SSH 101 - Mehr als nur telnet mit crypto

bios

GPN

now()

Überblick - Was machen wir heute

- Geschichte
- Einsatzszenarien
- Protokoll
  - Transport Layer
  - Authentification Layer
  - Connection Layer
- Konfiguration
- Clientkonfig
  - PubKey
  - SSH Agent
  - Port Forwarding
- Dunkle Magie

Disclaimer - Software & Versionen

- Verwendete Software: OpenSSH. One and only!!
- Protokollversionen: v2.

Es war einmal...

- Internet $\approx$ 23 Rechner
- Jeder vertraut Jedem
- Daher:
  - rlogin
  - telnet
  - rsh
- Keine Verschlüsselung
- Keine Authentifizierung
- Klartextpasswort übers Netz - Warum nicht?

Es war einmal. . .

- Einbruch in Uni Helsinki
- Tatu Ylönen erste Version 1995
- Firmengründung
- OpenSSH in OpenBSD 2.6
- aktuell: OpenSSH 6.0

Einsatzszenarien

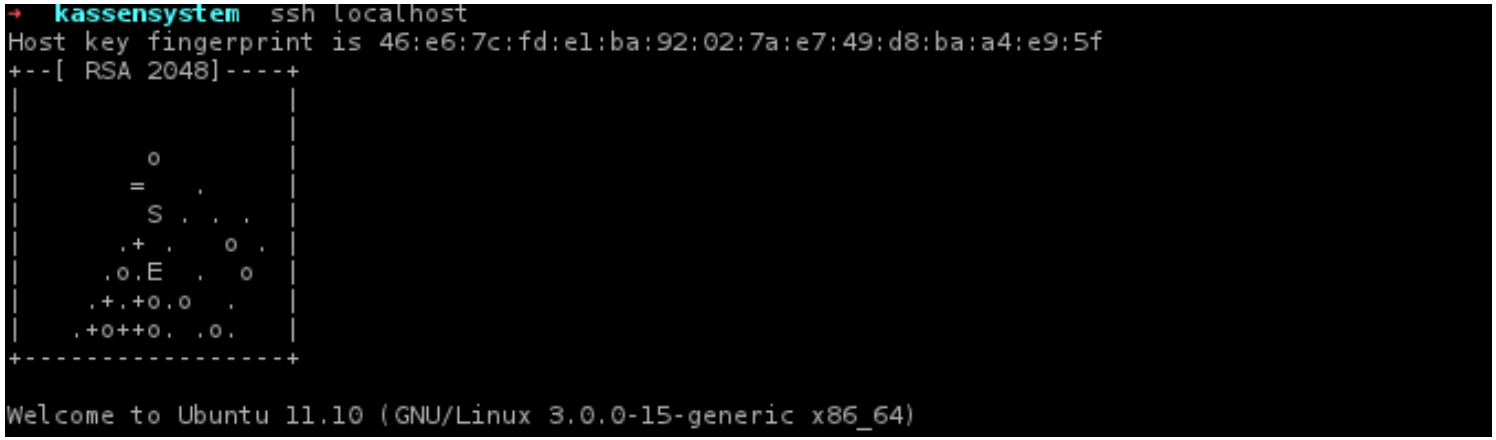
- gesicherte Kommunikation
- Remoteshell
- Umleiten von Streams
- Datentransfer
- Nutzerauthentifizierung

Protokollbestandteile

- Transport Layer (RFC 4253)
- Authentification Layer (RFC 4252)
- Connection Layer (RFC 4254)

Transport Layer

- Key-Exchange
- Verschlüsselung
- Kompression
- Integritätsprüfung



Authentification Layer

- Verschiedene Methoden zur Benutzerauthentifizierung
  - Password
  - Challenge Response (Keyboard-interactive)
  - public key
  - Verschiedenes über GSSAPI, u.a. Kerberos

Authentification Layer

- Password
  - Client fragt nach PW
  - Server antwortet nur auf Anfragen

Authentification Layer

- Public Key
  - Verschiedene Algorithmen
  - DSA, RSA gebräuchlich
  - x.509 theoretisch

