Profils Propres pour la Détection d'Intrusion

Yacine Bouzida et Sylvain Gombault

Département RSM GET/ ENST Bretagne
02, rue de la Châtaigneraie, CS 17607
35576 Cesson Sévigné CEDEX, FRANCE,
{Yacine.Bouzida,Sylvain.Gombault}@enst-bretagne.fr

Résumé Nous présentons, dans cet article, une nouvelle méthode de détection d'intrusion appartenant à la famille comportementale qui est basée sur l'analyse en composantes principales (ACP). Cette approche fonctionne en projetant les profils des utilisateurs sur un espace de traits qui peut décrire de signifiantes variations entre les profils. Ces traits sont connus sous le nom de *"profils propres"* car ils sont les vecteurs propres de l'ensemble des profils. L'opération de projection caractérise un profil utilisateur par une somme pondérée de l'ensemble des profils. Pour détecter si un profil est anormal, il suffit de comparer ses poids à ceux des profils utilisateurs connus. Les principaux avantages de cette méthode sont : (i) elle permet, au premier lieu, d'apprendre les profils utilisateurs ensuite déterminer si un nouveau profil correspond ou non à ceux des utilisateurs connus, (ii) son implémentation est très simple sur tous les systèmes ayant des mécanismes d'audit de sécurité et (iii) elle est robuste et permet de produire de très hauts taux de détection. Nous introduirons quelques formules nécessaires pour calculer les profils propres, ensuite nous décrirons l'algorithme de détection basé sur ces profils propres. Au départ, nous avons appliqué cette méthode sur un ensemble de profils des utilisateurs simulés, ensuite, nous avons utilisé un ensemble de profils des utilisateurs dans un réseau réel en analysant leur activité de navigation Web et les résultats expérimentaux sont très satisfaisants.

1 Introduction

Toute violation de la politique de sécurité d'un système informatique est vue comme un objectif potentiel d'intrusion [1].

La mise en place d'un IDS (Intrusion Detection System) demande la surveillance continue de ce qui se passe sur le système ou sur le réseau que l'on veut protéger. Cette surveillance est réalisée via certains mécanismes d'audit de sécurité.

Depuis le début des années 90, plusieurs outils de détection d'intrusion ont été développés et ont vu le jour. Ces outils aident les Officiers de sécurité (SSOs : Site Security Officers) à identifier d'éventuelles violations de la politique de sécurité. En général, il existe deux grandes familles de détection d'intrusion ; l'approche comportementale et l'approche par scénario.

Approche comportementale Cette approche est basée sur le comportement d'utilisateur et/ou application, on parle alors de profil utilisateur ou comportement d'une application. Elle a été proposée par Anderson en 1980 et reprise par Denning [2] en 1987. Anderson a proposé de décrire le profil utilisateur par un ensemble de mesures pertinentes modélisant au mieux son comportement afin de détecter par la suite toute déviation de son comportement habituel ainsi appris. Cette approche cherche alors à répondre à la question : "le comportement actuel de l'utilisateur et/ou application est il cohérent avec son comportement passé ?". (pour plus de détail des différents outils comportementaux, se référer à [5,6,7,8,12,13]).

Approche par scénario Cette approche cherche à retrouver des attaques connues dans le fichier d'audit. Elle nécessite donc une connaissance, a priori, des attaques bien définies. Cette approche cherche alors à répondre à la question : "le comportement actuel de l'utilisateur et/ou application contient-il une attaque connue ?". Dans ce cas, une construction d'une base de données d'attaques ou de scénarios d'attaques est nécessaire. (voir [4,9,10,11,14,16] pour d'éventuels outils de ce type d'approche).

Une détection d'intrusion hybride qui combine ces deux techniques en même temps peut être ajoutée comme une troisième approche des IDS.

L'analyse en composantes principales (ACP) [15] est une méthode populaire utilisée d'une manière excessive dans la réduction des dimensions d'espace et elle est utilisée dans plusieurs textes d'analyse multivariée. Elle est appliquée dans plusieurs domaines tels que la compression de données, le traitement d'images et la reconnaissance des formes.

