Advanced Usage of OpenSSH

Sean Cody

MUUG Presentation

September 9, 2008

12:49:24 AM

Who am I?

- Senior Systems Administrator for Prime Focus VFX Services (formerly Frantic Films VFX).
- Editor at The OpenBSD Journal (undeadly.org).
- Practical Paranoid
  - Gets claustrophobic in closed networks.
  - Enjoys a good challenge.

What we'll cover.

- Brief introduction to the OpenSSH world.
- A look at a few of some of the more esoteric but interesting features of OpenSSH.
- Getting the most out of your OpenSSH daemon.
- Some cute usage of OpenSSH to subvert the "real world" and survive hostile networks.

What I’ll Assume

- You’ve used a CLI before.
- You can read man pages.
- You have a good understanding of the fundamentals of ‘The Internet.’
- You’ll tell me when I screw up?

OpenSSH

OpenSSH

www.OpenSSH.com
RLOGIN
1982-99
LIVED LIFE
WITH JOY
TELNET
1971-99
WONT CRYPT
RSH
1982-99
++
Putting an end to unencrypted network logins

OpenSSH

- A suite of cryptographically secured connectivity tools.
- Comes in two flavours.
  - OpenSSH
  - OpenSSH-portable
- A crypto powered hammer in a world full of rusty nails.

Flavours

- OpenSSH-portable
  - Follows OpenSSH but contains patches to work on a variety of non BSD operating systems.
    - Like Linux, AIX, HPUX, Windows
  - Sometimes referred to as OpenSSH+PAM.
- Sometimes doesn’t get all the features of the parent project but tries really hard.

Problem

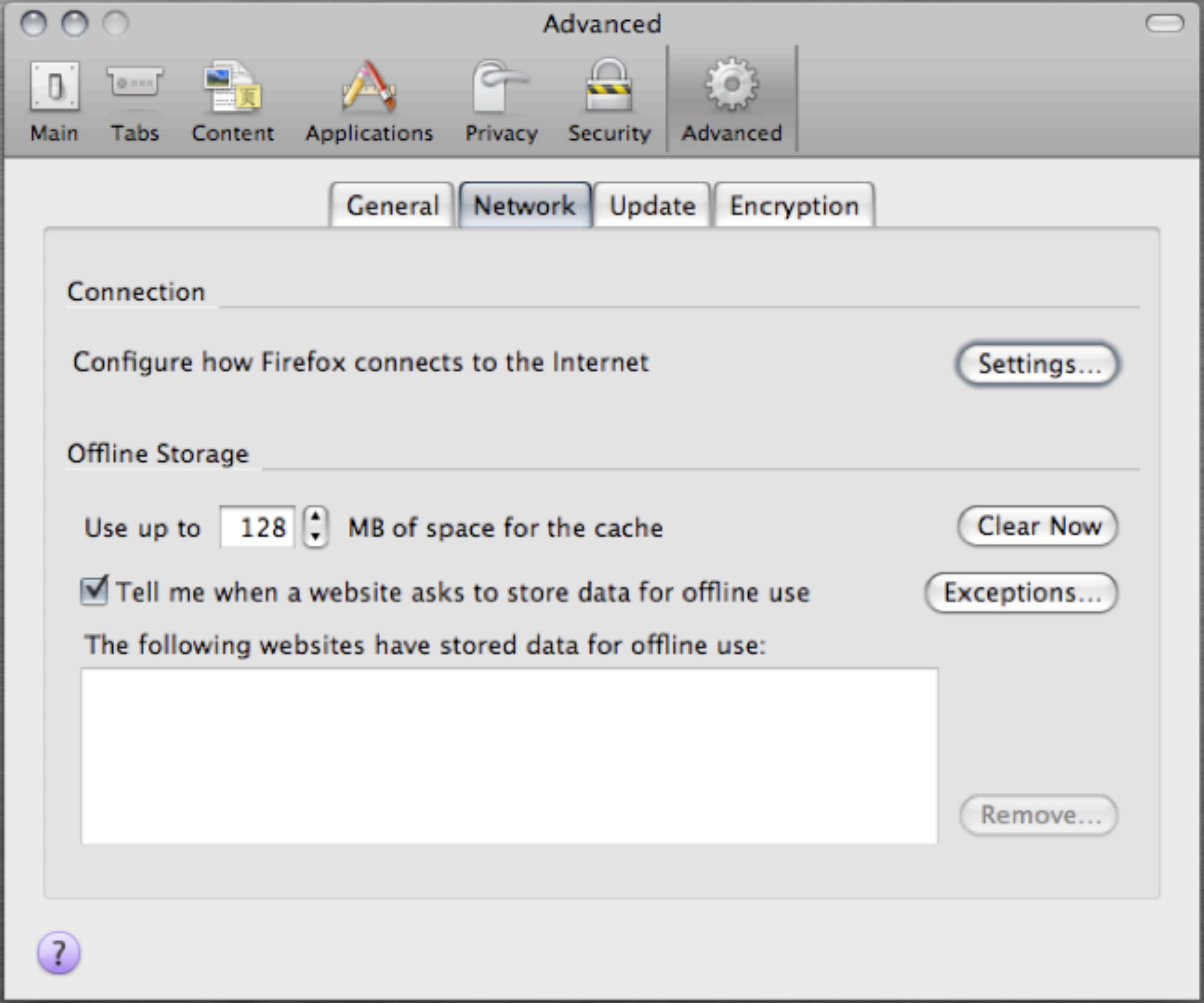
- If you decide to use a machine in a hostile network, how can you set it up to be useful yet still protect yourself from attacks and packet sniffing?
- ie. DefCon, badly setup conference, some random sketchy coffee shop/hot-spot.

