CASTAFIOR : Détection automatique de tunnels illégitimes par analyse statistique

M. Morel[2], F. Allard[1], R. Dubois[1], and P. Gompel[3]

[1] Thales Communications
160 Boulevard de Valmy – BP 82
92704 Colombes Cedex – France
firstname.lastname@fr.thalesgroup.com
[2] mathieu.c.morel@gmail.com
[3] pgompel@gmail.com

Résumé L'établissement de tunnels (HTTP, HTTPs ou apparentés) avec un serveur distant maîtrisé pour y faire passer des flux illégitimes est un des moyens les plus efficaces et les plus utilisés pour contourner la politique de sécurité d'un réseau. Les produits de sécurité actuels (pare-feux, serveurs mandataires, IDS...) sont généralement incapables de détecter ce genre de contournement. Une des solutions proposées dans la littérature consiste à utiliser des outils de classification statistiques pour identifier le protocole encapsulé dans un flux. Nous présentons ces techniques dans cet article et nous évaluons leur faisabilité à travers une preuve de concept. En particulier, un outil original permettant la détection des tunnels illégitimes à partir de quelques paramètres extraits des flux interceptés est décrit. Les résultats de manipulations expérimentales sur des bases de données réelles, présentés à la fin de cet article, montrent la viabilité des techniques utilisées. Un tel outil pourrait être mis en œuvre pour pallier les lacunes des IDPS et serveurs mandataires existants.

Mots-clés : Lutte informatique défensive (LID), CyberDefense, tunnels HTTPs, machine learning, HMM, arbres de décision, *RandomForest*.

1 Introduction

L'article s'organise de la manière suivante. Cette première partie introductive rappelle quelques notions élémentaires sur les tunnels applicatifs et les vulnérabilités qu'ils induisent. La deuxième partie explique l'intérêt de l'utilisation de méthodes d'analyse discriminante dans le cadre de la détection des tunnels illégitimes. Dans une troisième partie, un outil utilisant ces méthodes est décrit : CASTAFIOR. Enfin, la dernière partie présente les résultats obtenus en mettant en œuvre cet outil sur des bases de données de flux réels.

1.1 Principe de fonctionnement des tunnels applicatifs

Le filtrage des flux entrant et sortant d'un réseau de communication et la vérification de leur conformité à la politique de sécurité locale s'effectue généralement sur une passerelle, à l'aide de pare-feux, de serveurs mandataires et éventuellement d'IDS (*intrusion detection system*) ou d'IDPS (*intrusion detection and prevention system*). Les méthodes utilisées par ces outils pour détecter les infractions à la politique de sécurité sont les suivantes :

- filtrage de niveau réseau et transport (pare-feux) ;
- inspection du contenu des paquets au niveau applicatif (serveurs mandataires) ;
- utilisation d'analyse comportementale élémentaire pour identifier les comportements suspects (IDS ou IDPS comportementaux).

Néanmoins, il est généralement possible de contourner l'ensemble de ces filtrages en établissant un tunnel avec un serveur distant maîtrisé. Cette méthode consiste à encapsuler les données applicatives d'un protocole interdit à l'intérieur de trames d'un protocole autorisé à traverser la passerelle. Le serveur distant, contrôlé par l'utilisateur, extraie les trames encapsulées et les relaie vers leur véritable destination (cf. figure 1).

Par exemple, le logiciel *HTTPTunnel* [8] permet d'encapsuler des trames SSH ou *peer to peer* (P2P) dans des requêtes HTTP. De manière similaire, *Stunnel* [2] permet d'encapsuler la plupart des protocoles dans une connexion SSL/TLS, sur le port 443 (pour simuler une connexion HTTPs). Si l'on peut espérer qu'un serveur mandataire perfectionné détecte une anomalie dans les requêtes HTTP générées par *HTTPTunnel*, les flux générés par *Stunnel* sont quant à eux chiffrés, ce qui empêche toute inspection du contenu des paquets.

Notons que sur certains réseaux d'entreprise, pour permettre aux serveurs mandataires d'inspecter les données applicatives contenues dans les flux chiffrés, les passerelles sont configurées en coupure de chiffrement. Cette solution est cependant loin d'être pleinement satisfaisante, car elle empêche le chiffrement et l'authentification de bout en bout et pose donc de lourds problèmes de sécurité et de respect des données privées.

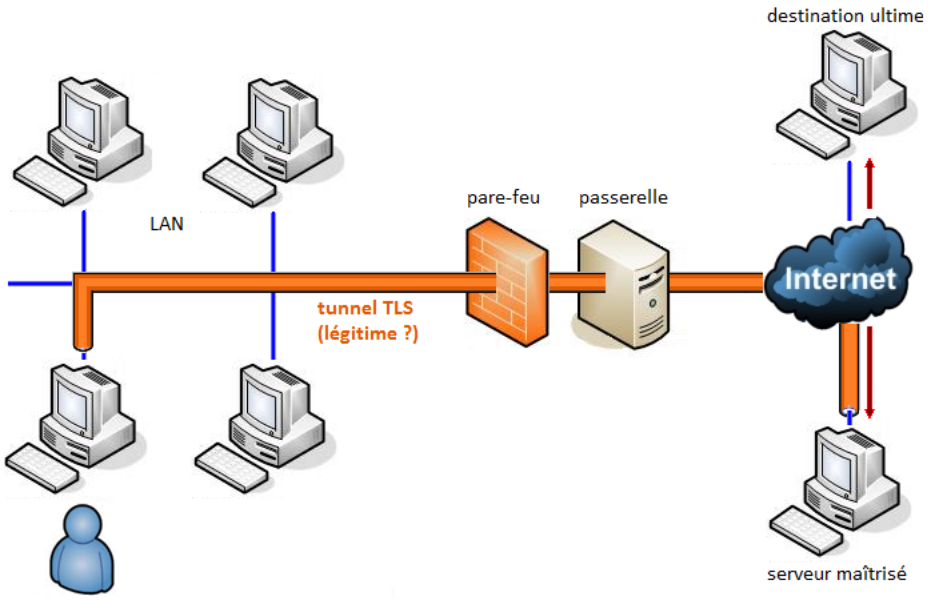

Figure 1. Un tunnel TLS échappe au filtrage de la passerelle du réseau

1.2 Vulnérabilités

Les deux logiciels précédemment cités à titre d'exemple sont librement téléchargeables sur Internet, et leur utilisation est simple pour un utilisateur ayant une connaissance élémentaire des réseaux IP. L'établissement d'un tunnel étant d'une facilité remarquable, ceux ci se multiplient et constituent aujourd'hui une des préoccupations principales des spécialistes de la sécurité des systèmes d'information. Une note du CERTA [11] souligne l'importance de ce problème.

Les tunnels sont souvent établis par des utilisateurs légitimes jugeant trop restrictif le filtrage appliqué par la passerelle de leur entreprise ou administration. Ils utilisent donc des tunnels HTTP/HTTPs (ces deux protocoles étant toujours autorisés) afin de pouvoir communiquer en SSH ou TELNET avec un serveur extérieur, voire pour pouvoir utiliser des logiciels de P2P ou de voix sur IP (VoIP). Les données encapsulées ainsi échangées échappent totalement au contrôle de la passerelle, introduisant une vulnérabilité dans le réseau. Ces données peuvent contenir des virus, le logiciel de *tunneling* lui même peut contenir une porte dérobée, ou encore l'utilisateur peut

être malveillant et utiliser le tunnel pour faire fuir de l'information. Enfin, notons que les tunnels sont aussi utilisés par des attaquants extérieurs pour communiquer avec une machine locale ayant été infectée par une porte dérobée, voire par des utilisateurs légitimes pour exploiter leurs machines depuis l'extérieur.

Les tunnels illégitimes constituent donc une brèche dans l'architecture de sécurité d'un réseau, difficilement détectable par les pare-feux, serveurs mandataires ou IDPS actuels.

2 Une nouvelle approche : utiliser des outils de classification statistiques

2.1 Principe général

Pour détecter la présence de tunnels illégitimes, l'utilisation des numéros de port s'avère inutile, et l'inspection approfondie du contenu des paquets est souvent source d'erreurs (les concepteurs des logiciels de *tunneling* étant particulièrement inventifs lorsqu'il s'agit de cacher des données dans un flux légitime), voire impossible si le flux est chiffré (cas des tunnels HTTPs). Pour pallier ce problème, une méthode originale consiste à utiliser des outils statistiques pour déterminer le protocole encapsulé dans un flux. Une fois ce protocole identifié, il suffit de se référer à la politique de sécurité du réseau pour décider du filtrage ou non du flux.

