Cours :

IFT-3830 : A-2023

Chargé de cours :

Daniel Ouimet

Travail :

T.P.-3

Date remise :

Samedi le 16 décembre 2023

Équipe:

Gueorgui Poklitar

Ion Hincu

tcpdumpscan.pl :

```perl
#!/usr/bin/perl

use strict;
use warnings;

# Initialisation des structures d'entreposage.
my %protocoles;
my %destinations;
my %uids_nfs;
my %reponses_arp;
my $temps_debut;
my $temps_fin;

# glob pour trouver le fichier tcpdump.* (peut trouver .txt, .log, etc.)
my @fichiers = glob 'tcpdump.*';
die "Fichier tcpdump n'est pas trouvé" unless @fichiers;

# Lecture du fichier tcpdump...
foreach my $fichier (@fichiers) {
    open(my $fh, '<', $fichier) or die "Impossible d'ouvrir $fichier: $!";

    while (my $ligne = <$fh>) {
        if ($ligne =~ /^\s*\d+\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(.*)$/) {
            my $temps = $1;
            my $source = $2;
            my $destination = $3;
            my $protocole = $4;
            my $info = $6;

            # Temps de début et de fin
            $temps_debut = $temps if not defined $temps_debut;
            $temps_fin = $temps;

            # Comptage des protocoles
            $protocoles{$protocole}++;

            # Comptage des destinations
            $destinations{$destination}++;

            # Extraction des UIDs pour le protocole NFS
            if ($protocole eq 'NFS' && $info =~ /uid:(\d+)/) {
                $uids_nfs{$1}++;
            }

            # Capture des réponses ARP
            if ($protocole eq 'ARP' && $ligne =~ /ARP\s+(\d+\.\d+\.\d+\.\d+)\s+is\s+at\s+(\S+)/) {
                $reponses_arp{"$1 -> $2"} = 1;
            }
        }
    }

    close($fh);
}
# Exportation des rapports.
generate_report('destinations.txt', \%destinations);
generate_report('proto.txt', \%protocoles);
generate_report('nfsuid.txt', \%uids_nfs);

# Impression des réponses ARP et la durée de temps.
my $total = $temps_fin - $temps_debut;
print "Réponses ARP:\n";
foreach my $reponse (sort{ lc($a) cmp lc($b) } keys %reponses_arp) {
    print "$reponse\n";
}
print "\nTemps de début et de fin: $temps_debut à $temps_fin / Total de $total secondes\n";

sub generate_report {
    my ($fichier, $donnees_ref) = @_;
    open(my $fh, '>', $fichier) or die "Impossible d'ouvrir $fichier: $!";

    foreach my $cle (sort { custom_sort($a, $b) } keys %{$donnees_ref}) {
        print $fh "$cle: $donnees_ref->{$cle}\n";
    }

    close($fh);
}

sub custom_sort {
    my ($a, $b) = @_;

    my %special_cases = (
        'broadcast' => 1,
        'spanning-tree-(for-bridges)_00' => 1,
    );

    my $a_special = exists $special_cases{lc($a)};
    my $b_special = exists $special_cases{lc($b)};

    if ($a_special == $b_special) {
        return lc($a) cmp lc($b);
    }

    return $a_special ? 1 : -1;
}
```

Voici un Screenshot du code perl utilisé dans le VM. On initialise d'abords les modes d'entreposage. On trouve un fichier qui commence par « tcpdump. » la fin du fichier n'est pas définie pour adresser l'éventualité qu'il soit dans un format diffèrent .txt ou .log toutes les options devrait fonctionner.

ls avant **tcpdumpscan.pl** :

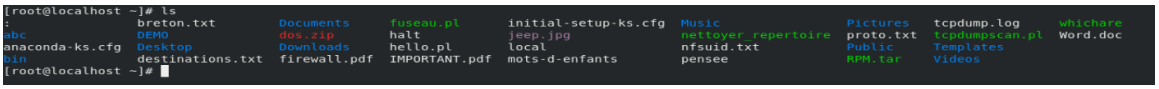
```
[root@localhost ~]# ls
:                breton.txt  dos.zip       halt                jeep.jpg        nettoyer_repertoire  RPM.tar          Videos
abc              DEMO        Downloads     hello.pl            local           pensee               tcpdump.log      whichare
anaconda-ks.cfg  Desktop     firewall.pdf  IMPORTANT.pdf       mots-d-enfants  Pictures             tcpdumpscan.pl   Word.doc
bin              Documents   fuseau.pl     initial-setup-ks.cfg Music          Public               Templates
[root@localhost ~]# vi tcpdumpscan.pl
[root@localhost ~]#
```

On fait le test pour démontrer que :

1. tcpdump.log existe dans le même endroit ou tcpdumpscan.pl habite.
2. Les fichiers (proto, destinations et nfsuid).txt … n'existent pas encore.

Après avoir donné la permission avec chmod… <u>On exécute le script!</u>

ls après **<u>tcpdumpscan.pl</u>** :



Ici on voit que :

1. Le script à effectivement générer les trois rapports qui n'existaient pas avant le lancement du script. (proto.txt, destinations.txt et nfsuid.txt)

