Sichere Internetkommunikation mit SSH (Secure Shell)

Name des Vortragenden: Oliver Litz
Matrikelnummer: xxxxxxx

Name des Vortragenden: Daniel Rößler
Matrikelnummer: xxxxxxx

Fachbereich: GIS
Studiengang: Praktische Informatik
Fach: Internet, Protokolle, Dienste
Betreuer: Wolfgang Pauly

Datum: 21.01.2000

Vortrag im Wintersemester 1999/2000
an der HTW des Saarlandes

Inhaltsverzeichnis

1. Was versteht man unter SSH ?

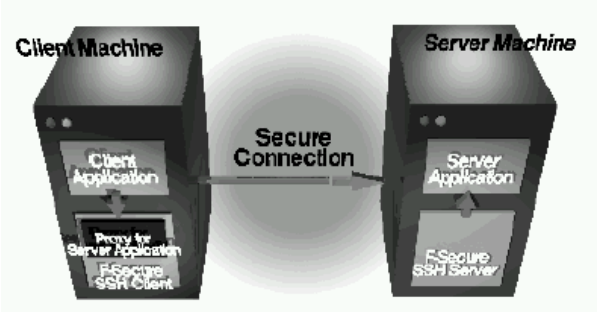
"Die SSH (Secure Shell) ist ein Programm mit dessen Hilfe man sich über ein Netzwerk auf anderen Computern anmelden, auf entfernten Computern Befehle ausführen und Dateien zwischen Computern übertragen kann. Es bietet "starke" Authentifizierung und sichere Kommunikation über unsichere Kanäle. Es ist konzipiert als Ersatz für rlogin, rsh und rcp." [Übersetzung aus SSH−FAQ, Kapitel 1.1]. Die SSH wurde ursprünglich von Tatu Ylönen an der TU Helsinki / Finnland entwickelt.

1.1 Wozu wird die SSH benötigt ?

Der gesamte Datenverkehr im Internet basiert auf dem TCP/IP Protokoll. In der momentan weltweit eingesetzten Version von IPv4 werden jedoch alle Daten unverschlüsselt übertragen. Die auf IP aufbauenden Protokolle treffen keine weiteren Schutzmaßnahmen zur Sicherung der Kommunikation. Sämtliche Kommunikation kann also abgehört werden, da die Übertragung über mehrere Computer stattfinden kann. An jedem dieser Punkte, besonders an wichtigen Knotenpunkten, ist es möglich durch sogenannte "Sniffer"−Programme den Datenverkehr nach bestimmten Kriterien automatisiert zu durchsuchen. Da ebenso Passwörter wie Email unverschlüsselt übertragen werden, ist es also leicht möglich, Zugang zu Kennungen zu erhalten, Emails mitzulesen oder Kreditkartennummern abzufangen.
Dies ist besonders bei Unternehmen kritisch, da leicht Betriebsgeheimnisse in die Hände von Konkurrenzunternehmen oder Geheimdiensten gelangen können. Ein gutes Beispiel hierfür ist das Echolon−System, welches sämtliche Telefon− und Internetkommunikation nach bestimmten Stichwörtern automatisiert abhört und Informationen sammelt. Aber auch der Staat könnte sich einen Überblick über die Gewohnheiten und Vorlieben jedes einzelnen Bürgers machen.
Eine weitere Gefahr liegt in der "Echtheit" der Verbindung. Woher weiß man, dass gerade mit diesem Rechner oder diesem Benutzer kommuniziert wird, oder ob die Identität nur vorgetäuscht ist. Die Eindeutigkeit eines Rechners ist durch die IP−Adresse gegeben, jedoch kann jemand eine falsche IP−Adresse vortäuschen (IP−Spoofing) und sämtliche Kommunikation dorthin abfangen. Ebenso können die Daten dann manipuliert werden und falsche Informationen weitergesendet werden (Man in the middle attack). Es ist auch möglich bestehende Verbindungen auf einen anderen Rechner zu entführen (Hijacking). Einem normalen Benutzer fallen solche Manipulationen nicht auf.

1.2 Leistungsmerkmale der SSH

Wie schützt man sich also vor solchen Angriffen ? Man benötigt ein System, das einen vor zwei verschiedenen Angriffsarten schützt. Die SSH gewährleistet die Vertraulichkeit der Daten durch "starke" Verschlüsselung. Dies bedeutet, dass es nicht ohne weiteres möglich ist, die Daten zu entschlüsseln, wie z.B. durch Analyse des Quelltextes. Eine "starke" Verschlüsselung muss einem Angreifer einen enormen Rechenaufwand zur Entschlüsselung entgegenstellen. Nur an den beiden Endpunkten der Kommunikation können die Datenpakete ver− und entschlüsselt werden. Die Authentizität der Informationen wird gesichert, d.h. jeder Endpunkt kann sich sicher sein, dass die Identität des anderen gewährleistet ist. Eine Verschlüsselung findet schon vor der Authentifizierung statt, somit werden auch keine Passwörter unverschlüsselt übertragen. Die folgende Abbildung 1 veranschaulicht eine SSH−Verbindung (entnommen aus F−Secure SSH, S. 2):



(Abbildung 1: SSH−Verbindung)

Die SSH bietet aber noch weitere interessante Möglichkeiten. Das Display eines entfernten Computers kann auf den lokalen Rechner umgeleitet werden. So ist es auch mit einem Windows−Client möglich UNIX−Programme auf dem lokalen Bildschirm zu benutzen. Port−Anfragen können weitergeleitet werden, wodurch man z.B. Emails sicher abrufen kann. Diese Technik wird als "Tunneling" bezeichnet. Ein lokaler Proxy−Server wird gestartet, der auf Anfragen an einen lokalen Port wartet. Eine solche Anfrage wird dann an den Port der entfernten Maschine über die SSH−Verbindung verschlüsselt weitergeleitet. Dies ist bei allen TCP/IP−Diensten wie HTTP, Telnet, POP3 oder SMTP möglich.
Bei langsamen Verbindungen kann es vorteilhaft sein, den kompletten Datenstrom zu komprimieren. Dies kann ebenso transparent geschehen wie die Ersetzung der Programme "rlogin", "rsh" und "rcp".