Les paramètres utilisés par ces outils statistiques doivent être observables quel que soit le flux considéré, et difficilement manipulables par un attaquant. On utilise en pratique la taille et le sens des paquets échangés entre le serveur et le client, ainsi que les temps séparant les arrivées de ces paquets. L'idée sous-jacente est que chaque protocole ou classe de protocole (HTTP, SSH, P2P, VoIP, etc.) induit un comportement caractéristique en termes de paquets de données [4] générés. Par exemple, un flux SSH sera composé majoritairement de "petits" paquets échangés dans les deux sens (les frappes au clavier, puis leurs réponses "*echo*"), tandis qu'un flux HTTP typique consistera en une requête de taille moyenne, suivie de la réponse du serveur sur plusieurs paquets de grandes tailles.

4. Dans tout l'article, le terme de "paquet" doit s'entendre au sens "paquet TCP transportant des données applicatives".

On suppose également que l'encapsulation d'un protocole dans un tunnel HTTP/HTTPs ne modifie pas notablement son comportement caractéristique, ou du moins que les comportements des différents protocoles encapsulés restent distinguables (cette supposition a été confirmée par des manipulations expérimentales sur une petite base de données). Partant de ces hypothèses, les méthodes de classification statistique par apprentissage suivent un schéma en deux étapes, schématisées sur la figure 2.

- dans une *phase initiale d'apprentissage*, à partir d'une base de données de flux pour lesquels le protocole encapsulé est connu, un modèle de chacun des protocoles que l'on souhaite savoir reconnaître est construit (en utilisant les paramètres cités précédemment) ;
- durant la *phase de classification*, les flux sont interceptés en temps réel. Les paramètres de classification sont extraits, puis comparés aux modèles statistiques précédents. Le protocole encapsulé dans le flux est alors "décidé", ce qui permet d'appliquer la politique de sécurité (lever une alerte, loguer le flux, le bloquer, etc.).

2.2 Une méthode contournable, mais à quel prix !

Il est évidemment possible pour un attaquant de contourner un système reposant sur la méthode précédente : pour cela, il lui suffit de générer un flux encapsulé dont les propriétés statistiques en termes de tailles de paquets et de temps inter-paquets sont proches de celles d'un flux autorisé. On peut imaginer avoir recours à des outils tels que *TCPReplay* [23], pour "modeler" un flux et lui donner les propriétés souhaitées. Cependant, une telle manipulation nécessite une certaine maîtrise technique, puisque les logiciels d'encapsulation existants n'offrent pas cette possibilité [5].

D'autre part, imposer à un flux de données certaines propriétés sur la taille des paquets échangés et les temps séparant ces paquets entraîne des contraintes fortes sur les débits en émission et en réception ainsi que sur la latence du flux. Par exemple, il est quasiment

5. Tout au plus certains logiciels de P2P proposent-ils une fonction d'obfuscation de protocole, consistant à introduire de l'aléa dans les tailles des paquets et les délais entre paquets. Cette méthode ne permettra évidemment pas pour autant au flux généré d'être classé comme HTTP par les outils statistiques.

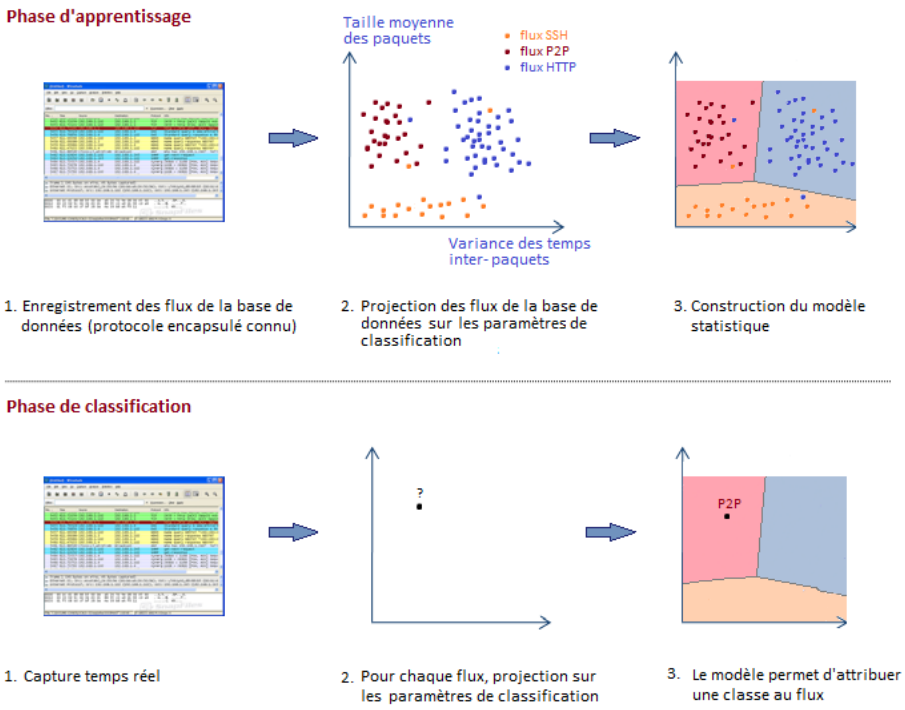

FIGURE 2. Principe de fonctionnement d'une méthode de classification par apprentissage

impossible de transformer un flux de type VoIP de telle manière qu'il "ressemble" à un flux HTTPs, tout en conservant une latence et un débit acceptables pour les utilisateurs. De même pour le protocole SSH, qui génère un très grand nombre de paquets client vers serveur rapprochés temporellement, ce qui est très éloigné du comportement normal d'un client HTTP.

S'il est donc possible de contourner un outil de détection de tunnels illégitimes basé sur une analyse statistique, ceci ne peut généralement être fait qu'au prix de lourds sacrifices sur le débit utile du protocole encapsulé. Une telle dégradation sera rédhibitoire pour de nombreuses applications. De plus la connaissance de l'existence d'un tel outil peut s'avérer dissuasive, par exemple par crainte de mesures disciplinaires envers l'utilisateur.

Notons en outre deux avantages non négligeables de cette méthode par rapport aux serveurs mandataires inspectant le contenu des paquets. D'une part, la classification statistique s'avère beaucoup plus rapide, ce qui limite l'engorgement des flux au niveau de la passerelle. D'autre part, les méthodes que l'on utilise ne nécessitent pas l'accès aux données applicatives des flux et ne posent donc pas de problème vis à vis du respect de la vie privé.

2.3 Rapide tour d'horizon de la littérature

L'utilisation de méthodes de classification statistiques pour identifier le protocole à l'origine d'un flux est un sujet abondamment étudié dans la littérature académique depuis 2005. De nombreux algorithmes statistiques ont été appliqués au sujet, avec des résultats souvent prometteurs. Cependant, ces études visent généralement des applications de type qualité de service (QoS) ou modélisation de flux réseau, et les problématiques de sécurité sont rarement traitées. On liste ci-dessous quelques-uns des travaux les plus représentatifs de la littérature.

Moore et Zuev [19] ont appliqué des méthodes *bayesiennes* à la classification des connexions TCP. Les résultats obtenus, bien que prometteurs (taux d'identification supérieur à 95%), sont biaisés par le fait que l'un des paramètres utilisé pour la classification est le numéro de port (ce qui est justement ce que l'on veut éviter dans notre cas).

Dans [18], Erman et Mahanti comparent les performances d'un algorithme d'apprentissage non supervisé (par "*clustering*" simple de type *k-means*) à celles d'un classificateur par la méthode naïve de Bayes. Cette fois, seuls 5 paramètres pour chaque connexion TCP sont utilisés (nombre de paquets, taille moyenne, durée moyenne de la connexion, etc.) pour classer les flots dans 9 classes de trafic (HTTP, SMTP, DNS, FTP, etc.). L'algorithme non supervisé donne des résultats sensiblement meilleurs (taux de classification correcte supérieur à 90%). Les mêmes auteurs présentent des résultats similaires dans [14].

L'article [25] constitue l'une des rares synthèses comparatives de plusieurs méthodes statistiques appliquées à la classification de flux de données. Williams montre que parmi de nombreux algorithmes

(méthode de Bayes, réseaux bayésiens, C4.5, arbres de Bayes, *Support Vector Machine*, etc.), les plus performants sont les SVM et C4.5.

D'autres études cherchent à caractériser les protocoles en se basant uniquement sur les premiers paquets échangés. C'est le cas de [6] et [13], dans lesquelles les auteurs utilisent les 5 premiers paquets de chaque connexion TCP pour en déterminer le protocole. Les outils statistiques utilisés sont des modèles de Markov cachés (HMM). Bien que performante et très rapide, cette méthode semble aisément contournable par un utilisateur mal intentionné qui simulerait un début de connexion normal.

Wright s'est intéressé à la classification des flux chiffrés dans [26]. Sa méthode, également basée sur l'utilisation de HMM, est applicable dans le cas où plusieurs connexions du même type sont ouvertes simultanément dans le même tunnel.

