Filtrage Efficace Pour la Détection d'Intrusions

Tarek Abbes, Adel Bouhoula, Michaël Rusinowitch

Filtrage Efficace Pour la Détection d'Intrusions

Tarek Abbes et Adel Bouhoula et Michaël Rusinowitch

LORIA, 615, Rue du Jardin Botanique, BP 101, 54602 Villers les Nancy Cedex
SUP'COM, Cité Technolgique des Communications 2083 Ariana Tunisie

Les systèmes de détection d'intrusions (IDS) sont indispensables pour compléter la protection des pares-feux. Cependant, ils ont des difficultés à traiter le haut débit et à mener parallèlement une analyse précise sur le contenu des paquets. Nous proposons dans cette article une nouvelle approche pour analyser le trafic réseau. Nous nous appuyons sur une organisation intelligente des règles de détection et sur des algorithmes de recherche simultanée de signatures. Cette méthodologie a été implantée dans le système de détection d'intrusions "Snort".

Mots clés: Détection d'Intrusions, Evasion, Snort, Filtrage

1 Introduction

Un syst`eme de d´etection d'intrusions analyse l'activit´e interne et externe du r´eseau et identifi e les sc´enarios d'attaques susceptibles de compromettre l'int´egrit´e du syst`eme informatique. Son d´eploiement est indispensable afi n de prot´eger convenablement le r´eseau et combler les lacunes des pares-feux. Il existe deux grandes familles de m´ethodes de d´etection d'intrusions. La premi`ere famille proc`ede par analyse comportementale (anomaly detection). Elle consiste `a mod´eliser les comportements normaux du syst`eme et ses utilisateurs. Une d´eviation par rapport au comportement normal peut r´ev´eler une attaque. Le nombre de fausses alertes engendr´ees est important mais la m´ethode peut r´ev´eler de nouvelles attaques non connues auparavant. Les techniques g´en´eralement employ´ees pour mod´eliser le comportement se fondent sur des outils statistiques et des algorithmes d'apprentissage automatique.

La deuxi`eme famille s'appelle "analyse par sc´enario" ou "misuse detection" et elle stocke dans des bases de donn´ees les signatures des attaques connues. Le nombre de faux positifs est diminu´e consid´erablement par rapport `a la premi`ere m´ethode mais cette technique se restreint aux attaques d´ej`a connues. Les syst`emes de d´etection d'intrusions (IDS) analysent les fi chiers d'audit pour retrouver les instances des sc´enarios connus. Les techniques d'analyse actuelles utilisent le fi ltrage (pattern matching), les syst`emes experts, les r´eseaux de P´etri, les algorithmes de "model-checking", etc.

Les IDS forcent les attaquants `a compliquer leurs plans d'attaques afi n de camoufler leurs activit´es malveillantes. Actuellement, ceux-ci envoient les paquets en d´esordre et exploitent le comportement des syst`emes d'exploitation `a g´erer les paquets superpos´es afi n de construire des suites de paquets dont la reconstitution conduit `a une attaque. Ils emploient par ailleurs des techniques d'insertion qui consiste `a envoyer des paquets qui seront ignor´es par l'hôte. Une autre technique d'´evasion consiste `a envoyer une attaque sur plusieurs paquets mais dont certains seront ignor´es par l'IDS.

Nous nous sommes int´eress´es dans cet article `a la d´etection par sc´enario et plus particuli`erement `a l'application des techniques de reconnaissance de motifs pour retrouver les signatures d'attaques. D'abord, nous avons regroup´e et ordonn´e les r`egles d'IDS pour pouvoir appliquer effi cacement les m´ethodes de reconnaissance simultan´ee de plusieurs motifs. Ensuite, nous avons impl´ement´e et test´e trois algorithmes de fi ltrage multiple. Enfi n, nous avons propos´e une nouvelle technique de d´etection, effi cace pour contrarier les techniques d'evasion. Afi n de valider nos id´ees, nous avons implant´e nos algorithmes dans le syst`eme de d´etection d'intrusions Snort d´evelopp´e initialement par Martin Roesch[Roe99].