Authentification Layer

- Keyboard Interactive
  - Server schickt Prompts
  - Client fragt ab und schickt Antwort zurück
  - OTP-Authentifizerung möglich:
    - S/Key
    - Hardwaretokens: Yubikey, SecureID

Authentification Layer

- GSSAPI
  - Anbindung an externe Provider
  - Kerberos, NTLM
  - Single-Sign-On möglich

Connection Layer

- Aufteilung in Kanäle
  - Channels
  - Channel requests
  - Global requests
- Jede SSH-Session kann mehrere Kanäle enthalten
- Auch mehrere Kanäle eines Typs sind möglich

Kanaltypen

- Verschiedene Standardchannel
  - Shell - Terminal, SFTP, Programmausführung
  - Direct-TCPIP - Client-to-Server
  - Forwarded-TCPIP - Server-to-Client
- Jede SSH-Session kann mehrere Kanäle enthalten
- Auch mehrere Kanäle eines Typs sind möglich

Kanaltypen

- Verschiedene Standardchannel
  - Shell - Terminal, SFTP, Programmausführung
  - Direct-TCPIP - Client-to-Server
  - Forwarded-TCPIP - Server-to-Client
- Jede SSH-Session kann mehrere Kanäle enthalten
- Auch mehrere Kanäle eines Typs sind möglich

Kanäle multiplexen

- Mehrere (gleiche) Kanäle sind möglich
- Session Reuse
  - Agent Forwarding, X11, Portforwarding, VPN
  - Datentransfer
  - Mehrere Shells auf den Rechner
- Nur einmal Verbindungsausbau nötig
- ControlMaster auto

Konfiguration

- Server
  - man sshd_config
  - /etc/sshd_config
- Client
  - man ssh_config
  - Optionen für ssh
  - /etc/ssh/ssh_config

Client Konfiguration

- Match auf verschiedene Angaben
- Konfig je nach
  - User
  - Hostadress
  - Group
  - (Server)adress

Client - Match Beispiel

- lokaler User bios
- Arbeitsnutzer fnord
- SSH Agent nutzen
- Match auf Hostnamen mit * :
  Host *.foo.bar Username fnord ForwardAgent yes

Client - Connection

- Ohne Einschränkung / Match auf alles:

```
VisualHostKey yes

Host *
ControlPath ~/.ssh/master-%r@%h:%p
ControlMaster auto
```

Pubkey Anmeldung

- Vorteil: Passwortloser Login
- Mit zsh: Autocompletion!!1!
- ssh-keysgen -b 2048 -t dsa
- Auf Ziel: /.ssh/authorized_keys
- Fingerprint: ssh-keygen -l -f PFAD
- in authorized_keys:
  - Command, ForceCommand
  - From - Patternmatching
  - no-port-forwarding,no-agent-forwarding
  - no-X11-forwarding, no-pty
- tunnel="0",command="sh /etc/netstart tun0" ssh-rsa AAAA...==

