Focus

Simple Event Correlator (SEC) : surveillance en temps réel des journaux de sécurité

Risto Vaarandi 

Degré de difficulté

Cette dernière décennie a vu l'émergence d'une technique de traitement des évènements, dans de nombreux domaines. Toutefois, les solutions existantes spécialisées dans la surveillance de journaux ne supportent pas très bien cette technique. Nous allons présenter dans cet article l'utilisation du logiciel SEC, lequel permet de surveiller et de mettre en corrélation des évènements à partir des journaux de sécurité.

Les journaux d'évènements jouent un rôle crucial dans la sécurité des systèmes d'information. Aujourd'hui, de nombreuses applications, systèmes d'exploitation, dispositifs réseau et autres composants de systèmes sont capables de rédiger des messages d'évènements liés à la sécurité dans des fichiers journaux. Véritable norme en termes d'enregistrement des évènements, le protocole BSD *syslog* est supporté par une grande majorité des systèmes d'exploitation ainsi que par un bon nombre de fournisseurs en équipements réseau. Ce protocole permet de paramétrer un serveur central de journaux chargé de recevoir et de stocker les messages d'évènements issus de l'ensemble du système informatique. Il existe également plusieurs installations de serveurs *syslog* flexibles et puissantes, pouvant être utilisées sur un serveur central de journaux, dont la plus connue est *Syslog-ng*. Dans la mesure où la consignation d'évènements dans des journaux est une pratique largement acceptée et complètement normalisée, il est fort probable qu'un incident de sécurité sur un système informatique engendre un ou plusieurs messages de journal d'évènements correspondants dans dans un fichier journal.

Dans la mesure où les messages d'évènements sont, dans la plupart des cas, ajoutés aux journaux d'évènements en temps réel puisqu'ils sont émis par les composants du système, les journaux d'évènements constituent une excellente source d'informations,

Cet article explique...

- La technique de corrélation d'évènements ainsi que les approches les plus utilisées pour appliquer cette technique.
- Les raisons qui ont poussé à développer une solution comme SEC, et les principales fonctionnalités de ce logiciel.
- L'utilisation de SEC pour une surveillance en temps réel des journaux d'évènements de sécurité.

Ce qu'il faut savoir...

- Les lecteurs sont sensés maîtriser le langage des expressions régulières.
- Il est recommandé de maîtriser les principes fondamentaux de Perl notamment pour la partie intitulée *Intégrer du code Perl personnalisé grâce aux règles SEC.*

idéale pour la surveillance du système, ainsi que des conditions de sécurité qu'ils soulèvent. Au cours de ces 10 à 15 dernières années, un certain nombre d'outils open source a été développé dans le but de surveiller les journaux d'évènements en temps réel, comme par exemple, Swatch et Logsurfer. Toutefois, la majorité de ces outils ne peuvent accomplir que des tâches très simples comme déclancher une alarme dès qu'un certain type de message a été ajouté à un fichier journal. D'autre part, de nombreuses tâches de traitement d'évènements fondamentales impliquent la technique de *corrélation d'évènements*. Cette technique désigne en réalité une procédure d'interprétation conceptuelle dans laquelle une nouvelle signification est assignée à un ensemble d'évènements apparus dans un intervalle de temps prédéfini [Jakobson et Weissman, 1995]. Une application logicielle capable d'implémenter cette technique de corrélation d'évènements existe sous le nom anglais de *event correlator* (ou *corrélateur d'évènements* en français). Pendant la procédure d'interprétation, le corrélateur peut créer de nouveaux évènements et dissimuler les évènements originaux à l'utilisateur final.

Afin de mieux illustrer l'importance de la corrélation d'évènements en termes de gestion de la sécurité, penchons-nous sur le traitement des évènements dits d'*échec de connexion*. Bien qu'un évènement isolé d'*échec de connexion* puisse révéler une tentative de détournement de mot de passe, il peut également traduire le fait que l'utilisateur ait accidentellement entré un mot de passe erroné. Il est donc impossible de configurer un outil de surveillance de fichiers journaux de manière à envoyer une alerte immédiate dès l'apparition d'un message de journal de type *échec de connexion*, dans la mesure où une telle action pourrait entraîner un nombre élevé de fausses alertes. Afin d'en réduire le nombre de fausses alertes, un ou les deux schémas de corrélation d'évènements suivants peuvent être appliqués :

- dès que l'évènement N *échecs de connexion pour l'utilisateur X* a été observé au cours des T dernières secondes, générer l'évènement *nombre excessif d'échecs de connexion pour l'utilisateur X*, et l'envoyer à l'administrateur de sécurité sous forme d'alarme,
- en cas d'apparition de l'évènement *échec de connexion pour l'utilisateur X* et si, après les T secondes suivantes aucun évènement *connexion réussie pour l'utilisateur X* n'apparaît, générer l'évènement *échec de connexion non suivi d'une connexion réussie pour l'utilisateur X*, et l'envoyer à l'administrateur de sécurité sous forme d'alarme.

Au cours de ces dix dernières années, un certain nombre d'approches ont été proposées afin de traiter la corrélation d'évènements, dont des méthodes fondées sur les règles [Froehlich et al., 2002], sur les livres de code [Yemini et al., 1996], sur des graphes [Gruschke, 1998], sur des réseaux de neurones [Wietgrefe et al., 1997 ; Wietgrefe, 2002], et sur les probabilités [Meira, 1997 ; Steinder et Sethi, 2002]. Il existe également un certain nombre de solutions de corrélateurs d'évènements disponibles sur le marché, comme HP ECS, SMARTS, NetCool, NerveCenter, LOGEC, et RuleCore.

La méthode basé sur les livres de code (utilisée par SMARTS) fonctionne comme suit : si un ensemble d'évènements `e1, …, ek` doit être interprété sous forme d'évènement A, alors l'ensemble `e1, …, ek` sera stocké dans le *livre de code* en tant que vecteur de bits dirigé vers A. Si le corrélateur doit mettre en corrélation un ensemble d'évènements, il cherchera le ou les vecteurs correspondants les plus proches dans le livre de code, et reportera la ou les interprétations correspondant au(x) vecteur(s). Avec la méthode fondée sur les graphes, l'analyste humain identifie l'ensemble des dépendances entre les composants du système (services, hôtes, dispositifs réseau, etc.), puis élabore un graphe dont chaque noeud représente un composant du système et chaque arête une dépendance entre deux composants. Lorsqu'apparaît un ensemble d'évènements, le graphe permet de localiser la source des évènements (par exemple, dix évènements de type *le serveur HTTP ne répond plus* ont été engendrés par l'échec d'un seul lien du réseau). La méthode fondée sur les réseaux de neurones consiste à former un réseau de neurones afin qu'il puisse identifier des anomalies dans le flux des évènements, notamment pour la détection de leur source.

