Détection d'intrusions dans un environnement collaboratif sécurisé

Nischal Verma*, François Trousset**
Pascal Poncelet***, Florent Masseglia****

*IIT - Guwahati, Assam, India - nischaliit@gmail.com,
**LGI2P- Ecole des Mines d'Alès, Parc Scientifique G. Besse, 30035 Nîmes, France - trousset@ema.fr
***LIRMM UMR CNRS 5506, 161 Rue Ada, 34392 Montpellier Cedex 5, France - poncelet@lirmm.fr
****INRIA Sophia Antipolis, route des Lucioles - BP 93, 06902 Sophia Antipolis, France
florent.masseglia@sophia.inria.fr

Résumé. Pour pallier le problème des attaques sur les réseaux de nouvelles approches de détection d'anomalies ou d'abus ont été proposées ces dernières années et utilisent des signatures d'attaques pour comparer une nouvelle requête et ainsi déterminer s'il s'agit d'une attaque ou pas. Cependant ces systèmes sont mis à défaut quand la requête n'existe pas dans la base de signature. Généralement, ce problème est résolu via une expertise humaine afin de mettre à jour la base de signatures. Toutefois, il arrive fréquemment qu'une attaque ait déjà été détectée dans une autre organisation et il serait utile de pouvoir bénéficier de cette connaissance pour enrichir la base de signatures mais cette information est difficile à obtenir car les organisations ne souhaitent pas forcément indiquer les attaques qui ont eu lieu sur le site. Dans cet article nous proposons une nouvelle approche de détection d'intrusion dans un environnement collaboratif sécurisé. Notre approche permet de considérer toute signature décrite sous la forme d'expressions régulières et de garantir qu'aucune information n'est divulguée sur le contenu des différents sites.

1 Introduction

Le déploiement des ordinateurs et des réseaux a considérablement augmenté les risques causés par les attaques sur les systèmes informatiques qui deviennent un réel problème pour les entreprises et les organisations. Alors qu'auparavant de nombreuses attaques se focalisaient sur les serveurs Web car ils étaient souvent mal configurés ou mal maintenus, les attaques les plus récentes profitent des failles de sécurité des services ou applications Web qui sont plus vulnérables Heady et al. (1990); Graham (2001); Escamilla (1998). Pour pallier ce problème, de nouvelles approches appelées Systèmes de Détection d'Intrusions (SDI) ont fait leur apparition. Installés sur les réseaux, ils ont pour objectif d'analyser le trafic de requêtes et de détecter des comportements malveillants (e.g. Prelude-IDS, Snort). Ils peuvent être classés en deux grandes catégories (*e.g.* McHugh et al. (2000); Proctor (2001)) : les *systèmes de détection d'anomalies* qui cherchent à détecter les attaques et les *systèmes de détection d'abus* qui,

après avoir spécifié le comportement normal sur un site, vont rechercher les comportements non reconnus. Dans le cadre de cet article, nous nous intéressons plus particulièrement aux systèmes de type détection d'anomalies. Leur principe général est le suivant : une comparaison est effectuée entre une nouvelle requête et les signatures d'attaques représentées sous la forme d'expressions régulières. Par exemple, une attaque qui cherche à récupérer le fichier des mots de passes d'une machine (e.g. *abc/../de/../../../fg/../etc/passwd*) pourra être détectée via l'expression régulière suivante : *(/[^./]*/..)*/etc/passwd*. Traditionnellement ces signatures sont obtenues à partir d'algorithmes d'apprentissage ou bien via des sites spécialisés (e.g. OSVDB Database (2008)). Même si ces systèmes sont largement utilisés aujourd'hui, le problème essentiel est qu'ils ne savent pas gérer des attaques qui n'appartiennent pas à la base de signatures. Ainsi, lorsqu'une requête n'est pas reconnue par l'IDS, une alarme est déclenchée afin de requérir une expertise extérieure.

Récemment de nouvelles approches de détection, appelées Collaborative Intrusion Detection Systems (CIDS) (e.g. Cuppens et Miege (2005); Zhou et al. (2007); Janakiraman et al. (2003); Locasto et al. (2005); Zhang et Parashar (2006)), ont été proposées. En comparaison avec les IDS isolés, ces CIDS offrent la possibilité d'améliorer considérablement les temps de réactions et l'efficacité des détections en partageant, entre les IDS répartis sur une ou plusieurs organisations, les informations sur les attaques. Le principe général de ces approches est d'échanger via des réseaux Pair à Pair des informations survenues sur les différents IDS. Cependant traditionnellement les informations échangées se limitent aux adresses IP sources d'attaques (e.g. Cuppens et Miege (2005); Janakiraman et al. (2003); Locasto et al. (2005)) et considèrent que les données peuvent librement être échangées parmi les pairs. Cette dernière contrainte est très forte : les entreprises, pour des raisons de confidentialités, ne souhaitent pas dire qu'elles ont été attaquées et donc ne veulent pas donner d'information sur les signatures utilisées. Dans cet article nous proposons une approche de détection collaborative sécurisée, appelée SREXM (*Secure Regular Expression Mapping*), qui garantit que les données privées ne seront pas divulguées. Via notre approche, les différentes expressions régulières contenues dans les différents sites collaboratifs peuvent être utilisées sans qu'aucune information ne soit divulguée à l'extérieur. Les sites collaboratifs ont toute liberté pour travailler avec des signatures d'attaques ou de non attaques. Ainsi, lorsqu'une nouvelle requête intervient, la réponse obtenue sera simplement qu'il s'agit d'une attaque, d'une non attaque ou qu'il n'est pas possible de répondre, i.e. les bases de données n'ont pas d'information pour permettre de décider. A notre connaissance, il existe très peu de travaux qui ont abordé la problématique de la sécurité dans un tel environnement collaboratif. Seuls les travaux de Wang et al. (2005); Locasto et al. (2005) ont considéré l'aspect à la fois collaboratif et sécurisé. Dans ce contexte, la sécurisation concerne principalement les informations sur les adresses IP et sur les ports utilisés et utilise des filtres de Bloom pour les échanges. Notre problématique est différente dans la mesure où nous souhaitons échanger des données plus complexes que les adresses IP et les ports utilisées : des expressions régulières.