L'ACP consiste à réduire un système complexe de corrélations en un plus petit nombre de dimensions [17]. Ayant un ensemble de vecteurs observés $\{v_i\},\ i \in \{1, \ldots, N\}$, les q principaux axes $\{w_j\},\ j \in \{1, \ldots, q\}$ sont les axes orthonormés sur lesquels la variation sous une projection est importante. Il est prouvé que les vecteurs w_j sont données par les q vecteurs propres dominants (i.e. ceux associés aux plus grandes valeurs propres) de la matrice de covariance $C = \sum_i \frac{(v_i - \overline{v})(v_i - \overline{v})^T}{N}$ tel que $Cw_j = \lambda_i w_j$, où $\overline{v}$ est la moyenne arithmétique simple. $u_i = C^T(v_i - \overline{v})$, de dimension q, est ainsi une représentation réduite du vecteur observé v_i.

Dans les sections qui suivent, nous présenterons une nouvelle méthode de détection d'intrusion comportementale basée sur l'analyse en composantes principales. Le paragraphe 2 présente l'approche des profils propres, le paragraphe suivant illustre les différentes étapes de la méthode. Le paragraphe 4 présente les premiers résultats expérimentaux sur un certain nombre de profils utilisateurs sous unix et le paragraphe 5 donne des résultats expérimentaux réels sur un fichier web log proxy. Enfin, le paragraphe 6 conclut l'article.

2 L'approche des profils propres

La plupart des travaux effectués sur l'approche comportementale ne se sont intéressés que sur les mesures d'un profil utilisateur qui sont importantes pour les utiliser dans l'algorithme de détection. Ceci nous a suggéré qu'une théorie d'information codant et décodant les comportements des utilisateurs peut fournir de nouvelles informations sur ces comportements contenant de signifiantes caractéristiques.

Dans le langage de la théorie de l'information, nous voulons extraire les informations essentielles dans un profil utilisateur, le coder d'une manière très efficace, ensuite comparer un comportement ainsi codé avec une base de données des comportements utilisateurs codés de la même façon. Une méthode simple pour extraire les informations contenues dans un profil consiste à capturer la variation dans une collection de comportements des utilisateurs et utiliser ces informations pour coder et comparer les profils des comportements utilisateurs.

Dans le langage mathématique, nous souhaitons trouver les composantes principales de la distribution des comportements, ou bien trouver les vecteurs propres de la matrice de corrélation de l'ensemble des profils utilisateurs, en considérant un comportement comme un point (ou vecteur) dans un espace de dimension égale au nombre des différentes métriques utilisées. Ces métriques sont des mesures qui peuvent être le temps CPU utilisé, le nombre de commandes introduites par l'utilisateur durant une session, le nombre de chaque type d'événement audité durant un intervalle de temps,...

Ces vecteurs propres peuvent être considérés comme un ensemble de traits qui caractérisent la variation entre les comportements des utilisateurs. Chaque position d'un comportement contribue plus ou moins pour chaque vecteur propre que nous appelons profil propre.

Chaque profil peut être représenté par une combinaison linéaire des profils propres. Chaque profil peut être aussi *approximé* en utilisant seulement les meilleurs profils —ceux correspondant au plus grandes valeurs propres— et qui comptent pour la plus grande variation dans l'ensemble des profils des utilisateurs.

Un profil normal d'un utilisateur, dans notre système, consiste en un ensemble de mesures statistiques telles que celles proposées par Denning [2] :
- le temps CPU utilisé,
- le nombre de mots de passe erronés introduits durant un intervalle de temps,
- le nombre de chaque type de commande introduite par l'utilisateur pendant un intervalle de temps,
- le nombre de fichiers ouverts pendant une période,
- le nombre d'applications exécutées durant une session,
- le nombre de page web visitées pendant une journé, ...

L'ensemble de ces mesures est collecté dans un vecteur de dimension n appelé vecteur du profil utilisateur représentant le profil de l'utilisateur durant une session ou un intervalle de temps choisi, où n est le nombre de mesures statistiques mentionné ci-dessus.

Si Γ est le profil qui correspond à un comportement d'un certain utilisateur, alors nous pouvons écrire

$$\Gamma = \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{pmatrix} \tag{1}$$

où $m_i, i = 1, \ldots, n$ sont les mesures caractérisant le profil utilisateur.

3 Les différentes étapes de la méthode

Le schéma général de notre système est décrit dans la figure (1).