Voici les résultats des trois rapports…

proto.txt :

```
AFP: 190
ARP: 62
DNS: 10
DSI: 12
HSRP: 9
HTTP: 69
IMAP: 45
NFS: 3249
NFSACL: 12
NLM: 42
Portmap: 22
RPC: 22
STP: 7
Syslog: 3
TCP: 6173
TLSv1: 10
YPSERV: 132
~
~
~
~
~
"proto.txt" 17L, 157C
```

destinations.txt :

```
132.204.90.167: 2
132.204.90.171: 1
132.204.90.176: 2
132.204.90.179: 2
132.204.90.217: 1
132.204.90.25: 3
132.204.90.47: 2
132.204.90.49: 1
132.204.90.50: 5
132.204.90.91: 2
132.204.90.92: 2
132.204.90.93: 3
132.204.90.94: 1872
132.204.90.95: 1
132.204.90.96: 1
132.204.90.97: 1
132.204.90.98: 92
132.204.90.99: 2
174.89.254.10: 1
205.151.3.63: 13
207.46.195.241: 6
207.46.204.238: 4
224.0.0.2: 9
65.55.226.140: 1
65.55.37.62: 1
AsustekC_2a:f2:81: 1
AsustekC_4c:0a:3c: 1
AsustekC_78:a4:dc: 1
AsustekC_78:a5:c3: 1
AsustekC_78:a7:39: 1
AsustekC_88:01:66: 1
e0:cb:4e:9b:66:70: 1
Intel_d3:cb:82: 1
Intel_d4:35:11: 1
Intel_d4:35:4e: 1
Intel_d4:37:71: 1
Intel_d4:38:44: 1
Intel_e1:78:56: 1
Intel_e1:a5:db: 1
Intel_e1:a5:e9: 1
Intel_e1:a5:ea: 1
Intel_e1:a6:37: 1
IntelCor_2a:16:c8: 1
IntelCor_2a:1b:df: 1
IntelCor_2a:1c:bb: 1
IntelCor_2a:1d:7b: 1
IntelCor_2a:e0:93: 1
IntelCor_2a:e7:d1: 1
IntelCor_2a:e8:0a: 1
IntelCor_2e:0f:2c: 26
IntelCor_37:7c:e1: 1
TyanComp_74:ea:6b: 1
Broadcast: 10
Spanning-tree-(for-bridges)_00: 7
```

nfsuid.txt :

```
0: 4
1380: 413
2220: 72
2477: 4
2559: 1
2844: 1
308: 1
3148: 2
4482: 5
4520: 170
4528: 2
615: 1
832: 3
~
~
~
~
~
~
~
~
~
~
~
"nfsuid.txt" 13L, 103C
```

proto.txt :

Va compter tous les signaux par type de protocoles utilisées. Le tcpdump compte exactement 10069 lignes. Si on fait la somme toutes les occurrences par type, on arrive exactement à 10069 occurrences. C'est-à-dire que chaque ligne à effectivement été lus, prise en compte et classé avec le bon type. Les protocoles sont en ordre alphabétique.

destinations.txt :

À noter le Screenshot de destination est effectivement limité par sa longueur. Voir le rapport complet destination.txt dans le zip du TP3… Cependant il est important de remarquer que toutes les adresses IP et MAC sont en ordre alphabétique. Par contre, il y a des exceptions, un tri additionnel pour des cas spéciaux. Remarquez que à la fin de la liste Broadcast et Spanning-tree-(for-bridges)_00 sont déplacer à la fin de la liste (pas dans l'ordre normal), car ils ne sont pas un adresse de destination spécifique IP ou MAC, cependant nous avons décider de les inclure question de ne pas omettre des packages lors de la vérification du compte total. Question de debugging, il est utile de les voir inclus dans le compte car sans eux le compte total ne serait que de 10052 lors de la vérification et pourrait entrainer une confusion du genre : « …pourquoi mon code n'arrive pas à lire toutes mes 10069 lignes du tcpdump.log? ». En réalité ils ne devraient pas être présent.

nfsuid.txt :

Voici tous les uid, en ordre alphabétique et non numérique. C'est pourquoi nous avons un UID 308 placer après un chiffre numérique techniquement plus grand dont 2844. Le 3 vient après le 2, aussi simple que ça, le TP mentionne ordre alphabétique seulement.

Résultat de **tcpdumpscan.pl** à la sortie du script:

```
[root@localhost ~]# perl tcpdumpscan.pl
Réponses ARP:
132.204.90.101 -> 00:16:76:d4:35:4e
132.204.90.102 -> 00:16:76:d3:cb:82
132.204.90.105 -> 00:16:76:d4:38:44
132.204.90.117 -> 00:16:76:e1:a6:37
132.204.90.118 -> 00:16:76:e1:a5:e9
132.204.90.123 -> 00:16:76:e1:a5:ea
132.204.90.127 -> 00:1c:c0:2a:e0:93
132.204.90.150 -> 00:1b:21:2e:0f:2c
132.204.90.160 -> 00:11:d8:2a:f2:81
132.204.90.170 -> 00:13:d4:78:a7:39
132.204.90.171 -> 00:13:d4:4c:0a:3c
132.204.90.175 -> 00:13:d4:78:a5:c3
132.204.90.176 -> 00:13:d4:78:a4:dc
132.204.90.96 -> 00:1c:c0:2a:16:c8
132.204.90.99 -> 00:1c:c0:2a:1b:df

Temps de début et de fin: 0.000000 à 13.423330 / Total de 13.42333 secondes
[root@localhost ~]#
```

Ici on voit que non seulement il y a des nouveaux rapports en format .txt qui sont générer, mais le script donne aussi un compte rendu de tous les réponses ARP qui effectivement trouve l'adresse MAC à partir d'une adresse IP. Seulement les résultats qui ont une adresse IP qui pointait vers une adresse MAC exacte ont été afficher. À noter, certaines paires se répètent dans le fichier la répétition n'est pas pris en compte donc si le case existe déjà il ne sera pas dupliqué. Les adresse IP sont en ordre alphabétique, de la même façon que nfsuid.txt…

Puis on voit le temps de début du tcpdump et le temps de fin du rapport. De ce fait le signal à durée 13 .42333 secondes au total.