CR RKHUNTER

Nicolas Vadkerti

4 décembre 2019

https://github.com/SlaynPool/CR_Rkhunter-

1 Définitions

1.1 Rootkit

Un rootKit est un, ou plusieurs logiciels dont le but est de péréniser un accés. Le plus souvent, il sert a dissimuler la porte ouverte sur une machine une fois l'accés à celle-ci obtenue.

1.2 0days

Une faille 0 days est un bug d'un logiciel qui laisse des failles sur une machines. On l'appelle 0days car il n'est pas encore patché voir connu par les dévellopeurs du logiciel. Les 0days sont donc très dangereuse. De fait, quand une faille 0days est découverte, on ne peut savoir depuis quand elle est realisable, ni si quelqu'un l'a exploité. On peut citer Hearthbleed qui utiliser une faille OpenSSL comme exemple d'ancienne faille 0 days qui à fais du bruit.

1.3 Virus

Le principe d'un virus Informatique est exactement le même q'un virus Biologique. On peut donc le définir comme un programe qui à pour but de se répliquer, souvent dans d'autres hôtes d'un réseau, dans le but de prendre le controle des machines. L'utilisations peut etre par exemple d'obtenir des bots pour des attaques par deni de service.

1.4 CVE

Common Vulnerabilities and Exposures Le CVE est un dictionnaire de faille Informatique connu maintenue par le MITRE, un organisme publique Americain. Ce dictionnaire sert de "norme " aux outils de sécurté comme les antivirus par exemple.

1.5 La notion de Hash

Le principe d'un hash, est simple à comprendre. L'idée est d'avoir un algorithme qui prend un fichier quelconque en entrée, et renvoie une signature de celui de taille constante et surout unique. L'idée est donc de, ci je calcule le hash d'un fichier, si je modifie mon fichier, le hash sera completement different. On peut donc s'assurer que personne n'a modifier mon fichier grace à ce hash. Un algorithme de hashage n'est plus considéré comme fiable, à partir du moment où pour deux fichiers differents, nous somme capable de générer des hashs partiellement similaires ou entierement similaires

2 Protection contre les Rootkit : le cas de rkhunter

2.1 Installation de rkhunter

```
[slaynpool@MiniZbeub]~$ cd .env/build/
[slaynpool@MiniZbeub]build$ cp ~/Telechargements/rkhunter-1.4.6.tar.gz .
[slaynpool@MiniZbeub]build$ tar -xvf rkhunter-1.4.6.tar.gz
...
[slaynpool@MiniZbeub]build$ cd rkhunter-1.4.6/
[slaynpool@MiniZbeub]rkhunter-1.4.6$ su - root
Mot de passe :
[root@MiniZbeub]/home/slaynpool/.env/build/rkhunter-1.4.6# ./installer.sh --install
```

Listing 1 – Installation de RKHUNTER

Commande pour utiliser RKHUNTER : Le fichiers de configuration principal de rkhunter est : `/etc/rkhunter.conf` Les logs générer par rkhunter sont ici : `/var/log/rkhunter.log`

```
# Effectuer les verifications avec rkhunter
rkhunter -c
# Mettre a jour les fichiers de rkhunter
rkhunter --update
# Verifier que la config de rkhunter est bonne
rkhunter -C
```

Listing 2 – Commade pour utiliser RKHUNTER

Pour Fonctionner, rkhunter connait les condensés des fichiers importants qu'il récupère en ligne. Cela lui sert donc pour vérifier l'integrité des fichiers crusiaux. De plus, il est capable de détécter si des chaines de caractère suspect sont present dans les binaires.

Les signatures de rkhunter sont stockés dans `/var/lib/rkhunter/db/`. Dans les faits, meme si les fichiers ne sont accessible seulement par root :

```
[root@MiniZbeub]/var/lib/rkhunter# ls -all
total 20
drwx------  4 root root 4096  4 dc.  11:33 .
drwxr-xr-x 47 root root 4096  4 dc.  11:33 ..
drwx------  4 root root 4096  4 dc.  13:34 db
drwx------  2 root root 4096  4 dc.  13:48 tmp
```

Listing 3 – Droits du dossier "signature"

on ne peut pas considérer q'un potentiel attaquant n'ai pas accés à l'user root. C'est pour cela qu'il est recommandé d'update la base. De plus, on peut imaginer stocké la db sur un stockage amovible, et le stocké dans un endroit sûr style un coffre-fort.

3 Mise au point d'un rootkit élémentaire

3.1 Modification de ls

Donc la partie "difficile " est de compilé coreutile, ou l'on trouve le src de ls :

```
git clone https://github.com/coreutils/coreutils
cd coreutils
#installer gnulib & gperf Cf votre distrib
./bootstrap
./configure
make -j6
```

Listing 4 – Compilation

Maintenant, nous pouvons modiifer les sources de ls :

```
 diff trucls.c src/ls.c
1582a1583
>
1918a1920
>       int quattre;
1921,1922c1923,1924
<                             "abcdfghiklmnopqrstuvw:xABCDFGHI:LNQRST:UXZ1",
<                             long_options, &oi);
---
>                             "abcdfghiklmnopqrstuvw:xABCDFGHI:LNQRST:UXZ142",
>                             long_options, &oi);
1927a1930,1940
>         case '4':
>
>           quattre = 1;
>           break;
>
>         case '2':
>           if ( quattre == 1 ){
>             printf("Hello IUT\n");
>             break;
>         }
```

```
>
```

Listing 5 – Ls custom

On recompile avec make et c'est gagné !

```
[slaynpool@MiniZbeub]coreutils$ ./src/ls -42
Hello IUT
ABOUT-NLS config.status  HACKING      NEWS         tests
aclocal.m4  configure      init.cfg     po           thanks-gen
AUTHORS   configure.ac   INSTALL      README         THANKS.in
autom4te.cache  COPYING        lib      README-hacking       THANKStt.in
bootstrap dist-check.mk  m4     README-package-renamed-to-coreutils  TODO
bootstrap.conf  doc        maint.mk     README-prereq      trucls.c
build-aux gl         Makefile     README-release
cfg.mk    gnulib         Makefile.am  README-valgrind
ChangeLog gnulib-tests   Makefile.in  scripts
config.log  GNUmakefile    man      src
```

Listing 6 – Ls custom stdout

On peut donc placer notre nouveau ls dans /bin et rexecuté rkhunter