



RAPPORT DE PROJET

2^{ÈME} BACHELIER EN INFORMATIQUE

Linux

Auteur :
Timothée SIMON
Florian GIARRUSSO
Fabio CUMBO

Enseignant :
Antoine MALAISE

The logo for Campus technique is a red square with the words 'Campus' and 'technique' in white, stacked vertically.

Année académique 2017 - 2018

Ce document est mis à disposition selon les termes de la licence Creative Commons
“Attribution - Pas d’utilisation commerciale 4.0 International”.



Table des matières

1 Introduction

1.1 Choix de la distribution

Pour le choix de la distribution nous avons commencé par mettre en place certains critères de recherche :

- La gratuité
- La stabilité
- La légèreté

Nos recherches nous montrent plusieurs choix possibles :

- RHEL : Payante
- Arch Linux : Pas la plus stable car elle fonctionne en rolling release
- Ubuntu Server et Debian : Ne possèdent pas de version core ce qui les rend plus lourdes
- CentOS : Répond au mieux à tous nos critères

Nous allons donc faire notre serveur sous CentOS car celui-ci répond à tous nos critères dans sa version Core.

1.2 Mode d'installation

Pour l'installation nous avons opté pour l'écriture de scripts pour chacune des fonctionnalités de notre serveur. Cela nous permet de nous rappeler de nos procédures et de toujours les comprendre dans quelques années. Il nous suffit maintenant de copier nos scripts sur une machine réelle ou virtuelle pour pouvoir immédiatement commencer à configurer Linux.

Bien évidemment nos scripts ne gèrent pas beaucoup d'erreurs et ne vérifient pas ce que l'utilisateur a entré, ils ne sont donc pas vraiment prêts pour tout usage mais sont très utiles comme notes.

1.3 Organisation du groupe

Pour nous organiser nous avons utilisé les outils de GitHub. Nous avons donc commencé par créer un repo (privé pour le moment mais nous le passerons sûrement en public après les examens) nous permettant de travailler ensemble sur notre code. Nous avons aussi utilisé la ToDo list de GitHub pour nous organiser dans notre développement.

1.4 Machine physique

Nous avons eu l'occasion d'utiliser une machine physique pour mettre en pratique nos scripts. Celle-ci est bien évidemment une machine de récupération, elle est composée de Intel Pentium 4, de 1Go de DDR2 et de deux disques dur en RAID 1.

2 Installation

Pour l'installation nous avons suivis les étapes suivantes :

- Mise en place de la machine avec l'iso bootable (sur machine virtuelle ou physique)
- Sélection des locale (English US)
- Modification de l'heure (Bruxelles)
- Partitionnement manuel (LVM avec encryption AKA LUKS, sauf pour la machine réelle, faute de puissance) :
 - /dev/sda1 monté sur /boot : 1024 Mo en XFS
 - LVM (encryptée)
 - centos-root monté sur / : 5320 Mo en XFS
 - centos-home monté sur /home : 1023 Mo en XFS
 - centos-swap monté comme partition de swap : 819 Mo
- Activation de la NIC (enp0s3)
- Nom de domaine
- Lancement de l'installation
- Mise en place du mot de passe root

3 Serveur NTP

Le protocole NTP (***N**etwork **T**ime **P**rotocol*) va permettre de synchroniser les horloges des ordinateurs connectés au même réseau local que celle du serveur de temps. Celui-ci devra synchroniser sa propre horloge en contactant un serveur de temps de référence à distance donc par internet. Cette synchronisation des heures permettra entre autres de ne pas perturber certaines applications utilisant l'horloge du système mais aussi pour donner plus de cohérence en cas de comparaison des messages de « logs » de plusieurs ordinateurs sur le réseau.

```
1 #!/bin/bash
3 source ../Common.sh
5 RootCheck
7 #Installation de NTP
  Installe ntp ntpdate ntp-doc
9
  #On stop le service ntpd
11 systemctl stop ntpd
13
  #On met le serveur à la bonne heure au préalable
  ntpdate be.pool.ntp.org
15
  #Configuration du serveur ntp
17 cp ntp.conf /etc/ntp.conf
19
  #Démarrage et activation du service
  Service ntpd
21
  echo "Le service NTP est maintenant installé et activé."
```

