LinuxFocus article number 278
http://linuxfocus.org

door Erdal Mutlu
<erdal(at)linuxfocus.org>

Over de auteur:

Erdal is een van de Turkse LF auteurs. Hij werkt op dit moment als systeembeheerder bij de Linotype Library. Hij is al een grote Linux fan vanaf dat hij op de universiteit zat, hij houdt van werken en ontwikkelen in deze omgeving.

Systeembeheer automatiseren met SSH en SCP

Kort:

Wanneer je veel Linux systemen beheert, heb je zeker enkele scripts nodig die je helpen bij het automatiseren van enkele processen. Je zult tijdens je dagelijkse werk gemerkt hebben dat je op veel systemen ongeveer dezelfde handelingen uitvoert. Misschien hebt je er al eens aan gedacht om deze processen op een of andere manier te automatiseren. Dit is vooral handig bij het onderhoud aan veel identieke Linux/Unix systemen. In dit artikel laat ik een manier zien om dit te doen met SSH gereedschappen.

Vertaald naar het Nederlands door:
Hendrik-Jan Heins
<hjh(at)passys.nl>

Inleiding

Het gaat erom een manier te vinden om bestanden te kopiëren van het workstation waar ik achter zit naar een aantal workstations of servers, en om daarna enkele commando's uit te kunnen voeren op die machines, zoals bijvoorbeeld het starten van een rpm installatie of het veranderen van systeemopties. Soms moeten we eerst enkele commando's uitvoeren op die machines en daarna enkele bestanden ophalen die het resultaat van de commando's kunnen zijn.

Om dit artikel te begrijpen heb je wat basiskennis nodig over programmeren op de commandoregel. Voor meer informatie hierover kan je het LinuxFocus artikel Programmeren in de shell van Katja en Guido Socher bekijken. Je hebt ook enige kennis van SSH gereedschappen zoals SSH-keygen, SSH-add, SSH, SCP en SFTP nodig. Er bestaat een gratis versie van SSH voor Linux, OpenSSH, die al deze

gereedschappen bevat. Er zijn ook man-pagina’s beschikbaar.

Waarom SSH gebruiken?

Het antwoord is een wedervraag: Waarom niet? Je kunt rsh-rcp of telnet-ftp, gebruiken, maar zij zijn niet geschikt voor gebruik in een onveilige omgeving, zoals internet en misschien je eigen intranet. SSH biedt veilige versleutelde communicatie tussen twee machines over een onveilig netwerk. Ik ga de beveiligingsimplicaties van het gebruik van deze gereedschappen niet bespreken. Kijk maar eens naar het artikel Door de tunnel van Georges Tarbouriech.
Maar in het verleden heb ik ook scripts gebruikt die gebaseerd waren op telnet/ftp.

Bestanden en Mappen Kopieren met SCP

Om een bestand uit een lokale map te copieren naar een andere computer, kan het volgende commando gebruikt worden:

scp /pad/naar/het/bestand1 gebruiker@remote_host:/dir_op_afstand/nieuwbestand

In dit voorbeeld wordt het bestand met de naam bestand1 gekopieerd van de lokale map naar eem andere computer "remote_host" (remote_host kan de naam of het IP-adres van de andere machine zijn) in /dir_op_afstand/ met een nieuwe naam voor het bestand, nieuwbestand. Nu wordt je gevraagd je te autoriseren als "gebruiker". Wanneer de autorisatie geslaagd is, en de gebruiker voldoende rechten heeft op de andere machine, wordt het bestand gekopieerd. Je kunt de doelbestandsnaam ook weglaten, in dat geval wordt het bestand gecopieerd met z’n eigen naam. In het kort betekent dit dus dat je bestanden kunt hernoemen tijdens het kopieren.
Het omgekeerde is ook mogelijk: Je kunt een bestand van een andere machine kopieren naar een lokale map:

scp gebruiker@remote_host:/dir_op_afstand/bestand /pad/naar/lokale/folder/nieuwbestand

Een andere leuke mogelijkheid van het scp commando is dat je mappen recursief (dus: met inhoud) kunt copieren door middel van de "-r" optie.

scp -r gebruiker@remote_host:/dir_op_afstand/ .