Listing 1. *Règle SEC permettant de mettre en corrélation les messages de type SNMP public access udp (Udp accès public SNMP) issus du journal d'évènements Snort IDS*

```
# Echantillon de lignes d'entrée mises en corrélation :
# Mar  1 00:36:32 snorthost.mydomain [auth.alert]
# reniflage [17725] : [1:1411:10]
# udp accès public SNMP [Classification : tentative de fuite d'informations]
# [Priorité : 2] : {UDP} 192.168.115.34:54206 -> 192.168.52.179:161

type=SingleWithSuppress
ptype=RegExp
pattern=snort\[\d+\]: \[[\d:]+\] SNMP public access udp.*\{UDP\} \
([\d\.]+):\d+ -> ([\d\.]+):\d+
desc=SNMP public access from $1 to $2
action=pipe '%s' mail -s 'Snort alert' root
window=300
```

L'approche fondée sur l'utilisation des règles est très répandue en termes de corrélation d'évènements. Elle est d'ailleurs utilisée par plusieurs produits tels que HP ECS ainsi que RuleCore. Dans le cas de cette approche, les évènements sont mis en corrélation selon un modèle de règles *condition → action* défini par l'analyste humain. La corrélation d'évènements fondée sur l'utilisation de règles reste très avantageuse selon les analystes humains qui la jugent généralement plus naturelle pour exprimer leurs connaissances sous forme de règles. Ainsi, par exemple, il est plus facile de décrire des relations temporelles entre évènements au moyen de règles, par rapport que par l'utilisation d'autres méthodes. Par ailleurs, contrairement à certaines autres méthodes de corrélation d'évènements (comme la corrélation fondée sur les réseaux de neurones, par exemple), la corrélation d'évènements fondée sur les règles demeure claire et transparente pour l'utilisateur final. Comme [Rich et Knight, 1991] l'expliquent dans leur ouvrage, si les utilisateurs finaux ne comprennent pas pourquoi ni comment une application parvient à son objectif, ils ont tendance à ignorer les résultats calculés informatiquement par l'application en question.

Listing 2. *Ensemble de règles SEC destiné à mettre en corrélation les échecs d'authentification sshd avec les messages d'authentification réussie sous Solaris*

```
# Echantillon de lignes d'entrée mises en corrélation :
# Apr  3 14:20:19 myhost sshd[25888]: [ID 800047 auth.error] erreur :
# PAM : échec d'authentification pour risto à partir de myhost2
# Apr  3 14:20:23 myhost sshd[25888]: [ID 800047 auth.info] accepté
# interaction clavier/pam pour risto de ssh2 192.168.27.69 port 9729

type=PairWithWindow
ptype=RegExp
pattern=sshd\[\d+\]: \[ID \d+ auth\.error\]\
 error: PAM: Authentication
 failed for (\S+) from \S+
desc=PAM authentication failed for $1
action=event PAM_AUTHENTICATION_FAILED_FOR_$1
ptype2=RegExp
pattern2=sshd\[\d+\]: \[ID \d+ auth\.info\]\
Accepted keyboard-interactive/pam for ($1) from \S+ port \d+ ssh2
desc2=PAM authentication successful for $1
action2=none
window=30

type=SingleWithThreshold
ptype=RegExp
pattern=PAM_AUTHENTICATION_FAILED_FOR_(\S+)
context=!USER_$1_ALREADY_COUNTED && !COUNTING_OFF
continue=TakeNext
desc=Ten authentication failures for distinct users have been observed
action=pipe '%s' mail -s 'PAM alert' root; create COUNTING_OFF 3600
window=600
thresh=10

type=Single
ptype=RegExp
pattern=PAM_AUTHENTICATION_FAILED_FOR_(\S+)
context=!USER_$1_ALREADY_COUNTED && !COUNTING_OFF
desc=Set up the "count once" context for user $1
action=create USER_$1_ALREADY_COUNTED 600
```

En dépit de l'émergence de cette nouvelle technique de traitement des évènements par corrélation dans de nombreux domaines (y compris la gestion des réseaux et de leur sécurité, la détection des intrusions, etc.), les solutions open source existantes spécialisées dans la surveillance des fichiers journaux ne supportent pas très bien cette technique. Malgré le fort succès rencontré par les systèmes de corrélation d'évènements aujourd'hui disponibles sur le marché, et largement utilisés par de nombreuses sociétés à envergure internationale, ces produits présentent encore un certain nombre de défauts. Tout d'abord, les systèmes existants sont toujours trop lourds avec une conception et une interface utilisateur alambiquées. En d'autres termes, le déploiement et la maintenance de tels produits nécessitent énormément de temps, ainsi qu'une formation utilisateur intensive. Par ailleurs, leur complexité et leurs exigences en ressources les rendent bien souvent inadaptés à des systèmes informatiques modestes ou dans le cas d'une corrélation d'évènements sur noeud avec des ressources informatiques limitées. Deuxièmement, dans la mesure où la plupart des systèmes existants sont distribués de manière commerciale, ils sont dépendants d'une plateforme. Il faut donc fournir aux clients les programmes binaires censés fonctionner sous un nombre limité de systèmes d'exploitation. Par ailleurs, plusieurs systèmes commerciaux ont été conçus uniquement pour fonctionner sur une plateforme de réseau particulière (comme par exemple, HP OpenView). Certains systèmes souffrent également de leur conception spécifique pour une gestion de réseau par défaut, rendant leur application dans d'autres domaines (dont la surveillance des journaux d'évènements) difficile. Enfin, les systèmes existants sont relativement coûteux. De nombreux organismes au budget limité sont incapables d'acquérir de telles solutions pour leurs tâches quotidiennes de sécurité et de gestion du système.

Nous allons donc vous présenter dans cet article l'outil SEC (Simple Event Correlator), solution open source développée par l'auteur pour

une corrélation d'évènements légère et indépendante des plateformes. Nous analyserons ensuite plusieurs exemples concrets illustrant les possibles utilisations de SEC afin de surveiller et de mettre en corrélation des évènements à partir de journaux de sécurité.

Principes fondamentaux de SEC

SEC est un outil open source de corrélation d'évènements ayant recours à une approche de règles pour le traitement des évènements. Cette approche a été choisie en raison de sa facilité à représenter les connaissances et de la transparence de son processus de corrélation des évènements. SEC a été conçu indépendamment des plateformes utilisées, afin de proposer une configuration simple et une architecture légère, applicables à un large choix de tâches de corrélation d'évènements, tout en consommant peu de ressources système.

Afin de garantir l'indépendance de cet outil par rapport aux plateformes des systèmes d'exploitation, l'auteur a décidé de rédiger SEC en Perl, dans la mesure où ce langage fonctionne sur presque la majorité des systèmes d'exploitation, et est désormais inclus dans de nombreuses distributions de systèmes d'exploitation. Les applications Perl peuvent ainsi fonctionner sur un large choix de système d'exploitation. Par ailleurs, les programmes Perl bien rédigés sont généralement rapides et n'utilisent que peu de mémoire.

SEC ne requiert que peu d'espace sur le disque et est très simple à installer. Sa taille actuelle ne dépasse pas les 250Ko, et ses paramètres de configuration sont stockés dans des fichiers textes tout à fait normaux (la taille de chaque fichier n'est généralement que de quelques kilo octets). Par ailleurs, dans la mesure où SEC est rédigé en Perl et ne dépend d'aucun autre package de logiciels, l'outil peut être utilisé immédiatement après avoir dézippé sa distribution source, et ce, sans aucune manipulation supplémentaire (comme une compilation, une liaison de source, ou l'installation d'autres logiciels).

SEC reçoit ses évènements d'entrée de flux de fichiers. Les fichiers normaux, les canaux nommés, et les données d'entrée standard sont généralement supportées en tant qu'entrées. Il est ainsi possible d'utiliser SEC en tant que solution de surveillance des journaux d'évènements afin de l'intégrer avec n'importe quelle application capable de rédiger ses évènements de sortie sur un flux de fichiers. Les applications dotées d'une interface de programmation pour la gestion des évènements peuvent également être intégrées au moyen de simples plugins, lesquels emploient les appels de l'interface de programmation pour lire le flux d'évènements de l'application, et le copier dans les données de sortie standard ou dans un fichier (un plugin d'échantillons destinés à HP OpenView Operations fait partie du package de SEC).

