Over Telnet, FTP, de r-serie en Open SSH

Gouda 23 juni 2006

Uitgangssituatie

Telnet en FTP

TELNET was een wonderlijke applicatie, het biedt interactieve toegang tot de meest uiteenlopende platforms. Tegenwoordig nog bruikbaar om een oude printer te configureren óf om er andere protocollen mee te debuggen. B.v. POP3:

```
telnet pop.xs4all.nl:110
user doerian
pass *****
list
retr 12
dele 12
quit
```

FTP doet zich voor als een TELNET-sessie, met opdrachten als: `help, dir, cd, get en put.` Voor elke `get` en `put` wordt er een aparte connectie opgezet om een bestand te transporteren. Bij *anonymous ftp* valt er voor de client niet zo veel te beschermen, maar voor de server-host des te meer! Anonieme FTP-klanten worden ingespert als asielzoekers in een gesloten afdeling die we *chroot* noemen.

de r-serie van BSD

In tegenstelling tot Telnet en FTP, die open staan voor alle systemen van de wereld, was de r-series. `rlogin`, `rcp` en `rsh`, bedoeld voor unixen onderling. Typische unix-gebruiken en -grappen lukken werken hier ook.

rcp is een uitbreiding van **cp**. De file- en directory-namen mogen voorafgegaan worden door een hostnaam met dubbele punt. `rcp filenaam systemB:bewaardir`

rsh start een shell op en ander systeem om een opdracht aldaar uit te voeren. *stdin* en *stdout* gaan van- en naar het lokale systeem.

```
rsh lincom df
rsh lincom who
rsh lincom who|sort
rsh lincom "who| sort"
rsh lincom who -Hu | rsh delcom lp
rsh lincom
```

De laatste start alleen een shell, zonder wachtwoord. Als dat niet is toegestaan vraagt start `rsh rlogin` die wel een wachtwoord vraagt. Als beveiliging van belang is moet de r-serie niet meer gebruikt worden!

voorbeeld	commentaar
`cp f1 f2`	Kopieer f1 naar f2. f1 en f2 kunnen , absolute padnamen zijn, beginnend met een '/' of relatieve, ten opzichte van de current directory. rcp doet precies hetzelfde.
`cp f1 f2 f3 f4 d1` `cp *.html d1`	De laatste padnaam (d1) móet een directory zijn waarin de f*n*-files en de html-files gekopieerd worden. Met rcp gaat dat precies zo.
`cp /tmp/f1 .`	Let op de punt, kopieer naar de current directory
`rcp lincom:f1 f2`	Kopieer `f1` op systeem lincom naar `f2` in je homedir
`rcp f1 f2 f3 f4 lincom:d1`	Kopieer de vier files naar de directory `d1` in je homedir op lincom
`rcp lzudir/*.html lincom:lzudir`	Kopieer de html-files naar de `www-public-dir` van de user janb.
`rcp lincom:lzud"/*.html" .`	Let op de quotes!

Noot: Voor bovenstaande opdrachten is het nodig dat de *user* een account heeft op *lincom* en daar in haar homedir een .rhosts file met de namen van de *hosts* waarvan de r-opdrachten toegestaan moeten zijn.

Geheimschrift

De cryptografie had een enorme ontwikkeling doorgemaakt sinds de eerste standaard, 'DES' en de eerste ideeën van Diffie & Hellman, beide in 1974. Rond 1994 verschenen de belangrijkste toepassingen: PGP van Phil Zimmermann, SSL (https) van Elgamal bij Netscape, CA's Verisign en Thawte .van RS&A en Shuttleworth en SSH van Tatu Ylönen.

Symmetrische encryptie

DES Digital Encryption Standard was al die tijd de enige standaard geweest. Maar DES is niet echt snel of veilig en werd door niemand vertrouwd. 3DES (3x DES) is veiliger maar uit der aard nóg trager. Het Belgische AES is tot nieuwe standaard (Advanced ES) verheven.

Het sleutelbeheer

Het probleem bij symmetrische encryptie is het distribueren en geheimhouden van de sleutels. Voor iedere correspondent heb je een aparte sleutel nodig. Binnen een groep van n personen of servers zijn dat er $\frac{1}{2}n*(n-1)$ sleutels die regelmatig vernieuwd moeten worden. In de praktijk niet haalbaar.