Enfin, Bursztein [9] a proposé une solution intéressante indiquant le protocole le plus probable correspondant à un flux, à partir de la combinaison de trois méthodes de classification (heuristique sur les numéros de ports, inspection du contenu du paquet, classification statistique du profil du flux). La classification statistique utilisée est cependant élementaire (utilisation de profils simples, avec moyenne et écart-type). Au contraire, notre étude se consacre uniquement à la classification statistique, et combine deux algorithmes afin d'en améliorer les performances. L'approche que nous proposons est donc complémentaire de celle de Bursztein.

Divers outils de classification statistiques ont donc été appliqués au problème qui nous intéresse. Malheureusement, les performances de ces différents algorithmes ne peuvent pas être comparées à la lecture de la littérature, chaque auteur utilisant une méthodologie, une base de données et des paramètres de classification différents.

3 Un outil de détection des tunnels illégitimes : CASTAFIOR

La classification statistique semble constituer l'évolution naturelle de l'état de l'art dans le domaine de la détection des tunnels illégitimes. Cependant, ces méthodes de classification ne sont pas en-

core intégrées dans les produits sur étagère et le sujet semble surtout avoir été traité par le monde de la recherche académique.

Afin de réaliser une preuve de concept de la technologie en adoptant un point de vue opérationnel, un outil de détection de tunnels a été développé en s'inspirant de l'état de l'art. Cet outil, appelé CASTAFIOR [6], combine de manière originale deux méthodes statistiques complémentaires :

- *l'algorithme RandomForest*, utilisé pour analyser le flux globalement, à l'échelle "macroscopique" ;
- *un banc de HMM (modèles de Markov cachés)*, utilisé pour exploiter l'information séquentielle du flux, c'est à dire les enchaînements de paquets au niveau "microscopique".

Cette section justifie le choix de ces méthodes et explique leur fonctionnement, ainsi que la façon dont elles sont combinées. L'architecture globale de CASTAFIOR est également présentée.

3.1 Analyse au niveau "macroscopique" : *RandomForest*

Afin de déterminer la méthode de classification la plus performante parmi celles décrites dans la littérature, ainsi que les paramètres de classification à extraire de chaque flux, une étude comparative a été menée à partir d'une base de données de l'université de Cambridge, décrite dans [20]. Cette base est constituée de 20 000 flux pour chacun desquels 250 paramètres ont été extraits. Le protocole à l'origine du flux est également indiqué.

L'ensemble de ces manipulations a été effectué en utilisant WEKA [24], une librairie de data mining *Java* développée par l'université de Waikato en Nouvelle Zélande.

Choix des paramètres de classification La première étape de la conception d'un système de classification consiste à déterminer quels sont les paramètres que l'on va utiliser pour classer les flux. Pour que ces paramètres soient évaluables quel que soit le flux TCP considéré, ils doivent tous être déduits des données contenues dans les couches 1 à 4 du modèle OSI. De plus, pour rendre le contournement du système plus ardu, on choisit des paramètres coûteux à modifier

6. CASTAFIOR : Classification par Apprentissage STAtistique des Flux et Identification du protocole d'ORigine

pour un attaquant (par exemple, on ne considère pas les flags TCP). Seuls les paramètres dérivés des tailles des paquets et des temps inter-paquets ont donc été retenus (par exemple : taille moyenne des paquets client vers serveur, variance des temps inter-paquets, etc.).

Parmi tous les paramètres encore envisageables, on souhaite n'en retenir qu'une dizaine pour assurer la rapidité de la classification. Le choix a été effectué par application d'un algorithme de sélection supervisée (*CFS-Subset+BestFirst*). Le principe de cet algorithme est de chercher un sous ensemble de paramètres avec un pouvoir discriminant maximal, tout en gardant une faible valeur d'intra corrélation.

Finalement, dix paramètres ont été sélectionnés : variance des tailles de paquets client vers serveur, temps minimal inter-paquets serveur vers client, nombre de paquets transmis client vers serveur, nombre d'octets transmis serveur vers client, etc. La figure 3 montre la projection des flux de la base de données sur trois de ces dix paramètres. On constate une forte dépendance vis à vis du protocole utilisé.

Choix de l'algorithme de classification Une fois les paramètres de classification déterminés, il s'agit de choisir l'algorithme à utiliser pour construire les modèles statistiques. Afin de déterminer l'algorithme le plus performant, six méthodes statistiques usuelles ont été comparées :

- méthode de Bayes naïve [27] ;
- arbre de décision C4.5 [7] ;
- forêt d'arbres de décision *RandomForest* [17] ;
- méthode *K-Means* [4] ;
- séparateurs à vaste marge (SVM) [21] ;
- modèle de mélange de gaussiennes (GMM) [12].

La méthodologie suivie pour chaque méthode consiste en une "*cross-validation*". On sépare la base de données en dix parties de même cardinaux, et on répète les opérations suivantes pour i variant de 1 à 10 :

- retirer le $i$ème ensemble de la base de données ;
- construire le modèle statistique (phase d'apprentissage) à partir des 9 ensembles restants de la base de données ;

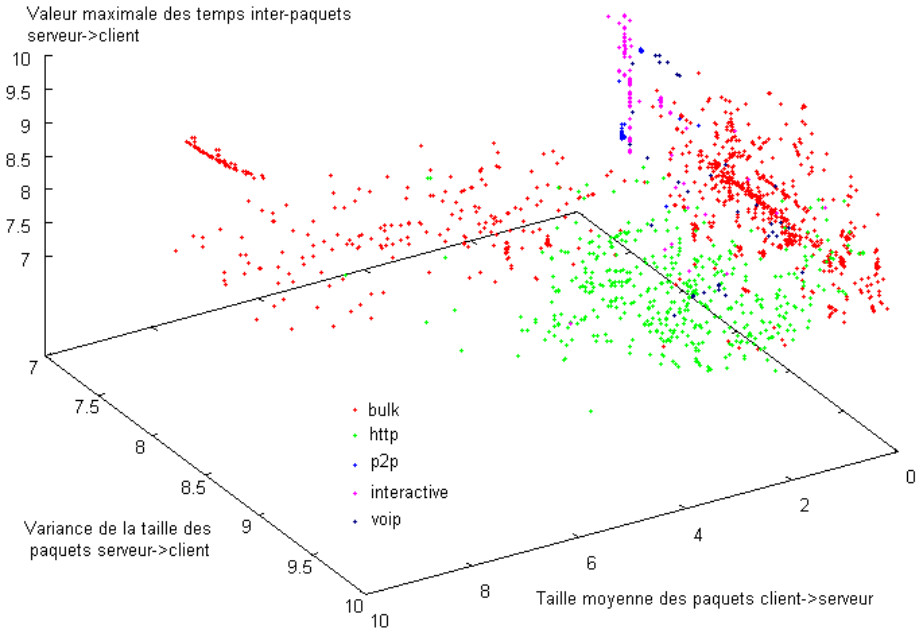

FIGURE 3. Projection des flux de la base sur trois des paramètres utilisés pour la classification (chaque point est un flux de la base de données)

– classer le $i$ème ensemble de flux de la base de données (phase de classification).

On peut ainsi évaluer pour chaque méthode la proportion de flux pour lesquels le protocole est correctement identifié. Les résultats obtenus sont représentés sur la figure 4.

On constate que l'algorithme *RandomForest* est celui qui donne les meilleurs résultats. C'est ce même algorithme qui s'est montré le plus robuste vis à vis de la taille de la base d'apprentissage, et il présente de bonnes performances en termes de complexité temporelle (un flux est classé en 20 microsecondes en moyenne par un programme Java non optimisé). Cet algorithme, qui n'avait jamais été appliqué jusqu'ici à la classification des flux, est celui que nous avons retenu par la suite. Il est brièvement décrit dans la section suivante.

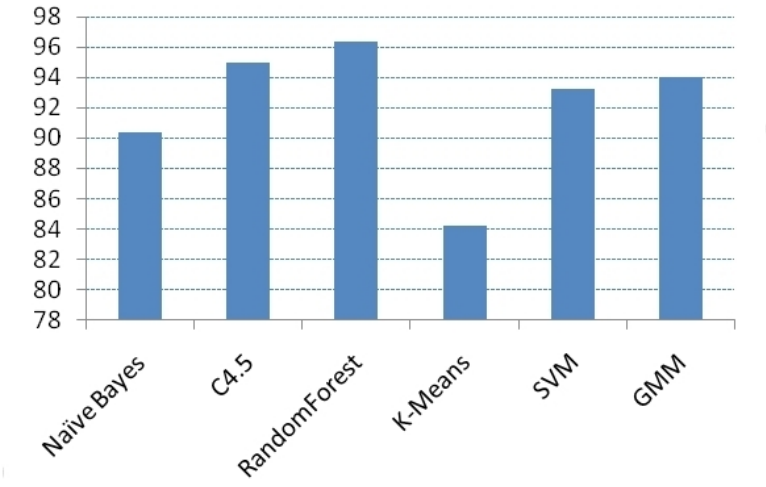

Figure 4. Pourcentage de flux bien classés obtenus pour six méthodes statistiques différentes

L'algorithme *RandomForest* L'algorithme *RandomForest* repose sur l'utilisation d'une forêt d'arbres de décisions aléatoires. Un exemple d'arbre de décision est présenté figure 5 : chaque nœud d'un tel arbre représente un test sur un des paramètres par rapport à une valeur discriminante, et chaque feuille de l'arbre représente un protocole. Pour classer un flux donné, il suffit de partir de la racine de l'arbre et de descendre dans les branches selon les résultats des tests. La feuille à laquelle on arrive est le résultat de la classification.