SEC peut produire des évènements de sortie en exécutant des commandes shell définies par l'utilisateur, en rédigeant des messages dans les fichiers ou dans les canaux nommés, ou en appelant des sous-programmes Perl précompilés, etc. Les évènements de sortie peuvent également être envoyés sur le réseau vers une autre instance de SEC, ce qui permet de configurer les schémas de corrélation d'évènements distribués. Par ailleurs, en dépit de l'absence d'interface graphique utilisateur sous SEC pour visualiser et gérer les évènements de sortie, il est plus sûr de diriger les évènements de sortie vers une application de gestion du système ou un cadre d'applications doté d'une telle interface graphique utilisateur (comme par exemple, HP OpenView Operations).

Les paramètres de configuration de SEC sont stockés dans des fichiers texte pouvant être créés et modifiés à l'aide de n'importe quel éditeur de texte. Chaque fichier de configuration contient une ou plusieurs règles, et les ensembles de règles fournis par les fichiers sont appliqués virtuellement en parallèle. SEC lit les données à partir des sources d'entrée ligne par ligne, et dès qu'une nouvelle ligne vient d'être lue, elle sera mise en correspondance avec les règles contenues dans le ou les fichiers de configuration.

Un des principaux aspect de SEC réside dans son *modèle de mise en corrélation des évènements*. SEC supporte les expressions régulières, les sous-chaînes, les sous-programmes Perl, ainsi que les valeurs vraies en tant que modèles. Le support des expressions régulières facilitent la configuration de SEC, puisque de nombreux outils UNIX (comme *grep*, *sed*, *find*, etc.) reposent précisément sur les expressions régulières, et de ce fait, la plupart des administrateurs de sécurité, de système et de réseau sont déjà familiers avec ce langage d'expressions régulières. Par ailleurs, dans la mesure où la majorité des outils de surveillance des journaux d'évènements utilisent le langage des expressions régulières pour la mise en corrélation des évènements, SEC peut être déployé en tant que solution de remplacement de surveillance de journaux à moindres efforts. À partir de la version 2.3.0 de l'outil, les évènements peuvent être passés vers des sous-programmes Perl précompilés pour leur reconnaissance, ce qui permet à l'utilisateur de configurer des schémas personnalisés de mise en corrélation d'évènements.

Outre le modèle de corrélation d'évènements, la plupart des règles sont définies au moyen d'une liste d'*actions*, ainsi que d'une expression booléenne facultative de *contextes*. Sont désignés par contextes SEC des entités logiques créées lors du processus de corrélation des évènements, dont chaque contexte possède une certaine durée de vie (soit finie, soit infinie).

Listing 3. *Ensemble de règles SEC destiné à consolider les messages d'alerte de priorité 1 issus de Snort IDS*

```
# Lignes d'entrée mises en corrélation :
# Apr  4 10:10:55 snorthost.mydomain [auth.alert] reniflage [18800]:
# [1:2528:14] Client SMTP PCT_tentative de surdébit de type Hello
# [Classification : tentative d'accès aux privilèges administrateur]
# [Priorité : 1]: {TCP} 192.168.5.43:28813 -> 192.168.250.44:25

type=Single
ptype=RegExp
pattern=snort\[\d+\]: \[[\d:]+\].*\[Priority: 1\]: \S+ \
([\d\.]+):?\d* -> [\d\.]+:?\d*
context=!ATTACK_FROM_$1
continue=TakeNext
desc=Priority 1 attack started from $1
action=create ATTACK_FROM_$1; \
        pipe '%s' mail -s 'Snort: priority 1 attack from $1 (alert)' root

type=Single
ptype=RegExp
pattern=snort\[\d+\]: \[[\d:]+\].*\[Priority: 1\]: \S+ ([\d\.]+):?\d* ->
                    [\d\.]+:?\d*
context=ATTACK_FROM_$1
desc=Priority 1 incident from $1
action=add ATTACK_FROM_$1 $0; \
        set ATTACK_FROM_$1 300 ( report ATTACK_FROM_$1 \
        mail -s 'Snort: priority 1 attack from $1 (report)' root )
```

Listing 4. *Règle SEC permettant de transférer des lignes strictement issues de /var/log/messages*

```
type=Suppress
ptype=TValue
pattern=TRUE
context=!_FILE_EVENT_/var/log/messages
desc=Pass only those lines that come from /var/log/messages
```

Les contextes peuvent être utilisés afin d'activer et de désactiver des règles de manière dynamique au moment de l'exécution. Par exemple, si une définition de règle a indiqué (X OU Y) en tant qu'expression de son contexte, alors que ni le contexte X, ni le contexte Y n'existe à un moment donné, la règle ne pourra pas être appliquée. Les contextes SEC agissent également en tant qu'entrepôts à évènements (c'est là une de leurs principales fonctions). Ainsi, les évènements dignes d'intérêt peuvent être associés à un contexte, et l'ensemble des évènements collectés fournit un traitement externe utilisé ultérieurement (ce concept a été emprunté à Logsurfer).

Aujourd'hui, SEC supporte neuf types de règles capables d'implémenter un certain nombre de scénarios de corrélation d'évènements parmi les plus répandus :

- *Single* – permet d'exécuter une liste d'actions dès l'observation d'un évènement mis en corrélation,
- *SingleWithScript* – identique au schéma *Single*, mais utilise également un script externe pour la corrélation,
- *SingleWithSuppress* – identique au schéma *Single*, mais ignore les évènements mis en corrélation consécutifs pendant *t* secondes,
- *Pair* – permet d'exécuter une liste d'actions sur un évènement A tout en ignorant les instances consécutives de A jusqu'à l'arrivée de B ; dès l'arrivée de B, exécute une autre liste d'actions,
- *PairWithWindow* – après avoir observé l'évènement A, attend pendant *t* secondes l'arrivée de l'évènement B ; si B n'arrive pas à temps, exécute une liste d'actions, sinon applique une autre liste d'actions,
- *SingleWithThreshold* – permet de compter les évènements d'entrée mis en corrélation pendant *t* secondes. En cas de dépassement du seuil défini au préalable, exécute une liste d'actions,
- *SingleWith2Thresholds* – identique au schéma *SingleWithThreshold*, mais doté d'un tour supplémentaire de secondes dans son décompte avec seuil décroissant,
- *Suppress* – permet de supprimer les évènements d'entrée mis en corrélation,
- *Calendar* – permet d'exécuter une liste d'actions à un moment particulier.

La plupart des définitions de règles SEC contiennent un paramètre appelé en anglais *event description string* (*chaîne de description d'évènement*) chargé de définir l'amplitude de la corrélation des évènements (consulter la partie intitulée *Règles SEC et opérations de corrélations d'évènements* pour plus de détails). Lorsqu'un évènement correspond à une règle, SEC calcule alors la clé de corrélation de l'évènement, en concaténant le nom du fichier de la règle, l'identifiant de la règle, et la chaîne de description de l'évènement. En cas d'opération de corrélation dotée de la même clé, l'évènement est mis en corrélation par cette opération. En cas d'absence d'une telle opération, et si la règle indique une corrélation d'évènement ponctuelle, SEC démarre alors une nouvelle opération avec la clé calculée. Il est intéressant de remarquer l'absence de correspondance un-à-un entre les règles et les opérations de corrélation d'évènements. SEC pourrait lancer plusieurs opérations pour une seule règle, mais les règles de type *Single*, *SingleWithS-*

cript, *Suppress*, et *Calendar* ne déclancheront jamais d'opération, puisqu'elles ne définissent pas de corrélation d'évènements sur une fenêtre temporelle.

