

SSH (Secure Shell)

Inledning

Det jag kommer att behandla i denna uppsats är bland annat vad SSH är för något, hur den används och tillämpningar på hur den kan användas i praktiken. Uppsatsen inriktar främst till personer med mycket datorvana och som läst första delen av kursen Tillämpad Datanätsteknik på Jönköpings Tekniska högskola. Tillämpningarna baseras för unix-plattformarna (Linux, Unix och Mac OS X), men det är inget krav att ha bekantat dig med dessa plattformar sedan tidigare. Även användare av Windows kan göra många av tillämpningarna i denna uppsats.

Vad är SSH?

Enligt Wikipedia skapades protokollet 1995 av en vid namn Tatu Ylönen. Han var vid tiden forskare vid Tekniska Högskolan i Helsingfors, Finland. Under samma år har redan 20 000 användare börjat använda SSH, mycket tack vare att han släppte programvaran helt gratis [1]. SSH ger möjligheten att koppla upp sig från sin hemdator, via internet och till en dator med SSH-server var som helst i världen och administrera denna via en terminal precis som om du hade tillgång till datorn fysiskt med skärm och tangentbord. Detta är i sig inget unikt, protokollet Telnet ger samma möjlighet och uppfanns redan på 1970-talet [2]. Men det som gör SSH till Telnets ersättare är att den är betydligt mycket säkrare. När en anslutning mellan två datorer upprättas med hjälp av SSH är den krypterad med symmetrisk kryptoalgoritm (både krypterar och dekrypterar med samma nyckel), vanligen med 3DES eller hmac-md5 beroende om man använder SSH-1 eller SSH-2. Trots att protokollet redan var populärt blev den godkänd som internetstandard så sent som år 2006 av IETF (Internet Engineering Task Force, de som publicerar RFC-dokumenterna [3]) [2].

Vad är SSH-1 och SSH-2?

Man kan säga att det är två olika versioner av SSH-protokollet. SSH-1 lanserades i början av året 1995 av Ylönen. Dock upptäckte man att första versionen hade ett flertal brister. Bland annat var det möjligt att göra en så kallad Man-in-the-middle attack, vilket innebär att en potentiell hacker kan ta sig in i en anslutning och agera som en mellanhand. Detta innebär att avlyssning av trafik är möjligt samt att den själv kan låtsas om att den är ena eller den andra datorn i uppkopplingen, och därmed kan den även på egen bevåg hämta information och köra program utan att offrets dator misstänker något.

För att komma till rätta med dessa problem förbättrades bland annat säkerheten kring de krypteringsnycklar som växlas mellan klient-datorn och server, samt införde säker integritetskontroll genom MAC [2](*message authentication code*) (vilket är ungefär som att jämföra med ett sigill på ett brev. Detta för att försäkra sig om att ingen annan mellan mottagaren och sändaren har läst meddelandet[8]). Dessa ändringar gjordes efter en granskning av protokollet redan 1996. Utöver ändringar i säkerhet infördes även möjlighet att använda flera sessioner (användare) på samma SSH-uppkoppling [2].

Ska man idag upprätta en SSH-server så ska man inte använda sig något annat än SSH-2. Det finns inga fördelar att använda SSH-1 jämfört med SSH-2. Enda anledningen att SSH-1-stödet fortfarande existerar beror på att det används fortfarande av många företag och organisationer.

När används SSH?

Idag används SSH huvudsakligen för följande uppgifter:

Fjärrstyrning av unix-plattformar

Troligen den mest förekommande arbetsuppgiften för SSH-protokollet är via en terminal fjärrstyra unix-baserade system. En terminal i unix-världen är motsvarigheten till Start → Kör → cmd i windows, eller DOS om det känns mer bekant.

Det som många administratörer på unix-system gillar med terminalen (bland annat) är att man kan administrera en dator enbart genom att skriva olika kommandon istället för att motsvarigheten på windows, att köra VNC eller Fjärrskrivbord som kräver att man har en god uppkoppling för att bilderna ska komma fram och inte man ska upplevas som att det är segt. Kör man en terminal och vill fjärrstyra unix-system krävs inte lika bra uppkoppling, eftersom att det är trots allt enbart text som skickas och därmed mindre data måste laddas ned.