Asymmetrische encryptie

Aan Diffie en Hellman danken we een methode om een geheime sleutel af te spreken zonder dat luistervinken die kunnen ontdekken. Maar omdat dit interactief werkt is DH voor email ongeschikt.

Rivest, Shamir en Adleman kwamen door DH op een “private key” algoritme: een sleutelpaar haarvan de ene ontcijfert wan de ander vercijferd heeft en omgekeerd. Je kunt er zowel mee beveiligen (encrypten) als authentiseren (signeren) maar alleen korte berichten want het is erg rekenintensief. RSA wordt daarom alleen gebruikt voor het versleutelen van synnetrische sleutels en cryptografische controlesommen.

SSH en Open-SSH

Opdrachten en GUI's

Opdrachten **ssh, scp, sftp**, vervangen **rsh** en **rcp, telnet** en **ftp** op een veilige manier. De syntaxis is als die van de r-serie. SSH is beschikbaar op vrijwel alle platforms: alle unixen, WIN*, de Mac en zelfs PalmOS. **Open-ssh omvat zowel** protocol 1 als 2 en is een OpenBSD-implementatie van het SSHprotocol dat ontwikkeld en onderhouden wordt door Tatu Ylönen.

ssh en **scp** zijn weliswaar CLI-opdrachten, maar SSH heeft de GUI-gebruikers ook heel wat te bieden. Er bestaan GUI's voor **ssh** , zoals zijn **putty** en **WinSCP** voor WIN*, **Cyber Duck** voor de MAC en **KSSH** en het **sftp**: protocol voor de Konqueror en andere browsers voor KDE. Ook de *midnight commander* maakt dankbaar gebruik van de SSH-mogelijkheden.

> *Wie de CLI heeft leren kennen zal zich ook op de GUI's snel thuis voelen.*

Voorbeelden

Vervang de 'r' van BSD's r-serie door een 's' en je hebt de veilige SSH-opdrachten! Voorbeelden:

1. ```
   ssh hansrood@xs4all.nl
   mutt
   exit
   ```

2. ```
   scp -p lincom:.ssh/id_dsa.pub .
   ```

3. ```
   tar -cf - . |ssh lincom “cat >/var/bups/delsuse$(date +%x_%X)”
   ```

4. `scp OpenSSH.html hansrood@xs4all.nl:WWW/`

1- mail lezen bij xs4all, 2- haal de file id_dsa.pub op van lincom. 3- maak een backup naar host delsuse 4. met deze opdracht ga ik dit document op het Net beschikbaar stellen.

Hoe veilig is het?

De standaard installatie levert zonder, verdere bemoeienissen, een zogenaamde '*password authentification.' op.* Deze methode is al veel veiliger dan *telnet* en *ftp* en de *r serie.*
   ```

Authentisering van de server

Voordat je je wachtwoord ingeeft lijkt het verstandig om eerst de echtheid van de server te controleren. De server stuurt een ondertekend bericht met daarin zijn publieke sleutel (de *Host-Key*). De eerste keer dat we ergens inloggen moeten we de die sleutel zelf controleren, vaak aan de hand van de vingerafdruk. Klopt die dan antwoord je met een volmondig “yes”. De *pub-key* wordt dan bewaard in *$HOME/.ssh/authorized_keys* en wordt vervolgens gebruikt om de ondertekening van de server te controleren. Daarmee voorkom je dus *Phishing*, *Trojanen* en de «*man-in-the-middle».* Vervalste ip-adressen, en dito routers en *naamservers* kunnen geen kwaad meer. Daarna wordt een *sessiesleutel* afgesproken (met *RSA*-encryptie of, in versie 2 met een *Diffie-Hellman-geheim*). Met die sleutel wordt verder alles versleuteld (standaard met de verouderde en trage 3DES).

Authentisering van de client

Vervolgens log je in op de gebruikelijke manier: je krijgt een prompt «Password:» en te tikt het wachtwoord voor de remote login-account. Dit levert op zich niet zo veel gevaar op omdat het versleuteld verstuurd wordt maar levert toch nog een aantal bezwaren op:

1. Wachtwoorden zijn nu eenmaal lastig, als je die vaak moet ingeven, vooral de veilige lange wachtwoorden.

2. Bij ssh-opdrachten in shellscripts en bij de meer geavanceerde toepassingen met tunneling zijn wachtwoorden ook erg lastig.

3. Wachtwoorden zijn nooit echt veilig, denk aan afkijken tijdens intikken (door mensen, apparaten of software), opschrijven, chantage en vooral ook uitproberen (de zogenaamde brute kracht ontcijfering).