Les actions de SEC n'ont pas été seulement conçues pour la génération d'évènements de sortie, mais permettent également de faire interagir les règles entre elles, afin de gérer les contextes, stocker les évènements, relier les modules externes d'analyse d'évènements à SEC, exécuter le code Perl sans passer par un processus séparé, etc. En combinant plusieurs règles avec des listes d'actions et des expressions de contextes appropriées, il est possible de définir des schémas de corrélation d'évènements bien plus complexes. Les parties suivantes comprennent des exemples détaillés illustrant l'élaboration d'ensembles de règles sous SEC afin de surveiller les journaux d'évènements de sécurité.

Surveillance des journaux de sécurité avec SEC

Cette partie sera consacrée à l'analyse de plusieurs exemples d'ensembles de règles ainsi que des aptitudes de traitement des évènements sous SEC. Les ensembles de règles exposés dans les exemples suivants ont été rédigés dans le but de surveiller des journaux d'évènements concrets : le journal d'évènements Snort IDS, le journal du système Solaris, */var/adm/messages*, ainsi que le journal d'erreurs du serveur Web Apache. Les ensembles de règles ainsi créés ont été testés sous la version 2.3.3 de SEC.

Afin de tester les ensembles de règles présentés dans cette partie, il est possible de télécharger SEC à partir de sa page d'accueil. Si vous souhaitez installer SEC à partir du package source, il vous suffit de dézipper la distribution (par exemple, `tar -xzvf sec-2.3.3.tar.gz`), puis de copier le fichier intitulé *sec.pl* de la distribution vers le répertoire approprié (par exemple, `cp sec-2.3.3/sec.pl /usr/local/bin`). La page d'accueil de SEC propose également des liens vers les packages de binaires de plusieurs plateformes selon différents systèmes d'exploitation.

Afin de lancer SEC en mode interactif pour la surveillance du fichier journal */var/log/messages* avec les règles issues de *my.conf*, vous pouvez utiliser la ligne de commande suivante :

```
sec.pl -conf=my.conf -input=/var/log/
                          messages
```

Afin de configurer SEC de sorte à surveiller ses données d'entrée standard (très utile en vue d'un test), utilisez la ligne de commande suivante :

```
sec.pl -conf=my.conf -input=-
```

Il est possible de définir plusieurs options `-input` et `-conf` dans la ligne de commande. Parmi les autres options généralement utilisées, citons par exemple `-log` (paramètre le fichier journal pour SEC), `-syslog` (configure SEC de manière à se connecter via *syslog*), `-debug` (paramètre le niveau de journalisation pour SEC), `-pid` (paramètre le fichier contenant l'identifiant du processus pour SEC), `-detach` (oblige SEC à se désassocier du terminal de contrôle afin de devenir un démon), et `-testonly` (permet de tester la validité des règles sans lancer SEC).

Règles SEC et opérations de corrélation d'évènements

Supposons que vous disposiez d'un fichier de règles intitulé *my.conf*, lequel contiendrait la règle exposée dans le Listing 1.

La règle *SingleWithSuppress* tirée du Listing 1 a été conçue dans le but de correspondre aux messages *SNMP public access udp* issus du journal Snort IDS. À chaque fois que le démon Snort observe un paquet de requête SNMP doté du champ de communauté *public* dans le réseau, il enregistre ce message dans son journal. Toutefois, dans la mesure où un certain nombre d'outils de gestion réseau sondent le même hôte de manière régulière pendant un court intervalle de temps, le message pourrait également être enregistré de manière répétée pour les mêmes adresses IP source et destination. Cette règle permet d'installer un scénario de corrélation d'évènements appelé *compression*. Les occurrences répétées d'évènements identiques sont réduites à un seul évènement. Le paramètre *ptype* de la définition de la règle indique que ce modèle de corrélation d'évènements relève d'une expression régulière, et le paramètre *pattern* (*modèle*) définit cette expression régulière. Le paramètre *desc* permet de définir la chaîne de description de l'évènement, le paramètre *action* la liste d'actions d'envoi d'alerte e-mail à l'utilisateur *root* local, et le paramètre *window* (*fenêtre*) la fenêtre de corrélation fixée à 300 secondes.

Lorsque l'expression régulière correspond à un ligne de données d'entrée, les variables spéciales $1 et $2 seront paramétrées dans les champs des adresses IP source et destination de la ligne d'entrée en question, puisque la structure de l'expression régulière pour ces champs est comprise entre parenthèses. SEC va ensuite calculer la clé de corrélation d'évènements en concaténant le nom du fichier de la règle, l'identifiant de la règle ainsi que la chaîne de description de l'évènement. Ainsi, si $1 est égale à 192.168.115.34 et si $2 équivaut à 192.168.52.179, la clé correspondante sera la suivante : `my.conf | 0 | SNMP public access from 192.168.115.34 to 192.168.52.179` (les identifiants des règles débutent à zéro et le symbole de la barre sert de séparateur). Si l'opération dotée de cette clé existe bien, SEC transmettra l'évènement d'entrée à l'opération. À l'inverse, si une telle opération n'existe pas avec la clé, SEC lancera une nouvelle opération d'une durée de vie de 300 secondes. Cette opération consiste à envoyer immédiatement une alerte e-mail à l'utilisateur *root* local au moyen de l'action intitulée *pipe*

(*canal*). La chaîne de description de l'évènement indiquée par des % sera ainsi transférée par le canal vers l'entrée standard de la commande `mail -s 'Snort alert' root`, après quoi l'opération ignorera les évènements suivants reçus de SEC pour une mise en corrélation. En d'autres termes la règle permettra de réduire les messages répétés de type *'SNMP public access udp'* issus des mêmes adresses IP source et destination en un seul message (le premier).

Inclure le nom du fichier de la règle ainsi que l'identifiant de la règle dans la clé de corrélation d'évènements permet de garantir le bon fonctionnement des opérations de corrélation d'évènements déclanchées par différentes règles. De même, en choisissant la valeur correcte pour le paramètre *desc*, l'utilisateur final peut modifier l'étendue d'application de la mise en corrélation des évènements. Ainsi, si la valeur du paramètre est égale à `SNMP public access from $1`, SEC réduira l'ensemble des messages dotés du même champ d'adresse IP source en un seul message, et ce, quelle que soit l'adresse IP de destination.

Enfin, il faut être d'autant plus attentif à l'utilisation des variables spéciales $1, $2, etc. intégrées à une définition de ligne de commande que le contenu de ces variables spéciales sera interprété par le shell comme le reste de la ligne de commande. Ainsi, si le paramètre *pattern* (*modèle*) équivaut à `sshd\[\d+\]: (.+)`, et si le paramètre *action* est égal à `shellcmd echo $1 >> myfile`, un utilisateur malveillant peut alors lancer une commande arbitraire à partir de SEC en entrant une ligne falsifiée de type `sshd[0]: `mycommand`` au moyen de l'utilitaire *logger*. Afin d'éviter de telles situations, les modèles de SEC chargés de paramétrer des variables spéciales pour les lignes de commande doivent être rédigés de manière à empêcher l'affectation de métacaractères shell ou d'autres données inattendues aux variables.