Het bovenstaande commando copieert de map "dir_op_afstand" en alle onderliggende mappen en bestanden van de externe machine naar de huidige werkmap met behoud van de map- en bestandsnamen.

Opmerking: Er wordt aangenomen dat je een SSH daemon hebt draaien op het externe systeem.

Extern Aanmelden met behulp van SSH

In de plaats van met rlogin of telnet kan je inloggen met SSH, wat veel veiliger is:

ssh erdal@helvetica.fonts.de

Afhankelijk van je configuratie word je of om een wachtwoord gevraagd of een "wachtzin". Hier verbinden we met de externe machine helvetica.fonts.de met externe gebruiker erdal. Het SSS commando bevat een groot aantal opties die je kunt gebruiken wanneer nodig. Kijk eens naar de man pagina's van SSH.

Commando's Uitvoeren met Behulp van SSH

Er bestaat een mogelijkheid om commando's uit te voeren op de externe machine met behulp van SSH:

ssh erdal@helvetica.fonts.de df -H

Het is eigenlijk net hetzelfde de syntax voor het inloggen. Het enige verschil komt na het hostnaam deel. Het commando (in dit voorbeeld "df -H") wordt uitgevoerd op de externe machine. De output van dit commando wordt getoond op jou scherm.

Verbinding Maken met een Externe Computer zonder Wachtwoord

In plaats van gebruik te maken van wachtwoord autorisatie, kan je een sleutelpaar (publiek/privé) gebruiken. Je moet je eigen sleutelparen genereren. Er is het SSH-keygen gereedschap dat gebruikt kan worden om deze sleutels voor SSH te genereren:

ssh-keygen -b 1024 -t dsa

Je wordt gevraagd naar een naam voor de private sleutel. Normaal gesproken is de naam van de publieke sleutel hetzelfde als die van de private sleutel, maar met het woord ".pub" erachter. Hier staat "-b 1024" voor het aantal bits in de te genereren sleutel. Wanneer je geen waarde opgeeft, zal de standaardwaarde worden gebruikt. Met "-t dsa" geef je aan welk type sleutel je wilt gebruiken. De mogelijke waardes zijn: "rsa1" voor protocol versie 1 en "rsa" of "dsa" voor protocolversie 2. Ik raad je aan om protocolversie 2 van SSH te gebruiken. Maar wanneer je oudere servers hebt die alleen protocol versie 1 odersteunen, zal je "-t rsa1" moeten aangeven om een bruikbaar sleutelpaar te genereren. Je kunt SSH dwingen om protocol versie 1 of 2 te gebruiken door respectievelijk "-1" of "-2" aan te geven.

Om je sleutel te kunnen gebruiken, moet je de publieke sleutel op de externe machine installeren. De inhoud van het publieke sleutelbestand moet worden gecopieerd naar $HOME/.ssh/authorized_keys of $HOME/.ssh/authorized_keys2. Wees wel voorzichtig en mix geen sleutels voor verschillende protocol versies. authorized_keys wordt gebruikt voor protocol versie 1. authorized_keys2 wordt gebruikt voor protocol versie 2. Wanneer je de publieke sleutel correct hebt geïnstalleerd, word je de eerste keer dat je verbinding maakt met die machine, gevraagd om je wachtzin op te geven en wanneer je het fout doet word je gevraagd het wachtwoord voor de externe gebruiker op te geven. Je kunt de verbinding beperken zodat er standaard alleen gebruik wordt gemaakt van autorisatie op basis van de publieke sleutel door het configuratie bestand van SSHd aan te passen. De bestandsnaam is /etc/ssh/sshd_config en de instelling die je moet wijzigen is 'PasswordAuthentication'. Verander de waarde van deze instellig

naar "no" (PasswordAuthentication no) en herstart SSHd.