Wichtig ist aber, dass auf beiden Endpunkten der Kommunikation SSH eingesetzt wird, ansonsten läuft auf einer Teilstrecke der Datenstrom unverschlüsselt weiter. Auf einem POP3−Server kann also nur die Email verschlüsselt abgerufen werden, wenn dort ein SSH−Daemon aktiv ist.
Zu beachten ist das nicht in allen Ländern "starke" Verschlüsselung erlaubt ist ! Auch in Deutschland gibt es immer wieder Diskussionen über Einschränkung von Kryptographie.

2. Verschlüsselungsalgorithmen

Eine Anwendung ist nur so sicher wie die darin eingesetzten Algorithmen. Da die grundlegenden Funktionen der SSH auf der Verschlüsselung und der Authentifizierung beruhen, müssen diese auf ihre Sicherheit besonders kritisch analysiert werden.
Folgende Verschlüsselungsalgorithmen werden von der SSH zur Verfügung gestellt:
IDEA, DES und 3DES, weiterhin unterstützt werden Blowfish, Twofish und Arcfour. Wir betrachten hier nur die ersten drei Algorithmen, da diese am häufigsten eingesetzt werden.

2.1 IDEA

IDEA geht auf die Entwicklung von Xuejia Lai und James Massey zurück. Sie entwickelten 1990 einen Verschlüsselungsalgorithmus namens PES (Proposed Encryption Standard). Dieser wies jedoch einige Schwachpunkte auf, die dann in der überarbeiteten Version IPES (Improved Proposed Encryption Standard) beseitigt wurden. Ab 1992 kennt man diesen Algorithmus unter dem Namen IDEA (International Data Encryption Standard).
IDEA ist ein symmetrisches Blockchiffre−Verfahren. Es werden immer ganze Blöcke mit Daten chiffriert oder dechiffriert, nicht Byte für Byte. Symmetrisch bedeutet, dass zur Verschlüsselung und zur Entschlüsselung der gleiche 128−Bit Schlüssel benutzt wird. IDEA chiffriert in acht Runden mit anschließender Ausgabetransformation. In jeder Runde werden Teilschlüssel benutzt. Zur Dechiffrierung wird ein ähnliches Verfahren rückwärts angewendet.
Ascom Systec hält ein Patent in den meisten Staaten, wodurch IDEA nur für nicht−kommerzielle Nutzung frei ist. Deshalb wird meist das patentfreie DES oder Triple−DES (3DES) verwendet.

2.2 DES und Triple DES (3DES)

DES ist wie IDEA ein symmetrisches Blockchiffre−Verfahren. DES (Data Encryption Standard) wurde 1975 von IBM und dem amerikanischen Geheimdienst entwickelt. Ein 56−Bit Schlüssel ist heute nicht mehr vor "Brute−Force−Attacks" (Ausprobieren aller Möglichkeiten) sicher. Deshalb wird meist 3DES mit einem 168−Bit Schlüssel eingesetzt. 3DES verwendet drei mal den selben DES−verschlüsselten Block mit drei verschiedenen Schlüsseln. Die wirkliche Schlüsselstärke liegt bei etwa 112−Bit. 3DES ist jedoch langsamer als IDEA.

2.3 Wieso ist IDEA, DES, 3DES und damit die SSH sicher ?

Durch die Offenlegung der Verfahren kann jeder eine kryptoanalytische Überprüfung durchführen. Alle drei Verfahren wurden lange getestet und von Experten überprüft. Dabei sind keine verborgenen Sicherheitslücken gefunden worden. Geht man von einer "Brute−Force−Attack" aus, benötigte man bei IDEA eine Milliarde Chips, die in einer Sekunde eine Milliarde Schlüssel durchprobieren um in ca. 10 Billionen Jahren alle möglichen Schlüssel zu erhalten. Statistisch gesehen könnte man den Schlüssel schon früher finden. Man wird aber weder solange am Leben sein, noch soviele Chips bauen können, die eine solche Leistung bieten.

2.4 Problem bei symmetrischen Verfahren

Bei den symmetrischen Verfahren wird zur Ver− und Entschlüsselung der gleiche Schlüssel benutzt. Dies bedeutet jedoch, dass beide Kommunikationspartner den gleichen Schlüssel besitzen müssen. Die Schlüssel müssen also vorher über unsichere Kanäle ausgetauscht werden. Hier entsteht also eine Unsicherheit, die die ganze Verschlüsselung absurd machen würde. Deshalb wird zur Schlüsselübertragung RSA oder DSA eingesetzt.

3. Authentifizierung

Authentifizierung bedeutet die Feststellung der Identität eines Benutzers oder Rechners, d.h. es wird überprüft, ob ein Benutzer oder Rechner wirklich der ist, der er vorgibt zu sein. Es gibt drei Arten der Authentifizierung die von SSH unterstützt werden. Welche benutzt wird kann über eine Option angegeben werden.