Listing 5. *Ensemble de règles SEC permettant de surveiller le journal du serveur local Apache au moyen d'une liste dynamique d'expressions régulières afin de transmettre les lignes appariées vers le serveur à distance syslog*

```
type=Single
ptype=SubStr
pattern=SEC_STARTUP
context=SEC_INTERNAL_EVENT
continue=TakeNext
desc=Load the Sys::Syslog module
action=assign %a 0; eval %a (require Sys::Syslog); \
eval %a (exit(1) unless %a)

type=Single
ptype=RegExp
pattern=(SEC_STARTUP|SEC_RESTART)
context=SEC_INTERNAL_EVENT
desc=Compile the logging routine and initialize the list of patterns
action=eval %syslog ( sub { Sys::Syslog::syslog('err', $_[0]); } ); \
       eval %a ( @regexp = ('192\.168\.1\.1', 'File does not exist:'); \
                 Sys::Syslog::openlog('SEC', 'cons,pid', 'daemon') )

# ligne d'entrée mise en corrélation :
# [Fri Mar 24 09:19:50 2006] [erreur] [client 192.168.1.1]
# ce fichier n'existe pas : /var/apache/htdocs/robots.txt

type=Single
ptype=PerlFunc
pattern=sub { foreach my $pat (@regexp) {\
 if ($_[0] =~ /$pat/) { return 1; } } return 0; }
desc=Forward the suspicious message line to remote syslog server
action=call %o %syslog $0
```

Élaborer des ensembles de règles SEC à partir de règles individuelles

L'ensemble de règles présenté dans le Listing 2 visant à traiter les échecs d'authentification et les messages d'authentification réussie est un exemple bien plus complexe sensé illustrer la possibilité de paramétrer des règles de manière à interagir via l'utilisation d'évènements et de contextes synthétiques. L'objectif de cet ensemble de règles consiste à éliminer les échecs d'authentification accidentels généralement suivis peu après d'une authentification réussie, et à comptabiliser les échecs d'authentification non-accidentels afin de détecter des tentatives de piratage sur un grand nombre de comptes en une période de temps très courte tout en distinguant ces tentatives dirigées contre un ou quelques comptes.

La première règle de type *PairWithWindow* a été conçue dans le but de faire correspondre des échecs d'authentification *sshd* avec des messages d'authentification réussie issus du journal système de Solaris, */var/adm/messages*. Dès que l'expression régulière indiquée dans le paramètre *pattern* (*modèle*) correspond avec un message d'échec d'authentification d'un utilisateur donné, la variable $1 sera paramétrée sur le nom de cet utilisateur. SEC lance ensuite une opération de corrélation d'évènements, laquelle attendra un message d'authentification réussie pour le même utilisateur pendant les prochaines 300 secondes. Si le message d'authentification réussie arrive dans ce laps de temps, aucune action ne sera déclanchée (puisque le paramètre *action2* est fixé sur `none`). Il est intéressant de noter qu'avec les règles de type *Pair**, il est possible d'utiliser les variables spéciales $1, $2, etc. dans le paramètre *pattern2*, autrement dit, le modèle de la seconde moitié de la règle

*Pair** peut être dynamique. En cas d'absence de message d'authentification réussie, l'opération génèrera un évènement synthétique appelé `PAM_AUTHENTICATION_FAILED_FOR_<username>` avec l'action *event* (*évènement*). Les évènements synthétiques de SEC sont traités comme des évènements d'entrée normaux lus à partir de fichiers journaux. Ils sont ajoutés à la suite des entrées puis comparés à l'ensemble des règles.

La deuxième règle de type *SingleWithThreshold* lance une opération de corrélation d'évènements chargée de mettre en correspondance puis de comptabiliser les messages `PAM_AUTHENTICATION_FAILED_FOR_<username>`. Si 10 messages ont été observés dans une fourchette de 600 secondes, l'opération envoie une alerte e-mail à l'utilisateur *root* local, et créée également le contexte `COUNTING_OFF` avec une durée de vie d'une heure, afin d'éviter l'envoi d'alertes à l'utilisateur *root* toutes les 10 minutes lorsque le scan du compte prend beaucoup de temps. L'expression indiquée dans le paramètre *context* de la définition de la règle est le suivant : l*e contexte USER_<username>_ALREADY_COUNTED n'existe pas et le contexte COUNTING_OFF existe bien* (dans les expressions de contexte de SEC, le signe `!` indique une négation logique, `&&` l'opérateur logique ET, et `||` l'opérateur logique OU). Ainsi, en présence d'un contexte de type `COUNTING_OFF` l'expression devient fausse et la règle ne correspondra à aucun évènement. Une fois l'évènement `PAM_AUTHENTICATION_FAILED_FOR_<username>` comptabilisé, il sera passé dans la troisième règle dans la mesure où le paramètre *continue* (*continuer*) de la deuxième règle prend la valeur `TakeNext`. La troisième règle a pour objectif de créer le contexte `USER_<username>_ALREADY_COUNTED`, et, puisque la durée de vie du contexte et la fourchette de décomptage sont identiques (soit 600 secondes), le nom de chaque utilisateur distinct ne peut qu'augmenter la valeur du compteur qu'une seule fois pendant le décompte (dès que le contexte a été créé pour un nom d'utilisateur, l'expression de contexte de la deuxième règle pour le nom de l'utilisateur sera considérée fausse). En d'autres termes, grâce à l'interaction entre la deuxième et la troisième règle, les alertes e-mail ne sont envoyées qu'en cas d'incidents dans lesquels sont impliqués dix comptes utilisateur distincts.

Utiliser les contextes SEC pour consolider les évènements

Les contextes SEC permettent non seulement d'activer et de désactiver les règles, mais peuvent également servir d'entrepôt à évènements. SEC dispose, pour ce faire, de l'action *add* (*ajouter*) lui permettant d'ajouter un évènement à l'entrepôt à évènements du contexte en question, de l'action *report* (*rapporter*) afin d'acheminer l'ensemble des évènements de l'entrepôt vers les données d'entrée standard d'une commande externe, ainsi que d'un certain nombre d'autres opérations de contexte (comme déplacer des données entre des contextes et des variables spéciales SEC, par exemple). Nous allons étudier, dans cette partie, un scénario très simple d'utilisation des contextes en vue d'agréger et de rapporter des messages d'alertes de Snort IDS.

Les messages d'alerte que le démon Snort enregistre sont classés selon leur niveau de priorité de 1 à 3 (1 représentant la priorité la plus élevée et 3 la plus faible), et chaque message dispose d'un champ d'adresse IP source et destination traduisant la source et la destination du trafic réseau suspicieux. En général, dès que Snort détecte un évènement correspondant à une certaine adresse IP source, l'évènement en question sera suivi peu de temps après d'autres évènements pour la même adresse IP (ce phénomène est particulièrement vrai pour les attaques menées à l'aide d'une boîte à outils dans le but de trouver le plus de vulnérabilités possibles dans le réseau de destination). Il est donc préférable de ne pas générer d'alerte sur chaque évènement, mais au contraire de consolider ces évènements dans un nombre limité de rapports.

L'ensemble de règles présenté dans le Listing 3 a été conçu dans le but de traiter les messages d'alerte de niveau 1 issus de Snort dotés du même champ d'adresse IP source (nous désignerons, dans la suite de la présente sous-section, par *attaque* une activité réseau déclanchant de tels messages). Dès la détection du premier message de priorité 1 pour une certaine adresse IP source, SEC enverra une alerte e-mail au début de l'attaque. Si aucun message de priorité 1 n'a été détecté pendant 5 minutes pour cette adresse IP, SEC considère qu'il s'agit de la fin de l'attaque, et envoie un rapport par e-mail contenant l'ensemble des messages journaux correspondant à la dite attaque.