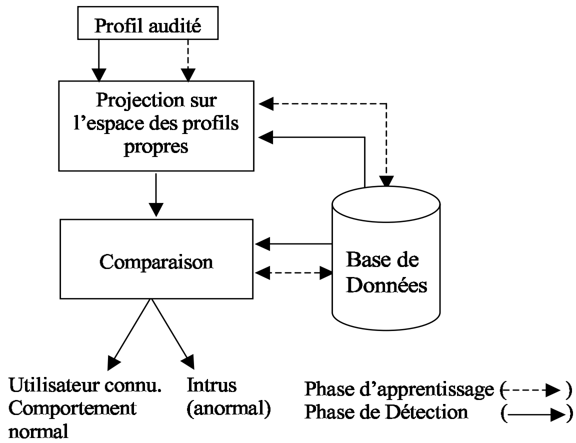

Fig. 1. Structure générale de notre système

Nous pouvons résumer les différentes étapes de cette méthode comme suit :

1. Un processus d'initialisation est nécessaire, il consiste à :
   (a) auditer l'ensemble des profils des utilisateurs du réseau,
   (b) calculer les vecteurs propres de cet ensemble,
   (c) extraire les profils traits des profils connus,
   (d) pour chaque classe (utilisateur), déterminer le vecteur trait de référence,
   (e) déterminer le seuil de chaque classe.
2. Un processus de classification est ensuite utilisé pour détecter si un nouveau comportement ainsi audité est normal ou non. Il comporte les étapes suivantes :

(a) projection du profil audité sur l'espace des profils propres,

(b) comparaison de son vecteur trait avec ceux des profils connus pour l'identification et la détection.

3.1 Le processus d'initialisation

Considérons un système comportant N utilisateurs. Chaque utilisateur est audité NU fois pendant des périodes différentes. Alors le nombre de profils est de $M = N \times NU$.

Profil moyen des utilisateurs

Soit l'ensemble des profils audités $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$. Le profil moyen Ψ de cet ensemble est défini comme étant le centre de gravité des mesures des profils utilisés pour l'apprentissage.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i \tag{2}$$

Profil caricature d'un utilisateur

Le profil caricature Φ_i est défini comme la différence entre le profil connu Γ_i et le profil moyen Ψ :

$$\Phi_i = \Gamma_i - \Psi \tag{3}$$

Calcul des profils propres

Les profils propres sont les vecteurs propres de la matrice de covariance C où

$$C_{(n \times n)} = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T = AA^T \tag{4}$$

et

$$A_{(n \times M)} = \frac{1}{\sqrt{M}} [\Phi_1 \Phi_2 \ldots \Phi_M] \tag{5}$$

Soit U_k le $k^{\text{ième}}$ vecteur propre de C associé à la valeur propre λ_k et $\mathbf{U} = [\mathrm{U_1 U_2 \ldots U_M}]$ est la matrice de ces vecteurs propres (profils propres). Alors

$$CU_k = \lambda_k U_k \tag{6}$$

tel que

$$U_k^T U_n = \begin{cases} 1 \text{ si } k = n \\ 0 \text{ si } k \neq n \end{cases} \tag{7}$$

le vecteur trait du profil est :

$$\Omega_i = U^T \times \Phi_i = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix} \tag{8}$$

Les poids $\omega_i, i = 1, \ldots, M$ décrivent la contribution de chaque profil propre dans la représentation du profil introduit, traitant les vecteurs propres comme l'ensemble de base des comportements des utilisateurs.

Ce vecteur trait peut être, par la suite, utilisé dans un algorithme de classification standard pour trouver la classe dans les comportements établis dans la phase d'apprentissage qui décrit au mieux ce nouveau comportement. La première idée qui vient à l'esprit pour déterminer quelle classe produit la meilleure description d'un vecteur en entrée est de trouver une classe dans l'espace des profils propres qui minimise la distance euclidienne avec le profil en entrée décrite dans le paragraphe suivant.

Si la taille d'un vecteur de profil est n (nombre de mesures considérées), la matrice C est de taille $n \times n$ et le calcul de n valeurs propres et vecteurs propres est une tâche très difficile à accomplir si on considère des centaines de mesures. Ainsi, une méthode faisable, en utilisant les équations (4) et (6), on peut obtenir

$$A\, A^T\, U_k = \lambda_k\, U_k \tag{9}$$

$$A^T A\, (A^T\, U_k) = \lambda_k\, (A^T\, U_k) \tag{10}$$

soit

$$Y_k = A^T\, U_k \tag{11}$$

alors

$$A^T A\, Y_k = \lambda_k\, Y_k \tag{12}$$

d'après l'équation (12), Y_k est un vecteur propre de la matrice $A^T A$ associé à la valeur propre λ_k.
Soit $X_k = \alpha_k Y_k$, donc X_k est aussi un vecteur propre de la matrice $A^T A$ d'après l'équation (11)

$$X_k^T\, X_k = (\alpha_k\, A^T\, U_k)^T\, (\alpha_k\, A^T\, U_k) = \alpha_k^2 \lambda_k\, U_k^T\, U_k \tag{13}$$