La difficulté de la méthode réside dans la construction de l'arbre de décision. Ceci se fait à partir de la base d'apprentissage, en utilisant un algorithme qui détermine récursivement pour chaque nœud le meilleur paramètre à utiliser et la valeur discriminante la plus pertinente (par minimisation de l'entropie inter-classes résultante).

L'algorithme *RandomForest* consiste à utiliser non pas un mais plusieurs arbres (une vingtaine en pratique) en introduisant un aléa lors de l'apprentissage de telle sorte que tous les arbres soient différents. Pour déterminer le protocole encapsulé dans un flux, on fait classer ce dernier par chacun des arbres de la forêt. Le pourcentage d'arbres ayant voté pour chacun des protocoles possibles est inter-

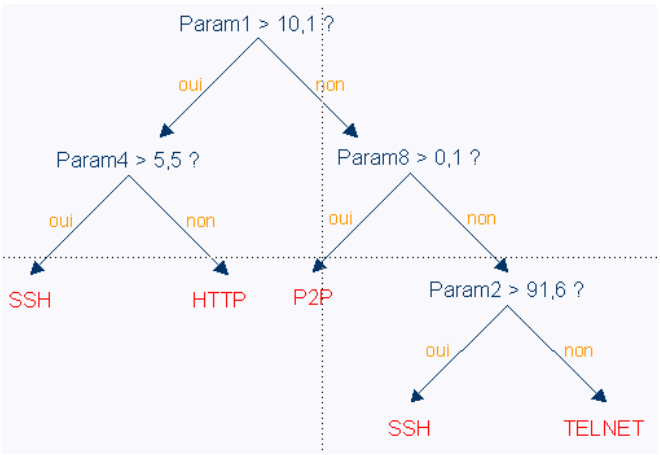

FIGURE 5. Un exemple d'arbre de décision

prété comme la probabilité que le flux appartienne à ce protocole. En particulier, le protocole choisi par une majorité d'arbres constitue le résultat de la classification.

3.2 Une méthode prenant en compte les séquences de paquets : les modèles de Markov cachés

La méthode de classification des flux *RandomForest* donne de bons résultats (voir partie 4). Cependant, elle utilise exclusivement les dix paramètres précédemment cités pour classer les flux. En particulier, comme ces paramètres sont des moyennes, variances, valeurs minimales ou maximales, calculés sur l'ensemble du flux, toute l'information concernant les enchaînements temporels de paquets est perdue. Or, la "signature" d'un protocole se retrouve également dans le déroulement temporel des échanges de données.

Ainsi, une frappe de touche en SSH sera presque systématiquement suivie d'un paquet "echo" du serveur. En revanche, en HTTP, une requête du client sera suivie de plusieurs paquets envoyés par le serveur, etc.

La figure 6 illustre ce principe. Pour chacun des paquets des flux de la base de données, sa position dans l'espace [taille du paquet]x[délai depuis le paquet précédent]x[rang du paquet dans le flux] est représentée. La répartition des points dépend clairement

du protocole utilisé, et on note une dépendance des deux paramètres taille/délai vis à vis du rang du paquet dans le flux. Cette illustration montre bien que les enchaînements temporels de paquets constituent également une signature du protocole utilisé.

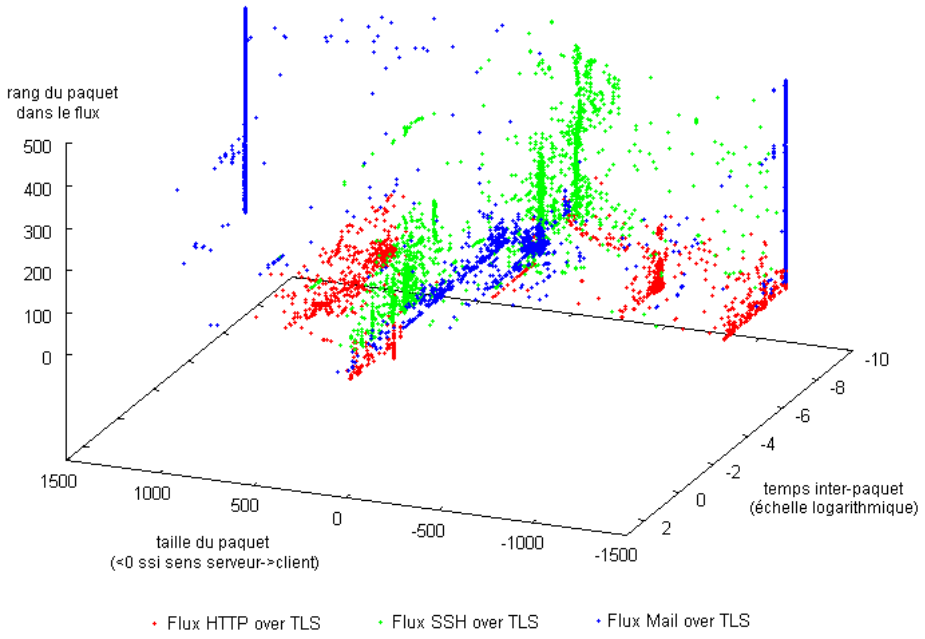

FIGURE 6. Représentation des tailles de paquets et des temps inter paquets en fonction du rang des paquets dans la connexion (chaque point est un paquet d'un flux)

Pour exploiter ces informations temporelles perdues par l'algorithme *RandomForest*, on utilise une autre méthode statistique, qui repose sur des modèles de Markov cachés (HMM). Grossièrement, un HMM est proche d'un automate à état. Chaque paquet du flux donne lieu à une transition dans l'automate, chaque transition ayant une certaine probabilité. A chaque état i de l'automate est associé une loi de probabilité $p_i(t, d)$ d'émission d'un paquet de taille t après un délai d. Le lecteur intéressé trouvera de plus amples informations sur les HMM dans [22].

En pratique, on utilisera un banc de HMM, c'est à dire que l'on construira un modèle de Markov caché pour représenter chaque protocole. Le "squelette" des HMM que l'on utilise est représenté sur la figure 7.

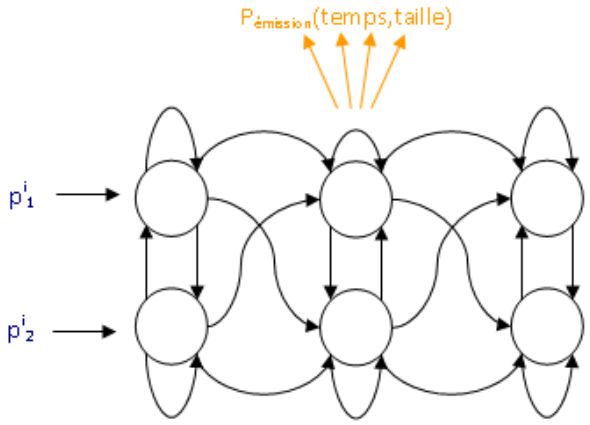

FIGURE 7. Représentation schématique des Modèles de Markov Cachés utilisés dans CASTAFIOR

Ainsi, en faisant l'hypothèse qu'un protocole est constitué d'une succession d' "états" pour lesquels les probabilités d'émission de paquets [taille, temps] sont à peu près constantes, chaque "colonne" des HMM utilisés représente un état protocolaire, et on autorise uniquement les transitions "vers la droite". Chaque HMM possède en outre deux états initiaux, situés sur la première colonne.

Étant donné les squelettes des HMM précédents, toutes les probabilités de transition et d'émissions de symboles sont calculées en utilisant la base de donnés d'apprentissage, en appliquant l'algorithme de Baum-Welsh.

Puisque l'on a construit un HMM pour chaque protocole que l'on souhaite savoir reconnaître, la probabilité qu'un flux appartienne au protocole i est donnée par la probabilité que ce flux ait été produit par le i^{eme} HMM (cette dernière est calculée par l'algorithme de Viterbi). Pour classer un flux, on cherche donc quel HMM donne la plus grande probabilité d'émission pour ce flux.

3.3 Combinaison des deux méthodes

L'outil de classification des flux CASTAFIOR qui a été développé intègre les deux algorithmes décrits précédemment. Une classification en parallèle des flux par ces deux méthodes permet de prendre en compte un maximum d'informations pour déterminer le protocole encapsulé. Le taux de bonne classification des flux devrait ainsi être amélioré, et le contournement du système par un attaquant rendu plus complexe.