Afin de stocker des messages de journal pour une certaine adresse IP `<ipaddress>`, l'ensemble de règles créé le contexte `ATTACK_FROM_<ipaddress>`. La première règle a pour but de détecter le premier évènement d'une attaque : la règle correspond à un évènement de priorité 1 identifié pour l'adresse IP source en question uniquement lorsque le contexte correspondant à cette même adresse IP n'a pas encore été créé. Après avoir mis en corrélation l'évènement, la règle créé le contexte puis envoie une alerte e-mail à l'utilisateur *root* local lui indiquant qu'une attaque vient de débuter. La seconde règle correspond à un message de journal de priorité 1 et l'ajoute à l'entrepôt à évènements du contexte correspondant au moyen de l'action *add* (*ajouter*). La variable spéciale $0 contient la ligne complète du message de journal correspondante. Ensuite, la règle emploie l'action *set* (*paramétrer*) afin d'étendre la durée de vie du contexte aux 300 secondes suivantes, et paramétrer l'action ON DELETE du contexte en question (`report ATTACK_FROM_$1 mail -s 'Snort: priority 1 attack from $1 (report)' root`). L'action ON

DELETE sera exécutée immédiatement avant l'expiration de la durée de vie du contexte, et le contexte est supprimé, lorsque, par exemple, aucun évènement de priorité 1 pour une adresse IP donnée n'a été observé au cours des 300 dernières secondes. L'action ON DELETE a recours à l'action *report* (*rapporter*) afin d'acheminer l'entrepôt à évènements du contexte vers la commande `mail -s 'Snort: priority 1 attack from $1 (report)' root` chargée d'envoyer les évènements collectés à l'utilisateur *root* local. Ainsi, les attaques composées de nombreux évènements seront rapportées en un seul message électronique. Par ailleurs, en cas d'attaque longue, l'utilisateur final recevra toujours un seul message électronique à temps l'informant du début de l'attaque.

Surveiller plusieurs fichiers

Exception faite de ses fonctionnalités avancées en corrélation d'évènements et en consolidation, SEC possède également un énorme avantage sur plusieurs autres solutions de surveillance de journaux bien connues : son aptitude à surveiller plusieurs fichiers journaux simultanément. Ainsi SEC permet de mettre en corrélation de multiples évènements issus de différentes sources. De même, en cas d'un nombre important de fichiers journaux présents sur le système, ils peuvent être traités dans un seul processus SEC, lequel permet non seulement d'économiser de l'espace dans la table des processus, mais également de faciliter la maintenance de SEC lui-même (par exemple, SEC n'aura qu'un seul fichier d'identifiant de processus et qu'un seul fichier journal). Il est très facile de configurer SEC de sorte à pouvoir surveiller plus d'une entrée. Il suffit pour cela de mentionner plus d'une option `-input` dans la ligne de commande ou de spécifier un nom de fichier contenant des caractères de remplacement dans l'option `-input` (ou de faire les deux).

Toutefois, lorsque les règles deviennent trop nombreuses, disposer de plus d'une source d'entrée peut engendrer des problèmes de performance et de transparence. Si plusieurs règles ont été conçues pour une seule source d'entrée, la mise en correspondance des lignes issues d'autres sources d'entrée avec de telles règles pourrait entraîner un surdébit considérable au moment de l'exécution. Dans le même ordre d'idée, si des lignes d'entrée correspondent accidentellement à la règle avec laquelle elles n'étaient pas sensées correspondre, des effets secondaires indésirables pourraient affecter le comportement de l'ensemble de règles de manière incompréhensible pour l'utilisateur final.

Afin de contourner de tels problèmes, SEC dispose de l'option de ligne de commande intitulée `-intcontexts`, laquelle oblige SEC à créer un *internal context* (*contexte interne*) dès la lecture d'une ligne issue d'une source d'entrée, et à supprimer le contexte une fois cette ligne comparée à l'ensemble des règles. Par exemple, si le nom de la source d'entrée est */var/log/messages*, le nom du contexte interne correspondant est `_FILE_`

Références

- Jim Brown. 2003. Travaux effectués avec SEC (Simple Event Correlator). *http://sixshooter.v6.thrupoint.net/SEC-examples/article.html,*
- P. Froehlich, W. Nejdl, M. Schroeder, C. V. Damasio, L. M. Pereira. 2002. Utilisation de la programmation logique étendue à la corrélation d'alarmes sur les réseaux de téléphones portables. Intelligence appliquée 17(2), pp. 187-202,
- Boris Gruschke. 1998. Gestion d'évènements intégrés : corrélation d'évènements au moyen de graphes de dépendance. Compte-rendu du 9ème Atelier International IFIP/IEE sur les systèmes distribués : opérations et gestion, pp. 130-141,
- G. Jakobson and M. Weissman. 1995. Gestion des télécommunications en temps réel : étendre la corrélation d'évènements aux contraintes temporaires. Compte-rendu du 4ème Symposium International sur la Gestion des Réseaux Intégrés, pp. 290-301,
- Dilmar Malheiros Meira. 1997. Proposition de modèle pour une corrélation d'alarmes dans les réseaux de télécommunication. Thèse de doctorat, Université fédérale de Minas Gerais, Brésil,
- Elaine Rich et Kevin Knight. 1991. Intelligence artificielle, 2ème édition. McGraw-Hill, ISBN 0-07-052263-4,
- John P. Rouillard. 2004. Analyse en temps réel de fichiers journaux avec Simple Event Correlator (SEC). Compte-rendu de la 18ème Conférence USENIX sur l'Administration des Systèmes, pp. 133-149,
- M. Steinder et A. S. Sethi. 2002. Détection des échecs de services bout à bout au moyen des réseaux bayesiens. Compte-rendu du 8ème Symposium IEEE/IFIP sur la Gestion et les Opérations Réseau, pp. 375-390,
- James Turnbull. 2005. Renforcer Linux. Apress, ISBN: 1-59059-444-4.
- Risto Vaarandi. 2005. Outils et techniques d'analyse des journaux d'évènements. Thèse de doctorat, Université de technologie de Tallinn, Estonie,
- Hermann Wietgrefe. 2002. Enquête et analyse pratique des méthodes de corrélation d'alarmes utilisées dans les réseaux d'accès GSM. Compte-rendu du 8ème Symposium IEEE/IFIP sur la gestion et les opérations réseau, pp. 391-404,
- Hermann Wietgrefe, Klaus-Dieter Tuchs, Klaus Jobmann, Guido Carls, Peter Froehlich, Wolfgang Nejdl, et Sebastian Steinfeld. 1997. Utilisation des réseaux de neurones pour la corrélation d'alarmes sur les réseaux de téléphones portables. Compte-rendu de l'Atelier International sur les applications de réseaux de neurones dans les télécommunications, pp. 248-255,
- S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, et D. Ohsie. 1996. Corrélation d'évènements rapides et puissantes. Magazine de communication de l'IEEE 34(5), pp. 82-90.

EVENT_/var/log/messages. Dans la mesure où le nom des contextes internes peut être utilisé dans les expressions de contexte des définitions de règles, l'utilisateur peut rédiger des règles sensées correspondre aux évènements uniquement issus de certaines sources d'entrée. Si l'utilisateur souhaite disposer de noms personnalisés pour ses contextes internes ou d'un seul nom pour plusieurs sources d'entrée, il peut spécifier ces noms grâce à l'option -input. Ainsi, les options -input=/var/log/syslog=SYSLOG -input=/var/adm/messages=SYSLOG obligent SEC à employer le contexte interne SYSLOG pour à la fois */var/log/syslog* et */var/adm/messages*.

Afin d'illustrer l'utilisation des contextes internes, observez la règle *Suppress* (*Supprimer*) tirée du Listing 4 au début du fichier de règles.

La règle *Suppress* de SEC supprime, comme son nom l'indique, les évènements mis en corrélation. Cette règle agit comme un filtre dont l'objectif est de ne pas transférer les évènements aux dernières règles dans le fichier de règles. Dans la définition de règle tirée du Listing 4, les paramètres *ptype* et *pattern* permettent de préciser que le modèle est une valeur vraie (TRUE) pouvant correspondre à n'importe quelle ligne. Toutefois, l'expression du contexte !_FILE_EVENT_/var/log/messages n'est vérifiée que pour les lignes non issues de */var/log/messages*. Par conséquent, la règle peut être utilisée au début du fichier de règles destinées à surveiller */var/log/messages*, puisqu'il ne transmet que les lignes pertinentes.