Comme $\quad U_k^T\, U = 1$
Alors

$$X_k^T\, X_k = \alpha_k^2 \lambda_k \tag{14}$$

Pour obtenir un vecteur X_k normé (*i.e.* $X_k^T\, X_k = 1$) il faut avoir

$$\alpha_k^2 \lambda_k = 1 \Rightarrow \alpha_k = \frac{1}{\sqrt{\lambda_k}} \tag{15}$$

D'après les équations (11) et (9), nous avons :

$$A\, Y_k = A\, A^T\, U_k \tag{16}$$

$$A\, Y_k = \lambda_k\, U_k \tag{17}$$

$$U_k = \frac{1}{\lambda_k}\, A\, Y_k = \frac{1}{\lambda_k\, \alpha_k} A\, X_k = \frac{1}{\sqrt{\lambda_k}} A\, X_k \tag{18}$$

D'où

$$U_k = \frac{1}{\sqrt{\lambda_k}} A\, X_k \tag{19}$$

Avec cette analyse, les calculs sont réduits d'une manière considérable, de l'ordre du nombre de mesures utilisées jusqu'à l'ordre du nombre de profils utilisés dans la phase d'apprentissage, car X_k est un vecteur propre de $A^T A$ qui est de dimension très réduite M.

Calcul des vecteurs traits Le vecteur trait Ω_i d'un profil Γ_i est obtenu en projetant son vecteur caricature Φ_i sur l'espace des profils propres

$$\Omega_i = U^T \times \Phi_i = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix}$$

où $\mathbf{U} = [\mathrm{U_1 U_2 \ldots U_M}]$
Ainsi, chaque profil est représenté par un ensemble de M' points (le vecteur trait).

Organisation en classes A chaque utilisateur lui correspond une classe composée des vecteurs traits obtenus en projetant les NU profils audités dans l'espace des profils propres. Chaque classe k est représentée par un vecteur trait de référence Ω^k :

$$\Omega^k = \frac{1}{NU} \sum_{i=1}^{NU} \Omega_i^k \tag{20}$$

Où Ω_i^k est le $i^{\text{ième}}$ vecteur trait du $k^{\text{ième}}$ utilisateur (classe).
La méthode la plus simple qui détermine la classe la plus proche du profil introduit Γ_i consiste à déterminer la classe k qui minimise la distance Euclidienne

$$\epsilon_k = \|\Omega_i - \Omega^k\|_2 \tag{21}$$

En plus, ce profil qui vient d'être audité sera affecté à la classe la plus proche si la condition suivante est vérifiée : $\epsilon_k < seuil\theta_k$ (θ_k : seuil choisi par expérience.) dans le cas contraire, le profil est classé comme intrus !

3.2 Le processus d'identification et de détection

Le processus de détection comporte les étapes suivantes

1. auditer un nouveau profil Γ_i à vérifier,
2. calculer son profil caricature $\Phi_i = \Gamma_i - \Psi$,
3. projeter le profil caricature Φ_i sur l'espace des profils. On obtient alors le vecteur trait :

$$\Omega_i = U^T \times \Phi_i = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix}$$

où $\mathbf{U}$ est la matrice des profils propres $U_i,\ i = 1, \ldots, M$

4. Déterminer la classe k qui minimise la distance $\epsilon_k = \|\Omega_i - \Omega^k\|_2$
   Si $\epsilon_k < \theta_k$ **alors**
   le nouveau profil observé est celui du $k^{\text{ième}}$ utilisateur
   Sinon
   le nouveau profil observé est anormal.
   Fsi

4 Résultats expériementaux préliminaires

Afin de montrer la rapidité, la robustesse et la simplicité de notre approche, nous avons considéré un simple exemple (qui peut être considéré comme une petite partie de notre système car on ne prend en compte que les occurrences des événements et quelques commandes) qui consiste en quatre types d'utilisateurs qui ont été utilisés dans [14,16] : l'utilisateur de mail, le novice, le développeur et l'utilisateur inexpérimenté. Les enchaînements qui suivent correspondent à des activités possibles pendant une courte période, de l'ordre de 30 minutes. Les mesures que nous considérons dans notre première expérimentation sont les différentes commandes[1] introduites par l'utilisateur durant un intervalle de temps.