../scripts/ntp/NTP.sh

```
# For more information about this file , see the man pages
2 # ntp.conf(5) , ntp_acc(5) , ntp_auth(5) , ntp_clock(5) , ntp_misc(5) ,
  ntp_mon(5) .
4
  driftfile /var/lib/ntp/drift
6
  #On ajoute un directory pour les logs
  logfile /var/log/ntp.log
8
  # Permit time synchronization with our time source , but do not
10 # permit the source to query or modify the service on this system .
  #restrict default nomodify notrap nopeer noquery
12
  # Permit all access over the loopback interface . This could
14 # be tightened as well , but to do so would effect some of
  # the administrative functions .
16 #restrict 127.0.0.1
  #restrict ::1
18
```



```

#On établis les règles pour l'accès au service , et sécurisation
20 #On empêche les machines distantes de modifier la configuration du
    serveur , protection ddos ,..
    restrict default nomodify nopeer notrap noquery
22
#On fait confiance à la machine elle-même
24 restrict 127.0.0.1 mask 255.0.0.0
#IPv6
26 restrict ::1
# Hosts on local network are less restricted.
28 #restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

30 # Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
32 #server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
34 #server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
36
#Synchronisation des horloges avec la pool zone de Belgique
38 server 3.be.pool.ntp.org
server 1.europe.pool.ntp.org
40 server 0.europe.pool.ntp.org

42 #On indique au serveur qu'il doit se synchroniser sur l'horloge locale
server 127.127.1.0
44
#serveur ""bidon en guise 'dIP fallback , au cas où la source de temps
    extérieure deviendrait momentanément indisponible. En cas
    'dindisponibilité du serveur distant , NTP continuera à tourner en se
    basant sur ce fonctionnement-là.
46 fudge 127.127.1.0 stratum 10

48
#broadcast 192.168.1.255 autokey # broadcast server
50 #broadcastclient # broadcast client
#broadcast 224.0.1.1 autokey # multicast server
52 #multicastclient 224.0.1.1 # multicast client
#manycastserver 239.255.254.254 # manycast server
54 #manycastclient 239.255.254.254 autokey # manycast client

56 #Enable public key cryptography.
#crypto
58
includefile /etc/ntp/crypto/pw
60
# Key file containing the keys and key identifiers used when operating
62 # with symmetric key cryptography.
#
64 keys /etc/ntp/keys

66 # Specify the key identifiers which are trusted.
#trustedkey 4 8 42
68
# Specify the key identifier to use with the ntpdc utility.
70 #requestkey 8

```

```

72 # Specify the key identifier to use with the ntpq utility .
   #controlkey 8
74
   # Enable writing of statistics records.
76 #statistics clockstats cryptostats loopstats peerstats
78
   # Disable the monitoring facility to prevent amplification attacks
   # using ntpdc
   # monlist command when default restrict does not include the noquery
   # flag. See
80 # CVE-2013-5211 for more details.
   # Note: Monitoring will not be disabled with the limited restriction
   # flag.
82 disable monitor

```

../scripts/ntp/ntp.conf

On déclare d'abord le fichier de dérive « driftfile ». Il va permettre de corriger les dérives de l'horloge système en l'absence de connexion réseau au serveur de référence. On déclare ensuite le répertoire et le fichier pour stocker les « logs » du service ntpd. On permet la synchronisation avec notre source de temps mais on interdit à la source de modifier ou d'interroger le service sur ce système. On autorise les accès sur l'interface de bouclage Ensuite, on se synchronise avec les serveurs NTP belge de référence. Pour résoudre les problèmes de charge on entre plusieurs adresse. Il s'agit d'un groupement de serveur et la redistribution se fait à l'aide du Round Robin DNS (association de plusieurs adresses IP à un FQDN) Enfin, on précise au serveur de se synchroniser sur l' « Undisciplined Local Clock » et on indique un faux « pilote » destiné à la sauvegarde de l'heure dans le cas où aucune source externe d'heure synchronisée n'est disponible.