Solution

- OpenSSH client contains a built in, on-demand SOCKS proxy!

-
  ```
  ssh -D1234 -n user@host
  ```
- Tell your web browser to use localhost:1234 as your proxy.
  - Bonus points for tunneling DNS over said proxy.
- This works for any application that can talk with a SOCKS proxy.

Solution (FireFox)



Connection

Configure how Firefox connects to the Internet

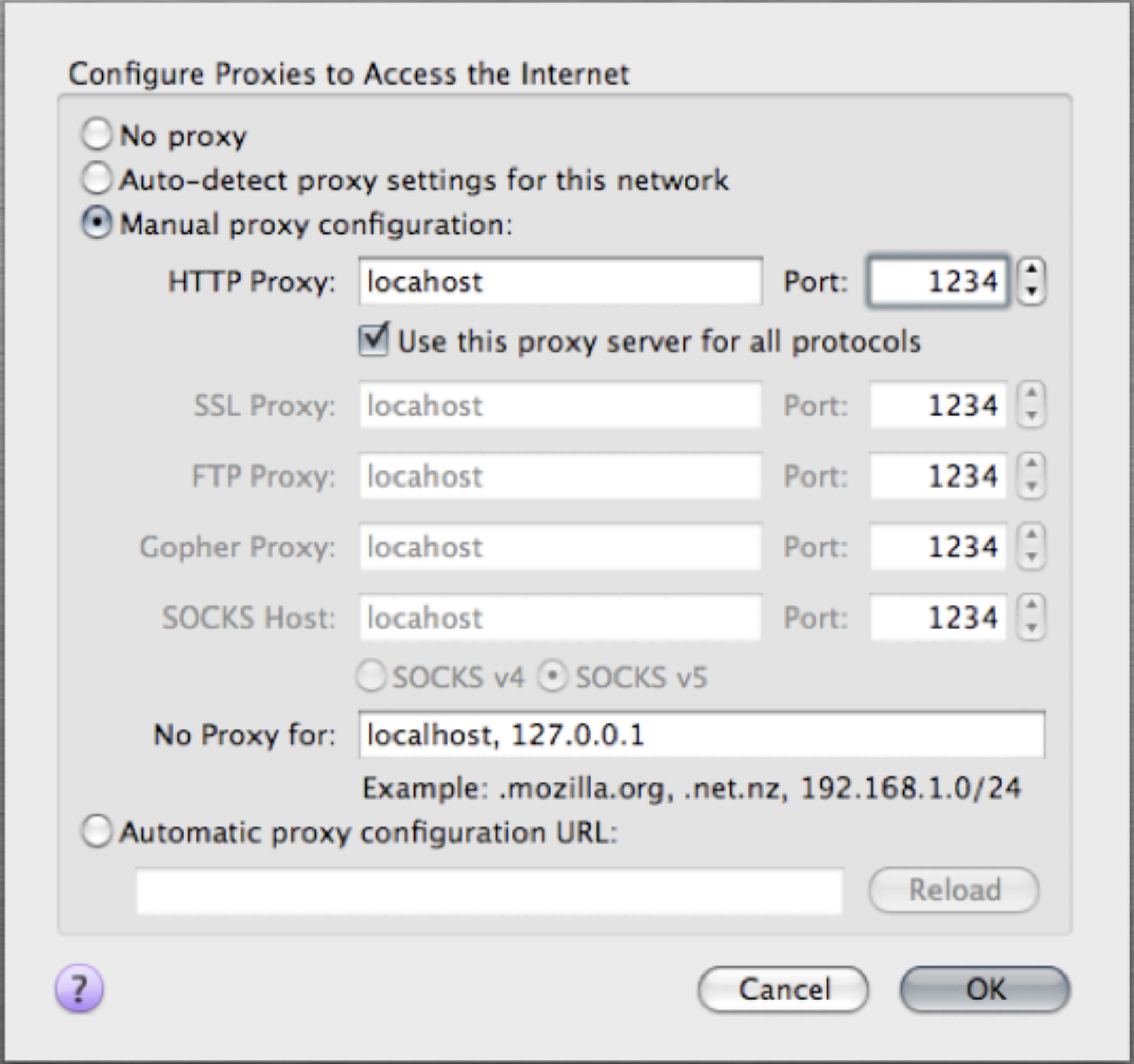
Offline Storage

Use up to 128 MB of space for the cache

☑ Tell me when a website asks to store data for offline use

The following websites have stored data for offline use:

Solution (FireFox)



Configure Proxies to Access the Internet

○ No proxy

○ Auto-detect proxy settings for this network

● Manual proxy configuration:

HTTP Proxy: locahost Port: 1234

☑ Use this proxy server for all protocols

SSL Proxy: locahost Port: 1234

FTP Proxy: locahost Port: 1234

Gopher Proxy: locahost Port: 1234

SOCKS Host: locahost Port: 1234

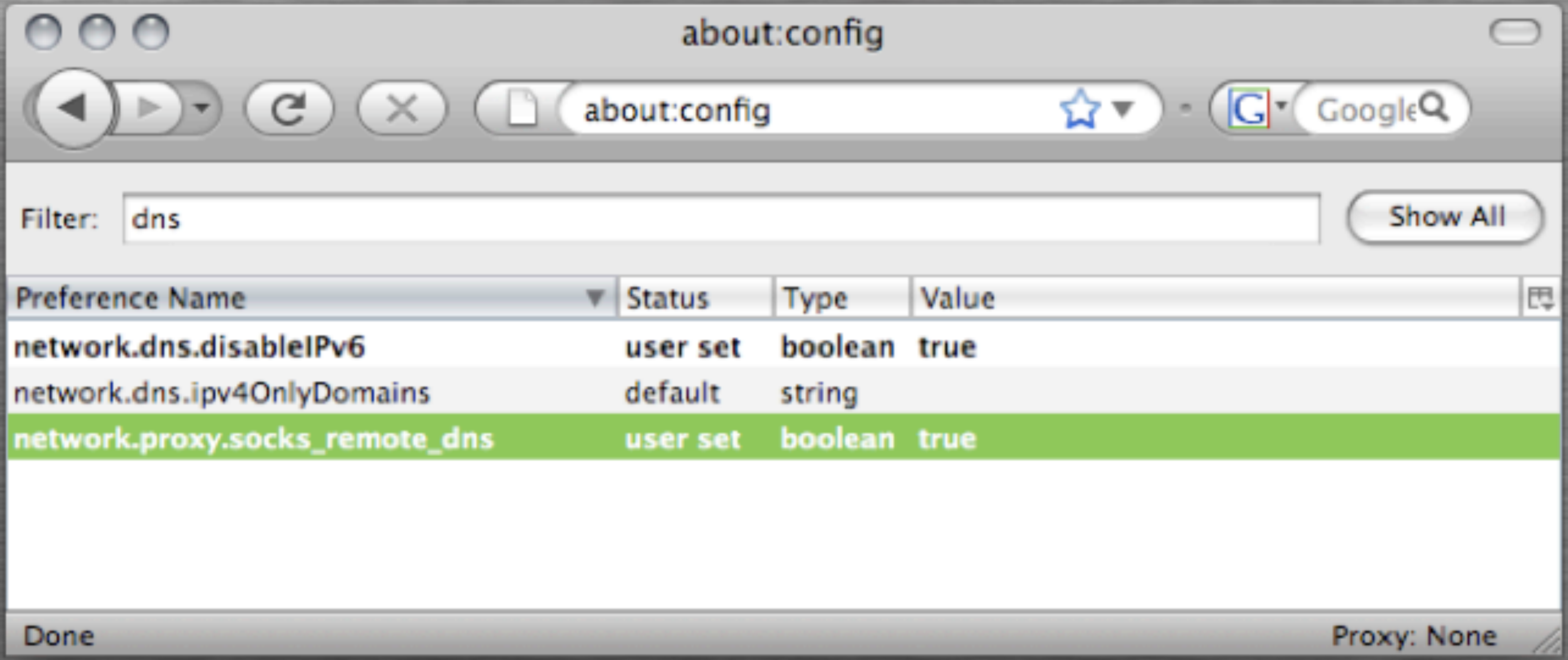
○ SOCKS v4 ● SOCKS v5

No Proxy for: localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

○ Automatic proxy configuration URL:

Solution (FireFox)



Filter: dns

Preference Name	Status	Type	Value
network.dns.disableIPv6	user set	boolean	true
network.dns.ipv4OnlyDomains	default	string	
network.proxy.socks_remote_dns	user set	boolean	true

Solution (FireFox)

- The “SwitchProxy” and “ProxyButton” make this configuration painless.
- Using a nice SSH-Agent will make the connections less painful.
  - On the mac there is SSHKeyChain
  - On other *nix hosts:
    - echo secure_browsing.sh > ssh -n -D8888:user@host && firefox &
    - use ssh-agent(1)

Problem

Problem

- In a low bandwidth/high-latency environment, how do you handle multiple connections to a remote server?