Définition de ces types de comportements Les différents profils utilisateurs sont décrits dans le tableau (1) après avoir transformé les commandes UNIX en événements d'audit AIX correspondant (pour plus de détail, voir [16]) pendant la session d'audit :

Application de la méthode des profils propres Appliquons les différentes étapes citées ci-dessus à ces différents utilisateurs :

Le processus d'initialisation

1. les profils générés par ces comportements d'utilisateurs (la base d'apprentissage) sont décrits dans le tableau (1),

2. le profil moyen de ces comportements est le suivant (en utilisant l'équation (2)),

   $\Psi^T = \frac{1}{4}\sum_{i=1}^{4} \Gamma_i^T =$

   $(0\,0\,0\,0\,0\,2.25\,1.75\,1.75\,0\,0\,0\,0\,1.75\,0\,0\,0.75\,26.5\,0\,2.5\,0\,0\,0\,0\,0\,0\,0.5\,1\,0.25)$,

3. calculer le vecteur trait (équation (3) pour chaque utilisateur $(\Phi_1, \Phi_2, \Phi_3, \Phi_4)$),
4. calculer la matrice A à partir de l'équation (5),

[1] Ces commandes sont traduites en évévenements d'audit AIX

	L'utilsateur de mail Γ_1	Le novice Γ_2	Le développeur Γ_3	L'utilisateur expérimenté Γ_4
user_login fail				
user log (23h à 6h)				
short_session				
use_su Ok				
use_su fail				
who, w, finger...		2	3	4
more, pg, cat,...		1	3	3
ls OK	2	2	3	
ls fail				
df, hostname, uname				
arp, netstat, ping				
ypcat				
lpr	4	1	1	1
rm, mv	4			
ln				
whoami,id				
rexec, rlogin, rsh			1	2
proc_Execute	16	18	55	17
Proc_SetPetri				
file_open fail		4	2	4
file_open fail cp				
file_open .netrc				
file_read lpr				
file_read passwd ...				
file_write passwd...fail				
file_write cp ok		1	1	
file_unlink rm	4			
file_mode				1

Tab. 1. les profils générés des différents utilisateurs

5. calculer les vecteurs propre de la matrice de covariance $C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = AA^T$: Ceci peut être fait en utilisant le résultat défini dans l'équation(19).

$$A^TA = \begin{bmatrix} 37.15 & 20.03 & -77.40 & 20.21 \\ 20.03 & 19.65 & -60.28 & 20.59 \\ -77.40 & -60.28 & 204.78 & -67.09 \\ 20.21 & 20.59 & -67.09 & 26.28 \end{bmatrix}$$

Les valeurs propres de cette matrice sont

$$\begin{bmatrix} 273.792 \\ 12.249 \\ 1.833 \\ 0 \end{bmatrix}$$

Leurs vescteurs propres correspondants sont

$$X_1(273.792) = \begin{bmatrix} -0.329 \\ -0.254 \\ 0.865 \\ -0.282 \end{bmatrix}, X_2(12.249) = \begin{bmatrix} -0.786 \\ 0.268 \\ -0.038 \\ 0.556 \end{bmatrix}, X_3(1.830) = \begin{bmatrix} 0.157 \\ -0.783 \\ 0.026 \\ 0.601 \end{bmatrix}$$

Les vecteurs propres $U_k, k = 1, \ldots, 3$ de la matrice de covariance $C = \frac{1}{M}\sum_{i=1}^{M} \Phi_i \Phi_i^T = A\,A^T$ sont obtenus à partir de l'équation (19). Dans cet exemple, nous avons considéré un seul profil pour chaque utilisateur. Ainsi, chaque profil correspond exactement à la classe qu'il forme $\Omega_k, k = 1, \ldots, 4$ $(car N = 4)$.
Le tableau (2) suivant présente la distance euclidienne entre les différents utilisateurs après projection (à partir de l'équation(21)).

	Utilisteur1	Utilisteur2	Utilisteur3	Utilisteur4
Utilisteur1	0	4.095	19.927	4.797
Utilisteur2		0	18.577	2.179
Utilisteur3			0	19.111
Utilisteur4				0

Tab. 2. la distance euclidienne entre les diffŕentes classes

Si nous essayons de représenter les quatre profils dans le nouvel espace engendré par les profils propres, leur projection est représentée dans la figure (2) suivante.