Dans la Section 2 de cet article, nous introduisons le syst`eme Snort et ses techniques de d´etection d'intrusions. Ensuite, nous expliquons la m´ethode utilis´ee pour regrouper les r`egles de Snort. Nous

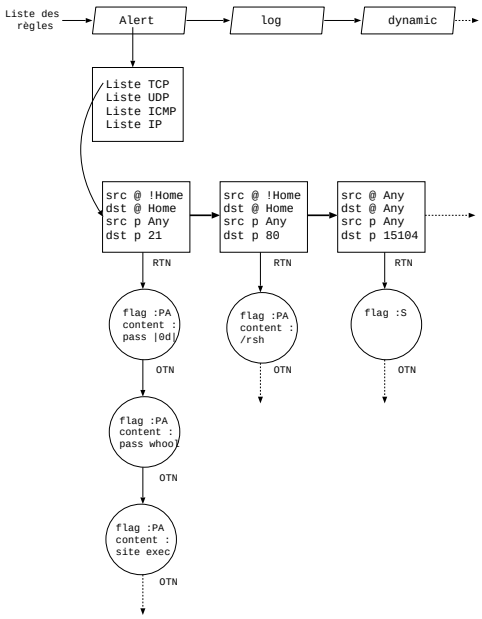

Fig. 1: Chaîne de détection de Snort

pr´esentons dans la Section 4 les algorithmes de reconnaissance de plusieurs motifs que nous avons implant´e dans Snort. La Section 5 met en œuvre notre nouveau proc´ed´e pour analyser le trafi c r´eseau. Enfi n, nous concluons notre travail dans la Section 6.

2 Introduction à Snort

Snort appartient `a la famille des d´etecteurs d'intrusions par sc´enarios. Il utilise une base de r`egles sp´ecifi´ee dans un simple fi chier de confi guration. Les r`egles peuvent être class´ees par type de trafi c. Ainsi, on trouve des r`egles sp´ecifi ques aux serveurs de noms (DNS), aux appels de proc´edures distants (RPC), aux services web, etc. L'administrateur confi gure le syst`eme de d´etection d'intrusions pour satisfaire la politique de s´ecurit´e du r´eseau qu'il administre. Il s´electionne alors les r`egles qui d´ecrivent les caract´eristiques des mauvais trafi cs contre lesquels il est indispensable de se prot´eger. Il peut par ailleurs formuler ses propres r`egles pour analyser un type de trafi c particulier. Un exemple de r`egles se pr´esente ainsi :

alert tcp !$HOME any → $HOME 23 *(msg: "Telnet Livingston DoS",*
flags :PA ;content : |*fff3 fff3 fff3 fff3 fff3*| *")* **(1)**

Snort commence la phase de pr´e-traitement par une analyse syntaxique des r`egles s´electionn´ees. Il construit une liste chaîn´ee `a 3 dimensions (Fig.1), prenant en compte le type de trafi c (TCP,ICMP,UDP,IP) et les caract´eristiques de la connexion (adresse source, adresse destination, port source, port destination). Chaque r`egle sera repr´esent´ee par une OTN (Option Tree Node) qui sauvegarde les options de cette r`egle (partie italique de la r`egle 1). Ces options servent `a la d´etection (valeur du TTL : time to live, valeur du TOS : type of service, valeur ID : identifi ant du paquet IP,...), au d´eclenchement des r´eponses suite `a la d´etection des attaques (contenus des mots cl´es REACT et RESP) et aux op´erations d'´ecriture dans les fi chiers logs (LOGTO : nom du fi chier log, msg : description de l'attaque,...). L'OTN est attach´ee `a une RTN (Rule Tree Node) partag´ee par plusieurs OTN et qui exprime la partie entête de la r`egle (partie non italique de la r`egle 1).

Apr`es avoir construit la liste des r`egles et initialis´e les interfaces r´eseaux, les plugins de d´etection, du post traitement, et d'affi chage, Snort commence son ´ecoute sur les interfaces r´eseaux sp´ecifi´ees. Il utilise pour cela la biblioth`eque pcap et envoie chaque paquet intercept´e `a la chaîne de d´etection. Il s´electionne ensuite les r`egles ad´equates et utilise leurs motifs pour reconnaître les signatures d'attaques.

Snort utilisait dans ses premi`eres versions la m´ethode naïve pour retrouver les motifs. Cette technique de reconnaissance a ´et´e am´elior´ee depuis. D'abord, l'algorithme de Boyer-Moore a permis d'acc´el´erer ´enorm´ement la recherche. Vu les grandes similarit´es observ´ees dans les motifs d'attaques (Fig. 2), l'int´erêt du fi ltrage multiple est apparu clairement. Varghese et Fisk [MF01] ont soulign´e l'importance de l'op´eration du fi ltrage dans Snort qui occupe d´esormais 31% du total du temps d'ex´ecution sur leurs corpus de tests. Afi n d'optimiser cette op´eration, ils ont introduit la m´ethode de fi ltrage de Boyer-Moore Horspool (BMH) et deux autres m´ethodes de fi ltrage multiple : la machine d'Aho-Corasick et l'algorithme Set-Wise Boyer-Moore. Ils ont r´eussi `a r´eduire le temps moyen de recherche d'un facteur de 4.6. Les deux algorithmes de fi ltrage multiple donnent presque les mêmes r´esultats. Les auteurs sugg`erent l'utilisation dynamique des 3 m´ethodes de reconnaissance. En effet, le BMH pr´esente les meilleurs r´esultats lorsqu'il s'agit d'un seul motif. En revanche, si l'ensemble de motifs est inf´erieur `a 100 alors l'utilisation du Set-Wise Boyer-Moore est la plus optimale. Tandis qu'avec un ensemble de motifs sup´erieur `a 100, l'algorithme d'Aho-Corasick semble être le plus ad´equat.