Filöverföring med SFTP eller SCP

FTP är ett underbart protokoll som gör det möjligt att på ett smidigt sätt att överföra filer mellan två datorer. För en administratör som vill hantera känsliga system-filer eller viktiga dokument vet att FTP tyvärr är ett protokoll som inte erbjuder mycket skydd, all trafik är avlyssningsbar för vem som helst. Lösning?

En lösning är att använda SFTP (SSH File Transfer Protocol) eller SCP (Secure Copy) som är så kallade filöverföringsprotokoll, möjligheten att skicka filer över en förbindelse [4]. Vilket av dessa två du använder sig av spelar ingen större roll eftersom att man normalt inte märker någon skillnad samt att de båda har en snarlik syntax om man skulle vilja skriva kommandona själv [5].

Att använda SFTP i ett FTP-program med stöd för protokollet, exempelvis FileZilla, går man ungefär samma sätt till väga när du ska ansluta dig till en beröringsskyddad FTP. Du fyller i adressen till servern, port-nummer, användarnamn samt lösenord (vilket är samma uppgifter när du gör en vanlig SSH-anslutning via terminal). Skillnaden blir att du väljer att det är en SFTP-anslutning istället för vanlig FTP. Nackdelen med SFTP är att normalt får man tillgång till hela systemet. Man förlorar med andra ord möjligheten att begränsa en användare så att den enbart får tillgång till filer i en viss mapp eller en del av ett system, vilket FTP kan göra.

Tunnling av andra anslutningar

Om vi analyserar ovanstående problem med FTP så ligger problemet i att förbindelsen mellan två datorer inte är krypterad. Hur gör man för att säkra förbindelsen?

Ett sätt är att använda FTPS (som står för SSL File Transfer Protocol och ska inte förväxlas med SFTP [6]), där man skaffar sig ett SSL-certifikat för att säkra anslutningen. Nackdelen är att ett certifikat genererat av en vanlig administratör innebär en säkerhetsrisk eftersom att det är upp till den som ska använda uppkopplingen att bedöma om certifikatet är riktigt. Vill man att det ska ske per automatik måste certifikatet genereras av Certificate authority (CA), vilket kostar pengar [7].

SSH löser samma problem, utan aktiv inblandning av användaren och utan att behöva öppna plånboken. Det man gör är att man upprättar en så kallad SSH-tunnel, vilket gör att den lyssnar efter en viss trafik från en dator och skickar den istället via en den tunnel som man har upprättat. Om man vill kan man se det som bilar som färdas på våra vägar. Vill

man följa en speciell bil från helikopter är det oftast inga problem, men säg att bilen åker igenom en tunnel istället kan inte helikoptern följa efter och det är näst intill omöjligt att se var bilen är någonstans i tunneln och vad den gör.

Hur används SSH?

Jag tänkte demontera hur SSH används inom de områden som protokollet är vanligast inom. Tanken är att dessa ska utföras som experiment eller som en guide till hur du kommer igång och använda SSH, men det går lika bra att enbart hänga med i texten.

Om du väljer att följa mina övningar bör du tänka på att de utgår från att de utförs på en unix-plattform. Jag har själv använt mig av Ubuntu Server 9.04 (linux) där jag installerade SSH-servern samt kört den SSH-klient som finns förinstallerat i operativsystemet Mac OS X version 10.6.2. Det fungerar även med andra versioner och plattformar än det jag kör här, men det kan inte garanteras att experimenten fungerar fullt ut på dessa.

När det gäller klient så finns det tillgängligt på samtliga plattformar. Som jag nämnde så finns SSH-klienten oftast förinstallerat med unix-baserade system (Linux, Unix och Mac). Windows har ingen SSH-klient som standard, dock går det att ladda ner ett program som heter Putty som är gratis att ladda ned och använda. Det finns även klienter till mobiltelefoner så som iPhone. Jag tänker inte gå igenom hur varken Putty eller mobila klient-ssh-applikationer fungerar eller hur man använder dessa.

Första steget, installera SSH-server

Det första du behöver fundera ut är på vilken dator du kan installera SSH-servern på. Servern ska installeras på den dator som du vill kunna ändra och köra de kommandon som du säger åt den att den ska utföra. Du bor i Jönköping och du har en server i Stockholm som du vill administrera eller fjärrstyra, då ska SSH-servern installeras på servern, såvida om du inte väljer att köra upp till Stockholm varje gång du vill göra en ändring på servern. Klienten ska finnas på den dator där du skriver dina kommandon, oftast din arbetsdator som du antagligen har i din bostad (detta fall Jönköping). När du har klurat ut detta kan du installera SSH-servern.