Les deux méthodes permettent de déterminer la probabilité qu'un flux donné appartienne à chacun des protocoles possibles. Il s'agit maintenant de combiner les probabilités renvoyées par chacune des méthodes pour en déduire un résultat final de classification et un indice de confiance sur ce résultat.

Etant donné un flux F, et un ensemble de protocoles possibles $a_1...a_n$, on dispose des deux vecteurs de probabilités $[p_1...p_n]$ et $[p'_1...p'_n]$, où p_i et p'_i représentent respectivement la probabilité que F appartienne au protocole a_i selon la méthode *RandomForest* et selon la méthode du banc de HMM.

Les résultats de classification des deux méthodes sont fusionnés par la formule de Bayes :

$$p''_i = \frac{\sum_{k=1}^{n} p_k.p(i \mid k) + \sum_{k=1}^{n} p'_k.p'(i \mid k)}{2}$$

où

- p''_i est la probabilité fusionnée que F appartienne au protocole a_i ;
- $p(i|k)$ (respectivement $p'(i|k)$) est la probabilité que le flux appartienne au protocole a_i sachant que la méthode *RandomForest* (respectivement la méthode du banc de HMM) a classé le flux comme appartenant au protocole a_k. [7]

On obtient ainsi un vecteur de probabilité $[p''_1...p''_n]$. On note i_{max} la valeur de i telle que $p''_{i_{max}}$ est maximal. Le flux est alors classé comme appartenant au protocole $a_{i_{max}}$, et l'indice de confiance associé est $p''_{i_{max}}$.

7. Ces probabilités peuvent être estimées par des manipulations sur la base de données d'apprentissage

3.4 Architecture globale de l'outil développé

La figure 8 présente l'architecture générale de CASTAFIOR. L'écoute du réseau s'effectue à l'aide du programme *TCPDump* [3]. Des scripts *perl* effectuent ensuite le démultiplexage des différents flux interceptés, et l'extraction des paramètres utiles à la classification. Les outils de classification proprement dits sont développés en *Java*, autour des librairies *Weka* [24] (pour *RandomForest*) et *JaHMM* [15] (pour le banc de HMM).

De plus, pour limiter le nombre de faux positifs [8], on applique un certain nombre d'heuristiques à l'historique des résultats de classification. Une heuristique peut ainsi consister à ne pas lever d'alertes si l'indice de confiance sur la classification est trop faible, ou si l'hôte dont il s'agit a toujours eu un comportement irréprochable auparavant, etc. L'application de ces heuristiques génère un rapport d'analyse, mis à jour en temps réel, contenant les alertes levées et leur niveau de criticité. Ce rapport peut être mis au format *Syslog* par exemple, pour des raisons d'interopérabilités.

4 Résultats expérimentaux

L'outil CASTAFIOR précédemment décrit a été mis en œuvre sur des bases de données de flux, pour évaluer ses performances dans un contexte opérationnel. L'objectif des manipulations effectuées est d'estimer la faisabilité en pratique de la technologie. Ainsi, on cherche à répondre aux question opérationnelles suivantes :

- Quel taux d'identification correcte des protocoles encapsulés peut on espérer obtenir ?
- Quel taux de faux positifs doit on envisager ?
- Combien de flux doit contenir la base d'apprentissage pour obtenir de bons résultats ?
- Combien de paquets d'un flux faut-il prendre en compte avant de le classer, pour obtenir un résultat de classification fiable ?
- Combien de temps dure la phase d'apprentissage ? Quelle est la durée moyenne de la classification d'un flux ?

8. Les faux positifs sont les véritables flux HTTPs qui sont classés de manière erronée comme tunnels illégitimes.

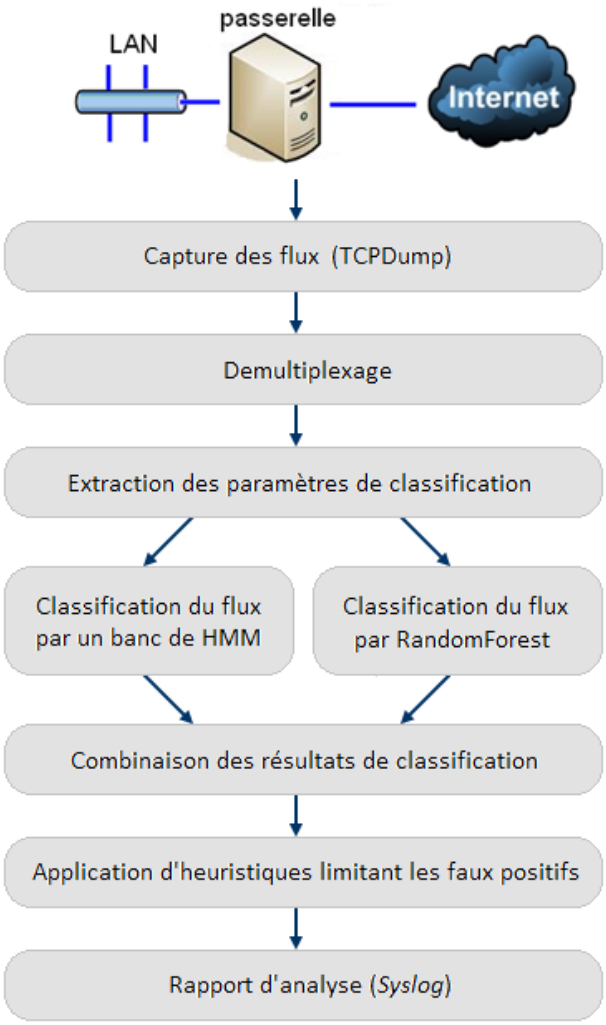

FIGURE 8. Architecture globale du système de classification de flux CASTAFIOR

4.1 Mise en œuvre sur une plateforme de test

En guise de manipulations préliminaires, l'outil CASTAFIOR a tout d'abord été installé sur une plateforme de taille réduite, composée de trois machines physiques *Debian* reliées par un switch *CISCO*. Ces machines représentent respectivement un réseau local, une passerelle et Internet. L'outil *Stunnel* est utilisé pour encapsuler

l'intégralité des flux échangés entre le "réseau local" et "internet" dans un tunnel TLS. Le rôle de la passerelle, sur laquelle est installé CASTAFIOR, est d'identifier à la volée le protocole à l'origine des flux capturés.

Une base d'apprentissage de quelques dizaines de flux a été générée manuellement sur cette plateforme, pour quelques protocoles (encapsulés dans TLS) : SSH, TELNET, HTTP, POP3, SMTP, SCP et IMAP. Dans un deuxième temps, des flux de données (toujours encapsulés) de différents protocoles ont été joués manuellement, le résultat de la classification par la "passerelle" étant observé en temps réel.

Le taux de bonne identification du protocole encapsulé obtenu lors de ces manipulations est de l'ordre de 100%. Cependant, ce résultat encourageant ne peut à lui seul constituer une preuve de concept valide. Tout d'abord, le nombre de flux générés est limité, ce qui diminue la confiance qu'on peut accorder au résultat obtenu. D'autre part, la plateforme utilisée constitue un environnement idéal, dans lequel les flux générés ne sont pas représentatifs de la diversité des flux d'un réseau de grande taille (pas de congestion réseau, un seul OS, un seul utilisateur, etc.).

Pour connaître le comportement de l'outil de classification face à des données plus réalistes, il convient donc de l'appliquer à une base de données de grande taille constituée de flux réels. La section qui suit présente la base de données qui a été utilisée par la suite.

4.2 Description de la base de données utilisée

Les manipulations des sections suivantes sont effectuées sur une base de données publique de flux réels fournie par le groupe de travail MAWI [1]. Cette base contient un enregistrement de l'intégralité des paquets transitant sur une ligne transpacifique Japon/USA de 150Mb/s, pendant une durée de 96 heures (du 30/03/2009 au 02/04/2009). Les données fournies sont anonymisées et les données applicatives ont été retirées.

Parmi l'ensemble des flux capturés, on ne garde que les connexions TCP sur Ipv4 entières. De plus, un échantillonnage de la base est effectué, afin de ne conserver qu'un nombre de flux raisonnable, et d'homogénéiser les effectifs des différents protocoles présents (sans

cela, la classe HTTP serait surreprésentée). On écarte également les protocoles trop peu représentés.

Afin de pouvoir quantifier les performances de l'outil de classification, il est nécessaire d'étiqueter la base de données, c'est à dire de connaître le protocole ayant généré chacun des flux. Puisque les données applicatives ne sont pas accessibles, les numéros de port sont utilisés pour étiqueter les flux[9]. Cette méthode empêche malheureusement d'identifier les protocoles utilisant des numéros de port dynamique (canal de données de FTP, P2P, etc.). Finalement, on a conservé uniquement les flux utilisant un des 9 numéros de port suivant, que l'on associe à leurs protocoles standards associés :

- port 80 : protocole HTTP ;
- port 443 : protocole HTTPs ;
- port 22 : protocole SSH ;
- port 25 : protocole SMTP ;
- port 53 : protocole DNS (transporté par TCP dans notre cas) ;
- port 22 : protocole FTP (canal de contrôle) ;
- port 445 : protocole *Active Directory* (service d'annuaire et de distribution de logiciels Windows) ;
- port 995 : protocole POP3 sécurisé ;
- port 2128 : protocole *Active Net Steward*, canal de contrôle (solution de sécurité réseau distribuée).