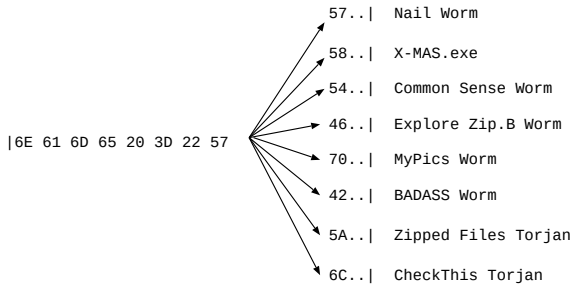

Fig. 2: Similarités dans les motifs d'attaques

Varghese et Fisk [MF01] ont rassembl´e les motifs de toutes les r`egles appartenant `a la même RTN. La disposition actuelle des OTN dans Snort les empêche de mieux factoriser les r`egles. Cette op´eration est vitale afi n d'optimiser la recherche dans la liste des r`egles. Elle permet aussi la construction effi cace d'un ensemble de motifs pour un groupe bien d´etermin´e de r`egles. Pour assurer cette factorisation, nous avons introduit une nouvelle m´ethode de compilation des r`egles de Snort. Cette m´ethode, permet de rassembler un groupe de r`egles ayant des caract´eristiques communes.

3 Restructuration des règles de Snort

Nous envisageons dans cette section de tenir compte des parties communes des r`egles de Snort afi n d'optimiser la recherche. Le r´esultat de cette factorisation est la cr´eation de deux nouveaux types d'OTN, appel´es OTN primaire et OTN secondaire. Les OTN primaires rassemblent les parties communes entre diverses r`egles. Les autres options s´ecifi ques `a chaque r`egle sont repr´esent´ees maintenant dans une OTN secondaires. Dans cette nouvelle architecture, nous appelons OTN ordinaire, chaque OTN qui repr´esente une r`egle n'ayant pas subi de factorisation. Cet OTN correspond parfaitement `a l'ancienne notion d'OTN.

Supposant que nous disposons des 4 r`egles suivantes :

R1: Alert TCP @src prt_src → @dst prt_dst (msg : "α", flags:A+, ttl:1, sid:x $_1$*)*
R2: Alert TCP @src prt_src → @dst prt_dst (msg : "β", flags:A+, ttl:2, sid:x $_2$*)*
R3: Alert TCP @src prt_src → @dst prt_dst (msg : "δ", flags:A+, ttl:3, sid:x $_3$ *)*
R4: Alert TCP @src prt_src → @dst prt_dst (msg : "θ", seq:0, sid:x $_4$*)*

Les quatre r`egles ont le même type "alert" et traitent le même flux "TCP". Elles ont par ailleurs la même adresse source, adresse destination, port source et port destination. Elles appartiennent par cons´equent `a la même RTN. Le sch´ema classique de disposition de ces r`egles avec Snort est dress´e dans la Fig.3 . En optant pour une factorisation d'´etats, nous obtenons une nouvelle physionomie de la chaîne d'OTNs. Le nombre de v´erifi cation de l'option "flags" est r´eduit de 3 `a 1. Ce gain sera plus important avec l'augmentation du nombre de r`egles.

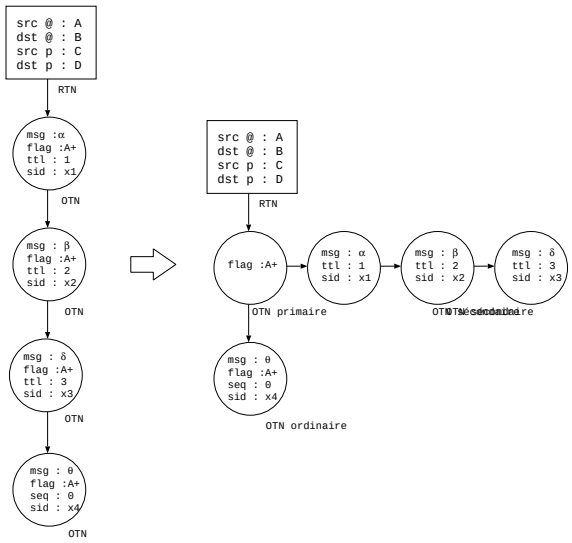

Fig. 3: Factorisation des règles de Snort

D'autre part en rassemblant les champs, nous pouvons introduire un ordre sur les options, ce qui nous m`ene `a affecter des pr´ec´edences entre les r`egles. D'autres avantages seront expliqu´es au fur et `a mesure dans ce papier, avantages qui sont en relation avec le fi ltrage de Snort.