Tot dit punt gaat alles goed. We hebben een veilige manier om bestanden te kopieren en om opdrachten uit te voeren op externe systemen. Maar voor de automatisering van bepaalde taken zou het handig zijn als we geen wachtwoorden hoefden in te geven. Anders valt er niets te automatiseren. Een oplosing kan zijn het opgeven van het wachtwoord in ieder script, maar dit is niet echt een goed idee. De betere methode is het gebruiken van de key-agent voor de wachtwoorden. SSH-agent is een programma dat private sleutels beheert om publieke sleutels te autoriseren. Start nu een key-agent:

ssh-agent $BASH

en voeg onze private sleutels eraan toe met behulp van

ssh-add .ssh/id_dsa

of

ssh-add .ssh/identity

id_dsa is de private DSA sleutel en identity is het sleutelbestand van RSA1. Dit zijn de standaard bestandsnamen voor sleutelgeneratie met behulp van SSH-keygen. Natuurlijk wordt je gevraagd om je wachtwoord voordat SSH-add je sleutel toevoegt aan de SSH-agent. Je kunt een lijst van deze sleutels zien met behulp van het volgende commando:

ssh-add -l

Als je nu een verbinding maakt met een server die jouw sleutel heeft in z’n autorisatie bestand, krijg je een verbinding zonder een letter in te typen! De SSH-agent handelt het autorisatieproces af.

Wanneer je een SSH-agent op de bovenstaande manier gebruikt, kan je hem alleen gebruiken in het venster waarin de SSH-agent is gestart, daarna moet je nog wat werk verzetten als je hem ook in andere vensters wilt gebruiken. Ik heb het volgende kleine script geschreven om de SSH-agent te starten:

```
#!/bin/sh
#
# Erdal mutlu
#
# Starting an ssh-agent for batch jobs usage.

agent_info_file=~/.ssh/agent_info

if [ -f $agent_info_file ]; then
 echo "Agent info file : $agent_info_file exists."
 echo "make sure that no ssh-agent is running and then delete this file."
 exit 1
fi

ssh-agent | head -2 > $agent_info_file
chmod 600 $agent_info_file
exit 0
```

Het bovenstaande script controleert of er al een bestand bestaat dat agent_info heet in de persoonlijke map van de gebruiker op de plaats waar normaal gesproken SSH gerelateerde bestanden staan. In ons geval is dat de map '.ssh/'. Als het bestand bestaat, wordt de gebruiker gewaarschuwd dat het er al is en wordt er melding gemaakt van de mogelijkheden. Wanneer de gebruiker geen draaiende SSH-agent heeft, dan moet hij/zij het bestand verwijderen en het script nogmaals starten. Het script start de SSH-agent en schrijft de eerste twee regels naar het bestand agent_info. Deze informatie kan achteraf door de SSH gereedschappen gebruikt worden. Vervolgens staat er een regel om de modus van het bestand te veranderen, zodat alleen de eigenaar van het bestand het kan lezen en ernaar kan schrijven.

Wanneer de SSH-agent draait, kan je je sleutels eraan toevoegen. Maar voordat je dit doet, moet je "source" starten met het bestand agent_info, zodat de SSH gereedschappen weten waar de SSH-agent is:

source ~/.ssh/agent_info of ~/.ssh/agent_info

En voeg je sleutels toe met SSH-add. Je kunt de volgende regels toevoegen aan je .bashrc bestand zodat iedere keer dat je een terminal venster opent, het bestand agent_info automatisch wordt "ge-sourced":

```
if [ -f .ssh/agent_info ]; then
. .ssh/agent_info
fi
```

WAARSCHUWING: Je moet de machine waarop je de SSH-agent en het geautomatiseerde script (zie verderop) gaat gebruiken wel goed beveiligen. Anders heeft iemand die toegang heeft tot jou account ook toegang tot alle servers waarvan je de SSH sleutels hebt. Alles heeft z'n prijs...