L'article est organisé de la manière suivante. Dans la section 2, nous présentons la problématique. Un aperçu général de l'approche est proposé dans la section 3. Les différents algorithmes utilisés sont décrits dans la section 4. La section 5 conclut en présentant différentes perspectives.

2 Problématique

Soient DB une base de données telles que $DB = DB_1 \bigcup DB_2 \ldots \bigcup DB_D$. Chaque base de données DB_i correspond à un tuple $< id, S_{exp} >$ où id correspond à l'identifiant de la base et S_{exp} est un ensemble d'expressions régulières. Toute expression régulière $exp_i \in S_{exp}$ est exprimée sous la forme d'un automate déterministe (e.g. Hopcroft et al. (2000)) représenté par le tuple $a_{exp_i} =< Etat, Trans, Init, Final >$. Dans a_{exp_i}, $Etat$ représente l'ensemble des états de l'automate, $Init$ l'état initial de l'automate, $Final$ l'ensemble des états finaux et $Trans$ l'ensemble des transitions de l'automate. Chaque transition est un quadruplet ($E_{initial}$, $Condition$, E_{final}, $Longueur$) signifiant que si l'automate est dans l'état $E_{initial}$, et que la condition $Condition$ est vérifiée alors l'automate passe dans l'état E_{final} et qu'un déplacement de $Longueur$ est effectué dans la chaîne à filtrer. Dans notre approche, nous associons également une valeur entière à chaque état final. Cette valeur spécifie si c'est une attaque ou non (booléen 0 ou 1), mais peut aussi fournir le type de l'attaque (entier).

Exemple 1. *Considérons l'expression régulière suivante : (/[^./]*/..)*/etc/passwd. L'automate associé est décrit dans la figure 1. Le tableau de gauche représente la matrice de transition et celui de droite les différentes conditions à respecter pour passer d'un état initial à un état final. Par exemple pour passer de l'état E_6 à l'état final, la condition à vérifier est que la portion courante de la chaîne, de longueur 6, soit "passwd".*

I	$cond_1$	E_1	1
E_1	$cond_2$	I	2
E_1	$cond_3$	E_2	1
E_1	$cond_4$	E_3	1
E_2	$cond_5$	E_2	1
E_2	$cond_6$	I	3
E_3	$cond_7$	E_2	1
E_3	$cond_8$	E_4	1
E_4	$cond_9$	E_2	1
E_4	$cond_{10}$	E_5	1
E_5	$cond_{11}$	E_2	1
E_5	$cond_{12}$	E_6	1
E_6	$cond_{13}$	I	2
E_6	$cond_{14}$	F	6

$cond_1$	/
$cond_2$	..
$cond_3$	[^./e]
$cond_4$	e
$cond_5$	[^./]
$cond_6$	/..
$cond_7$	[^./t]
$cond_8$	t
$cond_9$	[^./c]
$cond_{10}$	t
$cond_{11}$	[^./]
$cond_{12}$	/
$cond_{13}$	..
$cond_{14}$	passwd

FIG. 1 – *L'automate associé à l'expression régulière* exp

Définition 1. Soient DB une base de données telle que $DB = DB_1 \bigcup DB_2 \ldots \bigcup DB_D$ et une requête R, la problématique de la recherche d'expressions régulières dans un environnement collaboratif sécurisé consiste à vérifier s'il existe une expression régulière exp contenue dans DB pour laquelle $matching(exp, R) = TRUE$ tout en garantissant qu'aucune base ne fournisse directement d'information sur son contenu.

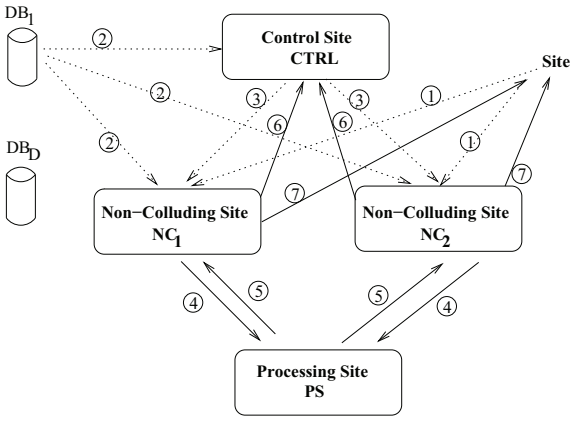

FIG. 2 – *L'architecture générale de* SREXM

3 L'approche SREXM

Dans cette section, nous proposons un survol de l'architecture sécurisée SREXM (*Secure Regular Expression Mapping*), définie pour pouvoir répondre à la problématique de la préservation de la vie privée dans le cas d'un environnement de détection collaboratif. Inspirée des travaux de Kantarcioglu et Vaidya (2002), cette architecture offre l'avantage de pouvoir réaliser les différentes opérations tout en garantissant qu'aucune des parties ne puisse avoir accès aux données privées des bases d'origine. Outre le site S chargé de fournir la requête à tester, l'architecture nécessite quatre sites *non collaboratifs* et *semi honnêtes* Goldreich (2000) : ils suivent le protocole correctement mais sont libres d'utiliser l'information qu'ils ont collectée pendant l'exécution du protocole. Ces sites indépendants collectent, stockent et évaluent l'information de manière sécurisée. Les différentes fonctions de ces sites sont les suivantes :

- **Le site de contrôle (Control Site)** $CTRL$: $CTRL$ est utilisé pour ordonnancer les différentes opérations de comparaisons de l'expression régulière. Pour cela, il interagit avec deux sites non collaboratifs NC_1 et NC_2.
- **Les sites Non Collaboratifs (Non Colluding Sites)** NC_1 **et** NC_2 : Ces sites symétriques collectent les données bruitées de toutes les bases ainsi que la requête bruitée à tester. Sous le contrôle de $CTRL$, ils interagissent avec PS et réalisent une série d'opérations de manière sécurisée sans pouvoir inférer ni des résultats intermédiaires ni du résultat final qui est retourné au site S.
- **Le site de calcul (Processing Site)** PS : Ce site est utilisé à la fois par NC_1 et NC_2 pour calculer de manière sécurisée les différentes fonctions et opérations. De manière similaire aux sites NC_1 et NC_2, PS ne peut pas déduire les résultats intermédiaires ou finaux des données qu'il traite.