Pubkey Anmeldung

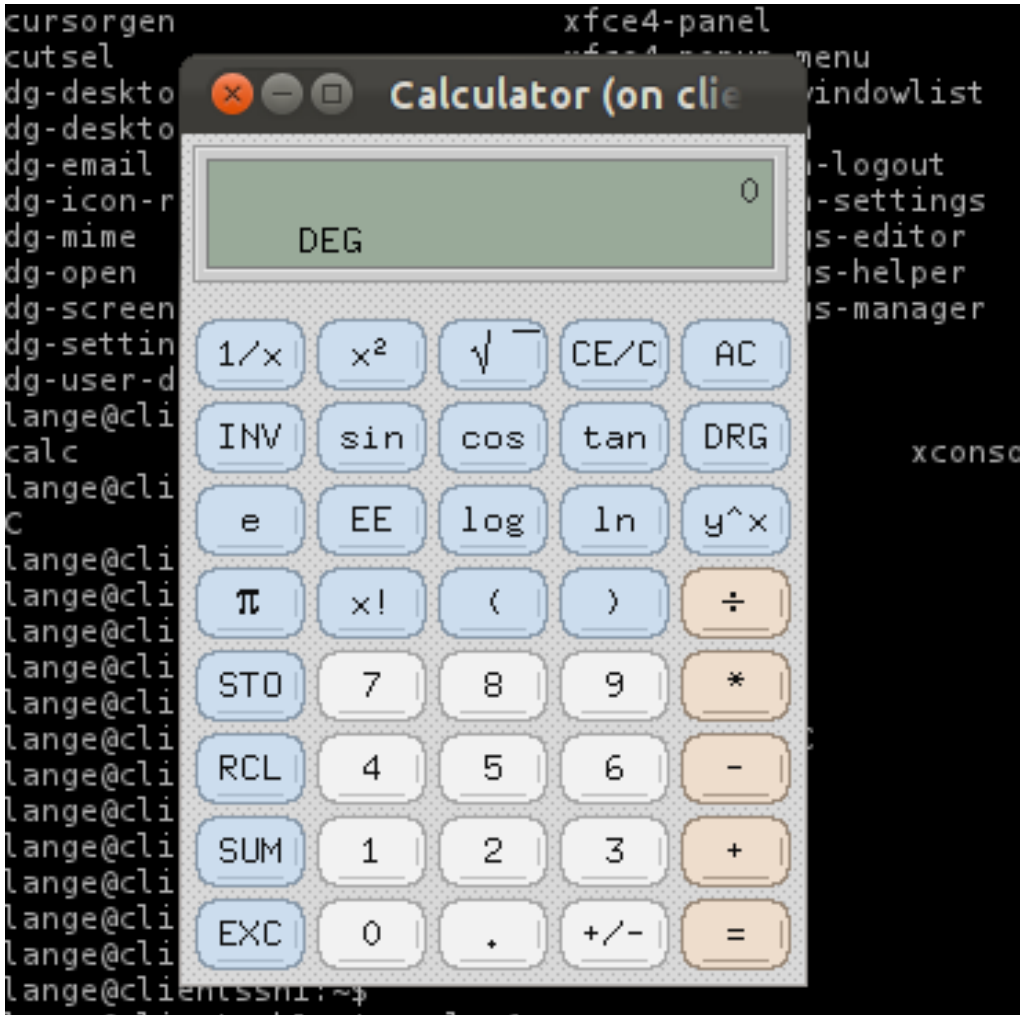
- Schutz gegen Plattenklau:
  - Plattencrypto
  - Besser/universaler: Passwort für Keyfile
- SSH-Agent / Keychain

SSH-Agent

- Läuft im Hintergrund
- setzt SSH_AGENT_PID, SSH_AUTH_SOCK
- ssh-add für neue Keys
- ssh-add -D zum Keys entfernen
- SSH-Agent / Keychain
- Aktivieren:
  - ForwardAgent yes
  - ssh -A
- schlecht bei unbekanntem System

X Forward

- $ssh -X user@server
- Programm auf Server starten
- warten ;)
- freuen



Ports forwarden - Lokal lauschen

- $ssh -L 8080:zielserver:80 proxysshserver
- Lokaler Port
- Zielserver, Portforwarding
- Anfrage - localhost:8080 - proxyssh - Zielserver:Port
- Als Service für andere: -g

Ports forwarden - Remote lauschen

- $ssh -R 2222:lokaler.server:22 ssh.server
- Anfrage - sshserver:2222 - mein.rechner - lokaler.server:22
- Für andere: GatewayPorts yes