Het Script

Nu wordt het tijd om uit te leggen hoe we enkele systeembeheer taken gaan automatiseren. Het gaat erom een aantal commando's uit te voeren voor een bepaalde lijst machines en om enkele bestanden te halen of te sturen naar deze machines. Dit moet een systeembeheerder regelmatig doen. Hier is het script:

```
#!/bin/sh

# Installing anything using Secure SHELL and SSH agent
# Erdal MUTLU
# 11.03.2001


#####################################################################
#                       Functions                                   #
#####################################################################
### Copy files between hosts
copy_files()
{
 if [ $files_file != "files_empty.txt" ];then
  cat $files_file | grep -v "#" | while read -r line
  do
   direction=‘echo ${line} | cut -d " " -f 1‘
   file1=‘echo ${line}     | cut -d " " -f 2‘
```

```
   file2=‘echo ${line}     | cut -d " " -f 3‘

   case ${direction} in
      "l2r") :  ### From localhost to remote host
          echo "$file1  --> ${host}:${file2}"
          scp $file1 root@${host}:${file2}
      ;;
      "r2l") :  ### From remote host to localhost
          echo "${host}:${file2}  --> localhost:${file2}"
          scp root@${host}:${file1} ${file2}
      ;;
      *)
          echo "Unknown direction of copy : ${direction}"
          echo "Must be either local or remote."
      ;;
   esac
  done
 fi
}

### Execute commands on remote hosts
execute_commands()
{
 if [ $commands_file != "commands_empty.txt" ];then
  cat $commands_file | grep -v "#" |  while read -r line
  do
   command_str="${line}"
   echo "Executing $command_str ..."
   ssh -x -a root@${host}  ${command_str} &
   wait $!
   echo "Execute $command_str OK."
  done
 fi
}

### Wrapper function to execute_commands and copy_files functions
doit()
{
 cat $host_file | grep -v "#" | while read -r host
 do
  echo "host=$host processing..."
  case "${mode}" in
        "1")
                copy_files
                execute_commands
                ;;
        "2")
                execute_commands
                copy_files
                ;;
        *)
                echo "$0 : Unknown mode : ${mode}"
                ;;
  esac
  echo "host=$host ok."
  echo "------------------------------------------------------------------"
 done
}

######################################################################
### Program starts here
```

```
#####################################################################

if [ $# -ne 4 ]; then
 echo "Usage : $0 mode host_file files_file commands_file"
 echo ""
 echo "mode is 1 or 2 "
 echo "    1 : first copy files and then execute commands."
 echo "    2 : first execute commands and then copy files."
 echo "If the name of files.txt is files_empty.txt then it is not processed."
 echo "If the name of commands.txt is commands_empty.txt then it is
 echo "not processed."
 exit
fi


mode=$1
host_file=$2
files_file=$3
commands_file=$4

agent_info_file=~/.ssh/agent_info
if [ -f $agent_info_file ]; then
 . $agent_info_file
fi

if [ ! -f $host_file ]; then
 echo "Hosts file : $host_file does not exist!"
 exit 1
fi

if [ $files_file != "files_empty.txt" -a ! -f $files_file ]; then
  echo  "Files file : $files_file does not exist!"
  exit 1
fi

if [ $commands_file != "commands_empty.txt" -a ! -f $commands_file ]; then
 echo  "Commands file : $commands_file does not exist!"
 exit 1
fi

#### Do everything there
doit
```

We bewaren het script als ainstall.sh (geautomatiseerde installatie) en proberen het te starten zonder parameters. We krijgen het volgende bericht:

./ainstall.sh

```
Usage : ./ainstall.sh mode host_file files_file commands_file

mode is 1 or 2
        1 : first copy files and then execute commands.
        2 : first execute commands and then copy files.
If the name of files.txt is files_empty.txt then it is not processed.
If the name of commands.txt is commands_empty.txt then it is not
processed.
```

Zoals het bericht al weergeeft: als je geen commando’s wilt uitvoeren, geef dan commands_empty.txt op

als bestandsnaam voor commands.txt en wanneer je geen bestanden wilt versturen, geef dan files_empty.txt aan als naam voor files_file. Soms hoef je alleen maar enkele commando's uit te voeren, terwijl je een andere keer alleen bestanden wilt versturen.