Les échanges de données entre les différents sites utilisent une fonction sécurisée $SEND^S(\overset{+}{V}|\overset{-}{V})'$ qui permet d'envoyer un vecteur de bits V = $\overset{+}{V} \oplus \overset{-}{V}$ à NC_1 et NC_2. Elle est définie de manière à envoyer $\overset{+}{V}$ à NC_1 et $\overset{-}{V}$ à NC_2 (ou inversement). Pour effectuer l'envoi

sécurisé, nous utilisons un nombre aléatoire R tel que $\overset{+}{V} = R$ et $\overset{-}{V} = V \oplus R$. Cette méthode est, par exemple, utilisée pour envoyer des données des différentes bases DB_i ou pour envoyer la requête du site S. Ainsi, le processus, décrit figure 2, débute de la manière suivante. Tout d'abord le site S envoie sa requête à NC_1 et NC_2 en utilisant la fonction $SEND^S$ (C.f. flèche numérotée 1 dans la figure 2). De manière plus précise, la requête R est considérée sous sa forme booléenne. Nous générons un vecteur aléatoire A_R ayant la même taille que la requête R et nous calculons un nouveau vecteur : $Z_R = A_R \bigoplus R$. Z_R est alors envoyé à NC_1 et A_R à NC_2 (ou vice-versa). Chaque base de données DB_i décompose la matrice de transition en trois tableaux : le premier correspond aux transitions, le second au conditions et le troisième aux longueurs des déplacement. Les indices de ces tableaux sont triés aléatoirement de manière à encoder la matrice. Les bases de données DB_i envoient d'abord à $CTRL$ la matrice de transition transformée puis, en utilisant $SEND^S$, envoient à NC_1 et NC_2 les conditions associées à la matrice ainsi que le tableau des longueurs dans l'ordre des indices fournis à $CTRL$ (C.f. flèche numérotée 2). A partir de ce moment l'évaluation de la requête est effectuée sous le contrôle de $CTRL$. Ce dernier demande à NC_1 et NC_2, via la fonction $NCOMPARE^S$, d'évaluer la condition d'indice i dans leur tableau de condition (C.f. flèche numérotée 3). A ce moment, NC_1 dispose d'une partie de la requête à tester, notée $\overset{+}{R}$, d'une partie des conditions, notée $\overset{+}{ST}_{R_i}$ ainsi que la position courante dans R. De la même manière NC_2 contient $\overset{-}{R}$, $\overset{-}{ST}_{R_i}$ et la position dans la requête, pos. Il ne reste donc plus, pour NC_1 et NC_2, qu'à extraire dans la requête, la sous chaîne commençant à la position pos et dont la longueur est égale à celle de la condition, i.e. $\overset{+}{ST}_{R_i}$ ou $\overset{-}{ST}_{R_i}$. L'étape suivante consiste à comparer les deux chaînes de manière sécurisée en envoyant à PS les différentes parties via la fonction $NCMP^S$ (C.f. flèche numérotée 4). Le résultat des comparaisons est réparti entre NC_1 et NC_2 et est retourné au contrôleur de manière sécurisé (C.f. flèches 5 et 6). Le processus est répété sous le contrôle de $CTRL$ tant que l'automate n'est pas terminé. De manière à maintenir le déplacement de la position dans la requête, lorsqu'une condition est vérifiée, i.e. les deux chaînes sont semblables, $CTRL$ utilise une opération sécurisée $INCR^S$ dont l'objectif est de décaler la position en fonction de la longueur associée à la transition. Pour cela, il indique à NC_1 et NC_2 l'indice correspondant dans le tableau des longueurs et ces derniers mettent à jour la valeur de pos. Lorsque l'automate passe dans un état final ou qu'il ne reconnaît pas la séquence, $CTRL$ agrège les différents résultats (attaque, non attaque) ou inconnu, via une fonction sécurisée $AGGREGATE^S$ et répartit le résultat final entre NC_1 et NC_2. Tant qu'il existe des données à traiter, ces derniers conservent le résultat agrégé pour les transmettre au site S en fin de traitement.

4 Les différents algorithmes sécurisés

Dans cette section, nous présentons les différents algorithmes utilisés dans l'approche SREXM. De manière à simplifier l'écriture des algorithmes, nous considérons les notations suivantes. Soit $(\overset{+}{X}|\overset{-}{X}) \leftarrow h^S(\overset{+}{Y_1}...\overset{+}{Y_n}|\overset{-}{Y_1}...\overset{-}{Y_n})$ un calcul triparti de n'importe quelle fonction h^S entre NC_1, NC_2 et PS où NC_1 possède une partie des entrées $\overset{+}{Y_1}...\overset{+}{Y_n}$ et obtient une partie du résultat $\overset{+}{X}$, et de manière similaire NC_2 possède une partie des entrées $\overset{-}{Y_1}...\overset{-}{Y_n}$ et obtient une partie des résultats $\overset{-}{X}$ à la fin du processus. Le résultat final est obtenu en appliquant l'opérateur

