Kapitel 16

SSH och IPMI

16.1 SSH

SSH betyder Secure Shell och är som en lång säker sladd till varje dator som du har konto på. Föregångarna telnet, rlogin och rsh ska man inte använda längre, utom vid speciella anledningar, eftersom de är helt okrypterade. SSH skapades först av ett finskt företag, men den version som numera oftast används är OpenSSH som har sitt ursprung i OpenBSD.

För SSH heter servern `sshd`, klient för inloggning `ssh` och för filöverföring finns `scp` och `sftp`. Exempel på användning:

```
ssh lois-lane.mc.hik.se
```

Där loggar man in som samma användare som du kör på den lokala maskinen. Och använder standardporten 22.

```
ssh -l ao lois-lane.mc.hik.se
```

eller: `ssh ao@lois-lane.mc.hik.se`

Med dessa två sätt kan man ange användarnamn, det går även att skriva `-l root` efter datornamnet.

```
ssh -p 2222 tintin.kau.se
```

Där anger jag en annan port, att byta portnummer i `/etc/ssh/sshd_config` gör att man slipper de flesta automatiska scanningarna och därmed får renare loggar. Men det höjer inte säkerheten egentligen, en motiverad attackerare kan lätt scanna alla portar. Om du kör SELinux på servern måste du ange att SSH ska få köra på annan port, kommando för det är:

```
sudo /usr/sbin/semanage port -a -t ssh_port_t
-p tcp 2222
```

Kommandot `semanage` finns i paketet policycoreutils-python på RHEL och CentOS.

16.2 scp

För att säkert överföra filer kan du använda såväl `scp` som `sftp`. Skillnaden är att `scp` är mer avsedd för enstaka filer och `sftp` mer som ett alternativ till FTP. Här är exempel på användning:

```
scp lois-lane.mc.hik.se:/etc/passwd .
```

Syntaxen liknar vanliga `cp`. Man måste alltid ange både källa och mål. I det här fallet kopieras `/etc/passwd` från lois-lane till filen `passwd` i katalogen där jag står på den lokala maskinen. Glöm inte kolon eller punkt.

```
scp /etc/passwd lois-lane.mc.hik.se:/tmp/passwd.a
```

Där kopierar jag min lokala passwd till en fil i /tmp på lois-lane.

Om sshd kör på annan port måste även scp veta om det. Till exempel med `scp -P 2222`. Märk väl att det är stort P för scp men litet p för ssh. Än bättre lösning är att ange port i
`$HOME/.ssh/config` gemensamt för alla klienter.

16.3 ssh_config

Som jag nämnde i förra stycket så kan du ha en fil som heter `config` liggande i katalogen `.ssh` i din hemkatalog. Formatet är så här:

```
Host=tintin
  HostName=tintin.kau.se
  Port=2222


Host=lois-lane
  HostName=lois-lane.mc.hik.se
  Port=722
```

Med de inställningarna kan man skriva `ssh tintin` utan att behöva ange `-p` eller domännamn. Det fungerar även för `scp` och `sftp`. Man kan både ha likamed-tecken och mellanslag. Indrag är frivilligt, det är varje förekomst av ordet Host som avgränsar posterna. Det går även bra att skriva nyckelorden med små bokstäver. Utöver HostName och Port brukar jag använda User och CheckHostIP=no (för virtuella servrar som delar IP-nummer men har olika portar).

För mer information se manualen med `man ssh_config`, observera att det är ett understreck där, liksom det är för `man sshd_config`.

16.4 SSH-nycklar

Om man inte gör något annat använder SSH samma lösenord som vid vanlig inloggning på maskinen, men man kan även använda publika nycklar. De sparas i en fil som heter `~/.ssh/authorized_keys` på servern du vill logga in på.