Voordat ik het script regel voor regel ga uitleggen, is hier een voorbeeld van gebruik: Stel dat je een tweede DNS server hebt toegevoegd aan je netwerk en dat je deze wilt toevoegen aan het /etc/resolv.conf bestand. Ga er voor het gemak van uit dat alle machines hetzelfde /etc/resolv.conf bestand bevatten. Het enige dat je nu hoeft te doen is het nieuwe resolv.conf bestand kopieren naar alle machines.
Allereerst heb je een lijst van alle machines nodig. We gaan alle machines opgeven in een bestand geheten hosts.txt. Het formaat van het hosts.txt bestand vereist dat iedere machine op een nieuwe regel wordt aangegeven, met de naam of het IP adres. Hier is een voorbeeld:

```
##############################################################################
#### Every line contains one hostname or IP address of a host. Lines that
#### begin with or contain # character are ignored.
##############################################################################
helvetica.fonts.de
optima.fonts.de
zaphino
vectora
#10.10.10.162
10.10.10.106
193.103.125.43
10.53.103.120
```

Zoals je kan zien in het voorbeeld, kan je volledige machinenamen aangeven of alleen het eerste deel. Vervolgens heb je een bestand nodig waarin je de te sturen bestanden in beschrijft. Er zijn twee mogelijkheden:

- De eerste mogelijkheid is een transfer van de lokale machine naar de machines die zijn aangegeven in het bestand hosts.txt. Dit is wat we in ons geval gaan doen.
- Vanaf iedere machine die is aangegeven in het bestand hosts.txt naar de lokale machine. Dit is om van iedere machine bestanden te verkrijgen. Bijvoorbeeld voor het gebruik met ons eenvoudige backup script, dat ik later in dit artikel zal beschrijven.

De te verzenden bestanden zijn aangegeven in een ander bestand. Dit slaan we op als files_file.txt. Het formaat van het bestand is als volgt: iedere regel bevat informatie over een te kopieren bestand. Er zijn twee mogelijke kopieer-richtingen: l2r (lokaal naar remote/extern) en r2l (extern naar lokaal). l2r is wanneer een bestand wordt gekopieerd van de lokale machine naar een externe machine. r21 is wanneer een bestand wordt gekopieerd van een externe machine naar de lokale machine. na het richtingscommando komen de bestandsnamen. Velden worden gescheiden door een spatie of een tab. Het eerste bestand wordt gekopieerd naar het tweede volgens het richtingscommando. De bestandsnaam voor de externe mchine moet volledig worden aangegeven, anders wordt het gecopieerd naar de persoonlijke map van de extern ingelogde gebruiker (meestal root). Hier is ons files_file.txt bestand:

```
##############################################################################
# The structure of this file is :
# - The meaning of the fileds are : is l2r (localhost to remote) and r2l
# (remote computer to local).
#       r2l  file1   file2
```

```
#         means copy file1 from remote (hosts specified in the
#         hosts.txt file) computer to localhost as file2.
#       l2r     file1   file2
#         means copy file1 from localhost to
#         remote (hosts specified in the hosts.txt file) computer as file2
#         file1 and file2 are files on the corrsponding hosts.
#
#       Note: the order of using local and remote specifies the direction
#       of the copy process.
##############################################################################
l2r     resolv.conf     /etc/resolv.conf
```

Zoals je kunt zien heb ik al een omschrijving van hoe de structuur van het bestand in elkaar zit bijgeleverd. Normaal gesproken lever ik deze beschrijving bij ieder files_file.txt bestand dat ik gebruik. Het is een eenvoudige maar goede manier van documenteren. In ons voorbeeld willen we het bestand resolv.conf kopieren naar een externe machine als /etc/resolv.conf. Ter demonstratie heb ik commando's om de eigenaar en de groep te veranderen bijgevoegd en laat ik het bestand /etc/resolv.conf zien na het kopieren. Commando's die moeten worden uitgevoerd, moeten in een afzonderlijk bestand geplaatst worden. We noemen het bestand met de commando's "commands_file.txt". Hier is ons commands_file.txt bestand:

```
##############################################################################
# The structure of this file is : Every line contains a command to be
#  executed. Every command is treated seperately.
##############################################################################
chown root.root /etc/resolv.conf
chmod 644 /etc/resolv.conf
cat /etc/resolv.conf
```

Het commando's bestand bevat commando's die op iedere externe machine (aangegeven in hosts.txt) worden uitgevoerd. De commando's worden sequentieel (achter elkaar) uitgevoerd, dus het eerste commando wordt als eerste uitgevoerd, daarna het tweede enz.