binaire XOR ($\bigoplus$) entre $\overset{+}{X}$ et $\overset{-}{X}$. Cependant, cela n'implique pas que NC_1 envoie directement $\overset{+}{Y_1}...\overset{+}{Y_n}$ à PS et reçoit le résultat $\overset{+}{X}$ de PS. En fait, NC_1 transforme ses entrées $\overset{+}{Y_1}...\overset{+}{Y_n}$ en $\overset{+}{Y_1'}...\overset{+}{Y_n'}$ via l'addition d'un bruit aléatoire uniforme et les envoie de manière sécurisée à PS. De manière symétrique, NC_2 envoie aussi ses entrées bruitées à PS. A la fin du calcul, les deux sites reçoivent le résultat partagé et bruité $\overset{+}{X'}$ et $\overset{-}{X'}$ de PS. Ce résultat intermédiaire pourra par la suite être utilisé comme entrée pour d'autres calculs. De la même manière, nous utilisons :

1. $g^S(\overset{+}{x}, \overset{+}{y}|\overset{-}{x}, \overset{-}{y}) \Leftrightarrow g^S(\overset{+}{x}|\overset{-}{x}; \overset{+}{y}|\overset{-}{y})$
2. Si $h^S()$ est une fonction à deux arguments alors $h^S(\overset{+}{X_1}, \cdots, \overset{+}{X_n}|\overset{-}{X_1}, \cdots, \overset{-}{X_n})$ correspond à $h^S(h^S(\cdots h^S(h^S(\overset{+}{X_1}, \overset{+}{X_2}|\overset{-}{X_1}, \overset{-}{X_2}); \overset{+}{X_3}|\overset{-}{X_3})\cdots); \overset{+}{X_n}|\overset{-}{X_n})$

4.1 L'algorithme $NCOMPARE^S$

Algorithme 1 : Algorithme $NCOMPARE^S$

Data : $(i|i)$ L'indice de la condition à tester qui est envoyé à NC_1 et NC_2 par $CTRL$.

Result : $(\overset{+}{b}|\overset{-}{b})$ deux booléens tels que $b = \overset{+}{b} \oplus \overset{-}{b}$ soit faux si STR_i se trouve à la position courante de la requête examinée et vrai sinon.

1. NC_1 calcule $Len_1 = length(\overset{+}{ST_{R_i}})$; NC_2 calcule $Len2 = length(\overset{-}{ST_{R_i}})$.
   // Par définition $Len_1 = Len_2$
2. If ($(pos + Len_1 > length(\overset{+}{R}))$||$(pos + Len_2 > length(\overset{-}{R}))$)
   then return $(\overset{+}{b}|\overset{-}{b}) = (1|0)$
3. NC_1 calcule $\overset{+}{S} = \overset{+}{R}_{pos} \cdots \overset{+}{R}_{pos+Len_1-1}$
4. NC_2 calcule $\overset{-}{S} = \overset{-}{R}_{pos} \cdots \overset{-}{R}_{pos+Len_2-1}$
5. avec PS, NC_1 et NC_2 calculent $(\overset{+}{b}|\overset{-}{b}) = NCMP^S(\overset{+}{S}|\overset{-}{S})$

L'évaluation de la condition $NCOMPARE^S(Str|Str)$ (C.f. Algorithme 1) d'une transition est commandée par le contrôleur $CTRL$. Il envoie à NC_1 et NC_2 l'indice *i* d'une chaîne dans le tableau des conditions dont NC_1 ne connaît que la partie $\overset{+}{ST_{R_i}}$ et NC_2 l'autre partie $\overset{-}{ST_{R_i}}$ de telle manière que la chaîne réelle soit : $STR_i = \overset{+}{ST_{R_i}} \oplus \overset{-}{ST_{R_1}}$. Chacun des sites NC_1 et NC_2 ne connaît également que sa partie de la requête à tester $(\overset{+}{R}|\overset{-}{R})$ ainsi que la position courante dans cette requête (pos). Après avoir extrait la partie de la requête R commençant à la position pos et ayant même longueur que STR_i, l'opération de comparaison est effectuée via $NCMP^S$. La fonction $NCMP^S(\overset{+}{S_1}, \overset{+}{S_2}|\overset{-}{S_1}, \overset{-}{S_2}) \rightarrow (\overset{+}{b}|\overset{-}{b})$ (C.f. section 4.4) permet de comparer des séquences de bits de même longueur $S_1 = \overset{+}{S_1} \oplus \overset{-}{S_1}$ et $S_2 = \overset{+}{S_2} \oplus \overset{-}{S_2}$ et retourne une valeur booléenne $b = \overset{+}{b} \oplus \overset{-}{b}$ telle que b est faux si S_1 et S_2 sont identiques et vrai dans le cas contraire. Le résultat final est retourné à $CTRL$.

Complexité : La complexité de l'algorithme $NCOMPARE^S$ est la même que celle de $NCMP^S$ (C.f. section 4.4).

$COMPARE^S$ ne permet pas à NC_1 ou NC_2 de connaître si le résultat de la comparaison est vrai ou faux. Ils ne peuvent normalement déduire que la longueur de la requête qui a été comparée avec succès (la valeur de pos). Cependant, même s'ils arrivaient à déduire la liste des indices des chaînes qui ont effectivement filtré la requête, étant donné qu'ils ne connaissent que des valeurs aléatoire contenues dans STR, ils ne peuvent en déduire au pire qu'une séquence aléatoire de longueur pos. Pour autant, ils ne peuvent pas en déduire si le filtrage a réussi ou non, ni connaître la valeur associée à l'état final dans le cas où le filtrage a réussi. Au niveau du contrôleur $CTRL$ aucune information sur la longueur de la partie filtrée ou sur la valeur de la requête ne peut être déduite. En effet, $CTRL$ n'a aucun accès aux valeurs (requête ou STR), il ne connaît que des indices et il en est de même pour les déplacement. La seule information qu'il connaît est le parcours effectué dans l'automate.