3.1 Factorisation des règles de Snort

Soit $D = \{x_1, x_2, .., x_i, ..., x_n\}$ l'ensemble d'options dans Snort. Nous voulons regrouper les r`egles selon l'ensemble $Y = \{y_1, y_2, .., y_i, ...y_m\} \subseteq D$ des mots cl´es. Nous pouvons obtenir alors jusqu'`a 2^m classes diff´erentes.

Nous assignons `a chaque x de D un poids p_i non nul si $x_i \in Y$ et 0 sinon. Nous notons $p = (p_1, p_2, ..., p_i, .., p_n)$ le vecteur r´esultant.

Chaque r`egle peut être projet´ee sur la base $x = (x_1 x_2, .., x_i, .., x_n)$ des options de Snort. Nous obtenons alors un vecteur $v = (e_1, e_2, .., e_i, .., e_n)_x$ tel que $e_i = 1$ si l'option $x_i$ existe dans la r`egle A et $e_i = 0$ dans le cas contraire. Le poids total d'une r`egle r, donn´e par $F(r)$, est la somme des poids des options qu'elle contient.

Il est ´evident que tous les ´el´ements d'une classe pr´esentent le même poids c'est `a dire $F(A) = F([A])$. En outre, il est judicieux d'assurer l'injectivit´e de $F$. Ainsi, `a chaque poids correspond une seule classe possible ce qui nous permet de classifi er les r`egles selon leurs poids respectifs. Afi n d'assurer l'injectivit´e de F il suffi t de choisir convenablement p (par exemple prendre $p_i = e^i$).

3.2 Adaptation de la chaîne de détection

L'IDS peut s'adapter dynamiquement et d'une façon autonome `a son environnement d'ex´ecution. En effet, une analyse des alarmes d´eclench´ees met en relief les signatures fr´equement utilis´ees. Il suffi t alors d'extraire p´eriodiquement les options de ces r`egles et de leur attibuer des poids plus importants. Les r`egles auront des poids totaux plus grands et se replaceront au d´ebut de la chaîne de d´etection. On optimise ainsi le temps des paquets en fonction de l'activit´e courante du r´eseau.

4 Reconnaissance multiple dans Snort

L'op´eration du fi ltrage est consid´er´ee comme la plus p´enalisante dans le cycle de d´etection [Roe99]. L'emploi de la reconnaissance simultan´ee de plusieurs motifs permet d'optimiser la recherche en profi tant des similarit´es de certains motifs (Fig.2). Par ailleurs, nous pouvons retarder cette op´eration apr`es la v´erifi cation

des options communes des r`egles ce qui ´evite parfois le d´eclenchement inutile du filtrage suite `a la violation d'un champ dans la partie commune. Nous avons opt´e pour le regroupement par d´efaut de toutes les r`egles contenant les options "content" et "listcontent". Ces deux champs, ainsi que leurs d´eriv´es (offset, depth, distance, within) sont incorpor´es alors dans des OTN primaires avec une mise `a jour de l'ensemble de motifs.

Nous avons impl´ement´e 3 m´ethodes de reconnaissance, `a savoir Forward DAWG, la machine d'Aho-Corasick et le MATCH-DAWG. La m´ethode d'Aho-Corasick a ´et´e d´ej`a appliqu´ee par Vaghese et Fisk [MF01], mais nous l'avons r´e-implant´e avec des structures de donn´ees semblables `a celles utilis´ees dans le Forward DAWG, pour faciliter la combinaison des deux m´ethodes et r´ealiser le MATCH-DAWG. Nous d´etaillons dans la suite ces trois m´ethodes en pr´ecisant leurs avantages.

4.1 Recherche avec les DAWG

Nous utilisons les graphes acycliques de mots (DAWG) comme une structure de base. Il s'agit en effet d'une structure de donn´ees compacte pour repr´esenter les sous s´equences des mots (voir par exemple [KR97]).

