Exploitation des logs pour des mesures pro-actives

quelques pistes

C. Labourdette[1]

[1]Centre de Mathématiques et de de Leurs Applications
Ecole Normale supérieure de Cachan / CNRS

Le 19 juin 2008 / TutoJres

Le contenu

le cadre

- Un administrateur système et réseau
- Un système essentiellement Linux
- Des dispositifs de surveillance
- Des dispositifs de sécurité
- Autant que possible une solution libre et Open Source

Les logs

A chaque service sa gamme de logs

- Le Courrier électronique
- Le web
- Les connexions (ssh)
- Les transferts de fichier
- Les IDS
- Les IPS
- Les firewalls
- Les applications maison

brute force sur ssh

Ssh fait son apparition en 1995 en Finlande et devient au fil du temps Le protocole/logiciel de connexion.
En 2005 grosse augmentation des attaques par brute force.
==> des logs du type :

log

sshd[14156] : Invalid user truc from xxx.xxx.xxx.xxx
sshd[14156] : Failed password for invalid user truc from xxx.xxx.xxx.xxx port 59505 ssh2

Que faire ?

En amont

Changer de port, rajouter des contraintes pour l'établissement (ssh-knock) ?
==> Compliqué lorsqu'il y a beaucoup d'utilisateur.

En aval

Réagir d'après les logs ?
==> Moins problématique pour les utilisateurs, mais se passe après l'attaque..

réaction

Principe simple : tentatives de connexion + utilisateurs inconnus
==> on bloque les connexions provenant de l'adresse
ATTENTION ! Ne pas laisser l'attaquant bloquer le service ...
Donc, principe de précaution : il faut être sûr pour agir, on attend plusieurs logs du même type pour bloquer une adresse.
Exemples de logiciels :
DenyHosts, Fail2ban, BlockHosts, SSH_Blocking, ...
Défaut : les attaques par log injection ?

DenyHosts

Programme en python (Phil Schwartz) :
Bloque les attaques ssh en ajoutant des instructions dans le fichier */etc/hosts.deny* (http ://denyhosts.sourceforge.net)
Comportement basique et configuration minimale :

- Fichier de config */etc/denyhosts.conf*
- Utilisation du cron
- Alertes par mail

fail2ban

Programme en python (Cyril Jaquier) :
Bloque les adresses qui rencontrent trop d'échec d'authentification, en ajoutant des règles dans */etc/hosts.deny* mais aussi au firewall (iptables)
http ://www.fail2ban.org
Il peut être utilisé pour tout service utilisant une authentification :

- Répertoire de config */etc/fail2ban*
- Notion de filtre et d'action associée
- Bannissement paramêtrable en temps

log injection

Imaginons un serveur ssh sur la machine 192.168.80.110

Injection

192.168.80.16 : $ nc 192.168.80.110 22
SSH-2.0-OpenSSH_4.7p1 Debian-12 ROOT LOGIN REFUSED hi FROM 1.2.3.4 *protocol mismatch*

résultat

sshd[26980] : Bad protocol version identification 'ROOT LOGIN REFUSED hi FROM 1.2.3.4' from 192.168.80.16

Attention aux expressions régulières !

La collecte

- syslog depuis 1986 normalisé par l'IETF depuis 2001
- aujourd'hui syslog-ng ...
- parfois des formats maison
- outils de surveillance dont le but est de générer des logs
- Le point important : centralisation et collecte des logs

Les outils d'analyse

- Il faut connaitre son système et son réseau
- importance de l'analyse !
- Pour comprendre les logs il faut les étudier
- De très nombreux outils permettant d'analyser les logs (de façon plus ou moins statique) : Logwatch, logsurfer, logsurfer+, swatch, 2swatch ...
- De nombreux produits commerciaux ...
- Mais aussi : SEC