4.2 L'algorithme $INCR^S$

La requête à tester est répartie dans NC_1 et NC_2 de manière sécurisée. La position du début d'une comparaison est connue par NC_1 et NC_2. La modification de cette position est commandée par $CTRL$ via l'algorithme $INCR^S(len|len)$[1]. Lors de l'envoi de l'automate à $CTRL$ et des données à NC_1 et NC_2, ces derniers reçoivent également un tableau trié de manière aléatoire indiquant les longueurs des déplacements à effectuer. L'objectif de ce tri est d'interdire qu'il puisse y avoir une correspondance directe entre les indices des conditions et des longueurs. L'algorithme $INCR^S$ ne fait qu'indiquer quel est l'indice du tableau à utiliser par NC_1 et NC_2 pour mettre à jour la position, pos, de la requête en cours d'analyse. Lorsque $CTRL$ commande une incrémentation, rien ne permet à NC_1 ou NC_2 de connaître la condition qui a déclenché le déplacement car $CTRL$ peut très bien faire faire des calculs inutiles. Du coté de $CTRL$ aucune information sur la longueur ne peut être déduite car il ne connaît que des indices sur le tableau des longueurs.

4.3 L'algorithme $AGGREGATE^S$

L'aggrégation des résultats consiste à conserver de manière sécurisée le dernier résultat obtenu par $CTRL$ lorsqu'un automate a reconnu la requête. L'objectif est qu'au travers d'$AGGREGATE^S$ (C.f. Algorithme 2), ni NC_1 ni NC_2 ne puissent savoir si l'automate a filtré la requête ou non. Pour cela, $CTRL$ positionne un bit à 1 si la requête a été filtrée et à 0 autrement. En fonction de la valeur de ce bit, l'information qui sera stockée dans un accumulateur entre NC_1 et NC_2 est soit la valeur de l'état final W_f, soit un vecteur aléatoire. Pour cela nous utilisons la fonction $\bigvee^S(\overset{+}{S_1}, \overset{+}{S_2}|\overset{-}{S_1}, \overset{-}{S_2}) \rightarrow (\overset{+}{V}|\overset{-}{V})$ qui réalise un OU bit à bit sur les séquences de bits S_1 et S_2 de même longueur et retourne la séquence V ainsi que $\bigwedge^S(\overset{+}{S_1}, \overset{+}{S_2}|\overset{-}{S_1}, \overset{-}{S_2}) \rightarrow (\overset{+}{V}|\overset{-}{V})$ qui réalise un AND bit à bit sur les séquences de bits S_1 et S_2 de même longueur et retourne la séquence V. A l'issu du processus SRexM, NC_1 et NC_2 envoient les valeurs stockées dans l'accumulateur à S. Ce dernier n'a plus qu'à effectuer un

[1] Par manque de place, nous ne décrivons pas le contenu de cet algorithme.

XOR entre les deux valeurs retournées pour obtenir le résultat.

Pour chaque expression régulière (automate), les valeurs V_f associées aux états finaux des bases sont encodées à l'aide de deux nombres aléatoires R_1 et R_2 en calculant $W_f = V_f \oplus R_1 \oplus R_2$. $CTRL$ connaît les W_f, NC_1 connaît R_1 et NC_2 connaît R_2. On considère que la longueur des W_f est identique pour toutes les bases.

Algorithme 2 : Algorithme $AGGREGATE^S$

Data : $Y = \overset{+}{Y} \oplus \overset{-}{Y}$ de longueur $n+1$ dont le bit zéro correspond au bit positionné par $CTRL$.
// $\overset{+}{A}$ et $\overset{-}{A}$ sont les valeurs d'agrégation conservées respectivement par NC_1 et NC_2.
// $n+1$ la longueur de A.

1. NC_1 calcule $\overset{+}{Z} = \overset{+}{Y} \oplus 0R_1$; NC_2 calcule $\overset{-}{Z} = \overset{-}{Y} \oplus 0R_2$;
2. $\forall k \in 1..n$: NC_1,NC_2,PS effectuent $(\overset{+}{B_k}|\overset{-}{B_k}) = \bigwedge^S (\bigvee^S (\overset{+}{Z_0}, \overset{+}{A_k}|\overset{-}{Z_0}, \overset{-}{A_k}) \; ; \; \bigvee^S (\neg\overset{+}{Z_0}, \overset{+}{Z_k}|\overset{-}{Z_0}, \overset{-}{Z_k}))$
3. NC_1, NC_2, PS calculent $(\overset{+}{B_0}|\overset{-}{B_0}) = \bigvee^S (\overset{+}{Z_0}, \overset{+}{A_0}|\overset{-}{Z_0}, \overset{-}{A_0})$
4. NC_1 et NC_2 effectuent respectivement $\overset{+}{A} = \overset{+}{B}$ et $\overset{-}{A} = \overset{-}{B}$.

Propriété 1. *L'algorithme $AGGREGATE^S$ interdit à NC_1 ou NC_2 de connaître les informations stockées dans l'accumulateur. De la même manière il n'est pas possible de savoir si la valeur de l'accumulateur a changé ou pas.*

Preuve : Les données $(\overset{+}{Y}|\overset{-}{Y})$ reçues par NC_1 et NC_2 sont bruitées par $CTRL$ de manière aléatoire et il est donc impossible de connaître la valeur de Y ou celle de Y_0, i.e. le premier bit indiquant si l'automate a atteint l'état final ou non. Les opérateur $\bigvee^S$ et $\bigwedge^S$ retournent des valeur bruitées aléatoirement à NC_1 et NC_2 ce qui signifie que du point de vue de NC_1 (respectivement NC_2), la séquence obtenue dans $\overset{+}{B}$ (respectivement $\overset{-}{B}$) est alétoire et donc indépendante des valeurs de $\overset{+}{Y}$ et $\overset{+}{A}$ (respectivement $\overset{-}{Y}$ et $\overset{-}{A}$). En particulier, même si NC_1 et NC_2 connaissent la valeur initiale de A (0 au début du processus), il leur est impossible de déduire si cette valeur a été changée ou non par $AGGREGATE^S$.