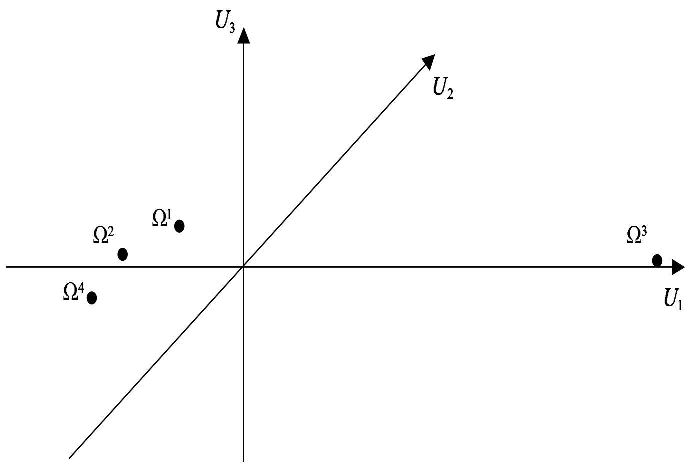

Fig. 2. La projection des profils des utilisateurs sur le nouvel espace des profils propres

5 Application de cette méthode sur un fichier web log

Ce paragraphe présente l'ensemble des tests pour mettre en œuvre le modèle de l'ACP et son application sur un fichier web log réel.

Plusieurs systèmes (voir pour l'instant [19]) sont implémentés pour des connaissances implicites à partir des fichiers web log mais aucun d'entre eux, à notre connaissance, ne s'est intéressé à répondre aux questions suivantes : le comportement de l'utilisateur change -t- il avec le temps, est il possible de caractériser les comportements utilisateurs en utilisant les fichiers web log et comment ?

Nous avons utilisé un fichier log (access.log) généré par SQUID qui est un logiciel de cache (web proxy cache) gratuit et open source [18]. Ce software est installé sur un pare-feu (firewall) reliant un réseau local contenant 6 machines d'utilisateurs qui sont des stagiaires à l'ENST-Bretagne.

Nous avons audité ces utilisateurs pendant un mois et effectué des tests sur ce fichier access.log ainsi généré qui est d'une taille d'environ 43 Méga Octets. Chaque entrée dans ce fichier a dix champs ayant le format suivant : [**Timestamp**] [**Elapsed-Time**] [**Client-IP**] [**Action**]/[**Code**] [**Size**] [**Method**] [**URL**] [**Hierarchy**]/[**Content Type**]
Avec :

1. Timestamp : L'horodatage indiquant l'heure de la requête, exprimé en seconde à partir du 1 janvier 1970 et avec une résolution en millisecondes,
2. Elapsed Time : Ce champ indique le temps écoulé de la requête en millisecondes,
3. Client-IP : L'adresse IP du client,
4. Action : Ce champ décrit comment la requête a été traitée localement dans le serveur proxy,
5. Code : Le code d'état envoyé au client comme réponse à sa requête,
6. Size : Ce champ enregistre le nombre d'octets transférés du proxy vers le client pour une requête donnée,
7. Method : La méthode http demandé par le client,
8. URL : L'URL du document demandé,
9. Hierarchy : Ce champ décrit où et comment le document est cherché,
10. Content : Le type du document spécifié.

Exemple :
985796546.734 1 192.168.xx.xx TCP_ MISS/200 207 GET http ://good.niagara .sightpath.com/images/homepage/smb-job-npm.gif - NONE/-image/gif

Avant de passer à la phase d'identification et de détection, le système doit acquérir un ensemble d'informations pour accomplir cette tâche. C'est ce qu'on appelle *la phase d'apprentissage*. Son rôle est de fixer les paramètres du système pour fournir les meilleurs résultats. C'est la tâche la plus importante et la plus délicate.

Elle est importante car les performances du système dépendent directement du choix de ces paramètres. Elle est délicate car il n'existe ni de formules ni de méthodes directes permettant la détermination de leurs valeurs, il faut les calculer par expérience.

Afin d'utiliser l'ACP, la matrice de corrélation doit être calculée. Pour ce faire, chaque élément du vecteur de profil Γ_i détermine le nombre de requêtes pour une URL donnée pendant une session d'audit (dans notre cas, une session consiste en une journée pendant la période d'activité des utilisateurs (de 08h00 à 19h00)). Ainsi, la taille du vecteur profil Γ_i dépend du nombre d'URL visitées par l'ensemble des utilisateurs choisis pour l'apprentissage durant une certaine période.