Le nombre de flux contenus dans la base de données pour chacun de ces protocoles apparaît dans le tableau 1.

Table 1. Répartition des flux de la base de données suivant les différents protocoles

Protocole	HTTP	HTTPs	SSH	SMTP	DNS	FTP	*Active Directory*	POP3s	*Net Steward*
Nombre de flux	2500	2500	2500	2500	2500	2500	1069	1503	1611

9. Ce choix, fait par défaut, n'invalide pas la démarche. En effet, l'immense majorité des flux capturés utilisent les numéros de ports standards. Par la suite, les numéros de ports ne sont évidemment pas utilisés pour la classification. La majorité des articles de la littérature utilisent également cette méthode d'étiquetage.

On remarquera que les flux utilisés pour ces manipulations sont pour la plupart des flux "clairs" non chiffrés. En effet, il n'existe malheureusement pas de base de données publique de flux chiffrés indiquant pour chaque flux le protocole l'ayant généré. Mettre en œuvre CASTAFIOR sur une base de flux clairs a néanmoins du sens, pour les raisons suivantes :

- les paramètres de classification utilisés par CASTAFIOR sont calculables indépendamment du fait que le flux soit chiffré ou non, puisque la classification n'utilise pas le contenu des paquets ;
- si ces flux étaient encapsulés dans un tunnel TLS, les paramètres de classification seraient légèrement modifiés par l'encapsulation. Cependant, le problème de la distinction des différents protocoles à partir des tailles des paquets et des temps inter-paquets resterait exactement le même. Une étude complémentaire, non présentée ici faute de place, a montré que l'impact de l'encapsulation TLS sur 8 des 10 paramètres utilisés ne devrait pas affecter les performances de classification ;
- les résultats obtenus sur les flux clairs peuvent être interprétés comme une borne supérieure des résultats qui peuvent être obtenus sur les flux chiffrés [10] ;
- Les manipulations préliminaires effectués sur la plateforme de test, avec des flux encapsulés dans TLS se sont montrées concluantes.
- certains résultats sont intéressants indépendamment du fait qu'ils soient obtenus sur des flux clairs ou chiffrés : le temps mis par CASTAFIOR pour classer un flux, les performances comparées des méthodes *RandomForest*, banc de HMM et de la méthode combinée CASTAFIOR, etc.

La suite de cette partie présente les résultats des manipulations effectuées sur cette base de données. En particulier, on présentera souvent par la suite les résultats obtenus en utilisant la méthode *RandomForest* seule, la méthode basée sur le banc de HMM seule, et la méthode combinant les deux précédentes, appelée CASTAFIOR.

10. Le chiffrement ayant *a priori* tendance à homogénéiser légèrement les paramètres, par exemple les tailles des paquets échangés qui seront arrondies à la taille du bloc de l'algorithme de chiffrement.

Ceci permet de mettre en évidence l'intérêt de la combinaison originale de ces deux algorithmes.

4.3 Résultats de classification obtenus

Pour évaluer les performances du système de classification, on utilise comme précédemment une méthode de *cross validation* avec 10 ensembles. Ainsi, tous les flux de la base de données sont classés une fois, sans jamais que l'apprentissage ait été fait sur les flux que l'on classe. La phase d'apprentissage est effectuée ici en utilisant 300 flux pour chacun des protocoles.

Les résultats obtenus sont présentés dans les tableaux 2, 3 et 4 sous forme de matrices de confusions [11].

Table 2. Résultats de la classification de la base de données par CASTAFIOR (méthode combinée)

HTTP	*HTTPs*	*SSH*	*SMTP*	*DNS*	*FTP*	*Act.Dir.*	*POP3s*	*NetStew*	
94.88	2.84	0.0	0.6	0.2	0.16	0.12	0.52	0.68	*HTTP*
1.72	**94.44**	0.0	1.8	0.12	0.4	0.44	0.92	0.16	*HTTPs*
0.0	0.0	**99.56**	0.2	0.0	0.0	0.08	0.16	0.0	*SSH*
0.0	1.72	0.0	**89.64**	0.08	3.32	4.0	0.4	0.66	*SMTP*
0.0	0.0	0.0	0.24	**99.68**	0.04	0.04	0.0	0.0	*DNS*
0.12	0.64	0.0	2.64	0.24	**94.96**	0.76	0.4	0.24	*FTP*
0.0	0.28	0.0	0.56	0.0	0.0	**99.16**	0.0	0.0	*ActiveDirectory*
0.67	1.06	0.0	0.67	0.0	0.06	0.0	**97.54**	0.0	*POP3s*
0.74	0.0	0.0	0.0	0.07	0.0	0.0	0.0	**99.19**	*NetSteward*

11. Les matrices de confusion se lisent de la manière suivante : pour la méthode combinée CASTAFIOR, 2.84% des flux appartenant au protocole HTTP ont été classés de manière erronée par l'algorithme dans la catégorie HTTPs. Les taux de bonne classification se lisent donc sur la diagonale.

Table 3. Résultats de la classification de la base de données par la méthode *RandomForest* seule

HTTP	*HTTPs*	*SSH*	*SMTP*	*DNS*	*FTP*	*Act.Dir.*	*POP3s*	*NetStew*	
93.08	4.36	0.0	1.08	0.04	0.24	0.08	0.36	0.76	*HTTP*
2.16	**91.36**	0.08	3.1	0.0	0.38	0.48	2.16	0.28	*HTTPs*
0.0	0.12	**99.44**	0.08	0.0	0.08	0.0	0.28	0.0	*SSH*
0.96	2.28	0.0	**91.12**	0.2	3.48	1.12	0.72	0.12	*SMTP*
0.0	0.0	0.0	0.32	**99.64**	0.0	0.04	0.0	0.0	*DNS*
0.08	0.6	0.0	3.0	0.0	**95.88**	0.2	0.24	0.0	*FTP*
0.19	0.09	0.0	0.47	0.0	0.0	**99.16**	0.0	0.09	*ActiveDirectory*
0.13	1.2	0.27	0.73	0.0	0.0	0.0	**97.67**	0.0	*POP3s*
1.37	0.06	0.0	0.0	0.0	0.0	0.0	0.0	**98.57**	*NetSteward*

Table 4. Résultats de la classification de la base de données par la méthode du banc de HMM seule

HTTP	*HTTPs*	*SSH*	*SMTP*	*DNS*	*FTP*	*Act.Dir.*	*POP3s*	*NetStew*	
85.08	7.88	0.0	0.64	0.44	0.28	1.0	1.48	3.2	*HTTP*
2.04	**92.68**	0.0	2.0	0.12	0.72	0.44	1.96	0.04	*HTTPs*
0.0	0.0	**99.44**	0.32	0.0	0.04	0.08	0.12	0.0	*SSH*
0.0	2.48	2.48	**72.16**	0.0	15.08	5.12	1.44	1.24	*SMTP*
0.0	2.12	0.0	0.16	**94.44**	0.04	2.28	0.2	0.76	*DNS*
0.12	0.88	20.32	7.76	0.028	**67.12**	2.2	0.88	0.44	*FTP*
0.0	0.56	0.0	2.06	0.19	0.65	**96.35**	0.09	0.09	*ActiveDirectory*
0.53	6.25	0.0	2.0	0.0	0.07	0.0	**91.15**	0.0	*POP3s*
0.93	1.55	0.0	0.12	0.93	0.5	0.74	0.0	**95.22**	*NetSteward*

On constate que CASTAFIOR donne des résultats légèrement meilleurs que chacune des deux autres méthodes prises indépendamment, ce qui confirme l'intérêt de cette approche multi niveaux.

Le taux d'identification correcte par CASTAFIOR du protocole de chaque flux est en moyenne de 96.19%, contre 95.81% pour *RandomForest* seul, et seulement 87.10% pour la méthode du banc de HMM.

Si l'on se place dans le cas d'un réseau classique où le filtrage de la passerelle autorise uniquement les flux HTTP et HTTPs à traverser, alors d'après les résultats précédents l'outil CASTAFIOR détecte au total 99.27% des flux illégitimes (contre 98.68% pour *RandomForest*, et 97.96% pour le banc de HMM). Les 0.73% de flux illégitimes classés comme HTTP ou HTTPs par l'outil constituent les "faux négatifs".

D'autre part, le taux de faux positifs, i.e. de flux HTTP ou HTTPs classés comme dangereux, sera en moyenne de 3.06% avec CASTAFIOR (contre 4.72% pour *RandomForest*, et 5.76% pour le banc de HMM). Ce taux de faux positifs peut être considéré comme encore trop élevé pour une utilisation en pratique, étant donné que l'immense majorité des flux traversant la passerelle seront des flux légitimes. Cependant, il devrait être possible de diminuer ce taux en utilisant la brique "application d'heuristiques" mentionnée précédemment (et non utilisée pour les manipulations présentées ici).