Complexité : La fonction $\bigvee^S$ est utilisée $2n+1$ fois et $\bigwedge^S$ n fois sur 1 bit. En reprenant la compexité des opérateurs $\bigvee^S$ et $\bigwedge^S$ (C.f. section 4.4) , $AGGREGATE^S$ NC_1 et NC_2 effectuent donc 34n+12 opérations binaires, calculent 6n+2 bit aléatoires envoient 12n+4 bits et reçoivent 10n+4 bits (données comprises). De son côté, PS effectue 12n+4 opérations binaires, calcule 3n+1 bits aléatoires, reçoit 12n+4 bits (6n+2 pour NC_1 et 6n+2 pour NC_2) et envoi 6n+2 bits (3n+1 à NC_1 et 3n+1 pour NC_2). Ceci est bien sûr à rapporter à la longueur des entrées (n+1 bits).

Remarques : Les deux mécanismes *bufferisation des données envoyées par les bases* et *agrégation des résultats* permettent de réaliser l'anonymisation des bases. En effet, même si le client peut détecter quelles sont les bases qui ont envoyé leurs données à SREXM, il ne peut pas en déduire quelle est celle qui a fourni le résultat final.Le retour de la valeur agrégée des

résultats peut être effectué dès qu'une expression est validée. Cependant, dans ce cas, NC_1 et NC_2 peuvent savoir quelle est la base qui a fourni la réponse. Pour que l'anonymisation soit efficace, il est donc nécessaire d'attendre, par exemple, que toutes les expressions de toutes les bases aient été considérées. Cette approche n'est malheureusement pas efficace car trop coûteuse en temps. Pour minimiser ce coût, il est bien sûr possible de retourner des valeurs intermédiaires au client chaque fois que n résultats ont été agrégés ce qui minimise le coût à $n/2$. En fait les deux mécanismes d'anonymisation ont des coûts différents : la bufferisation introduit essentiellement des coûts en espace alors que l'agrégation introduit des coûts en temps de calcul. Il est bien sûr possible de faire varier ces deux paramètres pour adapter le processus d'anonymisation en fonction des besoins et des coûts supportables.

Algorithme 3 : L'algorithme $\bigwedge^S$

Data : $(\overset{+}{X}, \overset{+}{Y} | \overset{-}{X}, \overset{-}{Y})$ sont des bits tels que $\overset{+}{X}$ et $\overset{+}{Y}$ appartiennent à NC_1, $\overset{-}{X}$ et $\overset{-}{Y}$ appartiennent à NC_2.

Result : $(A^R | B^R)$ sont tels que $A^R \bigoplus B^R = (\overset{+}{X} \bigoplus \overset{-}{X}) \bigwedge (\overset{+}{Y} \bigoplus \overset{-}{Y})$.

1. NC_1 et NC_2 génèrent et échangent quatre nombres aléatoires R_A, R'_A, R_B et R'_B tels que : $\overset{+}{X}' = \overset{+}{X} \bigoplus R_A$, $\overset{+}{Y}' = \overset{+}{Y} \bigoplus R'_A$, $\overset{-}{X}' = \overset{-}{X} \bigoplus R_B$ et $\overset{-}{Y}' = \overset{-}{Y} \bigoplus R'_B$.
2. NC_1 envoie $\overset{+}{X}'$ et $\overset{+}{Y}'$ à PS.
3. NC_2 envoie $\overset{-}{X}'$ et $\overset{-}{Y}'$ à PS.
4. PS calcule $\overset{+}{C} = \overset{+}{X}' \bigwedge \overset{-}{Y}'$ et $\overset{-}{C} = \overset{-}{X}' \bigwedge \overset{+}{Y}'$ ainsi qu'un nombre aléatoire R_{PS}.
5. PS envoie $A'_{PS} = \overset{+}{C} \bigoplus R_{PS}$ à NC_1 et $B'_{PS} = \overset{-}{C} \bigoplus R_{PS}$ à NC_2.
6. NC_1 calcule $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$
7. NC_2 calcule $B^R = B'_{PS} \bigoplus (\overset{-}{X} \bigwedge R'_A) \bigoplus (\overset{-}{Y} \bigwedge R_A) \bigoplus (\overset{-}{X} \bigwedge \overset{-}{Y}) \bigoplus (R_A \bigwedge R'_B)$.

4.4 Les algorithmes $NCMP^S$, $\bigwedge^S$ et $\bigvee^S$

Dans cette section, nous définissons trois algorithmes permettant de réaliser de manière sécurisée les opérations de comparaisons de chaînes. Le principe fondamental de ces algorithmes est d'ajouter un bruit aléatoire uniforme aux données qui pourra par la suite être supprimé du résultat final. Le protocole débute avec NC_1 et NC_2 qui modifient leurs données en les XOR-isant avec des valeurs aléatoires. NC_1 et NC_2 échangent ces valeurs. Les données modifiées sont alors envoyées (e.g. pour NC_2, $\overset{-}{X}' = \overset{-}{X} \bigoplus R_B$ et $\overset{-}{Y}' = \overset{-}{Y} \bigoplus R'_B$) à PS, qui peut alors calculer de manière sécurisée une opération $\bigwedge$ ou $\bigvee$. En fait, PS travaille sur des données bruitées et calcule $\overset{+}{C} = \overset{+}{X}' \bigwedge \overset{-}{Y}'$ et $\overset{-}{C} = \overset{-}{X}' \bigwedge \overset{+}{Y}'$. Ce dernier ajoute également un bruit aléatoire aux résultats intermédiaires afin d'éviter que NC_1 et NC_2 ne disposent du résultat final. A la fin de l'application des algorithmes, les sites non collaboratifs peuvent alors calculer le résultat final en supprimant le bruit qu'ils avaient ajouté. Par exemple, pour NC_1, l'opération suivante : $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$ peut