3.1 Authentifizierung über Passwort

Der Benutzer gibt bei der Einwahl das Benutzer−Passwort des Systems an. Die Verbindung läuft hier schon verschlüsselt über den Sitzungs−Schlüssel (Session−Key), das Passwort wird also nicht im Klartext übertragen.

3.2 Authentifizierung über die Dateien "/etc/hosts.equiv" und ".rhosts"

Ist der lokale Rechner in der Datei "/etc/hosts.equiv" auf dem entfernten Rechner eingetragen und der Benutzername identisch, oder ist der Benutzer und der lokale Rechner in der Datei ".rhosts" eingetragen, ist eine Einwahl ohne Passwortabfrage auf dem System möglich. Dies ist normalerweise sehr unsicher (IP−Spoofing) und wird nicht empfohlen. Die SSH authentifiziert die Rechner jedoch anhand des RSA−Host−Keys, wodurch eine höhere Sicherheit gewährleistet wird.

3.3 Authentifizierung basierend auf der Public Key (öffentlicher Schlüssel) Kryptographie

Die zwei bekanntesten Verfahren sind RSA (benannt nach Ron Rivest, Adi Shamir, Leonard Adleman) und DSA (Digital Signature Algorithm). Beides sind auf der selben Grundlage basierende asymmetrische Verfahren, d.h. es werden zwei voneinander unabhängige Schlüssel benötigt. Im folgenden wird RSA näher erläutert.

Bei der symmetrischen Verschlüsselung wird ein Passwort zur Ver− und Entschlüsselung verwendet. Dieses muss über unsichere Kanäle transportiert werden. RSA löst dieses Problem durch Verwendung von zwei Schlüsseln. Ein öffentlicher Schlüssel, der von jedem kopiert werden kann und ein geheimer Schlüssel, der unbedingt geschützt werden muss.

- Public Key (öffentlicher Schlüssel): dient zur Verschlüsselung
- Secret Key (Geheimer Schlüssel): dient zur Entschlüsselung

Beide Schlüssel werden zusammen angelegt und sind unteilbar !

Der öffentliche Schlüssel wird auf dem Server abgelegt, normalerweise im Unterverzeichnis ".ssh" des Benutzers. Der private Schlüssel wird entweder nur auf der lokalen Maschine oder auf anderen sicheren Speichermedien abgelegt.
Nach Anlage der Schlüssel kann sich der Benutzer einwählen. Dazu überträgt der Client dem Server den öffentlichen Schlüssel und teilt ihm mit, dass sich jemand anmelden will. Der Server überprüft ob der öffentliche Schlüssel zulässig ist. Wenn dies der Fall ist wird eine 256−Bit Zufallszahl generiert, die mit dem öffentlichen Schlüssel verschlüsselt wird. Dieser Wert wird zum Client geschickt. Der Client entschlüsselt die Zahl mit seinem privaten Schlüssel, berechnet eine 128−Bit MD5 Prüfsumme aus dem Ergebnis und sendet diese Prüfsumme zurück zum Server. MD5 (message digest) ist ein Hash−Algorithmus, der aus einem Text zwei kurze Prüfsummen berechnet. (Es wird nur die Prüfsumme übertragen, um zu verhindern, dass irgend jemand eine Klartextübertragung abhört). Wenn der Server bei der Überprüfung die gleiche Prüfsumme erhält, wird die Anmeldung des Clients akzeptiert. Andernfalls wird eine Einwahl über das Benutzer−Passwort versucht, wobei dieses jedoch nicht mehr im Klartext übertragen wird, da die Verbindung hier schon mit dem Sitzungs−Schlüssel verschlüsselt wird.
Bei dem Verfahren wird davon ausgegangen, dass nur der Client den richtigen Schlüssel kennt. Dies ist bei allen praktischen Anwendungen der Fall.

Der private RSA−Schlüssel sollte durch ein Passwort geschützt werden. Das Passwort kann ein beliebiger Text sein, dieser wird mit MD5 verschlüsselt um einen Schlüssel für IDEA zu produzieren. IDEA wird benutzt, um den geheimen Teil der Schlüsseldatei zu verschlüsseln. Um sich nun anmelden zu können braucht man nicht nur den Zugriff auf die Datei mit dem privaten Schlüssel, sondern auch das Passwort. Ohne Passwortschutz hängt die Anmeldung nur vom Besitz der Datei mit dem privaten Schlüssel ab. Die Anmeldung über das RSA Verfahren ist die sicherste Form, die von der SSH−Software unterstützt wird. Dieses Verfahren ist nicht auf das Netzwerk, auf Router, DNS−Server oder die Client−Maschine angewiesen.

Der RSA−Algorithmus ist seit 1978 bekannt und es gibt bis heute keine bekannten Sicherheitslöcher um das Verfahren zu überlisten. Dies wäre ein ähnliches mathematisches Problem wie die Primfaktorzerlegung von sehr großen Zahlen. Es gibt beispielsweise keine effektive Methoden um etwa eine 512−Bit Zahl in Primfaktoren zu zerlegen. Heutige Rechner würden Jahre dafür brauchen. Dies gilt auch für eine "Attacke" auf einen RSA−Schlüssel. Auch bei in Zukunft leistungsstärkeren Rechnern kann durch die Erhöhung der Schlüssellänge von z.B. 768−Bit auf 1024−Bit vorgebeugt werden. Die SSH erlaubt Schlüssellängen von 768−2048 Bit.