4 Connexion SSH

Pour l'administration à distance nous avons décidé d'utiliser SSH (*Secure **SH**ell*) car ce protocol est plus sécurisé que Telnet. Pour la connexion nous n'autorisons que l'authentification avec une clé RSA ou avec un mot de passe et un facteur de double authentification. Pour la double authentification nous avons décidé d'utiliser une clé Yubikey, ce facteur étant physique cela limite fortement l'accès aux personnes non-autorisées. Nous avons choisi de permettre une autre méthode d'authentification car l'utilisation d'une clé RSA seule est un risque en cas de perte du fichier et nous voulions toujours pouvoir être capable d'administrer notre serveur même si ce fichier est perdu ou corrompu.

```
#!/bin/bash
2
3 source ../Common.sh
4
5 RootCheck
6
7 ARG='Argument $1 "22"'
8
9 #Installation du paquet open-ssh (vérification) permettant de partager
10 un service ssh, par défaut il est normalement déjà installé
11 Installe openssh-server
12
13 echo "Port $PORT" >> sshd_config
14
15 # Utilisation du fichier de config
16 cp sshd_config /etc/ssh/sshd_config
17 cp sshbanner /etc/ssh/banner
18
19 #démarrage service sshd
20 Service sshd
21
22 iptables -A INPUT -p tcp --dport $PORT -j ACCEPT
23 iptables -A OUTPUT -p tcp --sport $PORT -j ACCEPT
```

../scripts/ssh/SSH.sh

Après avoir vérifié que l'utilisateur qui lance le script est bien root et installé le serveur openSSH si celui-ci n'était pas déjà installé, nous copions simplement le fichier de configuration et la bannière au bon endroit. Pour finir nous lançons et activons au démarrage le daemon.

5 Serveur DNS

Le serveur DNS (*Domain Name System*) permet de transformer un nom de domaine en adresse IP.

```
#!/bin/bash
2
source ../Common.sh
4
RootCheck
6
DOMAIN='Argument $1 "linux"'
8 SUBDOMAIN='Argument $2 "toto"'
10 IP='ip addr | grep 'state UP' -A2 | tail -n1 | awk '{print $2}' | cut -
    f1 -d '/' '
NETID='ip addr | grep 'state UP' -A2 | tail -n1 | awk '{print $2}' |
    cut -f1 -d '/' | cut -f1-3 -d '.' '
12 HN='hostname'
14 # Installation des paquet
Installe "bind" "bind-utils"
16
systemctl enable named.service
18 systemctl start named.service
#Ajout de l'adresse IP du seueur DNS si pas déjà fais dans le fichier
    de config réseau et dans /etc/resolv.conf "nameserver
    192.168.65.128"
20 systemctl restart network.service
#Modification fichier /etc/named.conf
22 #Création du fichier de zone "linux_forward.zone" et "linux_reverse.
    zone" dans var/named/ et configuration des fichiers
24 # Création du fichier named
26 echo "
options {
28     listen-on port 53 { 127.0.0.1;$IP; }; /*ajout de l'ip du serveur*/
    listen-on-v6 port 53 { ::1; };
30     directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
32     statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
34     allow-query { $NETID; localhost; }; /*ajout du reseau
        autorisé à query*/
    recursion yes; /*mis sur no */
36
    dnssec-enable yes;
38     dnssec-validation yes;

40     /* Path to ISC DLV key*/
    bindkeys-file "/etc/named.iscdlv.key";
42
    managed-keys-directory "/var/named/dynamic";
44
```

```

pid-file \"/run/named/named.pid\";
46 session-keyfile \"/run/named/session.key\";
};

48 logging {
50     channel default_debug {
52         file \"/data/named.run\";
54         severity dynamic;
56     };
58 /*Si domaine inconnu, aller voir dans named.ca*/

60 zone \".\" IN {
62     type hint;
64     file \"/named.ca\";
66 };

68 /*AJOUT DES ZONES*/

66 zone \"$DOMAIN.lan\" IN {
68     type master;
69     file \"/forward.$DOMAIN\";
70     allow-update { none; };
72 };

72 zone \"/2.0.10.in-addr.arpa\" IN {
74     type master;
75     file \"/reverse.$DOMAIN\";
76     allow-update { none; };
78 };

80 include \"/etc/named.rfc1912.zones\";
81 include \"/etc/named.root.key\";
82 \" > /etc/named.conf

84 echo \"
85 \\\$TTL 86400
86 @ IN SOA $HN.$DOMAIN.lan. root.$DOMAIN.lan. (
87     2011071001 ;Serial
88     3600 ;Refresh
89     1800 ;Retry
90     604800 ;Expire
91     86400 ;Minimum TTL
92 )
93 @ IN NS $HN.$DOMAIN.lan.
94 projet IN A $IP
95 $SUBDOMAIN IN CNAME $HN
96 \" > /var/named/forward.$DOMAIN

98 echo \"
99 \\\$TTL 86400
100 @ IN SOA $HN.$DOMAIN.lan. root.$DOMAIN.lan. (