être réalisée de manière sécurisée dans la mesure où il connaît ses propres éléments ($\overset{+}{X}$, $\overset{+}{Y}$ et R'_A) et les nombres aléatoires de NC_2 (R'_B et R_B). Le résultat final est alors $A^R \bigoplus B^R = A'_{PS} \bigoplus(\overset{+}{X} \bigwedge R'_B) \bigoplus(\overset{+}{Y} \bigwedge R_B) \bigoplus(\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus(R_B \bigwedge R'_A) \bigoplus B'_{PS} \bigoplus(\bar{X} \bigwedge R'_A) \bigoplus(\bar{Y} \bigwedge R_A) \bigoplus(\bar{X} \bigwedge \bar{Y}) \bigoplus(R_A \bigwedge R'_B)$ où $A'_{PS} \bigoplus B'_{PS} = (\overset{+}{X} \bigwedge R'_B) \bigoplus(\overset{+}{Y} \bigwedge R_B) \bigoplus(\bar{X} \bigwedge R'_A) \bigoplus(\bar{Y} \bigwedge R_A) \bigoplus(\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus(\bar{X} \bigwedge \bar{Y}) \bigoplus(R_A \bigwedge R'_B) \bigoplus(R_B \bigwedge R'_A) \bigoplus R_{PS} \bigoplus R_{PS}$.

Grâce à la propriété de l'opérateur XOR : $R \bigoplus R = 0$, nous obtenons le résultat désiré : $A^R \bigoplus B^R = \overset{+}{X} \bigwedge \overset{+}{Y} \bigoplus \overset{+}{X} \bigwedge \bar{Y} \bigoplus \bar{X} \bigwedge \overset{+}{Y} \bigoplus \bar{X} \bigwedge \bar{Y}$. Toutefois, cette opération n'est jamais réalisée par les sites non collaboratifs et le résultat final est partagé entre NC_1 et NC_2.

Algorithme 4 : L'algorithme $\bigvee^S$

Data : $(\overset{+}{X}, \overset{+}{Y} | \bar{X}, \bar{Y})$ sont des bits tels que $\overset{+}{X}$ et $\overset{+}{Y}$ appartiennent à NC_1, $\bar{X}$ et $\bar{Y}$ appartiennent à NC_2.

Result : $(A^R | B^R)$ sont tels que $A^R \bigoplus B^R = (\overset{+}{X} \bigoplus \bar{X}) \bigvee (\overset{+}{Y} \bigoplus \bar{Y})$.

1..5. Les 5 premières étapes sont similaires à celle de $\bigwedge^S$.

6. NC_1 calcule $A^R = A'_{PS} \bigoplus(\overset{+}{X} \bigwedge R'_B) \bigoplus(\overset{+}{Y} \bigwedge R_B) \bigoplus \overset{+}{X} \bigoplus \overset{+}{Y} \bigoplus(\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus(R_B \bigwedge R'_A)$.

7. NC_2 calcule $B^R = B'_{PS} \bigoplus(\bar{X} \bigwedge R'_A) \bigoplus(\bar{Y} \bigwedge R_A) \bigoplus \bar{X} \bigoplus \bar{Y} \bigoplus(\bar{X} \bigwedge \bar{Y}) \bigoplus(R_A \bigwedge R'_B)$.

Propriété 2. *L'algorithme $\bigwedge^S$ (resp. $\bigvee^S$) interdit à NC_1 d'apprendre des données privées de NC_2 et vice versa. En outre, la troisième partie PS ne peut rien apprendre sur leurs entrées privées.*

Preuve : A partir de l'algorithme, B'_{PS} est tout ce que NC_2 peut apprendre sur les données privées de NC_1. Grâce au bruit ajouté R_{PS}, NC_2 ne peut pas trouver les valeurs de $\overset{+}{X}$ ou $\overset{+}{Y}$. Etant donné que les rôles de NC_1 et NC_2 sont interchangeables, la même argumentation peut s'appliquer à NC_1 qui ne peut pas apprendre les données privées $\bar{X}$ ou $\bar{Y}$ de NC_2. En outre, en bruitant leurs entrées, NC_1 et NC_2 garantissent qu'aucune information privée n'est transmise à PS. Enfin, grâce à l'étape de prétraitement, PS ne dispose que d'un flot de valeurs qu'il ne peut pas distinguer de vraies valeurs ou de valeurs aléatoires.

Complexité : Pour l'opérateur $\bigwedge^S$, dix opérations doivent être réalisées ($6 \bigoplus$ et $4 \bigwedge$). Etant donné que deux opérations XOR sont réalisées dans l'algorithme $\bigvee^S$, nous avons au total 12 opérations. Pour chaque $\bigwedge^S$, NC_1 et NC_2 échangent 2×2 bits. A partir de NC_1 ou NC_2, 2×1 bits sont envoyés à PS et un bit retourné. En outre, NC_1 et NC_2 calculent tous les deux 2 bits aléatoires et 1 bit aléatoire est généré par PS.

La fonction $NCMP^S()$ compare deux vecteurs de bits à l'aide de la fonction sécurisée $\bigvee^S$. Le résultat de $NCMP^S()$ donne 2 bits. L'un est envoyé à NC_1 et l'autre est envoyé à NC_2. Un XOR de ces bits vaut 0 si les vecteurs sont similaires et 1 autrement.