Okay, nu heb je alle benodigde bestanden voor dit eenvoudige voorbeeld. Het enige dat nog moet worden aangegeven is de "modus" optie, deze geeft aan welke van de twee bestanden, commands_file.txt of files_file.txt, als eerste moet worden uitgevoerd. Je kunt de bestanden die zijn aangeven in files_file.txt eerst versturen en daarna alle commando's uitvoeren op de aangegeve machines, dit is mode 1. Maar je kunt ook eerst de commando's uitvoeren en daarna de bestanden sturen, dat is mode 2. Nu kan je het script als volgt met de juiste argumenten starten:

./ainstall.sh 1 hosts.txt files_file.txt commands_file.txt

Een kleine tip: ik begin de naam van een files.txt normaal gesproken altijd met 'files_een_kleine_omschrijving', zoals bijvoorbeeld "files_resolvconf.txt. Ik doe hetzelfde met de hosts.txt en commands.txt.

Nu is het tijd om wat uit te leggen over het script zelf. Het programma begint met het controleren van het aantal argumenten; wanneer dit meer of minder dan 4 is, wordt het gebruiksbericht getoond. Wanneer het aantal argumenten wel klopt worden de argumenten toegewezen aan de bijbehorende variabelen. Daarna wordt "source" uitgevoerd indien het bestand '~/.ssh/agent_info' bestaat. Dit bestand

bevat informatie over de draaiende SSH agent. Wanneer je geen agent draait, moet je de wachtwoorden handmatig ingeven, dus geen automatisering ;-). Daarna wordt er van ieder bestand (hosts, files en commands) nagegaan of het bestaat. Er bestat ook een speciale test voor files_empty.txt en commands_empty.txt. Wanneer je een dergelijke naam hebt aangegeven, is controle op het bestaan van het bestand niet nodig. Ik heb dit deel van het script veranderd tijdens het schrijven van dit artikel. Voorheen was het:

```
if [ -f $host_file -a -f $files_file -a -f $commands_file ]; then
 echo "$host_file $files_file $commands_file"
 doit
else
 echo "$host_file or $files_file or $commands_file does not exist"
 exit
fi
```

In dit geval moest ik bestanden hebben met de namen: files_empty.txt en commands_empty.txt. Maar dit was helemaal geen probleem aangezien ik maar in één map werkte.
Aan het einde komt de aanroep naar de functie 'doit'. Alles wordt geregeld met deze functie. De functie bevat een lus die bestaat uit 'cat' en 'while', die voor iedere aangegeven externe machine in het '$hosts_file' de copy_files en execute_commands functies aanroept volgens de 'mode'. Dus voor iedere externe machine wordt het werk zo gedaan. De 'host' variabele bevat de huidige naam of het IP adres van de externe machine.
Laten we eens kijken naar de copy_files functie. Deze functie controleert eerst of de waarde van 'files_file' gelijk is aan die van 'files_empty.txt' of niet. Wanneer de waarde gelijk is, gebeurt er niets. Wanneer de waarde niet gelijk is, dan wordt voor iedere regel in '$files_file', 'direction', 'file1' en 'file2' de variabelen die de richting van de kopie bepalen, respectievelijk de eerste en de tweede bestandsnaam volgens de waarde van de 'direction' variabele, gekopieerde met behulp van scp.
Laten we tenslotte eens kijken naar de execute_commands functie. De functie controleert of de waarde van het 'commands_file' gelijk is aan 'commands_empty.txt'. Wanneer de waarde hetzelfde is, gebeurt er niets. Wanneer de waarde niet hetzelfde is, dan wordt ieder commando in '$commands_file' in de achtergrond uitgevoerd op de externe machine met behulp van SSH. Na het uitvoeren van het SSH commando wordt er een pauze aangeroepen met de parameter '$!'. Dit commando zorg ervoor dat ieder commando in sequentiele vorm wordt uitgevoerd. '$!' vergroot het procesnummer met het meest recentelijk uitgevoerde achtergrond commando.
Dat is alles - eenvoudig, niet?