L'op´eration de reconnaissance de motifs avec les DAWG consiste `a chercher pour chaque caract`ere du texte, soit t[i], et `a l'´etat [u], une ´eventuelle transition ´etiquet´ee par t[i]. Si la transition existe (repr´esent´ee en trait contenu dans la Fig. 4), alors l'´etat courant est mis `a jour. Sinon on parcourt la chaîne des pointeurs de suffixes (repr´esent´ees en pointill´e dans la Fig. 4) jusqu'`a retrouver un ´etat avec une transition t[i] (ou atteindre l'´etat initial). La longueur de la partie du mot reconnu sera mise `a jour en fonction des transitions effectu´ees et lorsqu'on atteint une longueur ´egale `a la taille du motif, on s'arrête avec succ`es. La recherche des motifs avec les DAWG est lin´eaire en taille du motif et du texte.

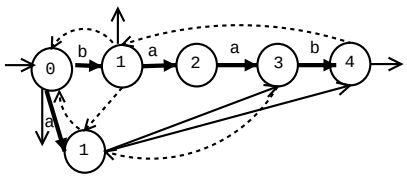

Fig. 4: DAWG du mot "baab"

4.2 Recherche avec la machine d'Aho-Corasick

L'algorithme d'Aho-Corasisk peut être vu comme une g´en´eralisation de l'algorithme de Knuth-Morris-Pratt pour faire une recherche multiple. Les ´etats de l'automate de recherche sont repr´esent´es par des nœuds de l'arbre des pr´efixes de chaque motif. La construction de cette machine se d´eroule en deux ´etapes. D'abord, on construit les ´etats et les transitions directes qui m`enent `a la reconnaissance du motif. Ensuite, on calcule une fonction d'´echec qui correspond au cas de non reconnaissance d'un caract`ere dans un ´etat donn´e. La fonction de transition finale sera la combinaison des transitions directes et des transitions d'´echec. La recherche du motif s'effectue ensuite en lisant le texte sur le graphe construit. Lorsqu'on atteint un ´etat final, on d´eclare retrouver le motif correspondant.

4.3 Recherche avec MATCH-DAWG

Cette m´ethode combine 2 algorithmes `a savoir la m´ethode d'Aho-Corasick et le Reverse Factor. La strat´egie consiste `a lire de droite `a gauche un segment du texte tant qu'il s'agit d'un facteur d'un des motifs. Ensuite, en utilisant le fait que cette partie du texte est contenue dans un motif, on le lit avec la machine `a ´etats finis d'Aho-Corasick du gauche `a droite afin de reporter les motifs trouv´es et calculer la longueur des sauts (Fig.5). Cet algorithme a ´et´e introduit par Lecroq dans [CCG93].

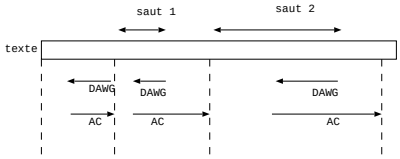

Fig. 5: Exécution de l'algorithme MATCH DAWG

5 Détection d'intrusions à la volée

5.1 Lutte contre l'évasion

Les attaquants utilisent divers moyens pour contourner les IDS[PN98]. Des pr´eprocesseurs ont ´et´e introduits dans Snort afi n de contrarier cette nouvelle g´en´eration d'attaques. On distingue en particulier les pr´eprocesseurs Frag2 et Stream4 [RG02]. Frag2 se limite par d´efaut `a 4 MB de m´emoire et 60 secondes de timeout pour rassembler les fragments IP. D'autre part, Stream4 est confi gur´e par d´efaut avec une limite de 8 MB de m´emoire et un timeout de 30 secondes pour rassembler les flux TCP.

Il est ind´eniable que ces pr´eprocesseurs ont am´elior´e le niveau de s´ecurit´e assur´e par Snort. Cependant, le stockage des paquets dans un espace m´emoire et la d´efi nition des timeouts peuvent rendre l'IDS vuln´erable aux attaques par d´eni de services et introduisent des retards dans la proc´edure de d´etection. Par ailleurs, l'allocation de l'espace m´emoire est tributaire des services support´es alors que les timeouts varient selon la capacit´e du r´eseau et la quantit´e de trafi c v´ehicul´e. En outre, les attaquants peuvent se synchroniser avec les timeouts pour r´eussir leurs exploits. Il en r´esulte que la fi xation de ces deux param`etres est une tâche assez cruciale.