Algorithme 5 : L'algorithme $NCMP^S$

Data : Une partie de V et de W est stockée dans NC_1 et l'autre partie est dans NC_2

Result : $(\overset{+}{R}|\bar{R})$ est tel que $\overset{+}{R} \oplus \bar{R} = 0$ si V = W sinon 1

1. NC_1 calcule $X \leftarrow \overset{+}{V} \oplus \overset{+}{W}$ où X = $(X_1, X_2, \cdots, X_l)$ et l est la longueur du vecteur V (ou W).
2. NC_2 calcule $Y \leftarrow \bar{V} \oplus \bar{W}$ où $Y = (Y_1, Y_2, \cdots, Y_l)$.
3. $(\overset{+}{R}|\bar{R}) \leftarrow \bigvee^S (X_1, X_2, \cdots, X_l | Y_1, Y_2, \cdots, Y_l)$

Complexité : Si l est la longueur des vecteurs comparés, alors CMP^S effectue l opérations $\oplus$ et utilise $l-1$ l'opération $\bigvee^S$. NC_1 et NC_2 effectuent donc $13l-12$ opérations binaires, calculent $2l-2$ bits aléatoires, reçoivent $4l-3$ bits (donnée comprise) et envoient $5l-4$ bits (résultat compris). De son côté, PS effectue $4l-4$ opérations binaires, calcule $l-1$ bits aléatoires, reçoit $4l-4$ bits et envoie $2l-2$ bits.

Propriété 3. *NC_1, NC_2 et PS n'ont aucune connaissance sur la valeur des données comparées ni sur le résultat de la comparaison.*

Preuve : Les données reçues par NC_1 et NC_2 sont bruitées au moment de l'envoi par des vecteurs aléatoires. De la même manière, toutes les valeurs retournées par $\bigvee^S$ sont elles aussi bruitées avec des bits générés aléatoirement. NC_1 et NC_2 ne voient donc que des valeurs aléatoires. Ils ne peuvent donc rien déduire sur les données et les résultats réels du calcul. Si PS conserve les résultats intermédiaires, il peut déduire une partie des nombres aléatoires qui ont été utilisés pour réencoder ses résultats envoyés à NC_1 et NC_2. Cependant, cela ne lui donne aucune information sur les données réelles.

5 Conclusion

Dans cet article, nous avons proposé une nouvelle approche de détection d'intrusion sécurisée collaborative. Via notre approche une requête peut utiliser les différentes connaissances des bases de données pour savoir si elle correspond à une attaque ou non. Nous avons prouvé que notre architecture garantissait qu'il n'était pas possible de connaître la base qui a fourni l'information et que tous les composants de l'architecture n'avaient pas la possibilité d'inférer de connaissances sur ces bases et sur la requête. Via notre approche, il est également possible de déduire le type d'attaque qui a eu lieu si celui-ci est précisé dans les bases de données. Les travaux que nous menons actuellement concernent d'une part l'étude de la suppression du site semi honnête $CTRL$ en répartissant les opérations à réaliser pour évaluer l'automate entre les différents autres sites. D'autre part, nous étudions l'optimisation de la gestion de l'automate (e.g. prise en compte de nouveaux opérateurs de conditions).

Références

Cuppens, F. et A. Miege (2005). Alert correlation in a cooperative intrusion detection framework. In *Proc. of the IEEE International Conference on Networks (ICON 2005)*, pp. 118–123.

Database, T. O. S. V. (2008). http ://osvdb.org/.

Escamilla, T. (1998). *Intrusion Detection : Network Security beyond the firewall*. John Wiley and Sons, New York.

Goldreich, O. (2000). Secure multi-party computation - working draft. *citeseer.ist.psu.edu/goldreich98secure.html*.

Graham, R. (2001). FAQ : Network intryusion detection system. *http ://www.robertgraham.com/pubs/network-intrusion-detection.html*.

Heady, R., G. Luger, A. Maccabe, et M. Servilla (August 1990). The architecture of a network level intrusion detection system. *Technical Report CS9020*.

Hopcroft, J., R. Motwanu, Rotwani, et J. Ullman (2000). *Introduction to Automata Theory, Languages and Computability*. Addison-Wesley.

Janakiraman, R., M. Waldvoge, et Q. Zhang (2003). Indra : a peer-to-peer approach to network intrusion detection and prevention. In *Proc. of the 12th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises*, pp. 226–231.

Kantarcioglu, M. et J. Vaidya (2002). An architecture for privacy-preserving mining of client information. In *Proc. of the Workshop on Privacy, Security, and Data Mining in conjunction with the 2002 IEEE ICDM Conf*, pp. 27–42.

Locasto, M., J. Parekh, A. Keromytis, et S. Stolfo (2005). Towards collaborative security and p2p intrusion detection. In *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, West Point, NY.

McHugh, J., A. Christie, et J. Allen (2000). Defending yourself : the role of intrusion detection systems. *IEEE Software*, 42–51.

Proctor, P. (2001). *Practical Intrusion Detection Handbook*. Prentice-Hall.

Wang, K., G. Cretu, et S. Stolfo (2005). Anomalous payload-based worm detection and signature generation. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection*.

Zhang, G. et M. Parashar (2006). Cooperative defence against ddos attacks. *Journal of Research and Practice in Information Technology 38*(1).

Zhou, C. V., S. Karunasekera, et C. Leckie (2007). Evaluation of a decentralized architecture for large scale collaborative intrusion detection. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, pp. 80–89.

Summary

Intrusion Detection System (IDS) uses misuse detection and anomaly detection to recognize any intrusions. Even if new efficient IDS are available they have to face on the following problem: what can we do when a request does not match a signature? Usually IDS send an alert to the supervisor. Nevertheless this signature was probably found in another web server and even if organizations may be willing to share their results they also want privacy of their data. In this paper we propose a new approach for detecting attacks in a collaborative way but by preserving the privacy of the collaborative organizations.