```

```

102         2011071001 ;Serial
        3600        ;Refresh
        1800        ;Retry
104        604800     ;Expire
        86400       ;Minimum TTL
106    )
    @      IN  NS      $HN.$DOMAIN.lan .
108    @      IN  PTR     $DOMAIN.lan .
    projet IN  A       $IP
110    15      IN  PTR     $HN.$DOMAIN.lan .
    " > /var/named/reverse.$DOMAIN
112
114 chown -v root:named /etc/named.conf
    systemctl restart named.service

116 named-checkconf -v /etc/named.conf
    named-checkzone $DOMAIN.lan /var/named/forward.$DOMAIN
118 named-checkzone $DOMAIN.lan /var/named/reverse.$DOMAIN

```

../scripts/dns/DNS_server.sh

Après avoir mis en place tout ce qui est nécessaire pour utiliser le DNS, nous copions le fichier de config de Named avec en modifiant l'adresse IP. Ensuite nous copions les fichier de zone reverse et forward customisée avec les arguments entrés par l'utilisateur. Finalement nous chargeons tout cela dans bind.

6 Serveur web

Pour le serveur web nous avons opté pour apache pour sa forte communauté.

```
source ../Common.sh
2
RootCheck
4
Installe httpd
6
Service httpd
8
DOMAIN='Argument $1 "toto.linux.lan"'
10 PORT='Argument $2 "2187"'
12 echo "ServerName $DOMAIN:$PORT" >> httpd.conf
echo "Listen $PORT" >> httpd.conf
14
cp httpd.conf /etc/httpd/conf/httpd.conf
16
echo "coucou" > /var/www/html/index.html
18
iptables -A INPUT -p tcp --dport $PORT -j ACCEPT
20 iptables -A OUTPUT -p tcp --sport $PORT -j ACCEPT
```

../scripts/web/apache.sh

Après avoir tout installé et customisé le fichier de config, nous le copions au bon endroit et créons une page web contenant "coucou". Pour finir nous ajoutons le bon port au firewall.

7 Serveurs de fichiers

Nous avons mis en place deux serveurs de fichier, un serveur NFS et un serveur Samba.

7.1 Serveur NFS

Le serveur NFS (*Network File System*) permettant à un utilisateur d'accéder à des fichiers sur un serveur de la même façon qu'il accède à des fichiers stockés en local.

```
#!/bin/bash
2 source ../Common.sh

4 RootCheck

6 s="./NFS.sh [DOSSIER] [ARGUMENTS] [IP]
DOSSIER: Dossier de partage (Default: /Partage)
8 ARGUMENTS: Arguments du partage nfs (Default: (rw,sync,no_root_squash,
no_subtree_check))
IP: Adresse ip du serveur (Default: adresse IP local de la machine)
10 "

12 Aide $1 $s
# Défaut du dossier de partage
14 DossierPartage='Argument $1 "/Partage"'
ARG='Argument $2 "(rw,sync,no_root_squash,no_subtree_check)'"
16 IP='Argument $3 \'ip addr | grep \'state UP\' -A2 | tail -n1 | awk '{
print $2}' | cut -f1 -d\'/\'\'\'

18 # Installation du serveur nfs
Installe nfs-utils

20
#création du dossier partagé si celui-ci n'existe pas encore
22 mkdir -p $DossierPartage

24 #Modification des permissions d'accès
chmod 755 $DossierPartage

26
echo "Le dossier $DossierPartage est maintenant créé"
28

30 #Activation et démarrage des services nfs au boot
Service rpcbind
32 Service nfs

34
#Configuration du fichier /etc/exports
36 echo "$DossierPartage $IP$ARG" >> /etc/exports

38 #On exporte le partage
exportfs -a
40 showmount -e
```

../scripts/nfs/NFS.sh

Après avoir vérifié que l'utilisateur qui lance le script est bien root et importé les variables en argument, nous installons le serveur nfs si celui-ci n'est pas déjà installé. Ensuite nous créons le dossier de partage avec les bons droits et activons les processus requis pour le serveur. Finalement nous ajoutons le dossier partagé ainsi que les arguments au fichier et partage et nous en informons le nfs.