Nous avons essay´e d'introduire une nouvelle m´ethode afi n de supporter la fragmentation et la segmentation du trafi c. L'id´ee consiste `a traiter le trafi c d`es sa r´eception mais en sauvegardant le r´esultat de l'analyse (appel´e trace) plutôt que les paquets proprement dits. Cela nous oblige `a adapter nos m´ethodes de reconnaissance de motifs pour supporter cette strat´egie. Une bonne gestion des listes de traces est aussi indispensable afi n d'´eliminer les traces devenues de l'information inutile pour la suite du processus de d´etection. Effectivement, chaque paquet qui arrive sera analys´e pour en d´eduire 5 propri´et´es :

- P1 : Le paquet contient une signature d'attaque.
- P2 : Le paquet est un sous mot d'une signature d'attaque.
- P3 : Le paquet contient un suffi xe qui est un pr´efi xe d'une signature d'attaque.
- P4 : Le paquet contient un pr´efi xe qui est un suffi xe d'une signature d'attaque.
- P5 : Le paquet ne comporte aucune partie d'une signature d'attaque.

On se concentrera dans un premier temps au cas d'une seule signature d'attaque `a chercher. On g´en´eralisera ensuite la m´ethode pour le cas de plusieurs signatures.

5.2 Recherche d'une seule signature d'attaque

Nous engendrons pour chaque paquet `a traiter, une structure appel´ee trace, qui contiendra les r´esultats du fi ltrage. Nous sauvegarderons en particulier :

- Lq : Liste contenant les coordonn´ees des sous mots du motif.
- Ls : Liste contenant les tailles des pr´efi xes qui sont des suffi xes du motif.
- Lp : Liste contenant les tailles des suffi xes qui sont des pr´efi xes du motif.

Nous d´erivons `a partir de ces connaissances des r`egles de d´etection. Par exemple, si nous avons un pr´efi xe de taille q, alors il suffi t pour reconnaitre tout le motif, d'avoir un suffi xe ou une liste de sous mots et un suffi xe, successeurs du paquet, ordonn´es et de taille totale ´egale `a |sig|-q avec |sig| la taille de la signature.

D'autre part il est judicieux de sauvegarder les diverses informations contenues dans le paquet et qui sont vitales pour la reconstruction du flux. On s'int´eresse en particulier `a la taille du paquet (l), le num´ero de s´equence (SN), l'identifi ant du paquet (ID), l'offset de l'entête IP (offset) et le bit du dernier fragment (MF).

Dans le cas d'un flux TCP, une autre information s'av`ere int´eressante qui est le premier num´ero de s´equence. Cette valeur repr´esent´ee par SN0 sera fi x´ee d`es l'ach`evement de l'op´eration de Three-Way Handshake et sert de rep`ere pour pouvoir ordonner les paquets. Par ailleurs, les num´eros des s´equences des paquets contenant les drapeaux RST et FIN seront d´eterminants pour rep´erer la fi n de la liste.

Nous calculons pour chaque paquet qui arrive les diff´erentes listes Lq, Ls et Lp. Nous g´en`erons ensuite les r`egles de d´etection et nous ins`erons la trace du paquet dans la liste des traces tri´ee par num´ero de s´equence contenu dans l'entête TCP ou l'offset du paquet IP (cas de la fragmentation). Si le paquet poss`ede un successeur ou un pr´ed´ecesseur imm´ediat alors ces r`egles seront ´evalu´ees. Elles peuvent être ´elimin´ees si elles ne sont pas satisfaites comme elles peuvent se transformer et transiter vers d'autres ´etats pour de futurs ´evaluations (Fig.6).

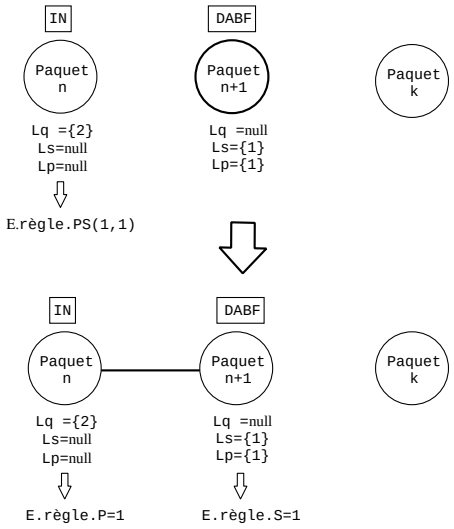

Fig. 6: Transition des règles

Cette m´ethode de d´etection analyse uniquement la trace du nouveau paquet reçu apr`es la r´eception des r`egles des traces voisines. Nous avons introduit alors la notion de jeton pour exprimer la conjonction entre les r`egles retenues dans des traces s´epar´ees. En effet, le jeton est une variable partag´ee par plusieurs r`egles et qui sera lib´er´ee partiellement par chaque r`egle ayant ´et´e v´erifi ´ee et d´efi nitivement si une seule r`egle d´etient le jeton. Si une r`egle n'est pas v´erifi ´ee, alors son jeton sera lib´er´e ce qui entraîne l'´elimination de toutes les r`egles qui lui sont attach´ees.