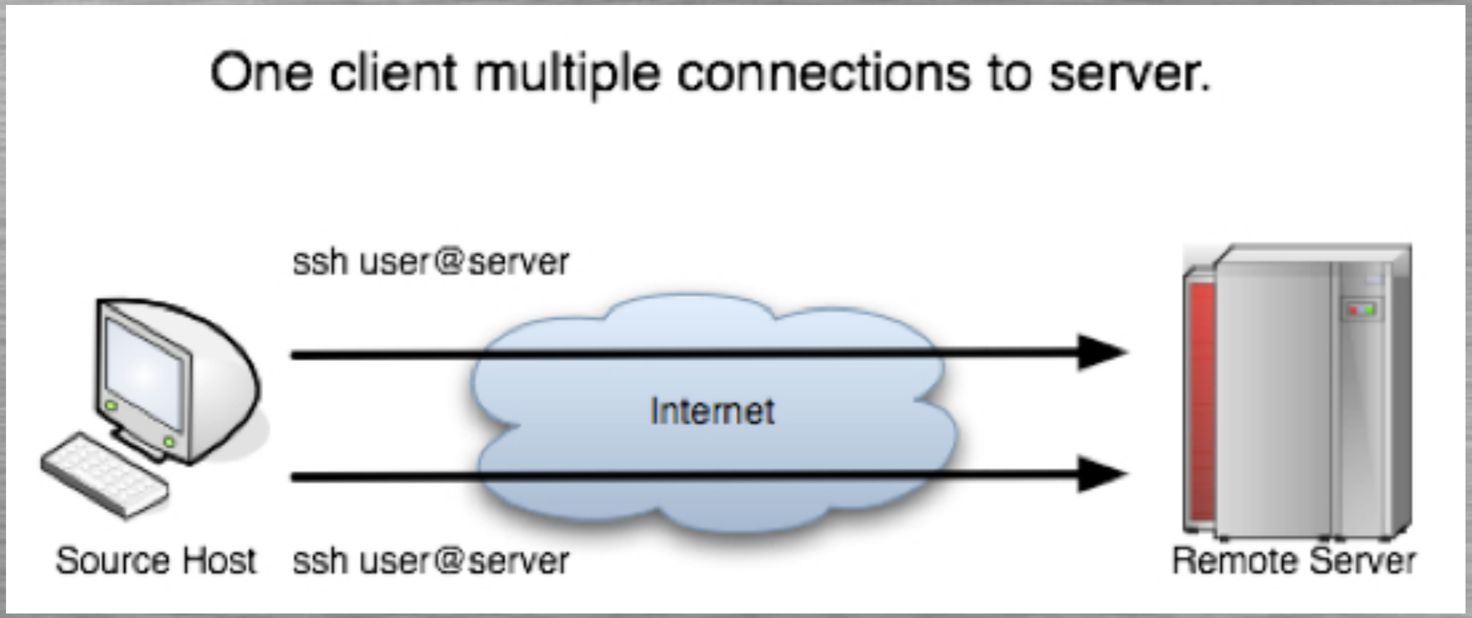
Problem

- In a low bandwidth/high-latency environment, how do you handle multiple connections to a remote server?
- The remote server also happens to be resource sensitive.