7.2 Serveur SMB

Le serveur SMB (*Server Message Block*) permet à des ordinateurs de partager des fichiers et des imprimantes entre eux.

```
#!/bin/bash
2
souce ../Common.sh
4
RootCheck
6 s=" ./SAMBA.sh DOSSIER UTILISATEUR NOM GROUPE
DOSSIER: Dossier de partage (Default: /Partage)
8 UTILISATEUR: Utilisateur possédant les droits (Default: Utilisateur
courant)
NOM: Nom du partage (Default: DOSSIER)
10 GROUPE: Nom du groupe utilisé pour le partage (Default: GroupePartage)
"
12
Aide $1 $s
14
#On crée le dossier de partage
16 DossierPartage='Argument $1 "/Partage"'
UserP='Argument $2 $USER'
18 NameP='Argument $3 $DossierPartage'
GroupePartage='Argument $4 "GroupePartage"'
20
#Installation samba
22 Installe samba
24 #démarrage et activation du démon au démarrage
Service smb
26
#création du dossier partagé si celui-ci n'existe pas encore
28 mkdir -p $DossierPartage
30 #Ajout du groupe sharedFolder contenant user1 et user2
groupadd $GroupePartage
32 useradd -g sharedFolder $UserP
34 #On gère les permissions du dossier partagé
chmod 770 $DossierPartage
36 chown -R $UserP:$GroupePartage $DossierPartage
```

```

38 echo "Veuillez choisir un mot de passe pour l'utilisateur $UserP"
smbpasswd -a $UserP
40
42 # Copie des réglages de samba
echo "
[{$NameP}]
44 path=${DossierPartage}
comment=Partage crée par un script
46 public=yes
force directory mode=777
48 force create mode=777
writeable=yes
50 browseable=yes" >> smb.conf
cp smb.conf /etc/samba/smb.conf
52
# Désactivation de SELinux (car autrement impossible de se connecter
  depuis le client , il faudra configurer tout ça )
54 setenforce 0
56 #Redémarrage du service smb
sudo systemctl restart smb.service

```

../scripts/samba/SAMBA.sh

Tout d'abord nous intégrons un menu d'aide et nous importons les variables passées en arguments. Ensuite nous installons le serveur samba si celui-ci n'est pas encore installé et nous démarrons ses services. Nous créons le groupe relatif au partage et nous y ajoutons l'utilisateur. Maintenant il nous faut créer demander le mot de passe du partage à l'utilisateur et créer la configuration. Finalement nous redémarons le daemon.