4. Je wachtwoord zou ook op onbetrouwbare ssh-servers kunnen uitlekken, je bent dus voor de beveiliging afhankelijk van andere systemen.

Hints voor de praktijk

1. Als je één of meer systemen vanaf het Internet toegankelijk maakt zul je merkem dat je dagelijks bezoek krijgt van “Koreaanse script-kiddies” en ander gespuis die lange rijen accountnamen uitproberen. Daar ben je vanaf zodra je het standaard poortnummer 22 vervangt door een hoger nummer. Hiervoor hoef je intern niets te veranderen, je zorgt dat je firewall, die de NAT verzorgt, het verkeer van buiten bestemd voor poort 34567 doorstuurt naar je sshd-server poort 22. En je vraagt je klanten om poort 34567 te gebruiken : ssh klant@jouwrouternaam -p 34567 .

2. Erg handig is dat je met ssh als root kunt inloggen (met telnet en rsh mag dat nooit). Helaas moet dat tegenwoordig toch weer ontraden worden. Daarnaast lijkt me niet meer dan logisch om accounts zonder wachtwoord uitsluiten van ssh-login . Dus voeg toe aan */etc/sshd_config* de regels:
   `PermitRootlogin no`
   `PermitEmptyPasswords no`

Veiliger

Bovenstaande methode, *ssh* met wachtwoord, werkt haast vanzelf: Op de meeste disrtibuties is het al aanwezig, anders installeer je zelf *ssh* en *sshd* voor de server. Het host-key-paar wordt daarbij gegenereerd. We kunnen het aan de client-kan nóg makkelijker en veiliger maken en helemaal geen wachtwoord meer gebruiken! We gaan voor de client-authenticatie ook encryptie gebruiken (RSA of DSA). We genereren daarvoor eerst een sleutelpaar:

```
janb@delsuse:~> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/janb/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/janb/.ssh/id_rsa.
Your public key has been saved in /home/janb/.ssh/id_rsa.pub.
The key fingerprint is:
62:70:cf:e9:32:5e:f1:d2:c7:14:2b:ad:c0:76:d5:8f janb@suscom
janb@delsuse:~> ls -lrt .ssh

totaal 12
-rw-r--r-- 1 janb users 837 2005-04-21 17:13 known_hosts
-rw-r--r-- 1 janb sshd 222 2006-06-21 17:43 id_rsa.pub
-rw------- 1 janb sshd 887 2006-06-21 17:43 id_rsa

janb@delsuse:~> cat .ssh/id_rsa.pub | ssh xs4all.nl “umask
177;test -d .ssh || mkdir .ssh;cat >>.ssh/authorized_keys”

janb@delsuse:~>
```

Géén wachtwoord of “wacht-frase”

Het *rsa*-sleutelpaar is hierboven aangemaakt zonder 'wachtfrase' en daarna toegevoegd aan de file ***xs4all.nl/ ~janb/.ssh/authorised_keys*** . Nu kunnen client én server elkaars digitale handtekening controleren en is er geen wachtwoord meer nodig. Controleer terdege of het werkt voordat je : *“PasswordAuthentication **no”*** configureert -houd een verbinding ondertussen open- want anders kom je er nooit meer in!! De voordelen zijn duidelijk, vooral ook als je geen user-intervention wilt, zoals bij *ssh*- en *scp*-opdrachten in shell-scripts.

De hele beveiliging berust nu dus op je login-wachtwoord. Dus uitloggen als je je systeem even wilt verlaten, of je lockt scherm en laat de screensaver ook een wachtwoord eisen. Anders kan elke voorbijganger even iets ondeugends uithalen op jou servers.

Een bezwaar is wel dat je private sleutels nu open en bloot in de *~/.ssh/id*-files staan en dat geeft niet echt een veilig gevoel. Maar ook daar is iets op gevonden!

Nóg veiliger: ***Passphrase*** en ssh-***agent***

Je kunt je private sleutels laten encrypten met een “passfrase” dat is een wat lang uitgevallen wachtwoord. Dan staat de sleutel nooit in bruikbare vorm op je schijf. Zelfs al wordt je systeem gestolen, dan nog zijn je private sleutels veilig, als je tenminste zo slim geweest bent om van je sleutelparen een kopie achter de hand te houden!

