Sicherer Mailabruf mit SSH (Secure Shell)

Name des Ausarbeitenden: Oliver Litz
Matrikelnummer: xxxxxxx

Name des Ausarbeitenden: Andreas Engel
Matrikelnummer: xxxxxxx

Fachbereich: GIS
Studiengang: Praktische Informatik
Fach: EDV-Fallstudie
Betreuer: Wolfgang Pauly, Prof. Dr. Reiner Güttler

Datum: 28.09.2000

Fallstudie im Sommersemester 2000
an der HTW des Saarlandes

Inhaltsverzeichnis

1. Einleitung

Ein Großteil des Datenverkehrs im Internet beruht auf dem Protokoll TCP/IP. Die wesentlichen Ziele in der Entwicklung dieses Protokolls waren eine permanente, transparente und verlässliche Verbindung herzustellen. Die Echtheit einer Verbindung ist nur durch die IP-Adresse gegeben, die einen Rechner eindeutig identifiziert. Aufgrund der damaligen geringen Größe des Internet bzw. ARPA-Netzes wurden keine sonstigen Sicherheitsmaßnahmen zur Identifizierung getroffen. Die weltweit eingesetzte Version IPv4 bietet noch heute keine Verschlüsselung oder "starke" Authentifizierung. Durch die netzartige Struktur des Internet ist es möglich, besonders an Knotenpunkten Übertragungen wie Passwörter oder Email abzuhören oder eine falsche Identität vorzutäuschen. Um diesem Missbrauch vorzubeugen, bietet die Secure Shell (SSH) eine "starke" Verschlüsselung und Authentifizierung. Durch TCP/IP-Tunneling ist es möglich, andere Dienste abzusichern. Diese Leistungsmerkmale der SSH möchten wir in der Fallstudie vorstellen und zeigen, wie man damit ohne Neuinstallation oder Aktualisierung an der im Betrieb befindlichen Software einen sicheren Mailabruf gewährleisten kann.

2. SSH im Überblick

"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].

Unter SSH wird sowohl ein kryptographisches Protokoll als auch eine konkrete Implementierung dieses Protokolls verstanden. Ursprünglicher SSH-Protokoll-Designer und Software-Autor ist Tatu Ylönen aus Finnland, der die Secure Shell an der TU Helsinki entwickelte und später die Firma SSH Communications Security Ltd. gründete.

Zum Funktionsumfang von Ylönens Secure Shell gehören:

- Login auf einer entfernten Maschine
- interaktive oder nichtinteraktive Ausführung von Kommandos auf der entfernten Maschine
- Kopieren von Dateien zwischen verschiedenen Rechnern eines Netzes.

SSH ermöglicht eine kryptographisch gesicherte Kommunikation über unsichere Netze und bietet ein hohes Sicherheitsniveau: zuverlässige gegenseitige Authentifizierung der Partner sowie Integrität und Vertraulichkeit der ausgetauschten Daten.

SSH ist als kompletter Ersatz der r-Utilities "rlogin", "rsh" und "rcp" gedacht. SSH bietet darüber hinaus aber noch weitere interessante Eigenschaften.

2.1 Interessante Eigenschaften

Die folgende Liste nennt einige interessante Eigenschaften der SSH:

- Durch die Verwendung einer auf dem asymmetrischen Kryptosystem RSA basierenden Authentifizierung des Servers werden mehrere Sicherheitslücken (z.B. IP- und DNS-Spoofing, IP Source Routing) geschlossen.
- Der Client ist sowohl durch systemweite als auch nutzerbezogene Konfigurationsdateien steuerbar. Damit kann der Administrator sinnvolle Voreinstellungen vornehmen, die dem gewöhnlichen Anwender die Arbeit stark erleichtern.
- Die Übertragung beliebiger Binärdaten zwischen den Maschinen ist möglich. Eine Kompression der Daten wird optional unterstützt.
- Es stehen verschiedene Verfahren zur Authentifizierung des Client gegenüber dem Server zur Verfügung.
- Im Standardfall erfolgt eine automatische und transparente Verschlüsselung der gesamten Kommunikation mittels eines der bei der Installation der Software ausgewählten symmetrischen Verfahren (IDEA, DES, Blowfish, usw.).
- SSH unterstützt den Schutz von X11-Verbindungen und beliebigen TCP-Verbindungen durch deren Weiterleitung über einen kryptographisch gesicherten Kanal.

- Jeder SSH-Server-Prozeß (Dämon) arbeitet mit zwei RSA-Schlüsselpaaren:
    1. Ein langlebiges Paar aus öffentlichem und privatem Host-Key dient der Identifikation der Maschine und ist für alle auf dieser Maschine laufenden Dämonen identisch.
    2. Beim Start kreiert jeder Server-Prozeß ein Server-Key-Paar, das er in bestimmten Intervallen (in der Regel nach jeweils einer Stunde) verändert, sofern es benutzt wurde. Dieses Paar wird niemals in einer Datei abgelegt. Die dynamischen Schlüssel sollen verhindern, dass ein Angreifer aufgezeichnete Sitzungen entschlüsseln kann, falls es ihm später gelingen sollte, in den Server einzudringen und dessen langlebige Schlüssel zu stehlen.
- Jeder Nutzer kann eine beliebige Anzahl von RSA-Schlüsseln zu seiner eigenen Identifikation verwalten und verwenden. Zu deren bequemer Handhabung steht der SSH-Agent zur Verfügung.

3. Sicherheit

Die Sicherheit einer Anwendung beruht auf den eingesetzten Algorithmen. Die grundlegenden Funktionen von SSH sind Verschlüsselung und Authentifizierung. Diese müssen deshalb in bezug auf ihre Sicherheit besonders kritisch analysiert werden.

3.1 Verschlüsselung

SSH setzt zur Verschlüsselung symmetrische Verfahren ein, wie z.B. IDEA oder 3DES. Diese Verfahren sind offengelegt und können durch jeden einer kryptoanalytischen Überprüfung unterzogen werden. Da bisher selbst Experten keine Sicherheitslücken entdecken konnten, sind diese Verfahren als sicher anzusehen. Außerdem halten die Verfahren Attacken, wie z.B. einer Brute-Force-Attack (Ausprobieren aller Möglichkeiten) stand. Symmetrische Verfahren sind im Vergleich zu asymmetrischen schnell, haben jedoch den Nachteil, dass zur Ver- und Entschlüsselung der gleiche geheime Schlüssel benutzt werden muss. Der Schlüssel muss also über unsichere Kanäle übermittelt werden.

3.2 Authentifizierung

Zur Authentifizierung, d.h. dem Feststellen der Identität eines Benutzers oder Rechners, unterstützt SSH ab Version 2.3.0 vier Möglichkeiten:

- Authentifizierung über Passwort
- Authentifizierung basierend auf Hostname
- Authentifizierung basierend auf Kerberos (experimentell)
- Authentifizierung basierend auf der Public-Key-Kryptographie

Die ersten beiden Methoden sind nicht gegen Angriffe sicher, die Authentifikation mittels Kerberos befindet sich noch im experimentellen Stadium, deshalb wird die Verwendung der Public-Key-Kryptographie empfohlen. Hierzu werden asymmetrische Verfahren eingesetzt, wie z.B. RSA oder DSA. Diese dienen zum einen zur Authentifizierung und zum andern zur Übertragung des Schlüssels für die symmetrischen Verfahren. Ein großer Nachteil der asymmetrischen Verfahren ist, dass sie im Vergleich zu symmetrischen sehr langsam sind. Sie bieten jedoch den entscheidenden Vorteil, dass kein geheimer Schlüssel mehr übermittelt werden muss. Vielmehr werden zwei verschiedene Schlüssel erzeugt:

- Public Key (öffentlicher Schlüssel)
- Secret Key (geheimer oder privater Schlüssel)

Der geheime Schlüssel muss beim Eigentümer (hier auf dem lokalen Rechner) verbleiben. Der öffentliche Schlüssel kann an jeden Kommunikationspartner (hier der entfernte Rechner) weitergegeben werden. Beide Schlüssel sind Gegenstücke, d.h. mit einem kann verschlüsselt, mit dem anderen entschlüsselt werden. Durch die Verwendung der Schlüssel kann eine Kommunikation ohne Passwort stattfinden, da der Benutzer sich mit seinem geheimen Schlüssel authentifiziert. Trotzdem sollte der Schlüssel mit einem Passwort gesichert werden, falls er durch unzureichende Sicherheitsmassnahmen in falsche Hände gelangen sollte. Es wird jedoch nie ein Passwort übertragen, sondern eine MD5-Prüfsumme. Diese wird mittels Public-Key-Kryptographie übermittelt. Die Public-Key-Kryptographie ist sicher, da sie einer Kryptoanalyse standhält. Ein sorgfältiger Umgang mit dem geheimen Schlüssel ist jedoch notwendig!

4. Verfügbarkeit

Das SSH-Protokoll wird durch die Communications Security [2] weiterentwickelt. Es existieren momentan zwei verschiedene Versionen von SSH. Die Entwicklung der älteren Version 1.x wurde eingestellt, aktuell ist die Version 2.x. Da beide Versionen ein unterschiedliches Protokoll verwenden, sind sie nicht zueinander kompatibel. Durch Festlegung eines herstellerunabhängigen SSH-Standards und Protokolls (Transport-Schicht-, Authentifizierungs- und Verbindungs-Protokoll) durch die IETF [5] ist es auch vielen Programmieren möglich, eigene, kompatible SSH-Software zu erstellen. Es existiert daher eine Vielzahl von SSH-Implementierungen, von denen wir die zwei wesentlichen vorstellen möchten. Diese unterscheiden sich nicht nur in der Implementierung, sondern auch in der Lizenzpolitik.

4.1 SSH-Implementierungen

- SSH der SSH Communication Security [2]
  Der offizielle Vertrieb der SSH geschieht durch die SSH Communication Security. Die SSH-Implementierungen 1.x und 2.x werden hier angeboten. Die Benutzung der Software ist ab Version 2.3.x für Benutzer eines freien Betriebssystems (genannt werden z.B. Linux oder FreeBSD) sowohl für private als auch für kommerzielle Zwecke kostenlos. Auch für Hochschulen (Professoren, Studenten oder Mitarbeiter) ist die Benutzung aller Versionen unentgeltlich. Für andere Betriebssysteme ist nur bei gewerblicher Nutzung eine kostenpflichtige Lizenz notwendig. Beide Versionen sind für gängige Betriebssysteme wie Windows oder Unix-Derivate verfügbar.