Eenvoudig een Backup maken van je configuratiebestanden

Hier is een geavanceerder voorbeeldgebruik van het script. Het gaat erom een backup te maken van de configuratiebestanden van je externe machines of servers. Hiervoor heb ik een klein script geschreven dat gebruikt maakt van ainstall.sh:

```
#!/bin/sh

server_dir=${HOME}/erdal/sh/ServerBackups

if [ ! -d $server_dir ]; then
 echo "Directory : $server_dir does not exists."
 exit 1
fi
```

```
cd $server_dir

servers=ll_servers.txt
prog=${HOME}/erdal/sh/einstall_sa.sh

cat $servers | grep -v "#" | while read -r host
do
  echo $host > host.txt
  $prog 1 host.txt files_empty.txt
  servers/${host}/commands_make_backup.txt
  $prog 1 host.txt files_getbackup.txt commands_empty.txt
  mv -f backup.tgz servers/${host}/backup/‘date +%Y%m%d‘.tgz
  rm -f host.txt
done

exit 0
```

Je moet een speciale map hebben die ’servers’ heet. In deze map moeten twee bestanden staan: files_getbackup.txt en ll_servers.txt. Hier is ’files_getbackup.txt’ :

r2l /root/backup.tgz backup.tgz

’ll_servers.txt’ bevat de namen of IP adressen van de externe machines die ge-backupt dienen te worden. Iedere naam in het bestand ’ll_servers.txt’ verwijst naar een map met dezelfde naam en in die map moet een bestand genaamd commands_make_backups.txt staan, dat een commando bevat om een /root/backup.tgz archief van de configuratiebestanden te maken van die machine. Verder is er nog een map genaamd backup. Alle backups van deze machine worden in die map opgeslagen. Wanneer de inhoud van ll_servers.txt het volgende is:

```
fileserver
dbserver
10.10.10.1
appserver
```

Dan is de mappen structuur van je ’$servers’ map als volgt:

```
servers
|-- files_getbackup.txt
|-- ll_servers.txt
|-- make_server_backups.sh
|-- 10.10.10.1
|   |-- backup
|   ‘-- commands_make_backup.txt
|-- appserver
|   |-- backup
|   ‘-- commands_make_backup.txt
|-- dbserver
|   |-- backup
|   ‘-- commands_make_backup.txt
|-- fileserver
    |-- backup
    ‘-- commands_make_backup.txt
```

En hier zijn enkele voorbeelden voor de commands_make_backups.txt bestanden:

tar cfz /root/backup.tgz /etc/samba /etc/atalk /etc/named.conf /var/named/zones

Het bovenstaande commands_make_backup.txt wordt gebruikt om een backup te maken van samba, atalk en nameserver instellingen en zone bestanden.

tar cfz /root/backup.tgz /etc/httpd /usr/local/apache

Het bovenstaande commands_make_backup.txt wordt gebruikt om een backup te maken van de apache server instellingen en bestanden.

tar cfz /root/backup.tgz /etc/squid /etc/named.conf

Het bovenstaande commands_make_backup.txt wordt gebruikt om een backup te maken van de squid proxy server en secundaire dns server instellingen.

Door de bovenstaande scripts te gebruiken en commands_make_backup.txt bestanden te maken zoals jij ze nodig hebt, kan je backups van de instellingen van je servers maken.

Conclusie

Het ainstall.sh script laat je enkele systeembeheerderstaken automatiseren. Het script is gebaseerd op eenvoudig gebruik van SSH gereedschappen. Je zult het script waarderen wanneer je veel identieke systemen beheert.

Bronnen

- SSH, The Secure Shell: The Definitive Guide, door Daniel J. Barrett and Richard Silverman.
- Door de tunnel, van Georges Tarbouriech.
- Programmeren in de Sheel, van Katja en Guido Socher.

Site onderhouden door het LinuxFocus editors team © Erdal Mutlu "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Vertaling info: en --> -- : Erdal Mutlu <erdal(at)linuxfocus.org> en --> nl: Hendrik-Jan Heins <hjh(at)passys.nl>

2005-01-14, generated by lfparser_pdf version 2.51