5.3 Gestion des listes de traces

Nous avons cr´ee 3 listes de traces pour les flux TCP, UDP et ICMP. Ces listes sont `a deux dimensions et contiennent des nœuds primaires et des nœuds secondaires. Les nœuds primaires sauvegardent les caract´eristiques de chaque flux. Ainsi, nous aurons les champs adresse source, adresse destination, port source et port destination pour les flux TCP et UDP et uniquement les adresses source et destination pour le trafi c ICMP. Les nœuds secondaires repr´esentent les traces et sont attach´es aux nœuds primaires correspondants. L'insertion dans la liste des nœuds secondaires se d´eroule en respectant l'ordre des traces afi n d'assurer le transfert correct des r`egles associ´ees. Ensuite, si l'ensemble des r`egles d'une trace T est vide et les deux traces qui la suive et la pr´ec`ede imm´ediatement existent, alors nous pouvons ´eliminer la trace T et chaîner

directement le pr´ed´ecesseur au successeur. Le transfert de r`egles s'´etablit toujours entre nœuds voisins et de l'ancienne trace vers la nouvelle. Ce sens est naturel puisque la s´emantique d'une r`egle d´ecrit ce qu'il suffi t de d´etecter chez les voisins imm´ediats pour reconnaître le motif d'attaque. Quelques sc´enarios de gestion de traces sont sch´ematis´es dans la fi gure 7.

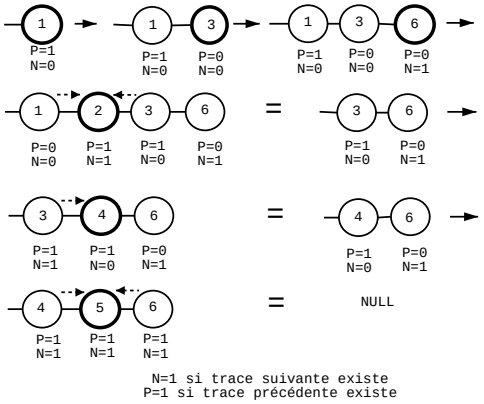

Fig. 7: Gestion des listes de traces

5.4 Recherche de plusieurs signatures d'attaques

Notre m´ethode de recherche d'une signature s'applique aussi `a plusieurs signatures `a condition d'adapter les algorithmes de fi ltrage et les structures de donn´ees repr´esentant les connaissances et les r`egles.

Supposons que nous disposons de k signatures, nous modifi ons alors nos structures de donn´ees de telle sorte que E.Tab_Lq repr´esente un tableau de taille k contenant des listes des coordonn´ees des sous mots de chaque signature. E.Tab_Lp, E.Tab_Ls et E.r`egle sont de même des tableaux de taille k et contenant respectivement pour chaque signature, les tailles des pr´efi xes, les tailles des suffi xes et les champs S, P et PS des r`egles.

5.5 Expérimentations

Nous avons test´e notre m´ethode de d´etection sur des trafi cs simul´es et des trafi cs r´eels. Le but du trafi c simul´e ´etait de valider la m´ethode de reconnaissance de motifs en essayant de d´etecter des phrases connues ins´er´ees dans des segments TCP et des paquets IP envoy´es en d´esordre. Le trafi c simul´e comporte ´egalement un grand nombre de fragments et de longues sessions TCP afi n de tester l'influence de ce type de trafi c sur nos algorithmes. Nous r´esumons dans le Tab. 1 les caract´eristiques des trafi cs analys´es et nous pr´esentons dans le tableau 2 les r´esultats obtenus.

Tab. 1: Caractéristiques du trafic

	Type	Size (Mb)	TCP (%)	UDP (%)	ICMP (%)	Autres (%)	# paquets Fragmentés	# flux TCP
log1	Sim.	2.10^{-3}	80	10	10	0	2	1
log2	Sim.	1.2	50	25	25	0	2000	1000
log3	Réel	102	90.1	6.7	3.1	0.1	0	1026
log4	Réel	222	75.3	17.8	6.7	0.2	1	4829
log5	Réel	10^3	75.3	17.8	6.7	0.2	46	18460

Il s'av`ere que notre m´ethode supporte parfaitement le processus de r´eassemblage des flux TCP et des fragments IP. Elle pr´esente ´egalement des temps nettement meilleurs lorsqu'il s'agit d'un trafi c trop fragment´e et des temps comparables en analysant des trafi cs r´eels. Par ailleurs, l'algorithme d'Aho Corasick pr´esente souvent de meilleurs r´esultats. Cela est dû `a la taille r´eduite des motifs.