- OpenSSH der OpenBSD-Entwicker [3]
  OpenSSH wurde von den OpenBSD Entwicklern 1999 freigegeben und ist kompatibel zu den SSH-Versionen der SSH Communication Security. Die verwendeten Algorithmen unterliegen keiner restriktiven Lizenz, wodurch OpenSSH im privaten als auch kommerziellen Umfeld kostenlos eingesetzt werden kann. OpenSSH baut auf anderen OpenBSD-Projekten auf, wie z.B. OpenSSL. Server und Client sind für Unix-Derivate verfügbar, wie z.B. Linux, Solaris oder FreeBSD.

Weitere freie SSH-Versionen für verschiedene Betriebssysteme sind auf der Internetseite von FreeSSH [4] zu finden.

4.2 Unterschiede

- Version 1.x und 2.x
  Die Protokollversionen 1.x und 2.x sind zueinander inkompatibel. Version 2.x stellt eine komplette Neuentwicklung des SSH-Protokolls dar. Es behebt viele Schwachstellen der Version 1.x. Gleichzeitig werden auch andere Verschlüsselungsverfahren benutzt, wie aus Abbildung 1 ersichtlich ist:

Algorithmus	***Version 1.x***	***Version 2.x***
RSA	Ja	Nein
DSA	Nein	Ja
DES	Ja	Nein
3DES	Ja	Ja
IDEA	Ja	Nein (kommerzielle: Ja)
Blowfish	Ja	Ja
Twofish	Nein	Ja
Arcfour	Nein	Ja
Cast128-cbc	Nein	Ja

Abbildung 1: Verschlüsselungsverfahren der SSH-Versionen 1.x und 2.x

- SSH und OpenSSH
  Da sowohl SSH als auch OpenSSH auf einem standardisierten Protokoll aufbauen, sind beide Versionen miteinander kompatibel. Bei wechselseitiger Nutzung von Client und Server müssen die angelegten Schlüssel konvertiert werden. Unterschiede zwischen SSH und OpenSSH veranschaulicht Abbildung 2:

SSH	***OpenSSH***
Kommerzielles Projekt	Open Source Projekt
Restriktive Lizenzpolitik[1]	Freie Lizenzpolitik
Protokollunterstützung für 1.x nur durch Installation von SSH 1.x	Ein Programm unterstützt die Protokolle 1.3, 1.5 und 2.0
Einfache Installation für Benutzer durch Skriptunterstützung	Schwierigere Installation für Benutzer, keine vorhandenen Skripte
Einsatz patentierter Algorithmen (in kommerzieller Version)	Einsatz freier Algorithmen
Support und Haftung für kommerzielles Produkt	Eingeschränkter Support (Newsgroups, Mailing-List)
OpenPGP-Unterstützung (GnuPgp)	-
Keberos-Authentifizierung (experimentell)	-
FTP-Server, restriktive Shell	-

Abbildung 2: Unterschiede zwischen SSH und OpenSSH

[1] Siehe Kapitel 4.1 (SSH)

4.3 Einsatz in der Fallstudie

Wir werden im folgenden den Einsatz der SSH-Version 2.x der Communication Security beschreiben. Obwohl diese Version einer restriktiven Lizenzpolitik unterliegt, haben wir uns aus folgenden Gründen für SSH und gegen das Open Source Projekt entschieden:

- Einfache Installation (auch durch den Endbenutzer)
- Einfache Benutzung durch Unterstützung von FTP und PGP.
- Kostenloser Einsatz auf freien Betriebssystemen
- Kostenloser Einsatz an Universitäten sowie für Privatanwender
- Erfolgreicher Einsatz von SSH schon seit Jahren, OpenSSH erst seit 1999
- Windows-Client mit komfortabler Oberfläche direkt vom Hersteller
- Support und Haftung für kommerzielles Produkt (für Firmen wichtig), Support für Universitäten siehe SANS-Projekt [6]

Die nachfolgenden Erläuterungen und Beispiele lassen sich jedoch ohne aufwändige Umstellungen auch auf OpenSSH anwenden, da beide Implementierungen die gleichen Befehle mit annähernd übereinstimmender Syntax verwenden. Wenn im folgenden von SSH gesprochen wird, ist immer die SSH-Version 2 gemeint.

5. Installation

SSH-Server und Client werden von der SSH Communication Security unter [2] zur Verfügung gestellt. Der SSH-Server wird von einer Vielzahl von Unix-Betriebssystemen wie Linux, FreeBSD, Solaris oder IRIX unterstützt. Der SSH-Client ist für Windows, Macintosh, OS/2, Unix und Java verfügbar.

5.1 Unix

Bei der Installation ist zu beachten, dass alle Benutzer den SSH-Serverdienst erst dann erfolgreich nutzen können, wenn die Installation durch den Administrator (root) erfolgt. Andernfalls kann sich nur der Benutzer einwählen, der auch den Serverdienst gestartet hat. Dieser muss dann einen anderen Port (>1024) als den Standardport (22) benutzen.

- Entpacken der Programmdateien mit:
  gunzip -c ssh-2.x.x.tar.gz | tar -xpvf -
- Lesen der Dateien "LICENSE" und "README" !
- Vornehmen von Konfigurationseinstellungen wie Installationspfad oder Verschlüsselungsalgorithmen (siehe ./configure --help) durch Aufruf von "./configure"
- Übersetzen des Quellprogramms mit "make".
- Durchführen der Installation mit "make install".

Bei der Installation wird ein Host-Key erzeugt. Dieser Host-Key ist bei jeder Sitzung gleich und identifiziert den Rechner eindeutig. Für jeden SSH-Dämon wird ein anderer Host-Key angelegt. Anhand des Host-Keys kann der Client die ordnungsgemäße Identität des Servers feststellen. Hiermit kann z.B. DNS-Spoofing erfolgreich verhindert werden. Der Host-Key befindet sich normalerweise im Verzeichnis "/etc/ssh2".

Nach der Installation stehen Client im Verzeichnis "$Installationspfad/bin" und Server im Verzeichnis "$Installationspfad/sbin" zur Verfügung.

5.1.1 Start des Client

Der Installationspfad der SSH sollte in den Standardpfad aufgenommen werden. Im Systemtechniklabor ist dies durch Anpassung der Datei "/export/home_ch/bin/profile.student.basis.97" durch Einfügen folgender Kommandos möglich:

```
PATH=$PATH:/Installationspfad-SSH/bin
MANPATH=$MANPATH:/Installationspfad-SSH/man
...
export PATH
export MANPATH
```

Abbildung 3: Anpassung der Pfadvariablen

In der Datei "$HOME/.ssh2/ssh2_config" können Standardeinstellungen für den Client vorgenommen werden. Dies sind z.B. der benutzte Verschlüsselungsalgorithmus, die benutzte Authentifikationsmethode, die Portnummer oder das X11-Forwarding. Siehe dazu auch die installierten Man-Pages unter "$Installationspfad/man". Zur Verwendung des SSH-Client müssen zuerst noch öffentlicher und geheimer Schlüssel erzeugt werden, wie in Kapitel 6 erläutert.

5.1.2 Start des Server

Zum sicheren Mailabruf ist es notwendig den SSH-Server auf dem gleichen Rechner zu installieren, auf dem sich auch der POP3-Server befindet. An der HTW ist dies der Rechner "mail-stl.htw-saarland.de". Der Server sollte beim Start des Betriebssystems aktiviert werden. Dies sollte über ein Startup-Skript erfolgen. Die Einbindung in den Internet-Dämon (inetd) wird nicht empfohlen, da das Hochfahren des Systems erheblich verzögert werden kann. Vorzugsweise ist das mitgelieferte Startskript "sshd2.startup" zu benutzen. Dieses wird im Verzeichnis "/etc/init.d/" abgelegt. Eventuell sind noch Pfadangaben an den Installationsort des Servers anzupassen. Im Verzeichnis "/etc/rc?.d/" befinden sich bei System V basierenden Betriebssystemen die Startskripte für die einzelnen Runlevels. Hierin ist das SSH-Startskript aufzunehmen. Von Hand kann der Server jetzt z.B. mit "/etc/init.d/sshd2.startup start" gestartet werden. Der Server steht nun für Anfragen auf Port 22 zur Verfügung. Zur Konfiguration des Servers wird die Datei "/etc/ssh2/sshd2_config" verwendet. Siehe dazu auch Anhang C.

5.2 Windows

Es gibt eine Reihe von Portierungen von SSH für die PC/Windows-Welt, leider unterstützen die wenigsten die Protokollversion 2. Der Client SSHWin-2.3.0 der SSH Communication Security ist einfach zu installieren und zu benutzen und unterstützt die Protokollversion 1 und 2. Die Lizenzbestimmungen aus Kapitel 4.1 sind zu beachten. Ein Download ist unter [2] möglich.

Nach Installation des Client "SSHWin-2.3.0" sind folgende Einstellungen im Menü "Connection" vorzunehmen, wie aus Abbildung 4 ersichtlich. Dieses Menü erreicht man durch Auswahl von "Edit" Þ "Settings" Þ "Host Settings" Þ "Connection". Als "Host Name" ist der Mailrechner "mail-stl.htw.uni-sb.de" und als "User Name" der Benutzername im Systemtechniklabor einzutragen, wie z.B. "aengel". Weiterhin notwendig ist die Einstellung der "Authentication Method" auf "Public Key" und der Portnummer auf "22".