Der Nachteil von asymmetrischen Verfahren wie RSA ist ihre Geschwindigkeit. Sie sind um vieles langsamer als symmetrische Verfahren wie IDEA.

4. Das SSH−Protokoll

Das von SSH verwendete Protokoll ist in die Anwendungsebene einzuordnen. Es baut also auf die darunterliegenden Schichten wie TCP/IP auf und kümmert sich nicht um die eigentliche Datenübertragung. Es ist ein binäres, paketorientiertes Protokoll, das einen "Bytestrom" überträgt. Das SSH−Protokoll ist in drei wichtige Protokolle untergliedert.

4.1 Das Transport−Schicht−Protokoll

Dieses Protokoll baut typischerweise auf einer TCP/IP−Verbindung auf, funktioniert allerdings auch mit einer UDP−Verbindung. Die Transport−Schicht unterstützt die verschlüsselte Anmeldung beim Server über verschiedene Verfahren (RSA , Passwort...). Außerdem ist es möglich komprimierte Daten zu übertragen. Die anderen Protokolle bauen auf dem Transport−Schicht Protokoll auf.

4.2 Das Authentifizierungs−Protokoll

Das Authetifizierungs−Protokoll baut auf dem Transport−Schicht−Protokoll auf. Über dieses Protokoll läuft die Anmeldung eines Benutzers beim Server. Es sind Anmeldungen von mehreren Clients gleichzeitig möglich. Voraussetzung für ein sicheres Arbeiten dieser Schicht ist natürlich, dass eine sichere Verbindung vorhanden ist. Dies ist durch das Transport−Schicht−Protokoll gegeben.

4.3 Das Verbindungs−Protokoll

Mit Hilfe dieses Protokolls werden die einzelnen Verbindungen weitergeschaltet. Portverbindungen werden weitergeleitet ("Tunelling"), Steuersignale können übertragen werden, Umgebungsvariablen werden weitergeleitet, das Umladen des Displays eines Rechners auf einen anderen ist möglich, Programme auf anderen Rechnern können gestartet werden usw. Das Verbindungs−Protokoll baut auf dem Authentifizierungs−Protokoll auf.

Zusammenfassend kann man sagen, dass die einzelnen Protokollschichten auch unabhängig voneinander arbeiten können, aber nur die drei Schichten zusammen ermöglichen eine sichere Verbindung, wie sie SSH anbietet.

5. Verfügbarkeit

Es gibt zwei bedeutende SSH−Implementierungen die angeboten werden:

5.1 SSH 1.x , SSH 2.x (http://www.ssh.org)

SSH 1.x ist für die meisten Betriebssysteme wie UNIX−Derivate und Windows verfügbar. Der Einsatz ist für den kommerziellen und privaten Gebrauch an keine Lizenz gebunden (außer USA). SSH 2.x ist eine Weiterentwicklung von SSH und für den kommerziellen Einsatz lizenzpflichtig. Es ist nicht zur SSH 1.x kompatibel. Das kommerzielle Programm wird von Datafellows [3] angeboten und heißt F−Secure.

5.2 OpenSSH (http://www.openssh.com)

OpenSSH wurde von den Open−BSD Entwickelern 1999 freigegeben und ist kompatibel zur SSH 1.x. Die verwendeten Algorithmen sind frei, wodurch SSH im privaten als auch kommerziellen Umfeld ohne Lizenzierung eingesetzt werden kann.

Für die SSH2 stehen als Windows−Clients z.B. das kommerzielle Programm F−Secure [3] oder das freie Programm Tera Term Pro [4] mit Modul TTSSH [5] zur Verfügung.
Im weiteren wird die SSH 2.x Implementierung betrachtet. Sie ist mit den anderen vergleichbar.

6. Installation und Generierung der Schlüssel

Kurzbeschreibung der Installation unter UNIX / LINUX
(unter Windows 9x Schlüsselgenerierung ähnlich):
Nach dem Entpacken der Programmdateien:
Die Installation wird zunächst auf dem Server durchgeführt.

- Aufruf des Programms "configure"
  Dabei werden die Umgebungsvariablen, für das spätere Kompilieren gesetzt. (Pfad des C−Kompilers usw.) Desweiteren wird festgelegt welche Verschlüsselungsalgorithmen verwendet werden sollen (RSA, IDEA usw.).

- Aufruf von "make"
  Die Programmdateien werden kompiliert.

- Aufruf von "make−install"
  Das Programm wird installiert. Dabei muss beachtet werden, dass nur bei Installation durch den Superuser es allen Benutzern ermöglicht wird SSH zu nutzen.
  Wenn es sich um die Erstinstallation auf dem Rechner handelt, wird der sogenannte Host−Key (privater und öffentlicher) generiert. Dieser Host−Key ist bei jeder Sitzung gleich und identifiziert den Rechner eindeutig. Wenn dieser Schlüssel verloren geht oder geändert werden muss, kann dies mit Hilfe des Kommandos "ssh−keygen2" geschehen. Es ist allerdings zu beachten, dass danach alle Clients den neuen Host−Key (nur den öffentlichen) erhalten müssen, da sie sonst den Server nicht mehr erkennen (der neue öffentliche Host−Key steht noch nicht in der Liste der bekannten Server). Für jeden SSH−Daemon wird ein anderer Host−Key angelegt.