SEC

- SEC : Simple Evenement Correlator
- Ecrit en perl (en 2000) par Risto Vaarandi
- permet entre autre de faire des ”corrélations” (le terme est impropre ...) entre des évènements
- Le produit OpenSource utilisé par tous
- Architecture logicielle simple (un seul programme)
- Configuration qui peut être très sophistiquée

Grandes lignes

- Portable (Perl) et intégrable (par exemple dans nagios)
- Une approche par utilisation de règles
- Evènements mis en “corrélation” par *condition –> action*
- Expressions régulières en Perl
- Réception évènements de flux de fichiers (qui peuvent être multiples)
- l’Action peut être l’exécution d’un sous-programme Perl
- Notion de contexte de corrélation

Règles

- Single : action dès l'observation d'un évènement mis en corrélation
- SingleWithScript : idem à Single mais scripts externe pour la corrélation
- SingleWithSuppress : idem à SingleWithScript mais ignore les évènements consécutifs pendant *n* secondes
- Pair : Soient deux évènements A et B, exécute des actions sur A en ignorant les instances multiples et exécute d'autres actions lorsque B se produit

Règles(suite)

- PairWithWindow : si observation de A attend *n* secondes que B se produise ; si B se produit on exécute des actions sinon on exécute d'autes actions
- SingleWithThreshold : Compte les évènements pendant *n* secondes si supérieur au seuil alors actions
- SingleWith2Threshold : idem au précédant mais il y à un deuxième tour de secondes avec un seuil décroissant
- Suppress : supprime les évènement mis en corrélation
- Calendar : exécute des actions à un moment donné

Type de Pattern

- SubStr : une sous-chaîne de caractères
- NSubStr : même chose que SubStr mais négation du résultat
- RegExp : une expression régulière
- NRegExp : même chose que RegExp mais négation du résultat
- PerlFunc : une fonction perl
- NPerlFunc : même chose que PerlFunc mais négation du résultat
- TValue : une valeur vraie ou fausse

Action

Les actions peuvent être multiples, il suffit de les séparer par un point virgule.
Il faut préciser le type de l'action
par exemple (la liste n'est pas exhaustive) :

- none : aucune action
- shellcmd : des commandes du shell (ou un script shell)
- spawn : idem au shellcmd excepté le fait que le résultat du shell est réinjecté dans SEC
- write : écriture dans un fichier
- pipe : un pipe sur la sortie standard

Action (suite)

- event : insertion d'un évènement dans SEC (feedback)
- call : appel d'une fonction perl
- assign : assign un texte à une variable
- eval : évaluation d'une suite de commandes en perl, le résulat est placé dans une variable

Contexte

Il est possible d'utiliser la notion de contexte nommé.
type=Single
ptype=RegExp
pattern=un
context=CONTEXTE_UN
desc=$0
action=write - Le contexte existe on ecrit un
type=Single
ptype=RegExp
pattern=go
desc=$0
action=create CONTEXTE_UN

Exécution

```
$sec.pl -input=- -conf=test3.sec
SEC (Simple Event Correlator) 2.4.2
Reading configuration from test3.sec
2 rules loaded from test3.sec
un
un
go
Creating context 'CONTEXTE_UN'
un
Writing event 'Le contexte existe on ecrit un' to file -
Le contexte existe on ecrit un
^C
```

Action et contexte

- create : création d'un contexte (il peut avoir une durée de vie)
- add : ajoutd'un évènement à un contexte
- report : les évènements d'un contexte sont donnés à une commande du shell (/bin/cat par exemple)
- fill : ajoute un évènement à un contexte après avoir effacé le contenu de celui-ci
- delete : effacer un contexte
- copy : les évènements d'un contexte sont copiés dans une variable

Exemple NFS

```
type=Pair
ptype=RegExp
pattern=NFS server (\S+) not responding
desc=$1 is not responding
action=shellcmd notify.sh "%s"
ptype2=substr
pattern2=NFS server $1 ok
desc2=$1 OK
action2=shellcmd notify.sh "%s"
window=3600
```