Dans notre expérimentation, nous avons utilisé six clients pour tester notre méthode. Le nombre de profils audités choisis pendant un mois était de 96 profils, soit 16 profils pour chaque utilisateur. Le reste des profils n'est pas sélectionné car ils correspondent à des journées de non activité (tel que les week-end et jours d'absences de ces utilisateurs).

Au départ, nous avons divisé cette base de profils en deux classes. La première, utilisée pour la phase d'apprentissage, est composée de 24 profils des trois premiers utilisateurs représentant leurs profils durant les deux premières semaines, soit 8 profils par utilisateur. Les 8 autres profils des 3 premiers clients sont utilisés pour tester la capacité de généralisation du système. Les 48 profils restant des 3 derniers clients, que l'on appelle ensemble non familier, sont utilisés pour évaluer la capacité du système pour rejeter les utilisateurs inconnus. Nous avons considéré les taux suivants pour présenter nos résultats :

Taux d'identification avec succès Le pourcentage de profils identifiés avec succès,
Taux de confusion ou faux négatifs Le pourcentage de profils qui ne sont ni identifiés avec succès ni rejetés,
Taux de rejet avec succès Définit le taux de profils non familiers qui sont rejetés,
Taux de faux rejet ou faux positifs Le taux de l'ensemble familier qui est rejeté.
Le tableau (3) suivant récapitule les résultats obtenus

	Ensemble Familier	
	Ensemble Appris	Ensemble non Appris
Taux d'identification avec succès	17/24 (70,83%)	10/24 (41,67%)
Taux de faux rejet ou faux positifs	7/24 (29,17%)	14/24 (58,33%)

Tab. 3. Les performances du sytème avec la première expérimentation

Ce tableau montre que les résultats obtenus ne sont pas intéressants, d'ailleurs 29,17 % des profils de l'ensemble familier ne sont pas reconnus. La cause de ces faux positifs est due en réalité aux changements des comportements des utilisateurs avec le temps.

Ceci nous a poussé à utiliser une autre stratégie qui consiste à auditer les utilisateurs pendant une période (nous avons choisi cette période égale à 4 jours successifs), ensuite nous introduisons les profils des utilisateurs de la journée qui suit pour la détection. Nous avons gardé trois utilisateurs pour la phase d'apprentissage. Au fur et à mesure que nous avançons d'une journée, la base d'apprentissage est mise à jour en considérant les profils des 3 utilisateurs pendant les quatre derniers jours d'audit choisis précédant la détection sauf dans le cas où le profil réel de l'utilisateur est détectée comme anormal. Ceci nous rappelle l'approche statistique de DENNING [2] qui détermine, à partir de n observations $x_1, x_2, \ldots, x_n$ d'une variable aléatoire x si une nouvelle observation x_{n+1} est anormale par rapport aux n observations précédentes.

Dans notre cas, chaque observation représente le profil d'un utilisateur audité pendant une journée. Le tableau (4) résume les différents résultats obtenus en utilisant cette stratégie.

	Ensemble familier		Ensemble non familier
	Profils Appris	Profils non Appris	
Taux d'identification avec succès	46/46 (100%)	34/36 (94.44%)	–
Taux de confusion ou faux négatifs	0%	0%	0%
Taux de rejet avec succès	–	–	100%
Taux de faux rejet ou faux positifs	0 (0%)	2/36 (5,56%)	–

Tab. 4. Les performances du sytème avec la deuxième stratégie

Dans ce deuxième tableau, nous avons testé notre système sur 36 profils, les douze premiers profils sont utilisés pour le premier apprentissage (4 profils par utilisateur).

Les capacités du système sur le fichier proxy web log

En utilisant un apprentissage régulier au fur et à mesure de l'évolution des profils des utilisateurs, la méthode n'a trouvé aucun problème pour reconnaître les profils qui lui ont été présentés. Concernant sa généralisation, elle a pu identifié 34 nouveaux profils sur 36 et rejeter tous les autres profils de l'ensemble non familier.

Cependant, le temps de traitement pour la phase d'apprentissage varie selon la taille du fichier de log audité pendant les 4 jours d'apprentissage. Dans notre cas, il varie entre 2 à 3 secondes en moyenne. Le temps de détection et d'identification est de l'ordre de la dixième de seconde car la détection consiste juste en la projection du nouveau profil sur la base des profils propres et le calcul de distance entre son vecteur trait et les différentes classes (3 classes pour les 3 premiers utilisateurs).