- Konfiguration und Testen
  Der Aufruf von "sshd2" startet den SSH−Daemon. Danach kann durch den Aufruf von "ssh2 localhost" festgestellt werden ob die ausführbaren Dateien richtig arbeiten.
  Um die Kommunikation zu testen, muss man SSH auf einem zweiten Rechner installieren.

- Generierung der Schlüssel
  Der Aufruf von "ssh−keygen2" erzeugt einen geheimen und einen öffentlichen Benutzer−Schlüssel. Durch die Option "−P" können die Schlüssel ohne Passwort erstellt werden, wie z.B. der Host−Key. Die SSH akzeptiert auch Schlüsselpaare die mit PGP erstellt wurden.

- Einbinden der Schlüssel in die SSH
  Der öffentliche Schlüssel muss auf dem entfernten Rechner in das Benutzerverzeichnis kopiert werden, üblicherweise ins Unterverzeichnis ".ssh" . In diesem Verzeichnis muss der Name des Schlüssels in die Datei "authorization" eingetragen werden. In dieser Datei werden alle Schlüssel eingetragen, die zur Einwahl berechtigen.
  Der private Schlüssel verbleibt **nur** auf dem lokalen Rechner. Er liegt üblicherweise im Unterverzeichnis ".ssh" des Benutzerverzeichnisses. Der Name des Schlüssels muss in die hier ebenfalls befindliche Datei "identification" eingetragen werden.
  Die benötigten Schlüsselwörter können aus der Online−Hilfe entnommen werden.

7. Ablauf der Kommunikation

Der Ablauf einer typischen Kommunikation zwischen Client und Server über SSH:

- Verbindungsaufnahme
- Authentifizierung des Servers
- Authentifizierung des Benutzers
- verschlüsselte Datenübertragung
- Verbindungsabbau

Dieser Ablauf ist bei jedem Betriebssystem gleich, es ist also unerheblich, ob es sich beim Client um eine Windows−Maschine, eine Unix−Workstation oder einen Macintosh handelt.

Der Client beginnt mit der Verbindungsaufnahme (standardmäßig auf Portnummer 22).
Der Server startet dann einen neuen SSH−Daemon und generiert ein neues Schlüsselpaar mit standardmäßig 768−Bit, bestehend aus öffentlichem und geheimen Server−Key. Dieser Server−Key wird in bestimmten Zeitabständen geändert (normalerweise nach jeweils einer Stunde) und nur in dieser einen Sitzung verwendet. Er wird nur im Hauptspeicher, nicht in einer Datei, gespeichert. Dadurch sollen Angriffe wie z.B. mit einem "Sniffer" verhindert werden, da auch nachträglich keine Entschlüsselung mehr möglich ist, selbst wenn der geheime Host−Key des Servers gestohlen wurde.

Der Server sendet seine beiden öffentlichen Schlüssel (Host−Key und Server−Key) an den Client. Der Client überprüft zunächst den Host−Key mit den Einträgen in seiner Liste der bekannten Server. Ist dies die erste Sitzung mit diesem Server, so kann der Client den neuen Schlüssel in seine Liste mit bekannten Host−Keys hinzufügen. Der Benutzer sollte jeden Neueintrag in die Liste des Clients bestätigen. Es wäre nämlich möglich, dass eine sogenannte "Man in the middle attack" vorliegt, wobei der neue Host−Key dann nicht zu dem Server gehören würde mit dem man eine Verbindung aufbauen will. Mit einer solchen "Attacke" könnte ein anderer Rechner in den Besitz vertraulicher Daten kommen.

Um dies zu verhindern sollte man dem Client den Host−Key des Servers auf anderem Wege zukommen zu lassen (zum Beispiel per Diskette oder per Email mit PGP−Signatur) und ihn von Hand in die Liste der bekannten Server eintragen.

Hat der Client den öffentlichen Host−Key des Servers angenommen, wird ein zufälliger Sitzungsschlüssel erzeugt. Diesen kodiert der Client mit Hilfe des öffentlichen Host−Keys und des öffentlichen Server−Keys. Das Ergebnis wird zurück an den Server geschickt. Dabei wird auch festgelegt welche Verschlüsselungsverfahren für die weitere Kommunikation benutzt werden. Den übertragenen Sitzungsschlüssel kann der Server nur mit Hilfe des geheimen Host−Keys und des geheimen Server−Keys entschlüsseln. Alle übertragenen Datenpakete werden ab diesem Zeitpunkt mit dem Sitzungsschlüssel und dem gewählten Verfahren verschlüsselt.

Anschließend wird der Benutzer durch eines der in 3. erläuterten Verfahren authentifiziert. Wurde die Authentifizierung erfolgreich abgeschlossen wird eine Shell zur Verfügung gestellt oder ein Kommando ausgeführt.

Durch den Verbindungsabbau (z.B. bei Eingabe von "exit") wird die Verbindung geschlossen, wenn keine Übertragungen mehr stattfinden.

8. Benutzung der SSH2

Die SSH2 stellt eine Reihe von Befehlen zur Verfügung die die "r−Utilities" ersetzen. Die möglichen Optionen der SSH2 gleichen denen der "r−Utilities", wodurch eine transparente Umstellung möglich ist.