8 Utilisateurs

Pour les utilisateurs nous avons optés pour un script interactif. Nous demandons à l'utilisateur de si l'utilisateur aura une clé Yubikey, s'il doit pouvoir se connecter en SSH et s'il faut lui appliquer un quota. Bien évidemment le script n'est lançable que par l'utilisateur root.

```
1 #!/bin/bash
  source ../Common.ssh
3
  RootCheck
5
  echo "Login :"
7 read LOGIN
  echo "Mot de passe :"
9 read -s PASS
  useradd $LOGIN -p $PASS
11
  echo "Voulez-vous configurer une Yubikey pour cet utilisateur ? (N/y)"
13 read C
  if [ "$C" = "y" ] || [ "$C" = "Y" ]
15 then
    echo "Veuillez connecter votre Yubikey"
17 read
    echo "Dans quel slot est configuré votre Challenge ?(1/2)"
19 read S
    if [ "$S" = "1" ] || [ "$C" = "2" ]
21 then
      SERIAL='ykinfo -s '
23      mkdir /home/$LOGIN/.yubico
      ykpamcfg -$$
25      mv /root/.yubico/challenge-* /var/yubico/$LOGIN-"${SERIAL: -7}"
    else
27      echo "Dans quel slot voulez-vous configurer votre Cahllenge ? (1/2)"
      read S
29
      echo "y" | ykpersonalize -$$ -ochal-resp -ochal-hmac -ohmac-lt64 -
      oserial-api-visible
31      SERIAL='ykinfo -s '
      mkdir /home/$LOGIN/.yubico
33      ykpamcfg -$$
      mv /root/.yubico/challenge-* /var/yubico/$LOGIN-"${SERIAL: -7}"
35    fi
  fi
37
  echo "Permettre la connexion en ssh sur cet utilisateur ? (N/y)"
39 read C
  if [ "$C" = "y" ] || [ "$C" = "Y" ]
41 then
    usermod -g sshallow $LOGIN
43 fi
45
  echo "Voulez-vous mettre un quota sur la partition home de cet
    utilisateur ? (N/y)"
```

```

47 read Q
echo "Avez-vous rajouté l'option defaults,usrquota,grpquota à la place
    de defaults sur la partition voulu dans le fichier /etc/fstab ? (N/
    y) "
read F
49 if ([ "$Q" = "y" ] || [ "$Q" = "Y" ]) && ([ "$F" = "y" ] || [ "$F" = "
    Y" ])
51 #On met le quota, le quotaon et quotacheck ainsi que l'edition du
    fichier /etc/fstab
    #doivent être fait préalablement
53 Installe quota

55 mount -o remount /home
    quotacheck -cugv /home
57 quotaon /home/
    edquota -u $LOGIN
59 #On affiche les quotas
    repquota -as
61 #Configuration periode de grace
    echo "Voulez-vous modifier la période de grâce ? (N/y)"
63 read P
    if [ "$P" = "y" ] || [ "$P" = "Y" ]
65 edquota -t
    fi
67 fi

69 echo "L'utilisateur $LOGIN a bien été créé"

```

../scripts/users/users.sh

```

1 #!/bin/bash
3 source Common.sh
# Activation de l'EPEL
5 yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
    latest-7.noarch.rpm

7 # Installation du module PAM
    Installe pam_yubico
9
cp ../Files/yubikey-auth /etc/pam.d/yubikey-auth
11 cp ../Files/sshd /etc/pam.d/sshd

```

../scripts/users/Yubico.sh

Nous avons laissé la possibilité de créer un quota sur la partition /home lors de la création d'un utilisateur. Pour que les quotas soient fonctionnels, il faut d'abord rajouter l'option usrquota et grpquota sur la partition /home comme si dessous

```

1 /dev/mapper/cento-home /home ext4 default,usrquota,grpquota 1 2

```

9 Gestion des sauvegardes

Nous utilisons un programme nommé BORG qui gère les sauvegardes de façon incrémentale sur un serveur distant ou sur une autre partition (par exemple un disque USB)

```
1 #!/bin/bash
3 source ../Common.sh
5 RootCheck
7 Chemin='Argument $1 "/Backup"' # Le chemin peut aussi être sous la
   forme user@hostname :/path/to/repo
8 PWD='Argument $2 "Test123*"'
9 DATE='date +%Y%m%d%H%M%S'
   tmpfile="/tmp/borgcron"
11
12 # Installation de python et de pip
13 Installe python
   pip install --upgrade setuptools
15
16 cd /tmp/
17 curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
   python get-pip.py
19
20 # Installation de borg
21 pip install borgbackup
23
24 # Inistialisation du repo borg
   borg init --encryption=$PWD $Chemin
25 borg create $Chemin:$DATE /
27
28 # Ajout de borg au crontab toute les heures
   crontab -l > $tmpfile
29 echo "0 * * * * export BORG_PASSPHRASE='$PWD';borg create $Chemin::'
   date +%Y%m%d%H%M%S' ' /" >> $tmpfile
   crontab $tmpfile
31 rm $tmpfile
```

../scripts/backups/backup.sh

Après avoir lancé la première sauvegarde on ajoute un crontab pour faire une sauvegarde tout les heures.