6 Conclusion

En résumé, cette nouvelle méthode apporte les idées suivantes :

1. Elle introduit une nouvelle méthode de détection d'intrusion dans l'approche comportementale qui permet une bonne classification des différents utilisateurs sous Unix et elle a démontré sa capacité d'apprentissage de profils et de généralisation en utilisant les fichiers log,
2. Elle présente une solution simple à exploiter pour le problème de détection d'intrusion en utilisant l'analyse en composantes principales.

Il est à noter que cette méthode peut facilement détecter les intrus de type masquerader [3]. D'ailleurs, si un utilisateur change de poste et garde toujours le même profil alors son vecteur trait sera proche de sa vraie classe. Dans ce cas, il suffit juste de vérifier l'adresse IP de cette classe pour savoir de quel utilisateur il s'agit. Ainsi, cet utilisateur est détecté comme masquerader.

Références

1. F. Cuppens et al. Recognizing Malicious Intention in an Intrusion Detection Process. Second International Conference on Hybrid Intelligent Systems, Santiago, Chili, December 2002.
2. D. Denning : An Intrusion Detection Model, IEEE Transactions on Software Engineering, Vol. 13 (2), 1987, pp. 222,232.
3. J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James. P. Anderson Co., Fort Washington, Pennsylvania, April 1980.
4. N. Habra, B. Le Charlier, A. Mounji, I. Mathieu, Asax : Software architecture and rule based language for universal audit trail analysis, in Y. Deswarte, G. Eizenberg, J.J. Quinsquater (Eds.), Proc. 2nd Symp. on Research in Computer Security (ESORICS), Toulouse, Berlin, Lecture Notes in Computer Science, vol. 648, Springer, Berlin, November 1992.
5. S.R Snapp et al. : DIDS (Distributed Intrusion Detection System), motivation, architecture, and early prototype, Proc. 14th National Computer Security Conf., Washington, DC, October 1991, pp. 167-176.
6. H. Debar et al. : A neural network component for an intrusion detection system, Proc. 1992 IEEE Computer Society Symp. On research in security and Privacy, Oakland, CA, May 1992, pp. 240-250.
7. D. E Denning, P. G. Neumann : Requirements and model for IDES, a real time intrusion detection expert system, Technical Report , Computer science Laboratory, SRI International, Menlo Park, CA 1985.
8. R. Jagannathan et al. : System design document : Next Generation Intrusion Detection Expert System (NIDES). Technical Report A007//A0014, SRI International, Ravenswood Avenue, Menlo Park, CA 94025, March 1993.
9. S. Kumar, E. Spafford : A pattern matching model for misuse intrusion detection, Proc. 17th National Computer security Conf. October 1994, pp. 11-21.

10. P. Porras, R. Kemmerer : penetration state analysis, a rule based intrusion detection approach, Proc. 8th Annual Computer Security Applications Conf., November 1992, pp. 220-229.
11. K. Ilgun : Ustat, a real time intrusion detection system for UNIX, Proc. IEEE Symp. On Research on Security and Privacy, Oakland, CA, May 1993, pp. 16-28.
12. S. Smaha, Haystack : an intrusion detection system, 4th Aerospace Computer Security Applications Conf. October 1988, pp. 37-44.
13. H. S. Vaccaro, G. E. Liepins, Detection of anomalous computer session activity, Proc. IEEE symp. On Research in Security and Privacy, 1989, pp. 280-289.
14. L. Mé : Gassata, a genetic algorithm as an alternative tool for security audit trail analysis, presented at RAID, the first international workshop on Recent Advances in Intrusion Detection, October 1998.
15. I. T. Jolliffe. Principal Component Analysis. 2nd Edition, New York : Springer Verlag, 2002.
16. L. Mé. Audit de Sécurité par Algorithmes Génétique, University of Rennes 1, PhD Thesis, Order N 1069, July 7th 1994.
17. H. Hotelling. Analysis of a complex statistical variables into principal components. Journal of Educational Psychology 24, pp. 417-441, 1933.
18. http ://www.squid-cache.org/
19. O. R. Zaïane, M. Xin, J. Han, Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs in Proc. ADL'98 (Advances in Digital Libraries), Santa Barbara, April 1998.