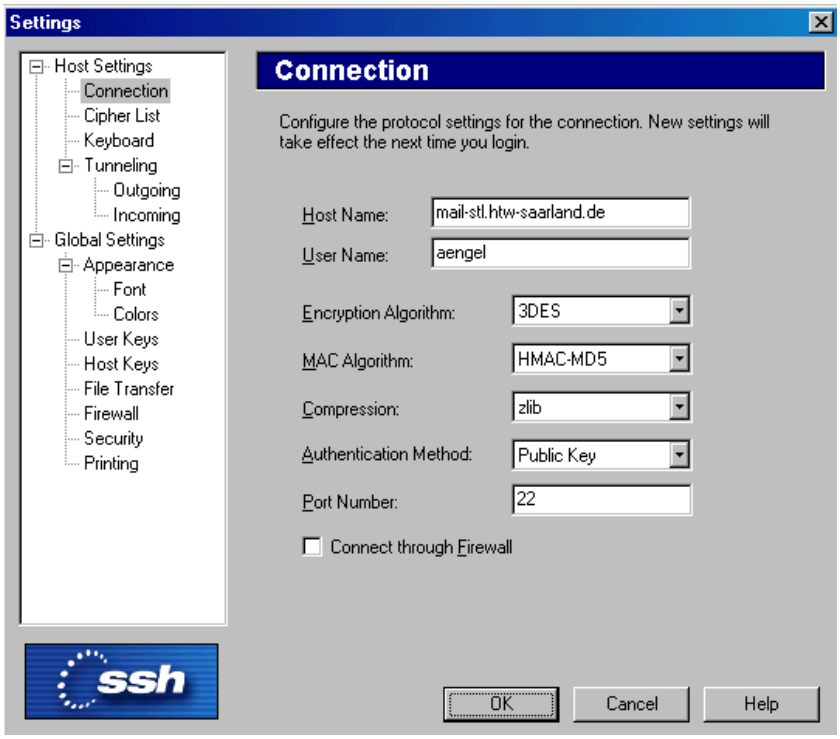

Abbildung 4: Verbindungseinstellungen

6. Generierung und Installation der Schlüssel

Zur Authentifizierung wird DSA empfohlen. Dazu muss der Benutzer einen geheimen und einen öffentlichen Schlüssel erzeugen. Dies kann auf einem Rechner des Systemtechniklabors erfolgen oder auf dem heimischen Rechner. Der geheime Schlüssel verbleibt dabei auf dem heimischen Rechner, hier auch lokaler Rechner (Local) genannt. Er muss unbedingt vom Zugriff anderer Personen geschützt werden. Der öffentliche Schlüssel muss in das Unterverzeichnis ".ssh2" des Benutzers im Systemtechniklabor kopiert werden. Dies kann mit einer Diskette erfolgen oder per Dateitransfer mit FTP. Zum Dateitransfer steht der Rechner "stl-d-04.htw-saarland.de" vom Zugriff außerhalb des HTW-Netzes zur Verfügung, vorausgesetzt man verfügt über eine Einmalpasswortliste. Da der öffentliche Schlüssel jedem zugänglich sein kann, müssen hier nicht unbedingt Sicherheitsmassnahmen ergriffen werden. Der öffentliche Schlüssel steht jetzt mittels NFS auch auf dem Rechner "mail-stl.htw-saarland.de" zur Verfügung, der zum Abruf von Email via POP3 auch von außerhalb des HTW-Netzes dient. Dadurch können nun SSH-Verbindungen zu diesem Rechner aufgebaut werden.

Außerdem sollte der öffentliche Host-Key des Rechners "mail-stl.htw-saarland.de", der diesen eindeutig identifiziert, zur Verhinderung eines Angriffes (z.B. mittels DNS-Spoofing) kopiert und in die Host-Key-Liste des lokalen Rechners aufgenommen werden. Dies erfolgt beim erstmaligen Zugriff auf den Rechner automatisch. Sicherer ist es den öffentlichen Host-Key auf Diskette zu kopieren. Er befindet sich auf dem jeweiligen Rechner unter "/etc/ssh2/hostkey.pub".

6.1 Unix

Der öffentliche und der geheime Schlüssel wird mit dem Befehl "ssh-keygen2" erzeugt und im Unterverzeichnis ".ssh2" des Benutzers abgelegt. Anschließend muss der öffentliche Schlüssel in die Datei "authorization" und der geheime Schlüssel in die Datei "identification" eingetragen werden.

Ein möglicher Ablauf zur Schlüsselgenerierung und Installation wäre folgender:

mkdir .ssh2	# Unterverzeichnis ".ssh2" erstellen
cd .ssh2	# Wechsel des Verzeichnisses
ssh-keygen2 htw-schluessel	# Schlüsselgenerierung, Name=htw-schluessel
echo "IdKey htw-schluessel" > identification	# Konfiguration der Datei "identification"
Öffentlichen Schlüssel "htw-schluessel.pub" auf den entfernten Rechner kopieren (z.B. mit Diskette, FTP)	

Abbildung 5: Schlüsselgenerierung und Installation auf dem lokalen Rechner (Local)

mkdir .ssh2	# Unterverzeichnis ".ssh2" erstellen
cd .ssh2	# Wechsel des Verzeichnisses
echo "Key htw-schluessel.pub" > authorization	# Konfiguration der Datei "authorization"
Öffentlichen Schlüssel "htw-schluessel.pub" in das Verzeichnis ".ssh2" kopieren	

Abbildung 6: Schlüsselgenerierung und Installation auf dem entfernten Rechner (Remote)

In der Datei "identification" sind alle Schlüssel eingetragen, mit denen man sich beim Server identifizieren kann. Sie befindet sich auf dem lokalen Rechner (zuhause). Der Aufbau der Datei ist in Abbildung 7 dargestellt.

```
IdKey schlüsselname1 (z.B. htw-schluessel)
IdKey schlüsselname2 (z.B. firma-schluessel)
...
```

Abbildung 7: Aufbau der Datei "identification"

In der Datei "authorization" sind alle Schlüssel eingetragen, die zur Einwahl berechtigen. Sie befindet sich auf dem entfernten Rechner (mail-stl). Der Aufbau der Datei ist in Abbildung 8 dargestellt.

```
Key schlüsselname1 (z.B. htw-schluessel.pub)
Key schlüsselname2 (z.B. firma-schluessel.pub)
...
```

Abbildung 8: Aufbau der Datei "authorization"

SSH stellt jedoch ein Skript zur Verfügung, welches den Vorgang der Schlüsselgenerierung und der Installation erheblich vereinfacht. Durch Aufruf des Befehls "ssh-pubkeymgr" wird der Benutzer interaktiv durch die Schritte der Schlüsselerzeugung und das Aufspielen des öffentlichen Schlüssel auf den entfernten Rechner geführt. "ssh-pubkeymgr" muss sowohl auf dem lokalen Rechner als auch auf dem entfernten Rechner durchgeführt werden. Abbildung 9 zeigt den Ablauf der Schlüsselerzeugung mittels "ssh-pubkeymgr".

Zur Benutzung im Systemtechniklabor der HTW empfehlen wir das selbstentwickelte Skript "ssh-htwkeygen" (Auszug in Abbildung 10). Es übernimmt die Schlüsselerzeugung und die Einrichtung der Datei "authorization" auf einem Rechner des Systemtechniklabors. Anschließend kann der Schlüssel auf eine Diskette kopiert werden. Auf den lokalen Rechner muss dann nur noch der geheime Schlüssel importiert werden. Das SSH-Befehlsverzeichnis sollte im Standardpfad enthalten sein.

```
> ssh-pubkeymgr

Checking for existing user public keys..
Couldn't find your DSA keypair.. I'll generate you a new set..
Running ssh-keygen2... don't forget to give it a passphrase!
Generating 1024-bit dsa key pair
  2 OOo.oOo.oOo.
Key generated.
1024-bit dsa, olitz@locutus, Fri Sep 01 2000 16:30:36 +0200
Passphrase :
Again :
Private key saved to /home/olitz/.ssh2/id_dsa_1024_a
Public key saved to /home/olitz/.ssh2/id_dsa_1024_a.pub
Creating your identity file..
Creating your authorization file..

Note: You'll need to edit this appropriately.
Creating your local host public key..
Adding your local host in case you don't want to go anywhere ;)
Do you want to add any hosts to your authorization file?(Default: yes)no
Skipping editing the authorization file..

All the new files are in your /home/olitz/.ssh2 directory.

Do you want to upload olitz@locutus key to a remote host? (Default: yes)yes
Upload to which host?
stl-d-04.htw-saarland.de
Which user account?
olitz
Where is the olitz's home directory?
(e.g. /home/anne, /u/ahc, etc.)
/export/home_98/olitz
Now running scp2 to connect to stl-d-04.htw-saarland.de..
Most likely you'll have to type a password :)
Host key not found from database.
Key fingerprint:
xehec-zazad-mizuh-zapok-fecyr-nizum-henef-pigic-vulyk-nobym-sexyx
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)? yes
Host key saved to /home/olitz/.ssh2/hostkeys/key_22_stl-d-04.htw-saarland.de.pubhost key for stl-d-04.htw-
saarland.de, accepted by olitz Sun Sep 03 2000 13:02:41 +0200
olitz@stl-d-04.htw-saarland.de's password:
olitz-stl-d-04.htw-saarland.de.pub          |  737B |  0.7 kB/s | TOC: 00:00:01 | 100%

Press return to upload to more hosts or Ctrl-D to exit.

Don't forget to run ssh-pubkeymgr on any remote hosts you sent
your public key to.

Done.
```

Abbildung 9: Schlüsselerzeugung mittels "ssh-pubkeymgr" auf dem lokalen Rechner

```
#!/bin/ksh
# @(#) ssh-htwkeygen SSH-Schlüsselerzeugung im STL – Auszug – siehe Diskette für vollständige Version

# Fehlerausgabe
function fehler { echo $1; exit 1; }

# Variablen
floppy_mount="volcheck"                 # Befehl zum Mounten der Diskette
floppy_umount="eject"                   # Befehl zum Umounten der Diskette
floppy_pfad="/floppy/floppy0"           # Mountpfad der Diskette
muster="-*"                             # Muster zur Argumentüberprüfung (z.B. -h | --help)

# Prüfe Argumente
if [[ ( "$1" == $muster || "$#" -gt 1 ) ]]; then
      echo "Aufruf: $0 [Schlüsselname]"; exit 1
fi

# Setze Schlüsselname
if [[ -z "$1" ]]; then
        key="htw-schluessel"    # Standardname des Schlüssels
else
        key=$1
fi

# Prüfe, ob schon ein Schlüssel existiert
if [[ ! -s "$HOME/.ssh2/$key" ]]; then
        echo "Erzeugung eines neuen Schlüssels"
        ssh-keygen2 $HOME/.ssh2/$key || fehler "Fehler beim Erzeugen des Schlüssels!"

        # Existiert Datei <authorization> schon ?
        if [[ -s "$HOME/.ssh2/authorization" ]]; then
                echo "Key $key.pub" >> $HOME/.ssh2/authorization
                echo "Datei <authorization> um Schlüssel <$key> erweitert"
        else
                echo "Key $key.pub" > $HOME/.ssh2/authorization
                echo "Datei <authorization> mit Schlüssel <$key> erzeugt"
        fi

        # Schlüssel auf Diskette kopieren
        echo "Schlüssel auf Diskette kopieren (Ja/Nein) ?\c"; read kopieren
        case "$kopieren" in
            "" | [jJ] | [jJ][aA])
                echo "Diskette einlegen.\c"; read diskette
                $floppy_mount || fehler "Fehler beim Mounten der Diskette!"
                cp $HOME/.ssh2/$key $HOME/.ssh2/$key.pub $floppy_pfad
                echo "IdKey $key" > $floppy_pfad/identification
                echo "Daten kopiert!"
                echo 'Schlüssel und Datei <identification> zuhause nach <$HOME/.ssh2> kopieren.'
                $floppy_umount || fehler "Fehler beim Umounten der Diskette!"
                ;;
            [nN] | [nN][eE][iI][nN])
                echo "Schlüssel <$key> kopieren und zuhause in Datei <identification> eintagen."
        esac
else
        echo "Schlüssel existiert schon."
        echo "Anderer Schlüsselname verwenden oder folgende Datei löschen:"
        echo "$HOME/.ssh2/$key"
fi
echo "ssh-htwkeygen beendet."
```

Abbildung 10: Schlüsselerzeugung mittels "ssh-htwkeygen" im Systemtechniklabor

6.1.1 Import von SSH-Schlüsseln

Der öffentliche und der geheime Teil eines SSH-Schlüssels wird in jeweils einer Textdatei gespeichert, wie z.B. der öffentliche in der Datei "htw-schluessel.pub" und der geheime in der Datei "htw-schluessel". Zum Import ist die öffentliche Schlüsseldatei auf den entfernten Rechner und die geheime Schlüsseldatei auf dem lokalen Rechner, in das Unterverzeichnis ".ssh2" des Benutzers zu kopieren. Anschließend sind die Dateien "authorization" und "identification" anzupassen, wie in Kapitel 6.1, Abbildung 7 und 8 beschrieben.