10 Mot de passe sur

Avec l'arrivée de CentOS 7.2, un nouvel utilitaire a été implémenté pour mettre facilement un mot de passe au chargement de grub. Il suffit d'utiliser la commande : **grub2-setpassword** Entrer le mot de passe et le confirmer. Enfin, il faut modifier le fichier : `/boot/grub2/grub.cfg` et supprimer le paramètre `--unrestricted`

```
1 ### BEGIN /etc/grub.d/10_linux ###
  menuentry 'CentOS Linux (3.10.0-862.3.2.el7.x86_64) 7 (Core)' --class
    centos --class gnu-linux --class gnu --class os --unrestricted
    $menuentry_id_option 'gnulinux-3.10.0-693.el7.x86_64-advanced-
      e47f19c5-8992-40a8-b004-05d4fbe5996c'
3
  content ...
```

11 Serveur FTP

Le protocole FTP (File Transfer Protocol) permet de transférer des fichiers d'un serveur à un client.

```
#!/bin/bash
2 source ../Common.sh
  RootCheck
4
  s='./FTP.sh [DOSSIER]'
6 DOSSIER: Dossier de partage (Defaut : /Partage)
  ,
8
  Aide $1 $s
10
  Installe vsftp
12
  DossierPartage='Argument $1 "/Partage"'
14
  systemctl start vsftp.service
16 systemctl enable vsftp.service
18 mkdir -p $DossierPartage
20 echo "# Allow all connections \nvsftpd: ALL\n# IP address range\nvsftpd:
    10.0.0.0/255.255.255.0" > /etc/hosts.allow
```

../scripts/ftp/FTP.sh

12 Anti-virus

```
#!/bin/bash
2 source ../Common.sh

4 RootCheck
#Check installation du support EPEL
6 Installe epel-release

8 #Installation de tous les composants de ClamAV

10 Installe clamav-server clamav-data clamav-update clamav-filesystem
    clamav clamav-scanner-systemd clamav-devel clamav-lib clamav-server
    -systemd

12 #Configuration du daemon Clam
#Copie du template dans le cas où l'on a pas de fichier de
    configuration
14 cp /usr/share/clamav/template/clamd.conf /etc/clamd.d/clamd.conf

16 #Activation de Freshclam pour garder la DB à jour

18 #Création du service freshclam et configuration
echo "# Run the freshclam as daemon
20 [Unit]
Description = freshclam scanner
22 After = network.target

24 [Service]
Type = forking
26 ExecStart = /usr/bin/freshclam -d -c 4
Restart = on-failure
28 PrivateTmp = true

30 [Install]
WantedBy=multi-user.target" > /usr/lib/systemd/system/clam-freshclam.
    service
32
#Démarrage et activation du service au démarrage
34 Service clam-freshclam.service

36 #Changement des fichiers service autrement clamd@.service ne démarre
    pas
#On renomme le fichier si on l'a pas déjà fais
38 if [ ! -e "/usr/lib/systemd/system/clamd.service" ];then
    mv /usr/lib/systemd/system/clamd@.service /usr/lib/systemd/system/
    clamd.service
40 fi

42 #On modifie le fichier clamd@scan.service et on change la référence
    vers
# /lib/systemd/system/clamd.service
44 echo ".include /lib/systemd/system/clamd.service

46 [Unit]
```



```

48 Description = Generic clamav scanner daemon
[Install]
50 WantedBy = multi-user.target" > /usr/lib/systemd/system/clamd@scan.
    service
52 #On modifie le fichier /usr/lib/systemd/system/clamd.service et on
    configure
    echo "[Unit]
54 Description = clamd scanner daemon
    After = syslog.target nss-lookup.target network.target
56
[Service]
58 Type = simple
    ExecStart = /usr/sbin/clamd -c /etc/clamd.d/clamd.conf --foreground=yes
60 Restart = on-failure
    PrivateTmp = true
62
[Install]
64 WantedBy=multi-user.target" > /usr/lib/systemd/system/clamd.service
66 #Démarrage et automatistion des services
    Service clamd.service
68 Service clamd@scan.service

```

../scripts/antivirus/antivirus.sh

Ce script va permettre d'installer l'antivirus sur le serveur pour améliorer sa sécurité. On vérifie d'abord que l'utilisateur exécute bien le script en tant que root, ce qui est nécessaire pour configurer les fichiers de ClamAV. On installe ensuite EPEL (Extra Package for Enterprise Linux) qui est un repo fournissant des packages additionnels pour les distributions de type RedHat/CentOs.