Tab. 2: Résultats des expérimentations

	Snort 1.9		Snort Modifi ´e			
	Alert	Temps(s)	Alert	FDAWG(s)	AC(s)	MDAWG(s)
log1	2	0.1	3	0.1	**0.1**	0.1
log2	0	2.4	1	0.11	**0.11**	0.12
log3	169	9.17	170	9.57	**7.62**	9.51
log4	1526	30.7	1525	32.39	**27.1**	31.35
log5	2974	89.76	3001	100.29	**83.9**	99.79

6 Conclusion

Le challenge des IDS actuels est de supporter parfaitement les connexions `a haut d´ebit en assurant une d´etection avec analyse d'´etats. Dans ce context, nous avons propos´e dans cet article une m´ethode en ligne qui r´ealise ce type de d´etection. La m´ethode d´ecouvre ´egalement les tentatives courantes d'´evasion employ´ees pour contourner les ´equipements de s´ecurit´e. Nous envisageons dans de futurs travaux compl´eter notre m´ethode de d´etection par une analyse protocolaire plus approfondie afi n de mieux connaitre la nature du trafi c et par la suite d'optimiser le processus de supervision.

References

[CCG+93] M. Crochemore, A. Czumaj, L. Gąsieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter. Fast practical multi-pattern matching. Rapport 93-3, Institut Gaspard Monge, Universit´e de Marne la Vall´ee, 1993.

[CL97] M. Crochemore and T. Lecroq. Pattern matching and text compression algorithms. In *Allen B. Tucker, Jr. (Editor-in-Chief), The Computer Science and Engineering Handbook, CRC Press, in cooperation with ACM, 1997*. 1997.

[Des02] N. Desai. Increasing performance in high speed nids, a look at snort's internals. URL: www.cis.udel.edu/zhi/www.docshow.net/ids/Increasing_Performance_in_High_Speed_NIDS.pdf, 2002.

[JM01] S. Staniford J. McAlerney, C. Coit. Towards faster pattern matching for intrusion detection. DARPA Information Survivability Conference and Exposition (DISCEX II'01), 2001.

[KNMH00] J. Kuri, G. Navarro, L. M´e, and L. Heye. A pattern matching based fi lter for audit reduction and fast detection of potential intrusions. In *Proceedings of the 3rd International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, LNCS v. 1907, pages 17–27, 2000.

[KR97] G. Kucherov and M. Rusinowitch. Matching a set of strings with variable length don't cares. *Theoretical Computer Science*, 178(1–2):129–154, 30 May 1997.

[KVVK] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. Stateful intrusion detection for high-speed networks. URL:citeseer.nj.nec.com/kruegel02stateful.html.

[MF01] G. Varghese M. Fisk. Fast content-based packet handling for intrusion detection. Rapport CS 20001-0670, University of California, San Diego, Departement of Computer science and Engineering, 2001.

[MH02] K. Wiley M. Hall. Capacity verifi cation for high speed network intrusion detection systems. In *Proceedings of the 5th International Workshop on the Recent Advances in Intrusion Detection (RAID'2002)*, LNCS v. 2516, pages 239–251, 2002.

[PCC] Victor K. Wei P. C. Chan. Preemptive distributed intrusion detection using mobile agents. In *Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02)*, page 103.

[PN98] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, January 1998.

[RG02] M. Roesch and C. Green. Snort users manual snort release. URL: www.snort.org/docs/SnortUsersManual.pdf, 2002.

[RL00] D.J. Fried J. Korba K. Das R. Lippmann, J. W. Haines. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Proceedings of the 3rd International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, LNCS v. 1907, pages 162–182, 2000.

[Roe99] M. Roesch. Snort — lightweight intrusion detection for networks. In USENIX, editor, *Proceedings of the Thirteenth Systems Administration Conference (LISA XIII): November 7—12, 1999, Seattle, WA, USA*, Berkeley, CA, USA, 1999. USENIX.

[SGVS99] R. Sekar, Y. Guang, S. Verma, and T. Shanbhag. A high-performance network intrusion detection system. In *ACM Conference on Computer and Communications Security*, pages 8–17, 1999.

[WDD00] A. Wespi, M. Dacier, and H. Debar. Intrusion detection using variable-length audit trail patterns. In *Proceedings of the 3rd International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, LNCS v. 1907, pages 110–129, 2000.