6.1.2 Import von OpenPGP-Schlüsseln

SSH unterstützt den Import von PGP-Schlüsseln, die nach dem OpenPGP-Standard erstellt wurden, z.B. von GnuPGP. Dazu muss auf dem lokalen Rechner die Datei "secring.gpg" aus dem PGP-Verzeichnis in das SSH-Verzeichnis kopiert werden. Die Datei "pubring.gpg" muss auf den entfernten Rechner kopiert werden. Unter Unix sind folgende Verweise nützlich, wenn GnuPGP installiert ist:

```
ln –s ~/.gnupg/secring.gpg ~/.ssh2/  (Local)
ln –s ~/.gnupg/pubring.gpg ~/.ssh2/  (Remote)
```

Abbildung 11: Verweise auf GnuPGP-Schlüsselringe

Folgende Daten sind auf dem lokalen Rechner in der Datei "identification" einzutragen:

PgpSecretKeyFile secring.gpg	Name des geheimen Schlüsselbundes
IdPgpKeyName name	Name des zu verwendenden Schlüssels
IdPgpFingerprint xxx	Fingerabdruck des zu verwendenden Schlüssels
IdPgpKeyId xxx	Id-Nummer des zu verwendenden Schlüssels

Abbildung 12: Datei "identification" mit PGP-Schlüssel

Folgende Daten sind auf dem entfernten Rechner in der Datei "authorization" einzutragen:

PgpPublicKeyFile pubring.gpg	Name des öffentlichen Schlüsselbundes
PgpKeyName name	Name des zu verwendenden Schlüssels
PgpFingerprint xxx	Fingerabdruck des zu verwendenden Schlüssels
PgpKeyId xxx	Id-Nummer des zu verwendenden Schlüssels

Abbildung 13: Datei "authorization" mit PGP-Schlüssel

Für den Benutzer "olitz" ist z.B. folgendes in der Datei "identification" einzutragen:

```
PgpSecretKeyFile secring.gpg
IdPgpKeyName Oliver Litz <olitz@htw-saarland.de>
IdPgpFingerprint DD28 3CC3 A462 3B15 A128  6DF5 0623 B512 A27B 3CD9
IdPgpKeyId 1024D/A27B3CD9
```

Abbildung 14: Beispiel für Datei "identification" mit PGP-Schlüssel

6.2 Windows

Zur Schlüsselgenerierung unter Windows wird nach Start des Programms SSH der Menüpunkt "Edit" Þ "Settings" Þ "Global Settings" Þ "User Keys" ausgewählt. Anschließend den Button "Generate New Keypair" anwählen. Die "Key-Length" (Schlüsselstärke) sollte mindestens "1024-Bit" betragen. Anschließend vergibt man einen Namen für die Schlüsseldatei, wie zB. "htw-schluessel". Zum Schutz des Schlüssels sollte man diesen mit einem Passwort sichern. Nach vollständiger Eingabe der Daten wird ein öffentlicher und ein geheimer Schlüssel generiert. Der öffentliche Schlüssel kann mit der Option "Upload Public Key" zum Rechner "stl-d-04.htw-saarland.de" gesendet werden.

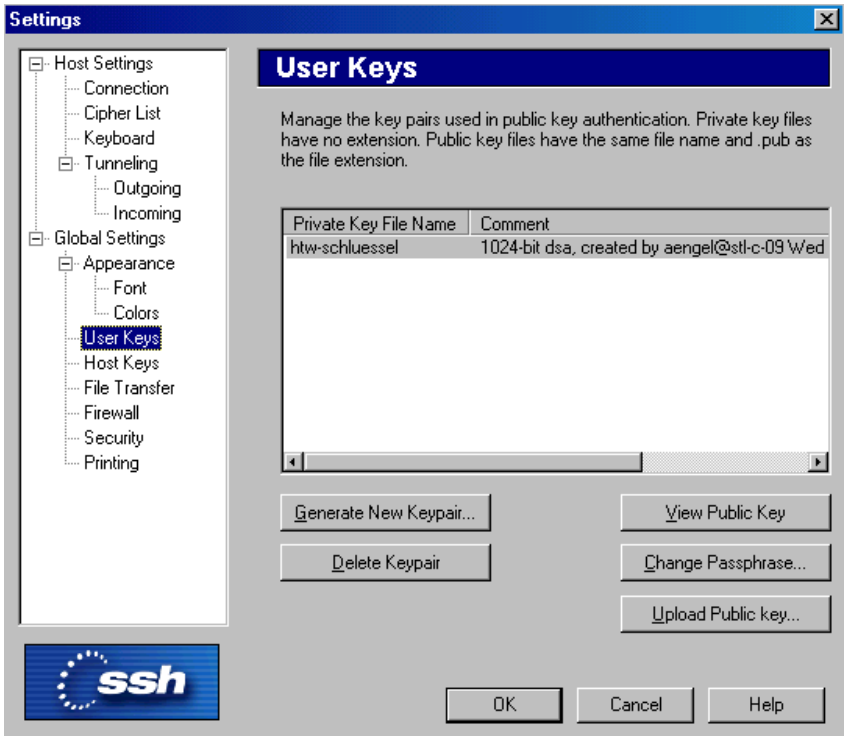

Abbildung 15: Schlüsselverwaltung

Nimmt der Benutzer zum ersten Mal Kontakt mittels SSH mit dem Rechner "mail-stl" auf, wird der öffentliche "Host-Key" übertragen. Er dient bei späteren Sitzungen dazu, die Identität des Hosts zu überprüfen. Fällt diese Prüfung einmal negativ aus, weigert sich der SSH-Client, weiter mit diesem Rechner zu kommunizieren.

6.2.1 Import von SSH-Schlüsseln

Um einen schon erstellten Schlüssel zu importieren, muss der jeweilige Schlüssel (geheim oder öffentlich) in einer Textdatei vorliegen. Der öffentliche Schlüssel befindet sich z.B. in der Datei "htw-schluessel.pub", der geheime Schlüssel in der Datei "htw-schluessel". Diese Dateien müssen in das Verzeichnis "UserKeys" kopiert werden. Dazu startet man den SSH-Client und wählt den Menüpunkt "Edit" Þ "Settings" Þ "Host Settings" an. Dort wird angezeigt, wo das Verzeichnis "UserKeys" auf dem Rechner zu finden ist. Durch Betätigung des "Browse"-Buttons öffnet sich ein Explorer-Fenster. In dieses können die Schlüssel dann kopiert werden. Standardmäßig ist das Verzeichnis unter "\Programme\SSHCommunications Security\Users\UserKeys" zu finden. Der Import von PGP-Schlüsseln ist in der Windows-Version nicht möglich.

7. Konzepte Mailabruf

Rechner, die über keine permanente Verbindung zum Internet verfügen, können nur Email empfangen, während eine Verbindung zu einem im Internet befindlichen Rechner besteht. Da aber nicht immer eine dauerhafte, kostenintensive Verbindung notwendig ist, besteht die Möglichkeit Email in einem Postfach auf einem dem Internet zugänglichen Rechner zwischenzuspeichern. Zum Abruf dieser hinterlegten Email dient das POP (Post Office Protokoll), üblicherweise in der Version 3 (POP3) eingesetzt. Zur Verwendung dieses Protokolls benötigt man einen Serverdienst auf Seite des Rechners mit permanenter Internetverbindung (Port 110) und einen Client auf Benutzerseite. Client und Server kommunizieren über POP3 miteinander.

7.1 Heutiges Konzept mit POP3

Zur Identifikation des Client beim Server über POP3, wird der Benutzername und das Passwort benötigt. Beides wird unverschlüsselt übertragen. Hier wird die Schwachstelle des POP3 ersichtlich, eine schwache Authentifizierung nur über Passwort. Auch das beim Datenverkehr im Internet zuständige Protokoll TCP/IP bietet hier keinen Schutz, da mit IPv4 alle Daten unverschlüsselt übertragen werden. Mit entsprechenden Programmen kann der Datenverkehr abgehört sowie Benutzername und zugehöriges Passwort angeeignet werden. Hierdurch können dann leicht Email abgefangen oder manipuliert werden. Besteht die Kennung nicht nur aus einem Postfach, sondern einer Login-Kennung, wie am Beispiel der HTW, entstehen weitere Gefahren. Einem Eindringling ist es nun möglich, aktiv auf einem Rechner des Systemtechniklabors tätig zu werden, d.h. Programme zu starten und neben Email auch weitere Daten zu analysieren oder zu manipulieren.

Der heute üblicherweise verwende Abruf mittels POP3 ist nach heutigen Gesichtspunkten also als veraltet und höchst sicherheitskritisch anzusehen.

7.2 Konzept mit SSH

Damit der Abruf von Email abgesichert werden kann, ist also eine eindeutige Identifikation sowie die Verschlüsselung des gesamten Datenverkehrs notwendig. Hier bietet sich die Benutzung von SSH mit den Leistungsmerkmalen (s. Kapitel 2) wie Authentifizierung, Verschlüsselung und "TCP/IP-Tunneling" an.

7.2.1 Tunneln von TCP/IP-Verbindungen

SSH bietet die Möglichkeit alle TCP/IP-Dienste, wie z.B. Telnet, FTP, HTTP oder POP durch sogenanntes "TCP/IP Tunneling" sicher zu übertragen. Dazu wird auf dem lokalen Rechner ein Proxy-Dienst eingerichtet, der auf dem lokalen Port auf Anfragen eines Client wartet. Anfrage und Daten werden dann über die sichere SSH-Verbindung zum entfernten Rechner übertragen. Idealerweise ist dies auch der Zielrechner. Ist dies nicht der Fall, werden Anfrage und Daten unverschlüsselt zum entsprechenden Port des Zielrechners weitergegeben.