Se 15.4 om du inte minns hur nyckelpar vid asymmetriskt krypto fungerar. På servern du vill logga in från kör du kommandot `ssh-keygen`. Det skapar ett nyckelpar i `.ssh`. Den privata nyckeln heter `id_rsa`, den ska du vara rädd om och inte sprida. Du får en fråga om en lösenordsfras (passphrase), det bör du använda till nycklar för interaktiv användning. Det kan vara ett lösenord, en mening eller en liten dikt om du är poetiskt lagd. Denna lösenordsfras tillhör den privata nyckeln och skickas aldrig till servern som

du loggar in på. Du får upprepa frasen som skydd mot felskrivning. Den publika nyckeln hamnar i en fil som heter `id_rsa.pub`. Den är inte lika viktig att skydda, men du bör inte sprida den för mycket. På servern som du vill logga in på lägger du den publika nyckeln i `authorized_keys`. Till exempel så här:

```
[ao@a ~]$ scp .ssh/id_rsa.pub f:
ao@f's password:
id_rsa.pub       100%  399     0.4KB/s   00:00
[ao@a ~]$ ssh f
ao@f's password:
Last login: Sat Feb 23 13:07:41 2013 from a
[ao@f ~]$ cat id_rsa.pub >> .ssh/authorized_keys
[ao@f ~]$ rm id_rsa.pub
[ao@f ~]$ chmod 600 .ssh/authorized_keys
[ao@f ~]$
```

Det går att göra på andra sätt, som att klipp-o-klistra i en editor eller använda `cat` och `ssh` med pipelines, men det här sättet kan vara enklare att förstå. Allra enklast är att använda `ssh-copy-id`. Anledningen till att jag använde `>>` är för att inte skriva över eventuella befintliga nycklar i `authorized_keys`. Men är du säker på att den är tom (inte finns) så går det förstås lika bra med `cp`. När nyckeln är på plats ska det gå att logga in utan lösenord. Så här:

```
[ao@a ~]$ ssh f
Last login: Thu Feb 28 01:17:56 2013 from a
[ao@f ~]$
```

Eftersom jag kör GNOME på datorn som heter a så fick jag frågan om lösenordsfras i ett annat fönster. Och det sparas i en nyckelring så nästa gång får man ingen fråga alls. Körs inget grafiskt gränssnitt på datorn du loggar in från får du frågan direkt efter prompten, till exempel om du loggar in från en server till en annan. Det ser ut så här:

```
[ao@a ~]$ ssh f
Enter passphrase for key '/home/ao/.ssh/id_rsa':
Last login: Thu Feb 28 01:32:04 2013 from a
```

```
[ao@f ~]$
```

Ett vanligt fel vid inloggning med publik nyckel är rättigheter på kataloger och filer. `.ssh` ska ha 700, `authorized_keys` ska ha 600 och båda ska förstås ägas av rätt användare. Inte heller får användarens hemkatalog vara skrivbar för andra. Ett annat vanligt misstag är att få med radbrytningar i `authorized_keys`. Om du kör `pico` måste du använda `pico -w` för att den inte ska lägga till nyradstecken.

Det går bra att använda samma nyckelpar till olika servrar, och att ha flera olika publika nycklar i samma `authorized_keys`.

Vill du ha automatisk inloggning, till exempel för cronjobb som backup, så kan du ange en tom lösenordsfras. Då kan alla som har tillgång till den privata nyckeln logga in utan lösenord. Det rekommenderas enbart från säkra till mindre säkra maskiner, t.ex. från en central backup-server bakom brandvägg och med få användarkonton till servrar ut mot Internet, inte åt andra hållet. Då trycker du bara enter två gånger vid `ssh-keygen`. Det är mycket praktiskt, men som sagt lite riskabelt. För att begränsa riskerna lite kan du i `authorized_keys` explicit lista vilka kommandon som får köras med t.ex. command=”/sbin/reboot” och från vilka adresser man får logga in med: from=”192.168.128.1”. Glöm inte citattecknen och skriv command= och from= först i samma rad som nyckeln. Testa ny inloggning innan du loggar ut, det är lätt att göra fel. Syntaxen för filen `authorized_keys` beskrivs i `man sshd_config` (inte i ssh_config).

