de formation.
- Dans le cas des chaînes de Markov, 32 767 requêtes sont nécessaires pour obtenir 1% de FPR. Il est à noter que lorsque 8191 demandes sont utilisées, le FPR est déjà de 1,7%
- En ce qui concerne le ML, les expériences montrent qu'avec 32 767 demandes de l'ensemble de données de la CSIC, le système atteint un FPR de 4,9%. Le taux diminue à 2,2% dans le cas de l'ensemble de données du ECML/PKDD lorsque 16 383 demandes sont utilisées.

En résumé, la méthode statistique et le système ML (dans le cas de l'ensemble de données du ECML/PKDD) sont ceux qui nécessitent le plus petit nombre de demandes de formation du système (16 383), sans décrémenter sa capacité de détection. Les chaînes de Markov et le système ML pour l'ensemble de données de la CSIC utilisent un double nombre de requêtes.

- **Les méthodes d'extraction de caractéristiques proposées, qui combinent des connaissances d'experts et des n-grams, ont amélioré les résultats de détection des deux techniques séparément.**

En outre, la sélection des fonctionnalités a été appliquée pour réduire les fonctionnalités non pertinentes et redondantes. L'utilisation d'un petit nombre de fonctionnalités entraîne une faible consommation de ressources.

En comparant les résultats des trois alternatives de combinaison proposées, à savoir combine-select, select-combine et select-n-gram- combine, on peut conclure que la première est l'option qui réduit le plus le nombre de fonctionnalités, mais elle obtient également les résultats les plus bas. Au contraire, select-n-gram-combine atteint les meilleurs résultats de détection, mais il a

besoin du plus grand nombre de fonctionnalités. Select-combine est un cas intermédiaire entre les deux autres alternatives.

En général, un nombre inférieur de fonctionnalités implique des temps de traitement plus courts. Cependant, il a été démontré que dans certains cas, comme pour le sous-ensemble select-combine de l'ensemble de données de la CSIC, la combinaison pouvait même réduire le temps de traitement.

- **Un nouvel ensemble de données accessible au public et étiqueté, l'ensemble de données de la CSIC, a été créé**

L'ensemble de données contient exclusivement du trafic HTTP et ses requêtes sont étiquetées comme normales ou anormales. Le fait que le jeu de données soit étiqueté permet d'évaluer les capacités de détection du système de détection Web. Plus précisément, l'ensemble de données contient environ 36 000 demandes normales de sable 25 000 anomalies. En outre, l'ensemble de données CSIC comprend des attaques Web modernes telles que l'injection SQL, le débordement de tampon, XSS, l'inclusion côté serveur, etc., satisfaisant la nécessité d'évaluer le comportement des systèmes face aux attaques modernes. Compte tenu de la vitesse rapide de l'évolution des attaques Web de nos jours, il est très important que les mécanismes de protection soient prêts à répondre de manière adéquate à ces menaces. Un autre avantage de l'ensemble de données est qu'il comprend des valeurs réalistes et qu'il n'est pas anonymisé.

- **Le comportement des techniques de détection proposées (méthodes statistiques, chaînes de Markov et arbres de décision) a été comparé à l'aide de l'ensemble de données de la SCCI pour étudier laquelle d'entre elles est la plus recommandable dans chaque scénario.**

Cette étude révèle que, en fonction du scénario et de l'objectif poursuivi, le système recommandable doit être choisi comme suit : lorsque l'objectif est d'obtenir des résultats de détection élevés, alors le système statistique est le plus approprié. C'est également l'option la plus appropriée lorsqu'un faible nombre de demandes est disponible. Cependant, si l'intérêt primordial est un temps de traitement peu élevé du système, alors les arbres de décision sont optimaux.

BIBLIOGRAPHIE

A. Tama, L. Nkenyereye, S. M. R. Islam and K. Kwak, "An Enhanced Anomaly Detection in Web Traffic Using a Stack of Classifier Ensemble," in IEEE Access, vol. 8, pp. 24120-24134, 2020, doi: 10.1109/ACCESS.2020.2969428.