Es gibt also zwei Möglichkeiten der Übertragung:

- sichere Übertragung: Zielrechner ist gleich dem entfernten Rechner
  Es besteht eine sichere SSH-Verbindung vom lokalen Rechner (Local) zum entfernten Rechner (Remote + Destination).

Abbildung 16: Sicheres TCP/IP-Tunneling

- sichere und unsichere Übertragung: Zielrechner ist ungleich dem entfernten Rechner
  Es besteht eine sichere SSH-Verbindung vom lokalen Rechner (Local) zum entfernten Rechner (Remote) sowie eine unsichere Verbindung vom entfernten Rechner (Remote) zum Zielrechner (Destination).

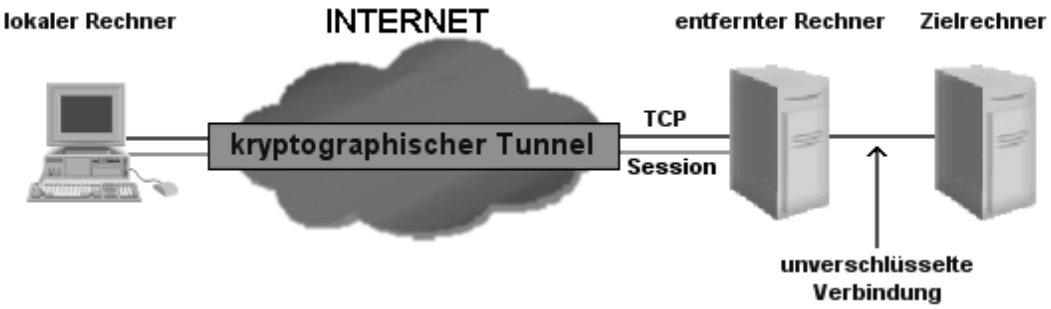

Abbildung 17: Sicheres und unsicheres TCP/IP-Tunneling

SSH unterscheidet zwischen "Local-" und "Remote-Tunneling". Beim "Local-Tunneling" wird auf dem lokalen Rechner auf die Verbindungsanfrage gewartet. Die Daten werden vom lokalen zum Zielrechner übertragen. Beim "Remote-Tunneling" wird auf dem Zielrechner auf die Verbindungsanfrage gewartet. Die Daten werden vom Zielrechner zum lokalen Rechner übertragen.

Hierdurch lassen sich auch leicht sichere virtuelle-private-Netzwerke (VPNs) realisieren.

7.2.2 Verwendung des TCP/IP-Tunneling zur Mailübertragung

Zur sicheren Mailübertragung werden POP-Anfragen mittels TCP/IP-Tunneling vom lokalen Rechner zum Mailrechner weitergeleitet. Dazu baut der Client zuerst eine SSH-Verbindung zum Server auf, der den entsprechenden SSH-Dienst zur Verfügung stellen muss. Schon bei der Authentifikation wird der Datenverkehr verschlüsselt (durch Session-Key des Servers). Nach der erfolgreichen Einwahl steht eine abgesicherte Verbindung zur Verfügung. Durch Tunneling wird nun auf Clientseite ein neuer Port zur Verfügung gestellt. Bei Zugriffen auf den Port werden diese über die SSH-Verbindung an den POP3-Port des Server (110) verschlüsselt weitergeleitet.

Demzufolge werden also keine Daten wie Benutzername oder Passwort mehr unverschlüsselt übertragen. Üblicherweise werden auf Clientseite Portnummern größer 1024 genutzt, da Portnummern kleiner oder gleich 1024 dem Systemverwalter zur Bereitstellung von Diensten vorbehalten bleiben.

Anwendungsprogramme zum Abruf von Email müssen also auf den Client und den neuen Port umgestellt werden. Manche Programme wie Netscape Communicator bis einschließlich Version 4.x sind fest auf Port 110 einprogrammiert. Um diese Programme zu nutzen, muss die SSH-Verbindung mit Systemverwalter- bzw. "Root"-Rechten aufgebaut werden, damit der Port 110 genutzt werden kann. Von diesem Vorgehen ist jedoch aus Sicherheitsgründen abzusehen. Ein Umstieg auf eine andere Clientsoftware ist hier zu empfehlen.

Es ist außerdem noch zu beachten, dass die Kommunikation nur zwischen den beiden Endsystemen der SSH-Verbindung verschlüsselt stattfindet. POP-Serverdienst und SSH-Serverdienst sollte also auf dem gleichen Rechner zur Verfügung stehen. Befindet sich der POP-Server außerhalb der SSH-Verbindung, wird zwischen dem SSH-Server und dem POP-Server eine unverschlüsselte Verbindung aufgebaut, wie in Abbildung 17 dargestellt.

8. Realisierung Mailabruf

Eingehende Email werden bei Benutzern des Systemtechniklabors unter "/var/mail/benutzername" gespeichert. Der Rechner "mail-stl.htw-saarland.de" stellt über einen POP3-Serverdienst die Möglichkeit zur Verfügung, eingegangene Email über einen POP-Client abzufragen.

Zur Realisierung des sicheren Mailabrufs muss der SSH-Dämon, wie in Kapitel 5 beschrieben, auf dem Rechner "mail-stl" installiert und aktiviert sein. An dem POP3-Server müssen keine Änderungen vorgenommen werden. Auf Client-Seite muss ein SSH-Client und ein POP-Client verfügbar sein. Im weiteren stellen wir eine Möglichkeit vor, den sicheren Abruf von Email auf aktuellen Betriebssystemen durchzuführen. Dabei nutzen wir das von SSH zur Verfügung gestellte Leistungsmerkmal "TCP/IP-Tunneling". Für andere Möglichkeiten, wie die Benutzung von "netcat" über SSH oder von Multi-User-POP, sei auf [7] verwiesen.

Ein Kommunikation mittels SSH findet immer nach folgendem Muster statt:

- Verbindungsaufnahme
- Authentifizierung des Servers
- Authentifizierung des Benutzers
- Verschlüsselte Datenübertragung
    - Anmeldung am POP-Server
    - Übertragung der Email
- Verbindungsabbau

Der Ablauf ist bei allen Betriebssystemen identisch.

8.1 Unix

Unix eignet sich insbesondere durch die gute Skriptfähigkeit der Unix-Shell für den komfortablen Mailabruf über SSH. Im folgenden wird davon ausgegangen, dass auf dem Unix-Client SSH, wie in Kapitel 5 erläutert, installiert und im Standardpfad eingetragen ist. Des weiteren wird für den Batch-Betrieb der POP-Client "fetchmail" [8] benötigt.

8.1.1 Manueller Verbindungsaufbau

Um eine Verbindung über SSH mit Tunneling zum Rechner "mail-stl" aufzubauen gibt es drei Möglichkeiten. Die Optionen können in der Kommandozeile übergeben werden, in einer Konfigurationsdatei gespeichert werden oder über ein grafisches Frontend eingestellt werden. Im folgenden wird beispielhaft die Benutzerkennung "olitz" im Systemtechniklabor verwendet. Ist die Benutzerkennung auf dem lokalen Rechner identisch mit der des Systemtechniklabors, dann kann die Eingabe optional erfolgen.

- Manueller Verbindungsaufbau mit Kommandozeilenoptionen

```
ssh2 +C –fo –l olitz -L 11000:mail-stl.htw-saarland.de:110 mail-stl.htw-saarland.de sleep 20
```

Abbildung 18: SSH mit Tunneling

Bedeutung der Optionen:

+C	Kompression der übertragenen Daten
-fo	Nach Aufbau der Verbindung als Hintergrundprozess im "oneshot"-Modus starten, d.h. SSH beendet sich selbst, nachdem das Tunneling beendet wird
-l username	Benutzername auf entferntem Rechner (Systemtechniklabor)
-L port:rechner:port	Tunneling des lokalen Port 11000 auf den Port 110 des Rechners "mail-stl"
sleep 20	Nach 20 Sekunden Wartezeit den SSH-Prozess beenden (optional)

- Manueller Verbindungsaufbau mit Konfigurationsdatei

Durch Verwendung der Konfigurationsdatei "ssh2_config" im Verzeichnis "$HOME/.ssh2" ist es möglich, durch Verwendung eines Alias, die Kommandozeilenoptionen zu ersetzen. Die in Abbildung 19 vorgenommenen Einstellungen ersetzen die Optionen aus Abbildung 18. Durch Aufruf von "ssh2 mail" kann die Verbindung manuell aufgebaut werden.

```
mail:
Host "mail-stl.htw-saarland.de"
User "olitz"
Compression "yes"
GoBackground "oneshot"
KeepAlive "yes"
LocalForward "11000:mail-stl.htw-saarland.de:110"
```

Abbildung 19: Konfigurationsdatei "ssh2_config"

- Manueller Verbindungsaufbau mit grafischem Frontend

Eine grafische Oberfläche stellt das Programms KSSH [9] für den KDE-Window-Manager zur Verfügung. Unter "Options" ist der Befehl aus Abbildung 18 einzutragen.

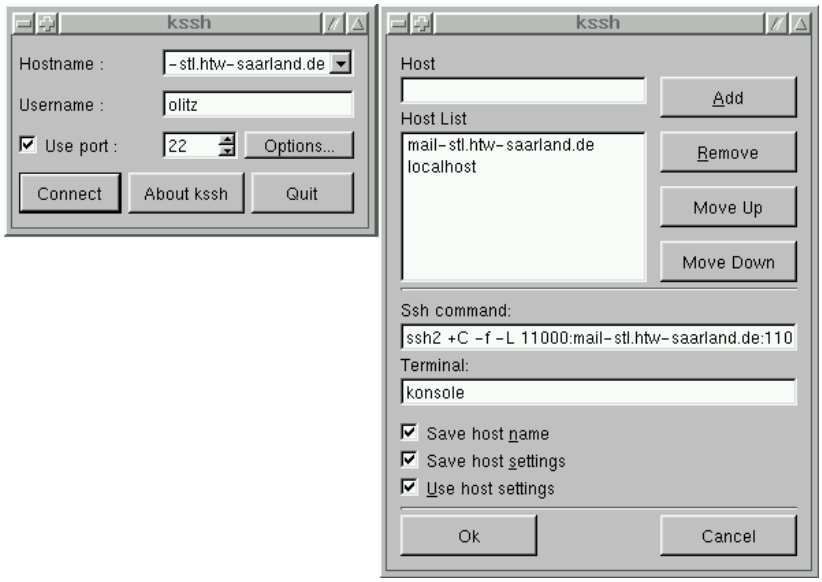

Abbildung 20: KSSH

Nach Aufbau der Verbindung und Abfrage des Schlüsselpasswortes, hier vereinfacht SSH-Passwort genannt, steht der lokale Port 11000 zum Mailabruf zur Verfügung. Zu beachten ist, dass für Benutzer nur Portnummern größer 1024 zur Verfügung stehen. Der Abruf kann nun über einen POP-Client mit folgenden Einstellungen stattfinden:

Protokoll:	POP3
Rechner:	localhost
Port:	11000

Abbildung 21: POP3-Einstellungen

Leider unterstützt nicht jeder POP-Client die Auswahl der Portnummer. Prominentes Beispiel ist hier Netscape, dessen Communicator fest auf Portnummer 110 einprogrammiert ist.