8.1 SCP2 − Secure Copy Client

Mit dem Secure Copy Client können Dateien zwischen zwei Rechnern über das Netzwerk verschlüsselt ausgetauscht werden. Einer der Rechner muss der lokale Rechner sein.
Aufruf: scp2 [Optionen] [Benutzer@]Rechner[#Port]:]Datei
[Benutzer@]Rechner[#Port]:]Datei oder Verzeichnis

8.2 SSH2 − Secure Shell Client

Der Secure Shell Client wird zur Einwahl auf einen entfernten Rechner benutzt. Auf diesem können dann Kommandos ausgeführt werden.
Aufruf: ssh2 [Optionen] entfernterRechner [auszuführenerBefehl]

Wichtige Optionen von SSH2:

−l user	Diesen Benutzernamen bei der Einwahl verwenden.
−f	Nach der Einwahl als Hintergrundprozess starten.
−p port	Portnummer des SSHD2−Daemon auf dem entfernten Rechner.
−L listen−port:host:port	Portumleitung vom lokalen Rechner auf entfernten Rechner. listen−port: Port des lokalen Rechners der umgeleitet werden soll. (als User sind nur Portnummern ab 1024 verfügbar !) host: Der entfernte Rechner. port: Port des entfernten Rechners auf den umgeleitet werden soll.
−R listen−port:host:port	Portumleitung von entferntem Rechner auf lokalen Rechner. listen−port: Port des entfernten Rechners der umgeleitet werden soll. (als User sind nur Portnummern ab 1024 verfügbar !) host: Der entfernte Rechner. port: Port des lokalen Rechners auf den umgeleitet werden soll.
+C	Die Komprimierung einschalten (bei anderen Versionen "−C").
−h	Ein Hilfsseite mit allen Optionen anzeigen.

8.3 SSHD2 − Secure Shell Daemon

Der Secure Shell Daemon ermöglicht die Einwahl eines Benutzers eines anderen Rechners über eine verschlüsselte Verbindung.
Aufruf: sshd2 [Optionen]

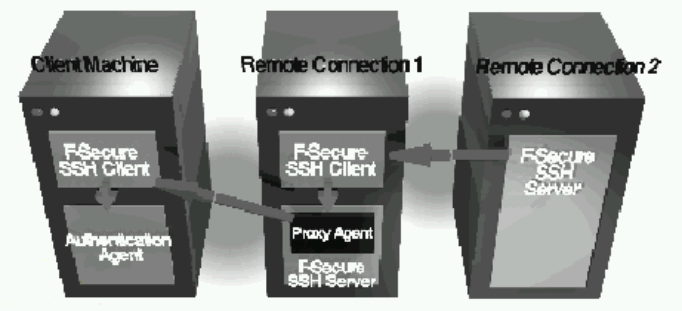
Einige Optionen von SSHD2:

−p port	Portnummer auf der der SSHD2−Daemon Anfragen entgegennimmt.
−i	Der SSHD2−Daemon wird von inetd gestartet.
−f	Ort der Konfigurationsdatei.

In der Konfigurationsdatei kann z.B. der gewünschte Verschlüsselungsalgorithmus, die Authentifizierungsart, die Anzahl der Fehlversuche bei falschen Passwörtern oder die Portnummer angegeben werden. Sie befindet sich standardmäßig im Verzeichnis "/etc/ssh2/sshd2_config".

8.4 SSH−AGENT2 − Authentication Agent

Der Authentication Agent verwaltet die Benutzerschlüssel und nimmt Authentifizierungsanfragen entgegen. Er bietet damit eine sichere Automatisierung des Einwahlvorgangs an.
Der geheime Schlüssel sollte immer durch ein Passwort geschützt sein, falls er einmal in die falschen Hände fallen sollte. Durch den Passwortschutz entsteht aber ein Komfortverlust und ein Problem. Man muss das Passwort bei jedem Aufbau einer SSH−Verbindung eingeben. Automatische Skripts die z.B. bisher über rsh − mit Hilfe der Einträge in "/etc/hots.equiv" oder ".rhosts" − laufen, müssten bei jeder Einwahl das Schlüssel−Passwort abfragen. Wird dem Authentication Agent ein Schlüssel übergeben (mit dem Befehl "ssh−add2"), wird das Schlüssel−Passwort abgefragt. Von nun an nimmt dieser alle Authentifizierungsanfragen entgegen und beantwortet sie. Der Authentification Agent läuft nur auf dem lokalen Rechner, wodurch der geheime Schlüssel auch nur auf diesem verbleibt. Auf weitere Anfragen wird nur von der lokalen Maschine geantwortet, wenn z.B. mehrere fortlaufende SSH−Verbindungen aufgebaut werden. Der oder die Schlüssel verlassen niemals den lokalen Rechner ! Folgende Abbildung 2 veranschaulicht dies (entnommen aus F−Secure SSH, S. 9) :



(Abbildung 2: Authentification Agent)

Eine Gefahr besteht jedoch in der Benutzung des Authentication Agent. Tritt ein Programmfehler oder ein Fehler im Betriebssystem auf, könnte der Inhalt des Speichers auf die Festplatte geschrieben werden. In diesem Speicherauszug ist es möglich das verwendete Passwort zu finden, vorausgesetzt jemand hat die Möglichkeit darauf zuzugreifen.