BERGHEL, H. The Code Red Worm. Communications of the ACM 44, 12 (Dec. 2001), 15–19.

DOYLE, J., SHROBE, H., AND SZOLOVITS, P. On widening the scope of attack recognition languages. http://medg.lcs.mit.edu/doyle/ publications/ Accessed 29 July 2002, 2000.

ECKMANN, S., VIGNA, G., AND KEMMERER, R. STATL: an attack language for state-based intrusion detection. Journal of Computer Security 10, 1–2 (2002), 71–103.

FYODOR, Y. 'SnortNet'—a distributed intrusion detection system, June 2000. http://snortnet.scorpions.net/snortnet.ps Accessed 5 August 2002.

KANSON, P. H. All public hospitals in Gothenburg Sweden crippled by Nimda. Forum on Risks to the Public in Computers and Related Systems 21, 67 (Oct. 2001). http://catless.ncl.ac.uk/Risks/21.67.html#subj13. Accessed 27 Dec 2002.

M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, 2018, pp. 131-134, doi: 10.1109/CFIS.2018.8336654.

ROBERTSON, W., VIGNA, G., KRUEGEL, C., AND KEMMERER, R. A. Using generalization and characterization techniques in the anomalybased detection of web attacks. In Network and Distributed System Security Symposium Conference Proceedings: 2006 (2006), Internet Society. http://www.isoc.org/isoc/conferences/ndss/06/proceedings/html/2006/papers/anomaly_signatures.pdf Accessed 12 February 2006.

Scully, Peter M. D. (2015). CSIC 2010 HTTP dataset in CSV format (for weka analysis). Intelligent Robotics Research Group, Dept. of Computer Science Aberystwyth University, Ceredigion, Wales. http://users.aber.ac.uk/pds7/csic_dataset/csic2010http.html

SDN-NFV Sec'18: Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function VirtualizationMarch 2018 Pages 1–6https://doi.org/10.1145/3180465.3180467

SEKAR, R., GUPTA, A., FRULLO, J., SHANBHAG, T., TIWARI, A., YANG, H., AND ZHOU, S. Specification-based anomaly detection: a new approach for detecting network intrusions. In Proceedings of the 9th ACM conference on Computer and communications security (2002), ACM Press, pp. 265–274.

Shannon, C. E. (2001). A mathematical theory of communication. SIGMOBILE Mobile Computing and Communications Review, 5(1):3–55. http://doi.acm.org/10.1145/584091.584093

T. S. Pham, T. H. Hoang and V. Van Canh, "Machine learning techniques for web intrusion detection — A comparison," 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE), Hanoi, 2016, pp. 291-297, doi: 10.1109/KSE.2016.7758069.

Top 10 Web Application Security Risks, https://owasp.org/www-project-top-ten/

VIGNA, G., ECKMANN, S., AND KEMMERER, R. Attack languages. In ISW 2000. 34th Information Survivability Workshop, 24–26 Oct. 2000, Cambridge, MA, USA (Piscataway, NJ, USA, 2000), IEEE, pp. 163–6.

Yeung, D.-Y. and Ding, Y. (2003). Host-based intrusion detection using dynamic and static behavioral models. Pattern Recognition, 36(1):229–243. http://repository.ust.hk/dspace/ bitstream/1783.1/2495/1/yeung.pr2003.pdf

Z. Zhang, R. George and K. Shujaee, "Efficient detection of anomolous HTTP payloads in networks," SoutheastCon 2016, Norfolk, VA, 2016, pp. 1-3, doi: 10.1109/SECON.2016.7506638.

Zhang, L. and White, G. B. (2007). Anomaly detection for application level network attacks using payload keywords. In Proceedings of the IEEE Symposium on Computational Intelligence in Security and Defense Applications CISDA 2007, pages 178–185.