Ein manueller Ablauf unter Unix würde also folgendermaßen aussehen:

- Aufruf von SSH mit Tunneling
- Evtl. Akzeptieren des Host-Keys
- Eingabe des SSH-Passwortes
- Start des POP-Client
- Abruf der Email mittels Tunneling über lokalen Port
- Eingabe des POP-Passwortes
- Beenden der Verbindung (SSH)

8.1.2 Automatisierung mittels "fetchmail"

Unter Unix gibt es die einfache Möglichkeit, alle Email vom Rechner "mail-stl" in die lokale Mailbox zu übertragen. Alle Mailprogramme können diese dann abrufen. Dazu kann das Programm "fetchmail" benutzt werden. Es tritt als POP-Client mit dem Mailserver in Verbindung und übergibt die übertragene Email standardmäßig dem Sendmail-Dienst auf Portnummer 25. Hierfür ist natürlich auf dem lokalen Rechner ein aktivierter Sendmail-Dienst notwendig. Sendmail stellt die Email dann dem lokalen Benutzer zu.
Die Konfigurationsdatei von "fetchmail" mit Benutzung von SSH sieht folgendermaßen aus:

```
defaults
user olitz is oliver here

poll localhost with protocol pop3 and port 11000:
preconnect "ssh2 -q -fo -l olitz -L 11000:mail-stl.htw-saarland.de:110 mail-stl.htw-saarland.de
sleep 20; sleep 5"
password PopPassword;
```

Abbildung 22: Konfigurationsdatei ".fetchmailrc"

Wie aus der Datei hervorgeht, stellt "fetchmail" standardmäßig eine Verbindung zum lokalen Rechner auf Port 11000 unter Verwendung des Protokolls "POP3" her. Auf dem POP-Server lautet der Benutzername "olitz", auf dem lokalen Rechner "oliver". Vorher wird aber das SSH-Kommando unter "preconnect" ausgeführt. Dieses aktiviert das Tunneling auf Port 11000. Während der Übertragung werden die Daten komprimiert und keine Meldungen ausgegeben. Auf die Ausführung von "fetchmail" wird sicherheitshalber noch fünf Sekunden gewartet, da auf langsameren Rechnern die Authentifizierung etwas länger dauern kann.

Durch Eingabe des Befehls "fetchmail" wird der Mailabruf gestartet. Zuerst muss aber noch das SSH-Passwort eingegeben werden. Das Passwort für den POP-Zugriff ist bereits in der Konfigurationsdatei von Fetchmail eingetragen. Diese Datei sollte die Rechte "600" besitzen, also nur für den Eigentümer les- und schreibbar sein !

Ein Ablauf mittels "fetchmail" würde also folgendermaßen aussehen:

- Aufruf von "fetchmail"
- Automatischer Aufbau der SSH-Verbindung
- Eingabe des SSH-Passwortes
- Automatischer Abruf der Email durch "fetchmail" mittels Tunneling über lokalen Port
- Automatisches Beenden der Verbindung (SSH)
- Start des Email-Client
- Abruf der Email über lokale Mailbox

8.1.3 Automatisierung mittels Authentication Agent

Durch Benutzung des Authentication Agent kann der Email-Abruf weiter automatisiert werden. Dem Agenten werden ein oder mehrere SSH-Schlüssel mit Passwort übergeben. Stellt nun ein Server eine Authentifizierungsanfrage nach diesem Schlüssel, antwortet der Agent selbstständig. Der Benutzer muss kein Passwort mehr eingeben. Passwort und Schlüssel verbleiben jedoch immer auf dem lokalen Rechner.

Durch den Agenten wird ein enormer Komfortgewinn erreicht, da das SSH-Passwort nur noch einmal eingegeben werden muss. Im folgenden übernimmt dies der Agent. Hierdurch ist es möglich, ein Skript zu benutzen, das alle 10 Minuten über den SSH-Tunnel Email abruft.

```
#!/bin/sh
ssh-add2 ~/.ssh2/schluessel
while true; do fetchmail --syslog; sleep 10m; done
```

Abbildung 23: Skript "getmail"

Mit "ssh-add2 schlüsselname" wird der jeweilige Schlüssel beim Agenten registriert. "ssh-add2" muss als Kind-Prozess von "ssh-agent2" gestartet werden. Das Programm "fetchmail" wird in einer Endlosschleife gestartet, wobei Status- und Fehlermeldungen an den Syslog-Dämon weitergegeben werden. Gestartet wird das Skript "getmail" (Achtung: Datei ausführbar machen!) durch Aufruf von:

"ssh-agent2 getmail"

Wird eine PPP-Verbindung aufgebaut, kann der Aufruf von "getmail" in der Datei "ip-up" erfolgen. In "ip-down" wird "getmail" dann entsprechend beendet. ("ip-up, ip-down" sind üblicherweise unter "/etc/ppp/" zu finden). Hierdurch werden nach Aufbau einer PPP-Verbindung alle 10 Minuten neue Email in die lokale Mailbox übertragen.

Ein Ablauf mittels "Authentication Agent" würde also folgendermaßen aussehen:

- Start Authentication Agent
- Schlüsselregistrierung mit SSH-Passwort beim Authentication Agent
- Aufbau der Internetverbindung
    - Automatischer Aufbau der SSH-Verbindung
    - Automatischer Abruf der Email durch "fetchmail" mittels Tunneling über lokalen Port
    - Automatisches Beenden der Verbindung (SSH)
- Beenden der Internetverbindung
- Start des Email-Client
- Abruf der Email über lokale Mailbox

8.1.4 Gefahren

So komfortabel die Lösung aus 8.1.3 nun auch ist, über einen großen Nachteil darf nicht hinweggesehen werden. Das POP-Passwort steht im Klartext in der Konfigurationsdatei ".fetchmailrc". Durch Fehler in Anwendungen, z.B. Browser mit Javascript, könnte die Datei übertragen werden. Außerdem befindet sich auch das vom Authentication Agent verwaltete SSH-Passwort im Speicher. Durch einen Absturz könnte das Passwort über den Speicherauszug ermittelt werden. Ist das System hinreichend sicher und die Rechte der Datei ".fetchmailrc" sorgfältig gesetzt, stellt dies eigentlich kein Problem dar. Trotzdem sollte man sich der Gefahr bewusst sein.

Eine wirklich sichere Lösung bedingt einen manuellen Ablauf wie in 8.1.1. Beide Passwörter sind bei jedem Abruf anzugeben. Dazu muss zuerst eine SSH-Verbindung mit TCP/IP-Tunneling aufgebaut werden, dann muss der Abruf über einen POP-Client erfolgen. Beide Schritte sind manuell auszuführen, und das jeweilige Passwort ist einzugeben.

8.2 Windows

Zum sicheren Mailabruf muss das TCP/IP-Tunneling aktiviert werden. Dazu sind unter "Host Settings" Þ "Tunneling" Þ "Outgoing" einmalig folgende Einstellungen vorzunehmen, wie in Abbildung 24 dargestellt.

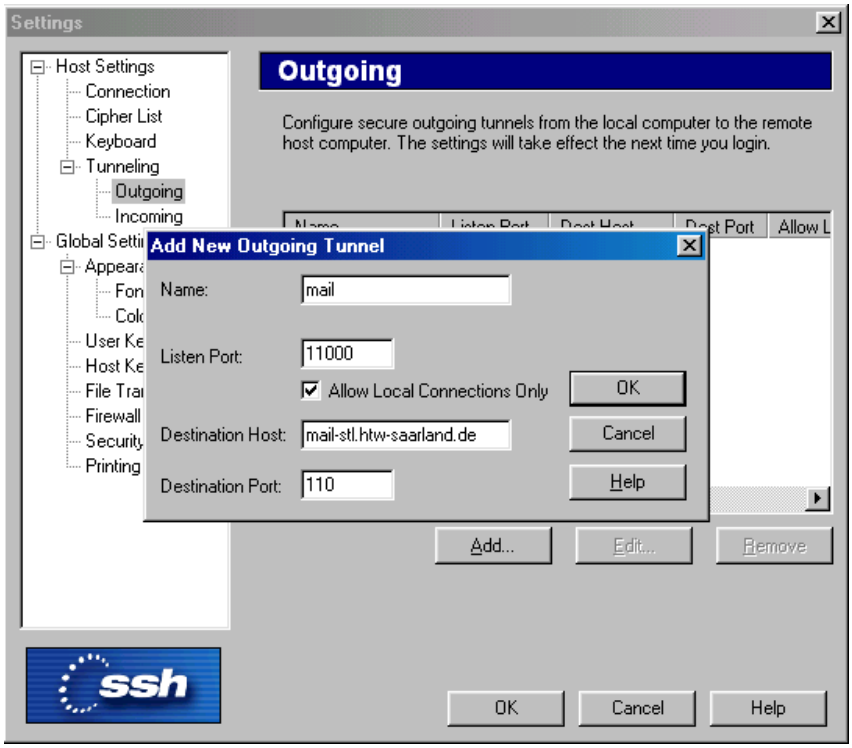

Abbildung 24: Aktivierung des Tunneling

Das Tunneling des lokalen Port 11000 auf den Port 110 des Rechners "mail-stl.htw-saarland.de" ist jetzt bei jeder Verbindung aktiviert.

Anschließend ist eine SSH-Verbindung zum Rechner "mail-stl.htw-saarland.de" aufzubauen. Nach Eingabe des SSH-Passwortes kann zum Mailabruf ein entsprechender POP-Client, wie z.B. Netscape, gestartet werden.

8.2.1 Mailabruf mittels Netscape

Zur Benutzung von Netscape [10] ist der Gebrauch der lokalen Portnummer 110 zwingend notwendig. Bei Programmen, die eine Auswahl der POP-Portnummer zum Mailabruf zulassen, sollte eine Portnummer größer 1024 gewählt werden. Die anderen Portnummern sollten für Dienste zur Verfügung stehen, die vom Systemverwalter initiiert werden. Im Vergleich zu UNIX ist die Benutzung der Portnummer 110 unter Windows, da sie dort auch dem Benutzer zur Verfügung steht, ein geringeres Problem.