Si l'option de la ligne de commande *–intcontexts* a été activée, SEC emploie le contexte interne _INTERNAL_EVENT sur les évènements synthétiques générés au moyen de l'action *event* (*évènement*). Toutefois, l'utilisateur final apprécierait parfois de disposer d'un autre contexte interne correspondant à un évènement synthétique. À titre de suggestion, il est possible de créer un canal nommé au moyen de l'outil *mkfifo*, de laisser SEC surveiller le canal nommé au moyen de l'option -input, puis d'utiliser l'action *write* (*rédiger*) à la place de l'action *event* (*évènement*) dans les définitions de règles. Par exemple, si le canal nommé */var/log/pipe* a été créé au moyen de mkfifo /var/log/pipe et si SEC a été lancé avec l'option de la ligne de commande -input=/var/log/pipe=SYSLOG, l'utilisation de action=write /var/log/pipe MY_SYNTHETIC_EVENT (ordonnant à SEC de rédiger la ligne MY_SYNTHETIC_EVENT sur */var/log/pipe*) permet de faire apparaître l'évènement MY_SYNTHETIC_EVENT dans l'ensemble du contexte interne SYSLOG.

Intégrer du code Perl personnalisé grâce aux règles SEC

Bien que les fonctionnalités de SEC évoquées plus haut permettent de rédiger des ensembles de règles adaptés à un large choix de scénarios possibles de corrélation d'évènements, il existe toujours des cas impossibles à traiter même en combinant ces fonctionnalités. Par exemple, les modèles de type *RegExp* ne permettent pas de spécifier une liste dynamique d'expressions régulières. De même, l'action *pipe* (*canal*) observée dans les précédents exemples sur les ensembles de règles implique la création d'un processus distinct pour une commande externe. Mais, lorsque l'action *pipe* est appelée une centaine de fois par seconde, le processeur va consacrer une quantité de temps considérable à passer par de nouveaux processus. Bien que SEC supporte les variables spéciales que l'utilisateur peut employer pour le stockage des valeurs, ces variables, similaires aux scalaires de Perl ainsi qu'à d'autres structures de données bien plus complexes (comme les listes et les tables de hachage sous Perl), ne peuvent être paramétrées avec de telles structures. Afin de résoudre ces problèmes, SEC supporte des modèles appelés *PerlFunc* (fonctions Perl définies par l'utilisateur destinées à mettre en corrélation les lignes d'entrée) ainsi que des expressions de contexte Perl, et les actions *eval* (*évaluation*) et *call* (*appel*) permettant de compiler et d'exécuter du code Perl personnalisé sous SEC.

L'ensemble de règles tirée du Listing 5 illustre l'emploi des actions *eval* et *call* et des modèles *PerlFunc*, ainsi que l'utilisation des modules Perl avec SEC ainsi que le paramétrage et l'accès aux structures de données Perl à l'aide d'un code personnalisé. L'ensemble de règles a été conçu dans le but de surveiller le journal d'erreurs du serveur Web Apache au moyen d'une liste dynamique d'expressions régulières, puis de transférer les lignes appariées vers le serveur à distance *syslog* sur lequel ces lignes pourraient être mises en corrélation par une autre instance de SEC. Afin d'économiser du temps processeur, cet ensemble de règles n'appelle pas l'utilitaire *logger* pour le transfert des lignes en tant que messages *syslog*, mais repose à la place sur les fonctions openlog() et syslog() du module Perl *Sys::Syslog*.

Afin de tirer avantage du module *Sys::Syslog*, celui-ci doit être chargé dès le lancement de SEC. Si SEC a été lancé avec l'option de ligne de commande -intevents, l'outil génèrera un évènement synthétique appelé SEC_STARTUP en tant que tout premier évènement au démarrage, paramètrera le contexte interne SEC_INTERNAL_EVENT pour l'évènement, puis le traitera avant tout autre évènement d'entrée. L'utilisateur peut ainsi rédiger des règles chargées d'exécuter diverses procédures de lancement. La première règle consiste à tenter de charger le module *Sys::Syslog* à l'aide des actions *assign* (*assigner*) et *eval* (*évaluer*). Elle commence par paramétrer la variable spéciale %a sur 0 grâce à l'action *assign*, puis évalue le code Perl require Sys::Syslog avec l'action *eval* (en interne, l'action *eval* appelle la fonction Perl eval()). Si le

résultat de l'action *eval* est bon, et si le module est bien chargé, la valeur 1 sera affectée à la variable spéciale %a (puisque cette valeur est retournée par le code validé `require Sys::Syslog`). À l'inverse, en cas d'échec de l'action *eval*, %a conservera sa valeur d'origine (0). L'action *eval* sera ensuite réutilisée afin de contrôler la valeur de %a. Si celle-ci est restée sur 0 (autrement dit, le module n'a pas pu être chargé), la fonction `exit(1)` est appelée depuis le code Perl exécuté par *eval*. Dans la mesure où son exécution se déroule au sein du processus SEC, `exit(1)` fermera SEC avec le code de sortie 1.

La deuxième règle a été conçue dans le but de mettre en corrélation les deux évènements internes `SEC_STARTUP` et `SEC_RESTART` (lorsque SEC a été lancé avec l'option `-intevents` et reçoit le signal SIGHUP, requête de reparamétrage de l'état interne et de chargement de la configuration, SEC génère un évènement synthétique appelé `SEC_RESTART` au moyen du contexte interne `SEC_INTERNAL_EVENT`). Après avoir observé un évènement apparié, la règle commence par utiliser l'action *eval* afin d'évaluer le code Perl `sub { Sys::Syslog::syslog('err', $_[0]); }`. Dans la mesure où ce code traduit la définition d'une fonction, l'action *eval* compilera la fonction puis retournera le pointeur vers le code compilé, lequel sera sauvegardé dans la variable spéciale %syslog. Cette fonction attend un paramètre d'entrée et utilise la fonction `syslog()` du module *Sys::Syslog* afin d'envoyer les paramètres d'entrée sous forme de message de niveau *err* vers le serveur *syslog*. La règle se chargera ensuite d'initialiser la liste *@regexp*, liste Perl contenant les expressions régulières. Comme *@regexp* est une liste globale, elle est accessible et modifiable au moyen d'appels ultérieurs vers l'action *eval*. (Afin d'éviter toute erreur avec le nom des variables dans le code SEC, un espace de nommage séparé appelé `main::SEC` est défini dans le code SEC, et l'action *eval* évalue toujours le code Perl personnalisé dans cet espace de nommage). Enfin, la règle ouvre la connexion *syslog* au moyen de la fonction `openlog()`, en paramétrant le nom du programme sur `SEC`, l'utilitaire de connexion sur `daemon`, et en activant les options sur `cons,pid` (connectez vous sur la console en cas d'échec de connexion normale puis incluez l'identifiant du processus à chaque message).

La troisième règle a pour objectif de mettre en corrélation les lignes d'entrée avec les expressions régulières issues de la liste *@regexp* initialisée par la deuxième règle (cette liste peut être modifiée par d'autres règles au moment de l'exécution). Cette règle a recours au modèle *PerlFunc* pour le processus de corrélation. La valeur du paramètre *pattern* doit être une définition de fonction Perl valide compilée au moment du chargement des règles. Dans le cas de la troisième règle, la fonction prend la ligne d'entrée (passée dans la fonction en tant que paramètre d'entrée $_[0]) puis effectue un scan sur la liste *@regexp* afin de détecter une expression régulière correspondante. Si une telle expression régulière est trouvée, la fonction retournera la valeur 1, sensée indiquer que le modèle *PerlFunc* correspond à la ligne d'entrée. Dans le cas contraire, la fonction retourne la valeur 0 en cas de non correspondance. Dans ce cas, la règle appellera une fonction Perl précompilée pour la connexion à *syslog* grâce à l'action *call* (*appel*). La variable spéciale %o permet de stocker la valeur de retour issue de l'appel vers la fonction, la variable spéciale %syslog contient un pointeur dirigé vers cette fonction, et $0 (qui contient la ligne d'entrée complète correspondante) représente le paramètre d'entrée de la fonction.