Att installera SSH-server i Ubuntu är barnsligt enkelt. Allt man behöver göra är att öppna en terminal och skriva kommandot nedan. I serverversionen av Ubuntu är den igång direkt efter att du har loggat in, men för desktop-versionen behöver man gå in i Program → Tillbehör → Terminal för att öppna ett terminalfönster. Nu skriv följande kommando:

```
sudo apt-get install openssh-server
```

Eventuellt behöver du ange det lösenord som du har till det konto som har administratörsrättigheter, men efter att den är klar med nedladdning och installation är allting klart och redo att köras. Du kan gå till nästa experiment. Vill man trimma säkerhetsinställningarna så är denna sida värd att läsa (<http://mysql-apache-php.com/ssh-attacks.htm>), men normalt behövs detta inte. Dock kan det vara bra att ändra porten från 22 (som ssh använder som standard) till en annan som du behagar.

För SSH-server på Mac kan följa denna guide (http://stocksy.co.uk/articles/Mac/ssh_on_mac_os_x/). Installation på andra linux operativsystem än Ubuntu kan variera, googla på "install openssh server <din linux os>". Vill man köra SSH-server på windows kan man försöka att installera MobaSSH.

Fjärrstyr SSH-servern

Nu till den roliga biten, att ta kontroll över en annan dator på distans. För att upprätta en

förbindelse skriver man följande i terminalen:

```
ssh adress.eller.ip-nummer.till.server -p 22 -l användarnamn
```

Ingående förklaring:

1. ssh – Namnet på programet som ska köras, dvs. SSH-klienten
2. adress.eller.ip-nummer.till.server – Det klienten behöver veta är vilken IP-nummer som servern har fått av Internetleverantören. Det går även bra med en vanlig domänadress (ex. hj.se) eller sitter man lokalt i samma nätverk som servern kan man också använda 192.168.X.Y.
3. -p 22 – Talar om vilken port som den koppla upp sig till. Port 22 är standard för SSH.
4. -l användarnamn – Vilket konto du ska använda dig av på servern för att göra dina ändringar. Det är samma typ av konto som man vanligtvis brukar använda och skapa som motsvarande i Windows. Innan man kan komma till skrivbordet ska man logga in på sitt personliga konto, på likande sätt är det även för unix-plattformar. När ett nytt konto skapas får den som standard även tillgång till SSH-inloggning (på unix-plattformar).

När du kör kommandot kommer du förmodligen att mötas av något med "RSA-key". Detta är bara en säkerhetsfunktion som gör att när du kontaktar en ny server för första gången måste du godkänna denna innan du får börja använda den. Allt du behöver göra är med andra ord skriva 'yes' för att bekräfta att servern är säker. Detta behöver du bara göra en gång när du kopplar mot en ny server för första gången.

Men något som du vanligtvis möter varje gång är att du behöver fylla i ett lösenord. Det är samma lösenord som för det användarnamn som du angav i kommandot. Det man bör tänka på är att när man fyller i lösenordet så ser det ut som att inget händer på skärmen. Detta är fullt normalt, och jag kan försäkra dig om att du verkligen fyller i lösenordet. Det man har gjort är att man har tagit bort asteriskerna (*) för att man inte ska kunna se antalet tecken som skrivs in.

Om lösenordet godkänns är du nu inloggad på servern. Prova nu att göra något på servern, så som att skriva några kommandon och se vad som händer. Exempel på kommandon (varje kolon (;) är ett kommando): ls, uptime, whoami, cat /etc/ssh/sshd_config.

Skapa en tunnel

I detta experiment tänkte jag visa hur man skapar en SSH-tunnel och hur man använder den i en partisk tillämplig. Jag hade tänkt att med hjälp av SSH-tunneln säkra all trafik som kommer till och från en webbläsare. I detta fall har jag valt att FireFoxs trafik ska bli säkrad, men det går förstås lika bra med vilken annan webbläsare som helst.

Första steget är att öppna upp Firefox och gå till webbplatsen <http://whatismyip.com/> och se vilket IP-nummer du har på ett ungefär. Detta har egentligen inget med skapandet av tunneln att göra, men jag kommer senare vilken betydelse detta steg har.