Maar nu ben je terug bij af en heb je dat vermaledijde wachtwoord weer terug, zal je zeggen. Daarvoor evenwel hebben we de *ssh-agent*, die, als een KDE Wallet, de passphrases beheert. *Ssh-agent* wordt meteen bij aanloggen opgestart (met *bash.rc* of *.xinitrc*) en vraagt eenmaal per sessie om de *passphrases*.

Met *ssh-add* geef je sleutelparen in bewaring. Het interface met de agent is een beetje merkwaardig: bij aanroep geef hij de shellopdrachten om twee environment-parameters te zetten die nodig zijn om met hem te communiceren. Zie hier hoe dat gaat.

```
delsuse:~ # ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-bNHTO10132/agent.10132; export
SSH_AUTH_SOCK;
SSH_AGENT_PID=10133; export SSH_AGENT_PID;
echo Agent pid 10133;
```

De ssh-agent wordt opgestart als daemon en schrijft twee shell-opdrachten naar *stdout.*

ssh heeft deze twee parameters nodig om de agent het authentiseren te laten afhandelen. Je geeft de *passphrase dus* nog maar één keer in, alleen bij het opstarten. Ook bij deze methode geldt dus dat je je console niet onbewaakt mag achterlaten!

Gebruik:

Als onderstaande je wat gecompliceerd overkomt... Excuses, het ís niet anders ;-)

De ssh-agent wordt op twee manieren opgestart:

met: `eval $(ssh-agent)`
daarmee worden de parameters gezet in het huidige shell-process en geëxporteerd naar alle kind-processen.

of met `ssh-agent xterm &`
zodat de `xterm` met de agent overweg kan.

of, meestal,: `ssh-agent /etc/X11/xinit/xinitrc`
waarna de gedurende de hele X-sessie de agent de ssh-agent beschikbaar is.

SSH-Trukendoos

Met rsh/ssh zijn de loop der tijd heel wat opvallende en bruikbare trucs bedacht. Hier twee veelgebruikte:

truc nummer 1 **X11-forwarding**

ssh lincom xterm
ssh -fX lincom netscape

Als je *ssh* start in een X-windowsomgeving kan je X-applicaties starten die dan de je lokale X-server gaan gebruiken. (**-f** geeft de gelegenheid om een wachtwoord in te geven, **`-X`** overrule-t de regel: '**X11Forwarding no**' in **`$HOME/.ssh/config`**

truc nummer 2: ***port-forwarding***

```
ssh -N -L 12345:localhost:110 hansrood@pop.xs4all.nl #als ik ssh-toegang tot
pop.xs4all zou hebben
```

```
ssh -N -L 12345:pop.xs4all.nl:110 hansrood@xs4all.nl #met alleen ssh-toegang tot mijn eigen xs4all-account
```

Daarmee zeggen we dat : “ De lokale poort (hier 1235) moet op de ssh-server (xs4all.nl of pop.xs4all.nl) worden doorgestuurd naar de popserver op poort 110.
Vervolgens moet ik bij mijn Kmail-pop-account invullen: *localhost* poort 12345. Dan haal ik voortaan op een veilige manier mijn mail op. Ook het wachtwoord wordt encrypt! Dit is slechts één voorbeeld van het gebruik van portforwarding voor veilige tunnels. Het kan voor veel protocollen worden toegepast. Je passeert er bovendien alle firewalls mee die poort 25-verkeer toelaten.

Samenvatting

1. *ssh, scp* en *sftp* vervangen de verouderde en onveilige TELNET, FTP en de **r-serie.**

1. Het eenvoudigst te installeren is de methode 'wachtwoord-authentisatie' . Deze methode geeft al een sterk verbeterde beveiliging.

2. Met weinig extra moeite voorkom je de risico's van het gebruik van wachtwoorden met de RSA/DSA authentisatie.

3. Met een *passphrase* en de *ssh-agent* voorkom je bovendien dat er private sleutels open en bloot op je harde schijf staan.

4. SSH is er voor CLI 'en GUI (GUI-clients en secure tunnels voor grafische applicaties)

Litteratuur:

1. C't 0607 root-server en sshd(130)

2. C't 0210 fish , DYNDNS +sshd iptables

3. LinMag 0510 ssh(24)

4. Linux Journal 0408 11 ssh-trucs

5. LFX 0510 SSH-key

6. LFX79 0605 p99

7. En natuurlijk de MAN(1) pages en RFC's rfc 4251/2/3