Exemple ssh

type=SingleWith2Thresholds
ptype=RegExp
pattern=sshd.* : Failed password for.* user .* from (\S+)
desc=Interdiction de ssh depuis $1
action=logonly ; shellcmd (/sbin/iptables -I FORWARD -s $1 -p tcp –dport 22 -j DROP ;/sbin/iptables -I INPUT -s $1 -p tcp –dport 22 -j DROP)
window=60 thresh=4 desc2=Autorisation du ssh depuis $1
action2=logonly ; shellcmd (/sbin/iptables -D FORWARD -s $1 -p tcp –dport 22 -j DROP ;/sbin/iptables -D INPUT -s $1 -p tcp –dport 22 -j DROP)
window2=7200
thresh2=0

Exemple portknocking

```
type=Pair
ptype=RegExp
pattern=\S+\s+\d+\s+\S+\s+(\S+)\s+/kernel : Connection \
attempt to (\S+) (\S+) :12345 from (\S+) :(\S+).*
desc=$0
action=write - Knock 1....
ptype2=RegExp
pattern2=\S+\s+\d+\s+\S+\s+(\S+)\s+/kernel : Connection \
attempt to (\S+) (\S+) :54321 from (\S+) :(\S+).*
desc2=$0
action2=write - Knock 2 ! Lancement sshd pour 30 s... ; \
shellcmd ./sesameouvretoi.sh ;
window=30
```

PortKnocking (suite)

Le fichier sesameouvretoi.sh contenant quelque chose comme :

sesameouvretoi.sh

```
# !/bin/sh
/usr/bin/sshd -d -g 15 > /dev/null 2>&1 &
pid=$ !
sleep 30
kill -TERM $PID
exit 0 ;
```

SEC usage

Quelques usages de SEC

- Séparer ou regrouper des logs
- Faire des rapports ou des statistiques
- “Nettoyer” les logs
- Prévenir en cas d’anomalie ou d’attaque
- Agir directement à la suite d’une succession d’évènement
- construire des chaines de traitement originales (portknocking, ...)

Comprendre et étudier

- Se fixer un objectif clair
- Analyser et comprendre tous les logs en relation
- Comprendre parfaitement les tenants et les aboutissants des applications
- Etudier en détail les protocoles mis en oeuvre
- Bien réfléchir à tous les impacts possibles avec d'autres applications
- Décrire le schéma, *[conditions] –> [actions]*, de l'administrateur avant de le copier
- Si possible construire des schémas et diagrammes temporels

Conseils

- Bien faire les choses dans l'ordre
- Simplicité avant tout
- Le principe de précautions est utile
- Il s'agit simplement d'automatiser un comportement humain, surtout pas de magie ;-)
- Attention aux outils maison
- Toujours simuler avant de rentrer en production

conclusion

- Il ne faut pas oublier que l'Informatique est un outil.
- Le but est d'automatiser le travail de l'administrateur.
- SEC est un véritable outil pour traiter et exploiter les logs.
- Aussi bien pour construire un traitement autonome qu'intégré.
- Toujours comprendre avant d'agir.

Documentation autour de SEC

- Article dans hakin9 N˚18 (N˚5/2006) de Risto Vaarandi
- Article dans MISC N˚22 (Nov/Déc 2005) : Des outils libres pour superviser la sécurité
- La page de Risto Vaarandi http ://kodu.neti.ee/ risto
- SEC sur sourceforge http ://simple-evcorr.sourceforge.net/
- La page personnelle de John P. Rouillard http ://www.cs.umb.edu/ rouilj/
- Un tutorial de Jim Brown http ://sixshooter.v6.thrupoint.net/SEC-examples/article.html
- Le chapitre 5 de Hardening Linux par James Turnbull http ://www.apress.com/resource/bookfile/2031