Unter Netscape sollten folgende Einstellungen noch vorgenommen werden:

- Konfigurationsmenü "Edit" Þ "Preferences" anwählen.
- Unter "Mail & Diskussionsforen" den Eintrag "Mail Server" anwählen.
- "Hinzufügen" rechts neben dem größeren Feld "Server für eingehende Mail" anwählen.
- Als Server-Name "localhost", als Server-Typ "POP3" und als Benutzername den Benutzernamen des Systemtechniklabors angeben, wie z.B. in Abbildung 25.
- Eingabe bestätigen mit "OK".

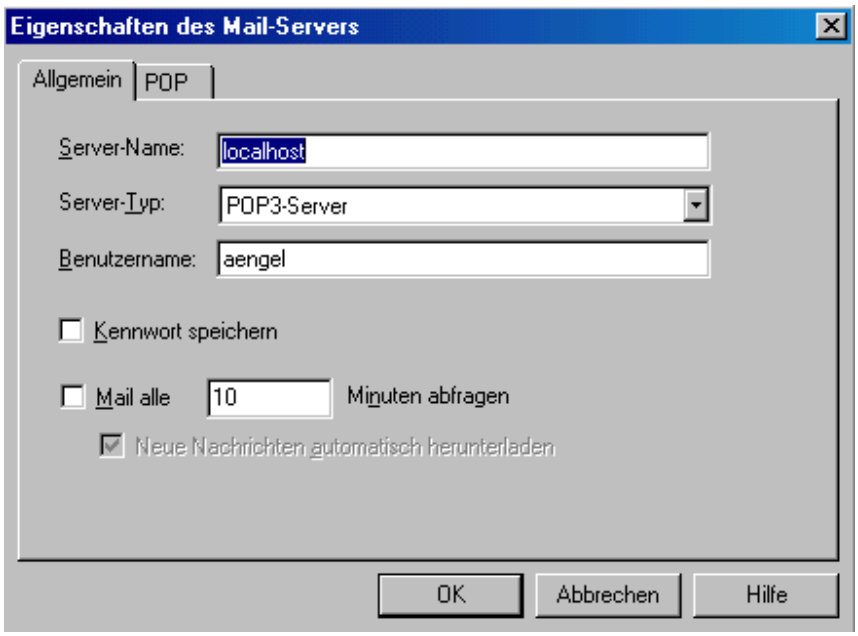

Abbildung 25: POP3-Konfigurationseinstellungen Netscape

Ein Ablauf mit Netscape würde folgendermaßen aussehen:

- Aufruf des SSH-Client mit Tunneling
- Eingabe des SSH-Passwortes
- Starten von Netscape
- Abruf der Email mittels Tunneling über lokalen Port (110)
- Eingabe des POP-Passwortes
- Beenden der Verbindung (SSH-Client)

8.2.2 Mailabruf mittels Pegasus Mail

Pegasus Mail [11] ist ein frei erhältlicher Email-Client. Der Vorteil dieser Anwendung ist, dass der Benutzer die Portnummer des Mail-Dienstes angeben kann. In der POP3-Konfiguration kann also der Mailabruf auf den Rechner "localhost" auf Portnummer 11000 erfolgen. Abbildung 26 zeigt die in Pegasus Mail vorzunehmenden Einstellungen.

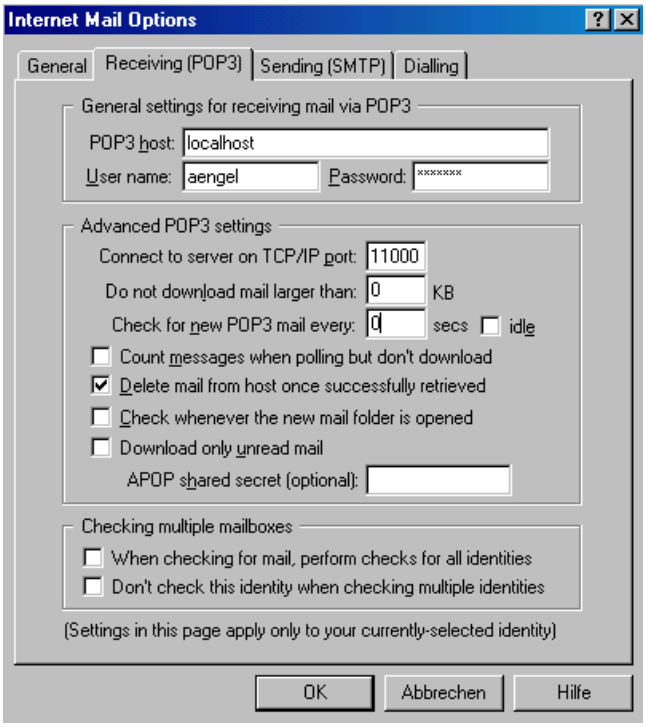

Abbildung 26: POP3-Konfigurationseinstellungen Pegasus Mail

9. Vor- und Nachteile

SSH sichert die Datenübertragung und die Authentizität des Rechners und Benutzers. Durch Nachlässigkeiten auf der Seite des Anwenders kann es jedoch zu Problemen und Gefahren kommen.

9.1 Vorteile

Neben der Verschlüsselung der Daten und der Authentifizierung der Rechner und Benutzer bietet die SSH ein breites Leistungsspektrum. SSH ist nicht nur ein sicherer Ersatz der "r-Utilities", sondern auch ein universelles Werkzeug zur Absicherung sämtlicher Kommunikation. Der große Vorteil der SSH ist, wenn sie einmal eingesetzt wird, können dadurch auch andere Dienste abgesichert und transparent weitergenutzt werden. Es ist keine Neuinstallation oder ein Update notwendig. In unserem Beispiel kann der vorhandene POP-Server ohne Änderungen seinen Dienst zur Verfügung stellen. Auch die auf Clientseite laufende Software ist durch einfache Umstellung des POP-Servers auf "localhost" weiterhin nutzbar. Die Anwendungen werden also sozusagen ohne ihr Zutun von außen durch die SSH abgesichert. Durch die Einbindung von PGP-Schlüsseln kann der Benutzer schon vorhandene Schlüssel und Passwörter nutzen. Durch die universelle Einsetzbarkeit müssen keine anderen zusätzlichen Dienste wie Secure-POP oder HTTPS gestartet werden, sondern nur der SSH-Dienst. Dadurch wird der Rechner, der Dienste zur Verfügung stellt, weniger belastet.

9.2 Nachteile

Durch die Benutzung von SSH zur sicheren Übertragung der Daten wird ein weiteres Passwort benötigt, welches den SSH-Schlüssel sichert. Bei Verbindungsaufnahme müssen deshalb zwei Passwörter eingegeben werden, einmal das SSH-Schlüsselpasswort und einmal das POP-Passwort des Mailservers. Dies ist ein großer Nachteil des vorgestellten Verfahrens. Es ist möglich, dieses Verfahren zu vereinfachen, jedoch erhöht man damit gleichzeitig die Sicherheitsrisiken.

9.3 Grundsätzliche Gefahren

Der Benutzer ist eine grundsätzliche Gefahr bei Verschlüsselungssystemen. Der Algorithmus kann noch so stark und effizient, aber trotzdem nutzlos sein, wenn der Benutzer nicht ordnungsgemäß mit der Auswahl und Aufbewahrung der Passwörter und Schlüssel umgeht.

Vergibt ein Benutzer kein Passwort zu seinem Schlüssel, besteht die Gefahr, dass ein Hacker sich des geheimen Schlüssels bemächtigt. Dadurch kann er ohne Passwort auf das System des Benutzers zugreifen. Dies ist z.B. durch Fehler in Web-Browsern möglich. Durch fehlerhafte Implementierungen von Java oder Javascript können Dateien übertragen werden, hier z.B. die SSH-Schlüsseldateien. Weiterhin kann sich auf unsicheren Betriebssystemen wie Windows 9x jeder Benutzer die Schlüsseldateien kopieren.

Viel gefährlicher jedoch ist der unachtsame Umgang mit Passwörtern. Diese sollten Angriffen mit Wörterbüchern standhalten können, also nicht aus Namen, Autonummern, Telefonnummern usw. bestehen. Außerdem sollten sie keinesfalls auf Bürounterlagen oder Notizzetteln aufgeschrieben werden und nie in der Schublade oder Brieftasche verbleiben.

Bei Firmen sollten im Umfeld des Benutzers ausreichende Sicherheitsmassnahmen getroffen werden, wie z.B. Abschirmungen gegen die Bildschirmabstrahlung und Verhinderung des Zugriffs auf die Hardware. Ansonsten kann die Bildschirmabstrahlung durch Antennen aufgefangen und damit der Bildschirminhalt reproduziert werden. Über den Zugriff auf die Hardware können Manipulationen am System vorgenommen oder auch Abhöreinrichtungen installiert werden. Durch einen effizienten Virenschutz kann das Einschleusen von trojanischen Pferden verhindert werden.

Eine weitere Gefahr besteht an den Endpunkten der Kommunikation. Hier liegen die Daten immer unverschlüsselt vor. Sie können also an beiden Endpunkten abgehört werden, z.B. durch den Administrator. Email können auf dem Weg zum Mailserver abgehört werden und sollten deshalb verschlüsselt werden, z.B. mit PGP. SSH ist hierfür nicht vorgesehen, sondern nur für die sichere Übertragung des Passwortes und der Daten von und zum Mailserver.

10. Fazit

Die universelle Einsetzbarkeit macht die SSH zu einer interessanten Lösung des sicheren Mailabrufs. Vorhandene Systeme können damit transparent abgesichert werden. Es sind keine neuen Dienste notwendig. Jedoch wird ein sorgfältiger Umgang mit den Authentifikationsmitteln (Schlüssel, Passwort) vom Benutzer erwartet. Der Nachteil der vorgestellten Vorgehensweise gegenüber Secure-POP mit SSL ist, dass der Benutzer zwei Passwörter zum Abruf benötigt. Einmal das SSH-Passwort und einmal das POP-Passwort. Er erhält aber eine abgesicherte Verbindung ohne eine Neuinvestition in Software tätigen zu müssen, abgesehen von SSH.