16.5 IPMI

IPMI betyder Intelligent Platform Management Interface. Det finns inbyggt på många serverklassmoderkort och är ett sätt att ”telnetta till BIOS”, alltså fjärrstyra hårdvara även utan fungerande operativsystem.

Med IPMI kan man starta, stänga ner och logga in på maskiner, helt oberoende om operativsystemets nätverk är uppe. Man tilldelar maskinen ett privat IP-nummer (RFC1918) och kan då logga in från andra maskiner på samma subnät. Det är bra att ha vid krascher och konfigurationsmisstag.

Det finns flera olika verktyg för IPMI, jag beskriver här IPMItool. Det som behövs finns i paketen ipmitool och OpenIPMI på Red Hat. I exemplet vill jag kunna fjärrstyra dator a från dator b.

Börja med att konfigurera IP-adress och konto på dator a, det kan man göra både från BIOS och Linux. Så här kan man göra det i Linux:

```
[root@a ~]# /etc/init.d/ipmi start
[root@a ~]# ipmitool lan set 1 ipsrc static
[root@a ~]# ipmitool lan set 1 ipaddr 192.168.200.10
[root@a ~]# ipmitool lan set 1 netmask 255.255.255.0
[root@a ~]# ipmitool user set name 2 admin
[root@a ~]# ipmitool user set password 2
Password for user 2:
Password for user 2:
[root@a ~]# ipmitool user enable 2
```

Sätt ingen default gateway om du inte måste, det är säkrare att IPMI enbart är åtkomligt från LAN:et. Lösenordet överförs i klartext för vissa varianter av protokollet, och det ska ju bara användas vid nödfall. Så välj något lösenord som du inte använder till andra saker, och spara det på ett säkert ställe, som root:s hemkatalog på servern du ska fjärrstyra från. Normalt sett bör man inte spara lösenord i klartext, men för IPMI kan det vara praktiskt att skriva det direkt i skripten, och köra `chmod 600` på dem.

För att kunna logga in behöver du aktivera ”console redirection” i BIOS på maskinen. Det sätter upp en virtuell serieport. För att komma åt GRUB lägger du till följande i grub.conf eller menu.list:

```
serial --unit=2 --speed=19200 --word=8 --parity=no \
   --stop=1
terminal --timeout=10 serial console
```

Kommentera bort eventuell splash. För den kärna du använder lägger du till `serial console=ttyS2,115200n8` och tar bort `rhgb quiet`. Med det tillägget ska du kunna se hela bootprocessen och få inloggningsprompt.

Då är det dags att prova från dator b. Om dator a har flera nätverksuttag så är det enbart ett som kan användas

för IPMI, så se till att det är rätt. På dator b kör du:
`sudo /sbin/ifconfig eth0:0 192.168.200.11`

Om allt är rätt ska du kunna köra:

```
ipmitool -I lanplus -H 192.168.200.10 -U admin \
-P hRttju65 sensor
```

Den ska visa värden på temperatur och fläkthastighet. Strängen efter -P är förstås ditt lösenord. Om det fungerar så har du satt upp nät och användare rätt. Nästa steg att kolla är SOL (serial over lan), det gör du med:

```
ipmitool -I lanplus -H 192.168.200.10 -U admin \
-P hRttju65 sol activate
```

Andra användbara kommandon är `power on` och `power reset` för att starta och starta om hela datorn. Lägg gärna in de du tänker använda i skript så du slipper leta på nät och i manualer vid skarpt läge. Lägg in IP-nummer i skripten eller i `/etc/hosts` så du säkert kommer ihåg dem när det behövs. Och som sagt tycker jag att man även kan spara lösenord i dessa skript, men det är en kompromiss med högre tillgänglighet än sekretess.

Datorn som du fjärrstyr från, b i mitt exempel, behöver enbart ha programmet ipmitool installerat, den behöver inte ha moderkort med IPMI-stöd. Men har du två eller flera servrar med IPMI så bör du sätta upp IPMI på alla så de kan fjärrstyras från varandra. Det ska räcka att datorn har ström för att de ska gå att komma åt.

HP har ett annat system som heter iLO (Integrated Lights Out), som har liknande funktioner.