8.5 SSH−ADD2 − Schlüsselaufnahme für Authentification Agent

Mit "ssh−add2" kann dem Authentification Agent ein Schlüssel hinzugefügt werden. Möglich ist dies nur, wenn "ssh−add2" als Kindesprozeß von "ssh−agent2" gestartet wird. Standardmäßig werden die Schlüssel aus der im Unterverzeichnis ".ssh2" des Benutzers befindlichen Datei "identity" dem Authentification Agent hinzugefügt.
Aufruf: ssh−add2 [Optionen]

8.6 SFTP − Secure FTP Client

Der Secure FTP Client bietet den gleichen Funktionsumfang wie ein gewöhnlicher FTP Client, jedoch werden die Daten verschlüsselt übertragen.
Aufruf: sftp [Optionen]

9. Praktische Beispiele

Die praktischen Beispiele finden in Bezug auf den Rechner "stl−d−04.htw−saarland.de" statt, im weiteren als "stl−d−04" bezeichnet, der sich im Systemtechniklabor der HTWdS befindet. Auf diesem Server läuft ein SSH2−Daemon auf Port 22.

9.1 Kopieren einer Datei

Kopieren der auf dem Rechner stl−d−04 befindlichen Datei "Vorlesung.doc" des Benutzers "olitz" in das aktuelle Verzeichnis des lokalen Rechners.
Befehl: scp2 olitz@stl−d−04:Vorlesung.doc .

9.2 Einwahl über SSH2

Einwahl von einem Client über SSH2 auf den Rechner stl−d−04 mit dem Benutzernamen olitz.
Befehl: ssh2 –l olitz stl−d−04

Das Display des entfernten Rechners wird schon jetzt automatisch auf die lokale Maschine umgeleitet. Programme können auf dem entfernten Rechner gestartet werden und die grafische Ausgabe erfolgt auf dem lokalen Rechner.
Folgender Befehl startet den Browser Netscape über die bestehende SSH−Verbindung, wobei die Ausgabe auf den lokalen Rechner umgeleitet wird:
Befehl: netscape &
Oder direkt: ssh2 –l olitz stl−d−04 netscape

Dies ist natürlich auch ohne SSH durch Setzen der DISPLAY−Variable und mit Freigabe über xhost möglich. Jedoch läuft die Verbindung über SSH verschlüsselt ab, die Daten können nicht unterwegs abgefangen werden.
Auch auf einem lokalen Windows−Rechner ist es über einen Windows−SSH Client möglich ein Programm mit grafischer Ausgabe auf dem entfernten Rechner zu starten und auf dem lokalen Windows−Rechner zu bedienen.

9.3 Einwahl mit angegebener Portnummer

Einwahl wie 9.2, jetzt jedoch auf einen SSH2−Daemon, der auf Port 1234 Anfragen entgegennimmt, bei gleichzeitig eingeschalteter Kompression.
Befehl: ssh2 −l olitz −p 1234 +C stl−d−04.htw−saarland.de

9.4 Einwahl mit Umleitung des FTP−Protokolls

Einwahl wie 9.2, jetzt jedoch mit Umleitung von Port 21 (FTP) des Rechners stl−d−04 auf Port 2100 des lokalen Rechners. Ist der Einwahlvorgang abgeschlossen, wird die Shell als Hintergrundprozess gestartet und der Befehl "sleep 10" (warte 10 Sekunden) ausgeführt.

Befehl: ssh2 −f −L 21:stl−d−04.htw−saarland.de:2100 stl−d−04.htw−saarland.de sleep 10

Mit folgendem Befehl kann nun eine FTP−Verbindung zu Rechner stl−d−04 aufgebaut und überprüft werden.
Befehl: telnet localhost 2100

9.5 Einwahl mit Umleitung des POP3−Protokolls

Einwahl wie 9.2, jetzt jedoch mit Umleitung von Port 25 (POP3) des POP3−Mailservers "mail−stl.htw−saarland.de" auf Port 2500 des lokalen Rechners.
Da auf dem Mailserver kein SSH−Daemon läuft kann nur eine Verbindung über stl−d−04 aufgebaut werden. Die Kommunikation von stl−d−04 mit dem Mailserver läuft **unverschlüsselt** ab. Erst zwischen stl−d−04 und dem lokalen Rechner ist die Verbindung wieder verschlüsselt.

Befehl: ssh2 −f −L 25:mail−stl.htw−saarland.de:2500 stl−d−04.htw−saarland.de sleep 10

Mit folgendem Befehl kann nun eine POP3−Verbindung zum Mailserver aufgebaut und überprüft werden:
Befehl: telnet localhost 2500

Unter UNIX oder Windows ist es nun auch möglich, Emails mit einem Email−Client wie z.B. Netscape unter folgenden Angaben abzufragen:
POP3−Server: localhost Port: 2500

9.6 Einwahl mit Umleitung des HTTP−Protokolls

Einwahl wie 9.2, jetzt jedoch mit Umleitung von Port 80 (HTTP) des Webservers "www1.htw−saarland.de" im Systemtechniklabor auf Port 8000 des lokalen Rechners.

Befehl: ssh2 −f −L 80:www1.htw−saarland.de:8000 stl−d−04.htw−saarland.de sleep 60

Auf dem lokalen Rechner kann nun unter einem Browser als URL eingegeben werden:
http://localhost:8000

Dadurch wird die Startseite des Systemtechniklabors auf dem lokalen Rechner geladen, auf den ein Zugriff nur innerhalb der Domäne "htw−saarland.de" möglich ist. Normalerweise kann von außerhalb auf diesen Server nicht zugegriffen werden. Dies ist hier jedoch jetzt möglich, da die Anfrage von "localhost" Port 8000 über die SSH−Verbindung zum Rechner stl−d−04 übertragen wird. Dieser stellt dann die Anfrage an Webserver und ist auch berechtigt, auf diesen zuzugreifen.
Ein Problem entsteht hier noch durch die auf dem Server befindlichen Links (Verweise), die alle als absolute Adressen vorliegen. Eine Anwahl eines Links bewirkt, dass der Browser versucht eine Verbindung zu "http://www1.htw−saarland.de" aufzubauen. Dies kann aber nicht gelingen, da jetzt wiederum versucht wird von außerhalb der HTW−Domäne auf diesen Rechner zuzugreifen.