Ensuite, au cas où on ne possède pas de fichier de configuration, on copie le template clamd.conf. On peut créer le service freshclam et le configurer de sorte qu'il vérifie 4 fois par jour la présence de mise à jour. On le démarre et on l'active au démarrage ensuite.

On renomme également le fichier /usr/lib/systemd/system/clamd@.service en /usr/lib/systemd/system/clamd.service autrement, par défaut le service ne fonctionne pas. Puis, il faut indiquer le bon chemin du clamd.service dans le fichier /usr/lib/systemd/system/clamd@scan.service et on peut configurer le fichier clamd.service. Les dernières lignes vont permettre d'activer et de démarrer automatiquement les services clamd et clamd@scan au démarrage.

```

1 #!/bin/bash
3 date=$(date +%d_%m_%Y_%H_%M)
5 echo "Arrêt du service freshclam et mise à jour"
7 service clam-freshclam stop
9 sudo freshclam

```

```

11 service clam-freshclam start
13 echo Service redémarré et mise à jour effectuée
14 echo Entrez le nom du repertoire ou fichier à analyser :
15 read nom
17 if [ -e "$nom" ];then
18     echo Analyse en cours ....
19
20     echo "
21         ----\n" >> $HOME/analyseVirus.log
22     echo " " >> $HOME/analyseVirus.log
23     echo "Analyse du $date\n" >> $HOME/analyseVirus.log
24     echo " " >> $HOME/analyseVirus.log
25     clamscan -l $HOME/analyseVirus.log -r $nom
26 else
27     echo Vous avez entré un nom de repertoire ou fichier inexistant
28 fi

```

../scripts/antivirus/analyseVirus.sh

Ce script va permettre d'exécuter une analyse virale d'un dossier ou fichier choisis par l'utilisateur qui lance le script. On met d'abord à jour la base de données de l'antivirus et puis on demande à l'utilisateur de spécifier la cible. Si elle existe, on procède à l'analyse.

13 Scripts communs

```
1 #!/bin/bash
3 # Vérification des erreurs
  set -e
5
  # Vérifie qu'un package est installé
7 function EstInstalle {
    if yum list installed "$@" >/dev/null 2>&1; then
9         true
    else
11        false
    fi
13 }
15 function Installe {
    if EstInstalle "$@"
17 then
        echo "$@ est/sont déjà installé"
19 else
        yum -y install "$@" &> /dev/null # Tentative d'installation du
        paquet
21
        if EstInstalle "$@" # Vérification que le paquet à été correctement
        installé
23 then
            if [ -z "$2" ]
25 then
                echo "$@ est maintenant installé"
27 else
                echo "$@ sont maintenant installés"
29 fi
31 else
            echo "Impossible d'installer le(s) package(s) \"$@\","
33 echo "Vérifiez votre connexion internet et réessayez !"
            exit
35 fi
37 fi
39 }
39 function Argument {
    # Si le premier argument est vide on retourne le deuxième
41 if [ -z "$1" ]
    then
43     echo "$2"
    else
45     echo "$1"
    fi
47 }
49 function RootCheck {
    if [ "$EUID" -ne 0 ]
51 then
```

```

53     echo "Vous devez lancer ce script en tant que root"
54     echo "Ré-essayez avec \"sudo ./\$0\""
55     exit
56 fi
57 }
58
59 function Service {
60     if ! systemctl restart $1
61     then
62         echo "Erreur lors du lancement du service"
63         exit 1
64     else
65         systemctl enable $1 &> /dev/null
66         echo "Le service à bien été lancé"
67     fi
68 }
69
70 function Aide {
71     if [ "$1" = "-h" ]
72     then
73         printf $2
74         exit
75     fi
76 }

```

../scripts/Common.sh

Dans ce fichier nous déclarons plusieurs fonctions utilisée dans plusieurs scripts. Les fonctions sont les suivantes :

- Installe : Installe un paquet si celui-ci n'est pas installé.
- Argument : Retourne le string passé en argument au programme et s'il n'y en a pas retourne l'argument par défaut.
- RootCheck : Vérifie que l'utilisateur qui lance le script est bien root.
- Service : Start puis enable un service.
- Aide : Retourne un string d'aide si l'utilisateur le demande avec l'argument -h.

14 Remerciement

Je remercie Terencio AGOZZINO pour avoir réalisé la mise en page de ce document en L^AT_EX.