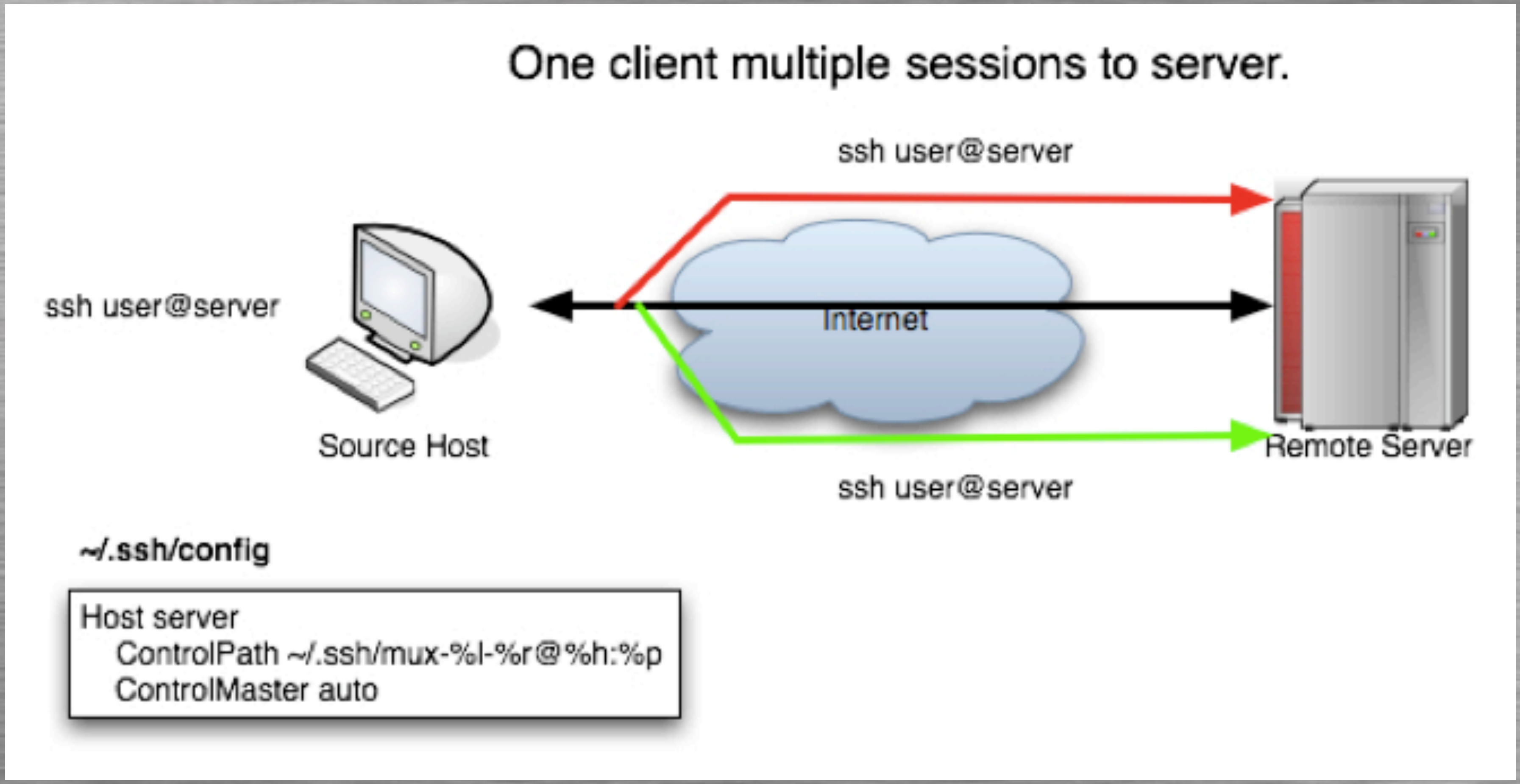
Solution

- We can use a single multiplexed session!
- One TCP socket, multiple sessions over said socket.

Solution


One client multiple connections to server.
ssh user@server
Internet
Source Host
ssh user@server
Remote Server

Solution



One client multiple sessions to server.

~/.ssh/config

```
Host server
   ControlPath ~/.ssh/mux-%l-%r@%h:%p
   ControlMaster auto
```

Problem

Problem

- How do you allow remote access to an internal subversion repository?

Problem

- How do you allow remote access to an internal subversion repository?
- Security and containment is important.

Problem

- How do you allow remote access to an internal subversion repository?
- Security and containment is important.
- How about restricting per user access to specific repositories?

Solution

- Use authorized_keys, with forced commands and a few extra options to limit ‘fringe utility.’
- authorized_keys file format:
    - OPTIONS TYPE KEY COMMENT
        - eg. no-pty ssh-rsa AAAA....a== sample