Notons néanmoins que la combinaison des deux méthodes permet une réduction de 35% du nombre de faux positifs par rapport à l'algorithme *RandomForest* seul.

4.4 Influence de la taille de la base d'apprentissage sur les performances

Une des principales difficultés de mise en œuvre des méthodes décrites dans cet article réside dans la nécessité de disposer d'une base de flux pour l'apprentissage. En effet, les performances obtenues lors de la classification seront nettement meilleures si la base d'apprentissage a été constituée sur le réseau où l'outil est utilisé. La constitution d'une telle base, à la charge du responsable du réseau, peut vite devenir fastidieuse si un grand nombre de flux est nécessaire pour chaque protocole.

Dans cette section, on fait varier le nombre de flux de chaque classe utilisés pour l'apprentissage et on trace l'évolution des performances de CASTAFIOR en fonction de ce paramètre.

En particulier, les figures 9, 10 et 11 montrent l'influence de la taille de la base d'apprentissage sur le taux de bonne classification, le taux de faux positifs ainsi que le taux de faux négatifs.

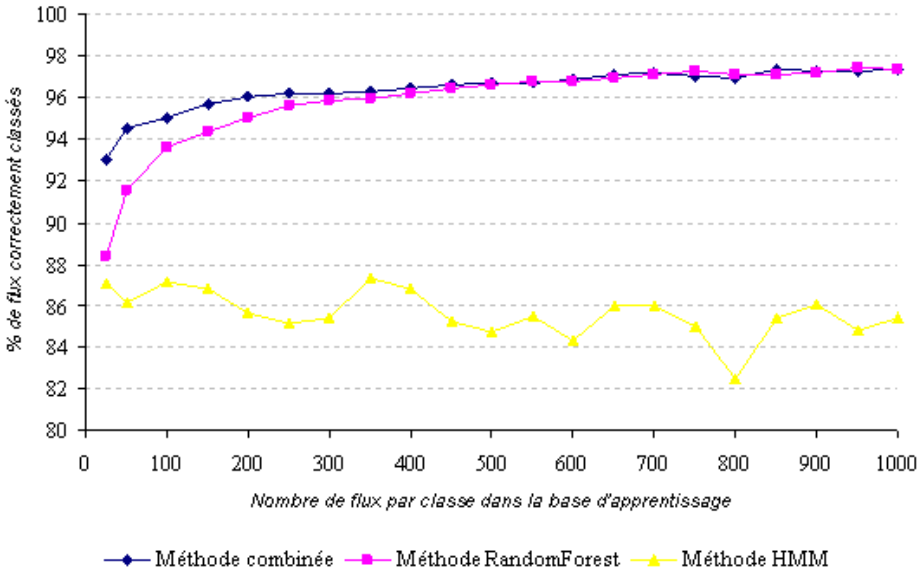

FIGURE 9. Evolution du taux de flux correctement classés en fonction de la taille de la base d'apprentissage (en nombre de flux par classe)

On constate que la méthode combinée CASTAFIOR est nettement plus performante que les deux autres méthodes lorsque l'apprentissage se fait sur des bases de données réduites.

D'autre part, pour des bases de données d'apprentissage importantes, les taux de bonne classification des méthodes CASTAFIOR et *RandomForest* sont proches, mais la méthode CASTAFIOR offre un taux de faux positifs inférieur. Ceci est dû au fait que CASTAFIOR commet de nombreuses erreurs de classification entre les flux HTTPs et HTTP (les deux classes étant similaires), mais ces erreurs n'affectent pas le taux de faux positifs (puisque ces deux classes sont autorisées).

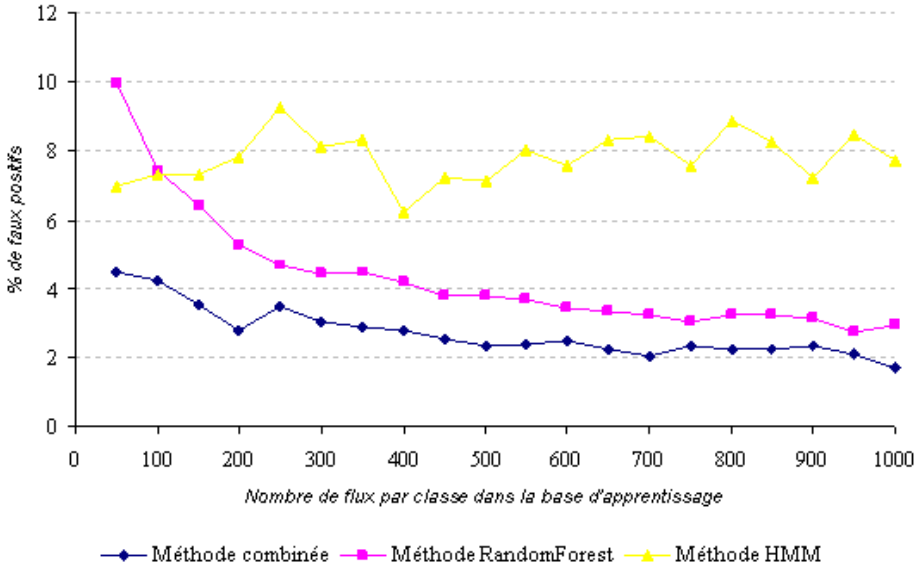

FIGURE 10. Evolution du taux de faux positifs en fonction de la taille de la base d'apprentissage

4.5 Influence du nombre de paquets pris en compte sur les performances

Pour l'ensemble des manipulations précédentes, pour chaque flux, l'intégralité de la connexion TCP était utilisée pour le calcul des paramètres et la classification. Cette approche est cohérente si l'objectif est de générer *a posteriori* un rapport contenant la liste des flux susceptibles d'être des tunnels non légitimes.

On peut également imaginer l'utilisation d'un tel outil dans un but préventif (et non plus seulement de détection). Dans ce cas, la classification doit être faite avant la fin de la connexion, pour pouvoir interrompre cette dernière le cas échéant. Par exemple, l'outil peut attendre qu'un nombre fixe de paquets soient échangés avant de prendre une décision de classification.

Dans cette section, on se place dans le cadre où seuls les n premiers paquets de chaque flux sont utilisés pour le calcul des paramètres et la classification. La figure 12 montre les performances globales obtenues en fonction de n.

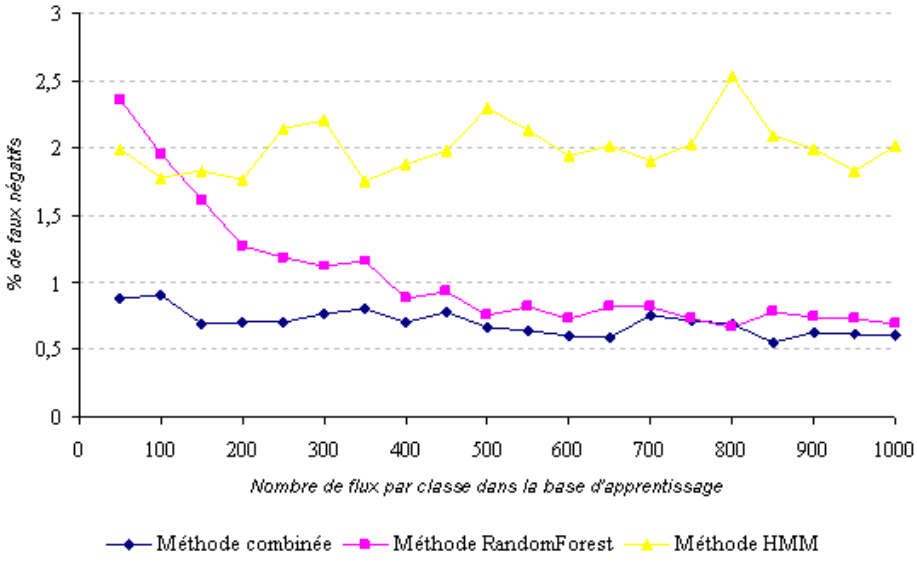

FIGURE 11. Evolution du taux de faux négatifs en fonction de la taille de la base d'apprentissage

Étonnamment, on constate que le nombre de paquets pris en compte affecte peu les performances de classification. Cela s'explique vraisemblablement par le fait que les différents protocoles considérés ont des comportement différents dès le début de la connexion, ce qui permet de les distinguer à partir d'un petit nombre de paquets échangés.

4.6 Performances temporelles

Une condition importante pour que l'outil puisse être mis en œuvre en pratique est qu'il soit suffisamment rapide pour pouvoir traiter les flux à la volée pendant la phase de classification. La durée de la phase d'apprentissage importe moins puisque cette étape se fait une seule fois, à l'installation de l'outil, sans contrainte de temps.