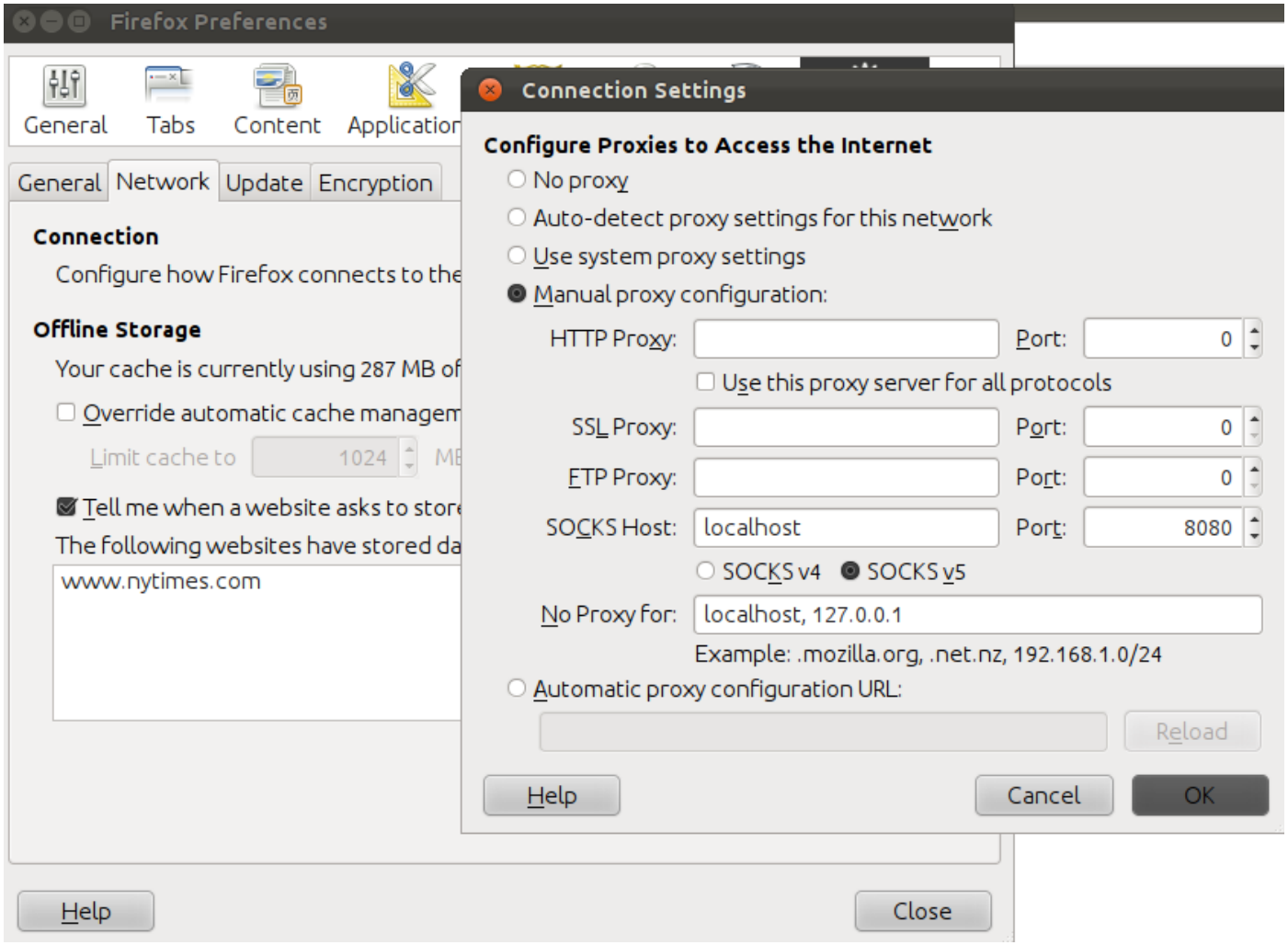
Ports forwarden

- Geht leider nur für TCP
- UDP mit Tricks:
  - SSH: ssh -L 5353:localhost:5353 user@server
  - Server: socat tcp4-listen:5353,reuseaddr,fork UDP:8.8.8.8:53
  - Client: socat -T15 udp4-recvfrom:53,reuseaddr,fork tcp:localhost:5353