Ainsi, l'ensemble de règles parvient à implémenter de manière efficace l'expression régulière dynamique correspondant à un journal d'erreurs d'un serveur Web, lequel ne peut être exprimé au moyen des modèles *RegExp*, et à transmettre les lignes appariées au serveur à distance *syslog* sans passer par un processus séparé pour une commande externe. Comme l'ensemble des fragments de code Perl employés dans la troisième règle sont compilés au démarrage de SEC, leur exécution au moment de l'exécution est aussi efficace que l'exécution du code SEC lui-même.

Performance et applications concrètes avec SEC

Bien que SEC soit écrit dans un langage interprété (et manque donc de rapidité et de mémoire par rapport aux programmes C compilés), cet outil est capable de gérer des centaines d'évènements par seconde tout en ne consommant que très peu de ressources. Lors d'un essai mené récemment sur 49,8 jours, deux instances de SEC ont été paramétrées de manière à fonctionner sur un serveur *syslog* Linux au moyen de deux processeurs Intel P4 Xeon de 3 GHz. La première instance était chargée de surveiller 20 fichiers journaux simultanément à l'aide de 243 règles configurées dans 22 fichiers de règles, alors que la seconde instance devait lire les entrées issues d'un canal nommé au moyen de 67 règles configurées dans 5 fichiers de règles. La première instance a traité 107 059 511 lignes d'entrée (soit 24,9 lignes par seconde en moyenne), et consommé 3,0% du temps de l'unité centrale et 8,1 Mo de mémoire. La seconde instance a traité 364 534 428 lignes d'entrée

Remerciements

Le présent travail reçoit le soutient de SEB Eesti Ühispank, et est financé par la bourse nationale estonienne n° SF0182712s06.

À propos de l'auteur

Risto Vaarandi est titulaire d'un doctorat en Ingénierie Informatique délivré par l'Université de Technologie de Tallinn, Estonie, en juin 2005. Depuis ces huit dernières années, Risto Vaarandi travaille chez SEB Eesti Ühispank en tant qu'ingénieur en développement informatique, et occupe également un poste à mi-temps de chercheur auprès de l'Institut d'Informatique de l'Université de Tartu, Estonie. Vous pouvez contacter l'auteur sur son site personnel à l'adresse suivante : *http://kodu.neti.ee/~risto*.

(soit 84,7 lignes par seconde en moyenne), et consommé 8,8% du temps de l'unité centrale et 6,1 Mo de mémoire.

La vitesse de traitement des évènements de SEC dépend largement de l'arrangement des règles. La performance de l'outil peut être améliorée de plusieurs façons. Dans la mesure où les lignes d'entrée sont comparées aux règles afin qu'elles soient définies dans un fichier de règles, placer les règles les plus souvent appariées en tête du fichier permet de gagner du temps de l'unité centrale. Dans le même ordre d'idée, lorsque un certain nombre de lignes d'entrée ne correspondent à aucune règle, disposer d'une règle *Suppress* pour de telles lignes en tête du fichier de règle engendre le même effet sur l'unité centrale. Si SEC a été configuré de manière à surveiller plusieurs sources d'entrée, il est possible d'utiliser les contextes internes (tels que décrits dans la partie intitulée *Surveiller plusieurs fichiers*) afin d'augmenter la vitesse de traitement des évènements de SEC. D'autres pistes permettant d'améliorer la performance de SEC consistent à rédiger des expressions régulières efficaces, ou à remplacer les modèles *RegExp* par des modèles *SubStr* autant que possible (ces derniers modèles sont en effet plus rapides).

Au cours de ces dernières années, SEC a été adopté par de nombreux organismes à diverses envergures et a été employé dans un certain nombre de domaines dont la surveillance des journaux d'évènements, la gestion des pares feu, la détection d'intrusions, et la gestion des réseaux (veuillez consulter la référence [Vaarandi 2005] pour des études de cas plus détaillées). SEC a été utilisé avec succès sur Snort IDS, Prelude IDS, le pare feu iptables, HP OpenView (NNM et Operations), Nagios, CiscoWorks, BMC patrol, SNMPTT, etc. SEC a également été appliqué à un large choix de plateformes de systèmes d'exploitation, dont Linux, FreeBSD, OpenBSD, Solaris, HP-UX, AIX, Tru64 Unix, Mac OS X, et Windows 2000.

Sur Internet

- *http://www.bmc.com/* – BMC Patrol,
- *http://www.cisco.com/* – CiscoWorks,
- *http://www.managementsoftware.hp.com/products/ecs/index.html* – HP ECS,
- *http://www.openview.hp.com/* – HP OpenView,
- *http://www.netfilter.org/* – Iptables,
- *http://www.logec.com/* – LOGEC,
- *http://www.cert.dfn.de/eng/logsurf/* – Logsurfer,
- *http://www.mysql.com/* – MySQL,
- *http://www.nagios.org/* – Nagios,
- *http://www.openservice.com/products/nervecenter.jsp* – NerveCenter,
- *http://www.micromuse.com/* – NetCool,
- *http://www.prelude-ids.org/* – Prelude IDS,
- *http://www.rulecore.com/* – RuleCore,
- *http://simple-evcorr.sourceforge.net/* – Simple Event Correlator,
- *http://www.smarts.com/* – SMARTS,
- *http://snmptt.sourceforge.net/* – SNMPTT,
- *http://www.snort.org/* – Snort IDS,
- *http://swatch.sourceforge.net/* – Swatch,
- *http://www.balabit.com/products/syslog_ng/* – Syslog-ng.

Conclusion

Nous venons de présenter dans le présent article l'outil SEC (Simple Event Correlator), solution open source spécialisée dans la corrélation d'évènements légère et multi-plateforme, au travers de plusieurs exemples concrets illustrant les possibles utilisations de SEC afin de surveiller en temps réel des journaux d'évènements de sécurité. Toutefois, en raison des contraintes spatiales inhérentes à l'édition, de nombreuses fonctionnalités de SEC n'ont pu être exposées ici. Les lecteurs intéressés pourront donc consulter la documentation en ligne de SEC pour une description plus poussée. Ils pourront également consulter plusieurs autres sources d'informations disponibles sur SEC. L'archive des règles SEC chez BleedingSnort (*http://www.bleedingsnort.com/sec/*) contient un certain nombre d'exemples d'ensembles de règles applicables à divers scénarios (comme, par exemple, la corrélation d'évènements sur Snort ou la gestion du pare feu iptables). *Working with SEC – the Simple Event Correlator* [Brown 2003] est un tutorial en ligne où les lecteurs pourront non seulement lire une excellente introduction sur SEC mais s'entraîner également sur un grand nombre de questions avancées comme l'intégration de SEC avec MySQL. Le chapitre 5 de *Hardening Linux* [Turnbull 2005] explique comment utiliser SEC afin de surveiller les fichiers journaux *syslog*. Enfin, un article plus récent présentant une bibliothèque d'ensembles de règles très utiles vient d'être publié. Une application de SEC menée à l'Université du Massachusettes à Boston y est notamment décrite [Rouillard 2004]. ●