Anhang A:

SSH2-Befehlsübersicht und Konfigurationsdateien

A.1 SSH2-Befehlsübersicht

ssh2	Secure-Shell-Client, dient zur Verbindungsaufnahme ähnlich "rsh"
sshd2	Secure-Shell-Dämon, stellt Serverdienst zur Verfügung
sftp2	Secure-Ftp-Client, dient zur Dateiübertragung
sftp-server2	Secure-Ftp-Server, Serverdienst zur Dateiübertragung
scp2	Secure-Copy-Client, ermöglicht Dateiaustausch ähnlich "rcp"
ssh-keygen2	Secure-Keygenerator, dient zur Erzeugung der RSA-Schlüssel
ssh-add2	Schlüsselaufnahme für den Authentication Agent
ssh-agent2	Authentication Agent zur Schlüsselverwaltung
ssh-askpass2	X11 Utility zur komfortablen Abfrage des Schlüsselpasswortes
ssh-signer2	Signiert Authentifizierungspakete
ssh-probe2	Durchsucht Netzwerk nach SSH2-Servern
ssh-pubkeymgr	Erstellt SSH2-Schlüssel und Konfigurationsdateien
ssh-chrootmgr	Stellt die Möglichkeiten einer restriktiven Shell zur Verfügung

A.2 Konfigurationsdateien

$HOME/.ssh2/authorization	Konfigurationsdatei zur Einwahl
$HOME/.ssh2/identification	Konfigurationsdatei zur Identifikation
$HOME/.ssh2/ssh2_config	Konfigurationsdatei des SSH2-Client
$HOME/.ssh2/hostkeys	Verzeichnis mit öffentlichen Host-Keys der SSH2-Server
$HOME/.ssh2/schlüsselname	Geheimer SSH2-Schlüssel
$HOME/.ssh2/schlüsselname.pub	Öffentlicher SSH2-Schlüssel
$HOME/.fetchmailrc	Konfigurationsdatei von "fetchmail"
/etc/ssh2/sshd2_config	Konfigurationsdatei des SSH2-Server
/etc/ssh2/ssh2_config	Globale Konfigurationsdatei des SSH2-Client
/etc/ssh2/hostkey	Geheimer Host-Key des SSH2-Server
/etc/ssh2/hostkey.pub	Öffentlicher Host-Key des SSH2-Server

Anhang B:

SSH2: Wichtige Befehle mit Optionen

Im folgenden Kapitel werden die am häufigsten verwendeten Befehle mit ihren Optionen vorgestellt. Beschreibungen und andere Optionen können der Online-Hilfe unter Windows oder den "Manual-Pages" unter Unix entnommen werden.

B.1 Secure Shell

Der Secure Shell Client wird zur Einwahl auf einen entfernten Rechner benutzt. Auf diesem können dann Kommandos ausgeführt werden.

Syntax:
ssh2 [Optionen] entfernterRechner [auszuführenderBefehl]

Optionen:

-l user	Diesen Benutzernamen bei der Einwahl verwenden.
-f	Nach der Einwahl als Hintergrundprozess starten.
-p port	Portnummer des SSHD2-Dämon 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. (für Benutzer sind nur Portnummern größer 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. (für Benutzer sind nur Portnummern größer 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.

Beispiel: Einwahl auf Rechner "mail-stl" mit Portnummer "1234", Benutzername "olitz", eingeschalteter Kompression, Start als Hintergrundprozess, lokaler Portumleitung auf "mail-stl".
ssh2 -f –p 1234 –l olitz +C -L 1100:mail-stl:110 mail-stl

B.2 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.

Syntax: scp2 [Optionen] [Benutzer@]Rechner[#Port]:]Datei [Benutzer@]Rechner[#Port]:]Datei oder Verzeichnis

Optionen:

-r	Unterverzeichnisse einschließen
-u	Quelldateien löschen (move)

Beispiel: Kopieren der auf dem Rechner stl-d-04 befindlichen Datei "Vorlesung.doc" in das aktuelle Verzeichnis.
scp2 olitz@stl-d-04:Vorlesung.doc .

B.3 Secure Shell Dämon

Der Secure Shell Dämon ermöglicht die Einwahl eines Benutzers eines anderen Rechners über eine verschlüsselte Verbindung.

Syntax: sshd2 [Optionen]

Optionen:

-p port	Anfragen werden auf dieser Portnummer entgegengenommen.
-i	Der SSHD2-Dämon wird von inetd gestartet (nicht empfohlen).
-f	Ort der Konfigurationsdatei.

Beispiel: Start des SSH2-Dämon mit der Konfigurationsdatei unter "/etc/ssh2/sshd2_config".
sshd2 –f /etc/ssh2/sshd2_config

B.4 Secure File Transfer Client

Der Secure FTP Client bietet den gleichen Funktionsumfang wie ein gewöhnlicher FTP Client, die Daten werden jedoch verschlüsselt übertragen.

Syntax: sftp2 [Optionen] [user@]host[#port]

Beispiel: Aufbauen einer gesicherten FTP-Verbindung zum Rechner "stl-d-04".
sftp2 stl-d-04

Anhang C:

Konfigurationsdatei des SSH-Dämon

Die Konfigurationsdatei des SSH-Dämon befindet sich unter "/etc/ssh2/sshd2_config". Einige wichtige Optionen werden in nachfolgender Tabelle vorgestellt. Weitere Optionen können den Man-Pages entnommen werden. Wenn nicht anders angegeben ist dem Schlüsselwort das Argument "yes" oder "no" mitzugeben.

AllowAgentForwarding	Weiterleitung von "Authentification Agent"-Verbindungen
AllowedAuthentications	Liste der erlaubten Authentifikationsarten in gewünschter Reihenfolge, z.B.: publickey, password, hostbased, kerberos
AllowTcpForwarding	Weiterleitung von TCP-Verbindungen
AllowUsers	Einwahl folgender Benutzer erlauben, wenn nicht in DenyUsers enthalten, wie z.B.: olitz, sj*, s[:isdigit:]##, s(jl|amza)
Ciphers	Verschlüsselungsalgorithmus, z.B. 3DES, IDEA, Blowfish, usw.
DenyHosts	Einwahl von den folgenden Hostnamen verbieten, z.B.: evil.org, aol.com
DenyUsers	Einwahl der folgenden Benutzer verbieten, wie z.B.: skuuppa, warezdude
ForcePTTYAllocation	Zuweisung eines TTY, auch wenn ein Kommando übergeben wird
ForwardX11	Weiterleitung der X11-Anzeige
ListenAddress	Server bindet sich an diese IP-Adresse (bei Multihomed Hosts)
MACs	Message Authentication Code, z.B. hmac-sha1, hmac-md5, usw.
MaxBroadcastsPerSecond	Anzahl der max. angenommen UDP-Broadcasts pro Sekunde
MaxConnections	Maximale Anzahl entgegengenommener Verbindungen
PasswordGuesses	Anzahl erlaubter Versuche für die Eingabe des Passwortes (Default: 3) : 1,2,3, usw.
PermitRootLogin	Einwahl des Administrators (root) mit SSH erlauben
PrintMotd	Ausgabe der Datei "/etc/motd" bei interaktivem Login
RandomSeedFile	Name des RandomSeed-Files
RequiredAuthentications	Liste der auf jeden Fall zu durchlaufenden Authentifikationsarten, wie z.B.: publickey, password
RequireReverseMapping	Schlägt DNS fehl, weise Verbindung ab (yes) Schlägt DNS fehl, prüfe ob IP-Adresse (Remote) erlaubt ist (no)
Ssh1Compatibility	Kompatibilität zu SSH1 (dafür muss SSH1 installiert sein)
Ssh1Path	Zugriffspfad für SSH1
Subsystem-sftp	Zugriffspfad des Secure File Transfer Client
SyslogFacility	Facility Code beim Loggen von Meldungen, z.B. Daemon, Auth
VerboseMode	Meldungen zur Fehlersuche ausgeben

Folgende Konfiguration der Datei "sshd2_config" wäre auf dem Rechner "mail-stl" möglich:

```
## General settings
        VerboseMode                             no
        AllowCshrcSourcingWithSubsystems        no
        ForcePTTYAllocation                     no
        SyslogFacility                          AUTH

## Network settings
        Port                    22
        ListenAddress           0.0.0.0
        RequireReverseMapping   no
        MaxBroadcastsPerSecond  0

## Crypto settings
        Ciphers                 AnyCipher
        MACs                    AnyMAC

## User settings
        PrintMotd               yes
        CheckMail               yes
        UserConfigDirectory     "%D/.ssh2"

## Public key configurations
        HostKeyFile             hostkey
        PublicHostKeyFile       hostkey.pub
        RandomSeedFile          random_seed
        IdentityFile            identification
        AuthorizationFile       authorization
        AllowAgentForwarding    yes

## Tunneling configurations
        AllowX11Forwarding      no
        AllowTcpForwarding      yes

## Authentication methods
        PasswordGuesses         3
        AllowedAuthentications  publickey

## User restrictions
        PermitRootLogin         no

## subsystem definitions
        subsystem-sftp          /SSH-Installationspfad/bin/sftp-server2
```

Anhang D:

Ergänzung der Internetseite des Systemtechniklabors

Auf der Diskette, die der Fallstudie beiliegt, ist die Internetseite des Systemtechniklabors beispielhaft um zwei Kapitel erweitert worden.

- Internetzugang via SSH (Secure Shell)
  Das Kapitel beschreibt die Einrichtung des SSH-Zugangs.
- Sicherer Mailabruf mit SSH
  Das Kapitel beschreibt die Vorgehensweise zum Abruf der Email mit SSH.

Weiterhin ist die Fallstudie im HTML-Format enthalten sowie das Skript "ssh-htwkeygen".

Abbildungsverzeichnis

Literaturverzeichnis

1. Anne Carasik, Steve Acheson
   SSH-FAQ (Häufig gestellte Fragen), Revision 1.3, Juni 2000
   http://www.employees.org/~satch/ssh/faq

2. SSH Communications Security Corp.
   Fredrikinkatu 42
   FIN-00100-Helsinki
   http://www.ssh.com

3. OpenBSD Project (http://www.openbsd.org)
   OpenSSH
   http://www.openssh.com

4. FreeSSH Project
   http://www.freessh.org

5. The Internet Engineering Task Force (IETF)
   http://www.ietf.org

6. SANS Institute
   5401 Westbard Ave. Suite 150
   Bethesda, MD 20816
   http://www.sans.org

7. Eric S. Raymond <esr@snark.thyrsus.com>
   Fetchmail FAQ
   http://www.tuxedo.org/~esr/fetchmail/fetchmail-FAQ.html

8. Eric S. Raymond <esr@snark.thyrsus.com>
   Fetchmail, POP-Client
   http://www.tuxedo.org/~esr/fetchmail/

9. Andrea Rizzi <rizzi@kde.org>
   KSSH
   http://www.geocities.com/bilibao/kssh.html

10. Netscape
    Netscape Communicator
    http://www.netscape.com

11. David Harris
    Pegasus Mail, POP-Client
    http://www.pegasus.usa.com