On présente dans le tableau ci-dessous la durée moyenne de classification d'un flux par CASTAFIOR (calculée à partir de la classification de l'ensemble de la base de données), ainsi que le temps de construction des modèles statistiques (phase d'apprentissage).

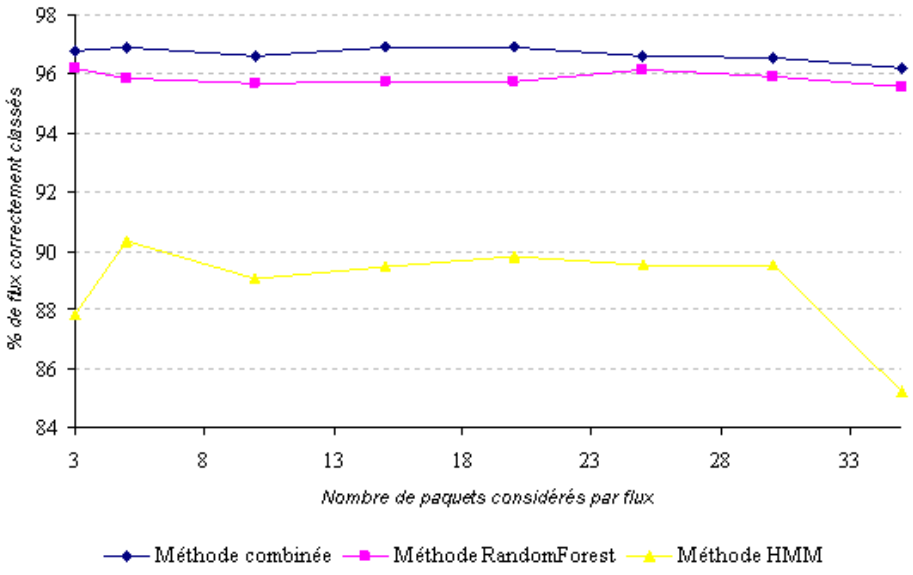

Figure 12. Evolution du taux de flux correctement classés en fonction du nombre de paquets utilisés pour la classification de chaque flux

Ces résultats sont surtout donnés à titre indicatif car dans le cadre de cette preuve de concept, les programmes utilisés sont écrits en Java et ne sont pas optimisés en termes de performances. La machine utilisée pour ces manipulations est une machine Debian (*Lenny 5.0.2*) équipée de deux cœurs Pentium 4 cadencés à 3.06GHz. La JVM utilisée est *Open-JDK Runtime Environment 1.6.0.*

Table 5. Durée de l'apprentissage et de la classification pour chacune des méthodes

Méthode	*RandomForest* seul	Banc de HMM seul	CASTAFIOR
Durée de la phase d'apprentissage (2500 flux)	1 143 ms.	41 106 ms.	44 063 ms.
Durée moyenne de classification d'un flux	223 μs.	201 μs.	427 μs.

La méthode complète CASTAFIOR est logiquement celle qui nécessite le plus de temps pour l'apprentissage et la classification, puisqu'elle intègre les deux autres algorithmes. Cependant, on constate que le temps d'apprentissage reste très raisonnable (moins d'une minute) et que la classification d'un flux se fait très rapidement (une demi-milliseconde), par rapport à la durée moyenne d'une connexion TCP (de l'ordre de la seconde). On peut imaginer que ces performances seront largement améliorées par un programme utilisant un langage bas niveau, et tournant sur une machine performante.

Par ailleurs, l'implémentation actuelle de CASTAFIOR effectue séquentiellement la classification *RandomForest* puis la classification par banc de HMM avant de fusionner les résultats. Les deux algorithmes pourraient être appliqués en parallèle, puisqu'ils sont totalement indépendants. Ceci ramènerait théoriquement le temps d'exécution aux alentours de 250μs, sur une machine multi-coeurs.

5 Conclusion

Face aux déficiences des outils de sécurité actuels en termes de détection des tunnels illégitimes, certains chercheurs du monde académique proposent d'utiliser des méthodes d'analyse statistique des flux pour déterminer le protocole encapsulé dans un tunnel chiffré. L'étude présentée dans cet article constitue une preuve de concept de cette nouvelle technologie.

Ainsi, un outil de détection des tunnels mettant en œuvre de façon originale deux algorithmes de classification statistiques a été décrit. Cet outil a été implémenté, puis appliqué à des bases de données de flux réels. Les résultats obtenus sont encourageant puisque pour plus de 96% des flux, le protocole encapsulé est correctement identifié et la politique de sécurité peut être appliquée. En revanche, le taux de faux positifs reste élevé (de l'ordre de 3%), ce qui peut gêner les utilisateurs légitimes. Des techniques permettant de réduire ce taux ont été proposées. Les performances temporelles de la méthode, bien qu'indicatives, semblent également prometteuses.

Le sujet traité étant vaste, cette étude ne peut être exhaustive. En particulier un certain nombre de réserves peuvent être émises

sur la technologie, et mériteraient d'être traitées par des travaux complémentaires :

- la constitution de la base d'apprentissage permettant la construction des différents modèles est fondamentale. Les auteurs ont constaté empiriquement que les performances des outils de classification sont fortement dégradées si cette base n'a pas été enregistrée sur le réseau où l'outil est utilisé. L'étude de méthodes permettant de réduire cette dépendance au site, ou le développement d'un outil permettant la constitution rapide d'une base de données apporteraient un éclairage sur ce point ;
- le comportement d'un outil de classification statistique face à un flux d'un protocole inconnu dans sa base d'apprentissage est imprévisible. Une modification de l'algorithme de classification pour détecter statistiquement un tel flux devrait cependant être possible ;
- une étude complémentaire portant sur les moyens de contournement de ce type d'outil est nécessaire. En particulier, il s'agit d'évaluer les contraintes réelles sur le débit utile et la latence induites par ce contournement.

Références

1. Mawi working group traffic archive. http://mawi.wide.ad.jp/mawi.
2. Stunnel universal ssl wrapper. http://www.stunnel.org.
3. Tcpdump/libpcap. http://www.tcpdump.org.
4. M. Aizerman, E. Braveman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control 25*, pages 821–837, 1964.
5. Alcatel-Lucent. Brevet wo 2009/021892 a1, procédé et installation de classification de trafics dans les réseaux ip, février 2009.
6. L. Bernaille and R. Texeira. Early recognition of encrypted applications. In *PAM2007*, 2007.
7. L. Breiman. Random forests. In *Machine Learning 45 (1)*, pages 5–32, 2001.
8. L. Brinkhoff. Gnu httptunnel. http://www.nocrew.org/software/httptunnel.html.
9. E. Bursztein. Probabilistic identification for hard to classify protocol. In *WISTP'08*, 2008.
10. A. Dainotti, W. de Donato, A. Pescape, and P. Salvo Rossi. Classification of network traffic via packet-level hidden markov models. In *IEEE GLOBECOM'08*, 2008.

11. Centre d'Expertise Gouvernementale de Réponse et de Traitement des Attaques Informatiques. Tunnels et pare-feux : une cohabitation difficile. In *note d'information CERTA-2001-INF-003-001*, 2001.
12. I. Dinov. Expectation maximization and mixture modeling tutorial, california digital library. In *Statistics Online Computational Resource, Paper EM_MM*, 2008.
13. M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Detection of encrypted tunnels across networks boundaries. In *ICC2008*, 2008.
14. J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *ACM SIGCOMM'06*, 2006.
15. J.M. François. Jahmm, hidden markov model : an implementation in java. http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm.
16. W. Li, M. Canini, A.W. Moore, and R. Bolla. Efficient application identification and the temporal and spatial stability of classification schema. In *Computer Networks 53*, 2009.
17. S.P. Lloyd. Least square quantization in pcm. In *IEEE Transactions on Information Theory 28 (2)*, pages 129–137, 1982.
18. A. Mahanti, J. Erman, and M. Arlitt. Internet traffic identification using machine learning. In *IEEE GLOBECOM'06*, 2006.
19. A.W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS'05*, 2005.
20. A.W. Moore, D. Zuev, and M.L. Crogan. Discriminators for use in flow-based classification. Technical report, Université de Cambridge, 2008.
21. J.R. Quinlan. C4.5 : Programs for machine learning. In *Morgan Kaufmann Publishers*, 1993.
22. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE 77 (2)*, pages 257–286, 1989.
23. Aaron Turner. Tcpreplay. http://tcpreplay.synfin.net/trac.
24. université de Waikato. Weka 3 : data mining software in java. http://www.cs.waikato.ac.nz/~ml/weka/index.html.
25. N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. In *ACM SIGICOMM'06*, 2006.
26. C.V. Wright, F. Monrose, and G.M. Masson. Toward better protocol identification using profile hmms. In *Technical report*, 2005.
27. H. Zhang. The optimality of naive bayes. In *FLAIRS conference*, 2004.