9.7 Einwahl mit Umleitung des Proxys

Einwahl wie 9.2, jetzt jedoch mit Umleitung von Port 3128 (Proxy) des Proxy−Servers "www−proxy.htw−saarland.de" auf Port 3128 des lokalen Rechners.

Befehl: ssh2 −f −L 3128:www−proxy.htw−saarland.de:3128 stl−d−04.htw−saarland.de sleep 60

Auf dem lokalen Rechner muss die Proxy−Einstellung des Browsers für das HTTP−Protokoll auf folgende Einstellung geändert werden:
Proxy für HTTP: localhost Port: 3128

Dies löst das Problem der absolute Links aus 9.6. Nun kann ganz transparent auf den Systemtechniklaborserver "www1.htw−saarland.de" zugegriffen werden. Sämtliche HTTP−Verbindungen laufen nun über den HTW−Proxy.

10. Wovor schützt die SSH nicht ?

SSH sichert den Datenverkehr und die Authentizität des Rechners und Benutzers. Geht jedoch der Benutzer nachlässig mit sensitiven Informationen wie Schlüsseln oder Passwörtern um, wird die SSH nutzlos. SSH bietet einen guten Schutz auch bei Verlust des Schlüssel−Passwortes oder des geheimen Schlüssels. Fällt einem Hacker eines von beiden in die Hände benötigt er immer noch den anderen Teil. Allein mit dem Schlüssel−Passwort oder dem geheimen Schlüssel kann er nichts anfangen.

Wird jedoch **kein** Schlüssel−Passwort vergeben, besteht die Gefahr, dass der geheime Schlüssel einem Hacker, der das System unterlaufen hat, in die Hände fällt. Dieser kann den geheimen Schlüssel benutzen und sich ohne Schlüssel−Passwort zur fremden Benutzerkennung einwählen. Es sollte also immer ein Schlüssel−Passwort verwendet werden.
Diese Gefahr besteht auch, wenn der geheime Schlüssel und das Schlüssel−Passwort sich auf einem System befinden zu dem mehrere Benutzer uneingeschränkten Zugriff haben (z.B. Windows 9x).
Bewahrt der Benutzer das Passwort der Zugangskennung (nicht das Schlüssel−Passwort) an unsicheren Stellen auf (wie die Brieftasche), kann die Kennung leicht unterlaufen werden ohne dass die SSH überhaupt zum Zuge kommt. Das Passwort sollte ein "starkes" Passwort sein, also nicht einfach aus Name oder Autonummer bestehen.
Die SSH schützt also einen Benutzer nicht vor sich selbst !

Weitere Angriffspunkte sind bedingt durch die technischen Eigenschaften von Computern. Die Bildschirmabstrahlung kann mit einer in der Nähe befindlichen Antenne aufgefangen und so das Bild reproduziert werden. Sämtliche Aktionen des Benutzers können damit überwacht werden. So können sensible Daten nach außen gelangen. Eine Passworteingabe ist hier meist nicht mitzulesen, da diese nicht auf dem Bildschirm angezeigt wird. Aber ein sog. "trojanisches Pferd", d.h. ein Programm welches unbemerkt auf einem Rechner im Hintergrund läuft, kann sämtliche Eingaben des Benutzers protokollieren und später zur Verfügung stellen.

Ein weiterer Angriffpunkt besteht darin, dass an den Endpunkten der Kommunikation der gesamte Datenverkehr verschlüsselt bzw. entschlüsselt wird. Er liegt hier also immer entschlüsselt vor und kann von jemanden der die entsprechenden Rechte besitzt protokolliert werden. Dies wäre z.B. einem "korrupten" Superuser möglich, der sich dadurch leicht Zugang zu anderen Systemen verschaffen kann.

11. Fazit

Die SSH bietet einen sehr guten Schutz bei Übertragung wichtiger Daten. Dies wird durch eine "starke" Verschlüsselung und die Authentizität des Rechners und Benutzers sichergestellt. Zusätzliche Leistungsmerkmale wie "TCP−IP Tunneling" machen die SSH zu einem mächtigen Werkzeug. Voraussetzung ist aber der sorgfältige Umgang des Benutzers mit Passwörtern und Schlüsseln.

Literaturverzeichnis

1. Anne Carasik, Steve Acheson
   SSH−FAQ (Häufig gestellte Fragen), Revision 1.1, September 1999
   http://www.employees.org/~satch/ssh/faq

2. Data Fellows Ltd.
   F−Secure SSH (SSH 2.0 for Windows NT and Windows 95), Gummerus Printing, Jyväskylä 1998
   http://www.datafellows.com

3. Data Fellows Ltd.
   F−Secure, kommerzieller F−Secure Client und Server
   http://www.datafellows.com/f−secure

4. Takashi Teranishi
   Tera Term Pro, Windows Terminal Emulationsprogramm
   http://hp.vector.co.jp/authors/VA002416/teraterm.html

5. Robert O'Callahan
   TTSSH (Tera Term SSH), SSH−Erweiterungsmodul für Tera Term Pro
   http://www.zip.com.au/~roca/ttssh.html