Skapa nu tunneln genom att öppna en terminal och skriv följande kommando:

```
ssh -D 2222 -C <användare>@adress.eller.ip-nummer.till.server -p 22 -N
```

Ingående förklaring:

1. ssh – SSH-klienten

2. -D 2222 – Skapa en dynamisk koppling med port 2222 (du kan välja vilken annan port du vill, dock väj inte en reserverad port så som 80, 21, 22, 25 osv.)
3. -C <användare> @ [...] till.server – Uppgifter om användarnamn som man ska logga in på servern med samt adressen till servern. Snabel-a (@) ska alltid finnas där för att kunna urskilja användarnamnet respektive adressen till servern.
4. -p 22 – Port 22 används
5. -N – Kör inga vidare kommandon på servern. Djupare än så går jag inte igenom på denna. Dock är det starkt rekommenderat att ha kvar denna.

Om allt går som planerat har du nu skapat en tunnel. Öppna Firefox igen och gör följande:

1. Gå till Proxy-inställningarna

Mac

Firefox → Inställningar... → Avancerat → Nätverk → Inställningar...

Windows och Linux

Redigera → Inställningar... → Avancerat → Nätverk → Inställningar...

2. Ställ in följande inställningar

- Bocka i "Manuell proxykonfiguration"
- Under "SOCKS-värld:", fyll i **localhost**
- På "Port:" (på samma rad som "SOCKS-värld:") skriver du **2222**, om du inte själv valt en annan port.
- Bocka i "SOCKS v5"

3. Klicka OK och återvänd till webbläsarfönstret

När du nu uppdaterar eller försöker att gå in på sida igen kommer IP-nummret från första steget att ha ändrats till samma ip-nummer som till den server som du har anslutit SSH-tunneln till.

"Vad har hänt?", kanske du undrar. I och med att all trafik till och från webbläsaren går via tunneln innebär det även också att servern sköter all trafik till och från på det öppna Internet. Med andra ord, när du vill gå in på whatismyip.com kommer webbläsaren skicka förfrågan genom tunneln och till servern. Servern har blivit instruerad att enbart skicka trafiken vidare. Servern för whatismyip.com tar emot förfrågan och tar nu reda på vilken IP-nummer som trafiken kommer från har. Eftersom att den enbart kan se att trafik kommer från servern och inte trafiken skickades ursprungligen så tror den att det är servern som ursprungliga avsändare och därmed får man som svar servens ip-nummer när det har laddat in i webbläsaren. Vi har med andra ord skapat oss en enkel proxy-server. Vill du verifiera det bara berör Firefox så kan du öppna en annan webbläsare (exempelvis safari eller Internet Explorer) och gå in på whatismyip.com.

OBS! Även om trafiken från din arbetsdator till servern är krypterad så innebär *inte* att trafiken från servern till whatismyip.com är säker eftersom det är ingen tunnel upprättad mellan dessa. Med andra ord är denna trafik full läsbar för utomstående.

Vill du prova andra applikationer (så som FTP) istället för en webbläsare kan du göra det

utan problem. Enda kravet är att den har möjlighet att ställa in proxy med hjälp av SOCKS v5. Då öppnar du först en tunnel och sedan går du till programmets inställningar och fyll i SOCKS-inställningarna på ett motsvarande sätt som ovan.

Källförteckning

- [1] "Telnet", Källa <http://en.wikipedia.org/wiki/Telnet> [2009-11-10]
- [2] "SSH", Källa <http://sv.wikipedia.org/wiki/SSH> [2009-11-10]
- [3] "Internet Engineering Task Force",
Källa <http://sv.wikipedia.org/wiki/IETF> [2009-11-13]
- [4] "SFTP", Källa <http://sv.wikipedia.org/wiki/SFTP> [2009-11-11]
- [5] "SCP", Källa <http://sv.wikipedia.org/wiki/SCP> [2009-11-11]
- [6] "SSL File Transfer Protocol", Källa <http://sv.wikipedia.org/wiki/Ftps> [2009-11-11]
- [7] "SSL-certifikat", Källa <http://sv.wikipedia.org/wiki/SSL-certifikat> [2009-11-11]
- [8] "Analysis and improvement of message authentication codes in
OpenSSH",
Källa <http://valchev.net/peter/cpsc503/proposal/UMACproposal.pdf> [2009-11-15]