Solution - Server

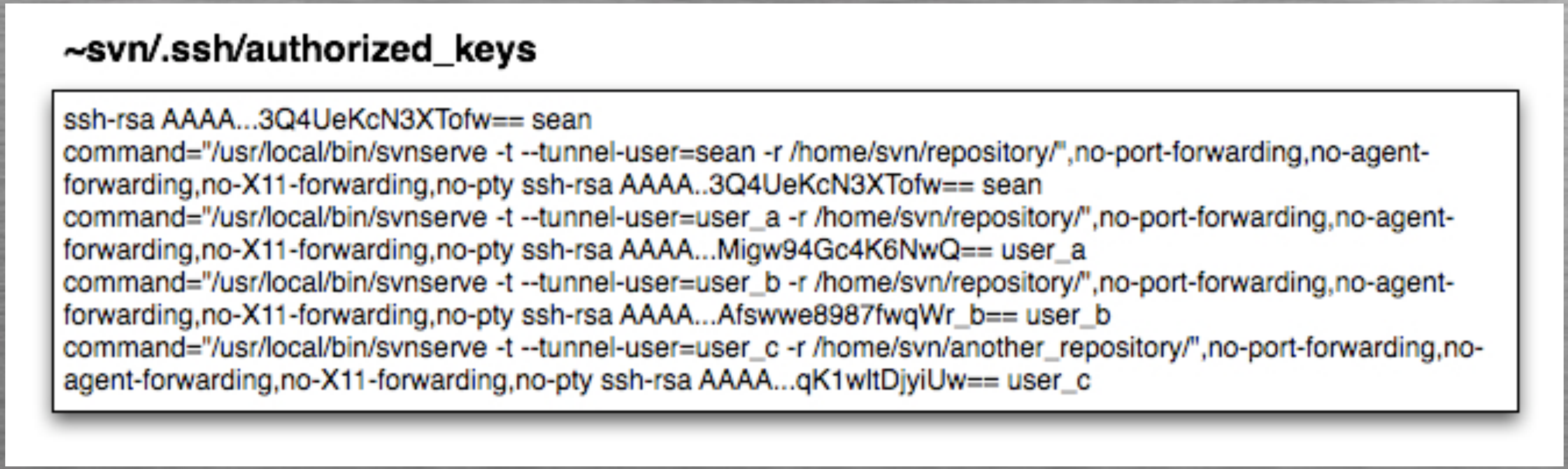
- Add a user called ‘svn’ whose home is /home/svn/
- su - svn

  mkdir -p ~svn/.ssh/

  mkdir -p ~svn/repository/

  touch ~svn/.ssh/authorized_keys

  svnadmin create ~svn/repository/

- 

~svn/.ssh/authorized_keys

```
ssh-rsa AAAA...3Q4UeKcN3XTofw== sean
command="/usr/local/bin/svnserve -t --tunnel-user=sean -r /home/svn/repository/",no-port-forwarding,no-agent-
forwarding,no-X11-forwarding,no-pty ssh-rsa AAAA..3Q4UeKcN3XTofw== sean
command="/usr/local/bin/svnserve -t --tunnel-user=user_a -r /home/svn/repository/",no-port-forwarding,no-agent-
forwarding,no-X11-forwarding,no-pty ssh-rsa AAAA...Migw94Gc4K6NwQ== user_a
command="/usr/local/bin/svnserve -t --tunnel-user=user_b -r /home/svn/repository/",no-port-forwarding,no-agent-
forwarding,no-X11-forwarding,no-pty ssh-rsa AAAA...Afswwe8987fwqWr_b== user_b
command="/usr/local/bin/svnserve -t --tunnel-user=user_c -r /home/svn/another_repository/",no-port-forwarding,no-
agent-forwarding,no-X11-forwarding,no-pty ssh-rsa AAAA...qK1wltDjyiUw== user_c
```

Solution - Client

- Each client must setup their ssh key identity and their public key must be the key in the server’s authorized_keys file.
- Connecting to the repository is as easy as

  -
    ```
    svn co svn+ssh://user_a@server/path_to_repository/
    ```

  -
    ```
    env SVN_SSH="ssh -i /Users/sean/.ssh/svn" svn co \
        svn+ssh://user@server/path_to_repository/
    ```

Problem

Problem

- How do you provide desk side support to a user who is on the other side of the world on a foreign network ?

Problem

- How do you provide desk side support to a user who is on the other side of the world on a foreign network ?
- The user is also NAT’d (possibly multiple times) behind some random firewall (or firewalls).