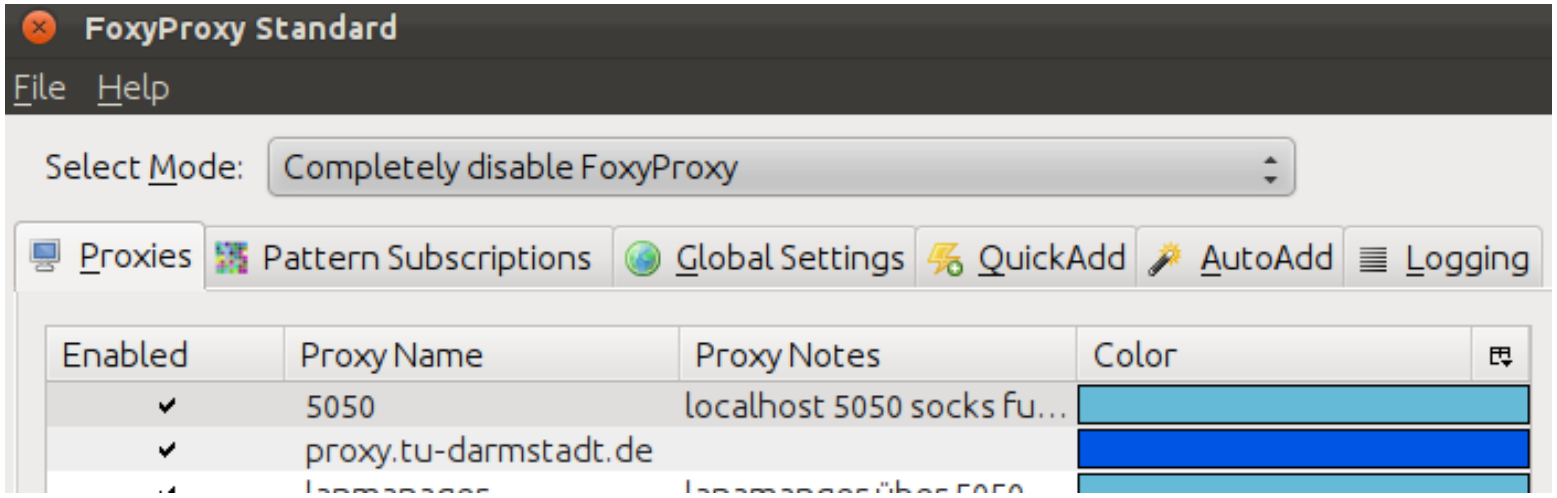
Weiteres Tunneln

- SOCKS-Proxy
  - ssh -D 8080 usermein.proxy.server
  - In Browser/Clients eintragen

Weiteres Tunneln



Weiteres Tunneln



Weiteres TUNneln

- TUN-Device
  - ssh -w any user@mein.proxy.server
  - Danach Netz konfigurieren
- SSH als Netcat
- ssh -W zielserver:port user@server
- SSH over SSH

Raustunneln über Proxy

- SSH auf 443
- ssh -P mein.proxy.server user@mein.ziel.server
- Proxy zu intelligent?
  - ProxyCommand corkscrew mein.proxy.server 8080
  - ProxyCommand proxytunnel -p proxyserver:8080 -u proxyuser -s proxypasswort -d

Dateien über SSH schieben

- Copy&Paste
- scp
- sftp
- rsync!!1!
- Dateien durch SSH pipen

```
tar zcf - FOO | ssh user@server 'tar zxf -'
ssh user@server 'tar zcf - BAR' | tar zxf -
tar zcf - BAR | ssh user@server 'cat - > stuff.tar.gz'
```

sshuttle

- Keine Lust auf tun-Configs?
- Kein root auf Remote Server
- Lokal iptables oder ipfw
- “where transparent proxy meets VPN meets ssh”
- Bitte mittippen:

```
git clone git://github.com/apenwarr/sshuttle
./sshuttle -r user@server  0.0.0.0/0 -vv --dns
```

- Keine Installation auf Server
- Dunkle Magie!!

Mobile Shell

- mosh
- Muss auf Server installiert werden
- Baut SSH Verbindung auf zum Passwort austauschen
- UDP Pakete -¿ Ideal zum Zug fahren
- Zustandsbasiert

```
git clone https://github.com/keithw/mosh
cd mosh
./autogen.sh
./configure
make
```

- mosh user@server
- Dunkle Magie!!

Überlebt!

- Fertig!
- Mist erzählt?
- Ideen? Was vergessen?
- Propaganda für euer Tool fehlt?
- Mail: bios@chaos-darmstadt.de