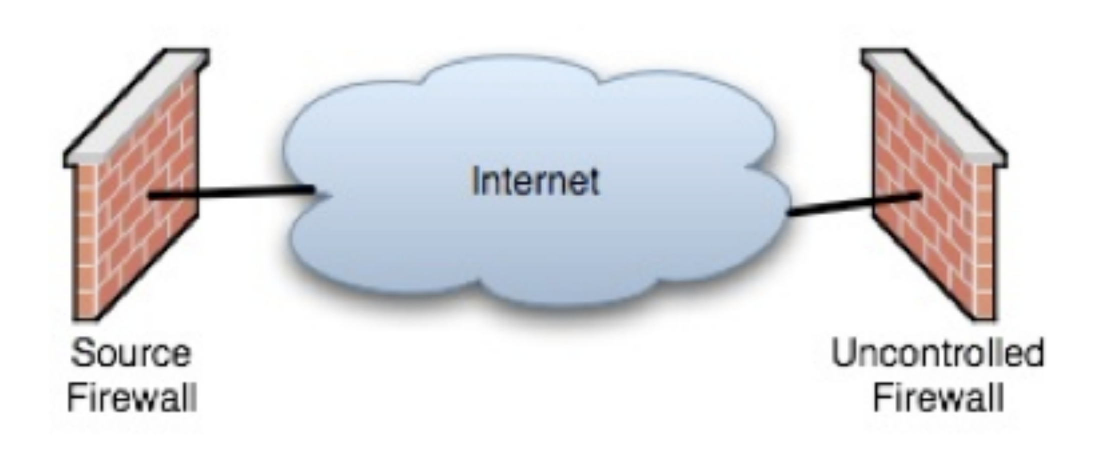
Problem

- How do you provide desk side support to a user who is on the other side of the world on a foreign network ?
- The user is also NAT'd (possibly multiple times) behind some random firewall (or firewalls).
- The solution needs to be 'average user' friendly.

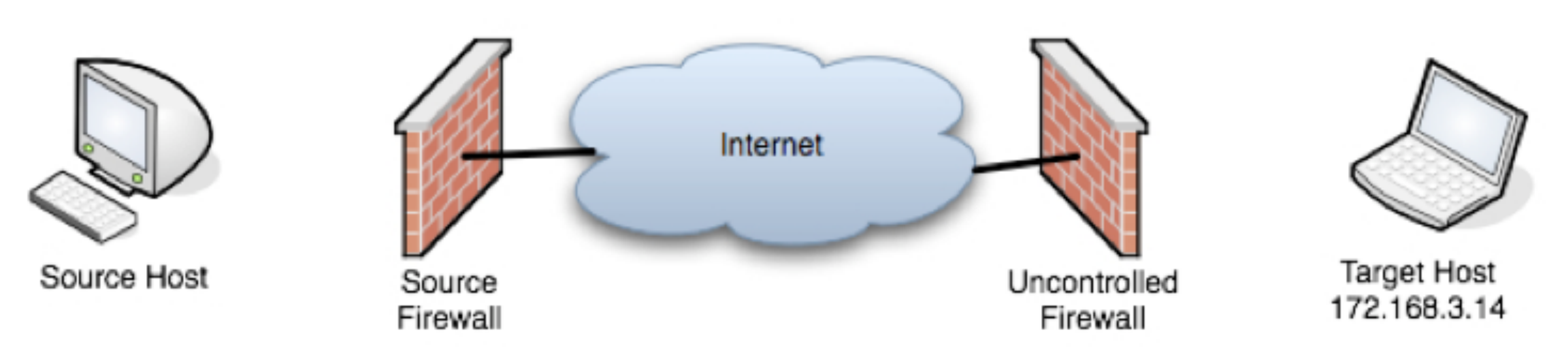
Solution

A reverse SSH tunnel using an intermediary SSH server!

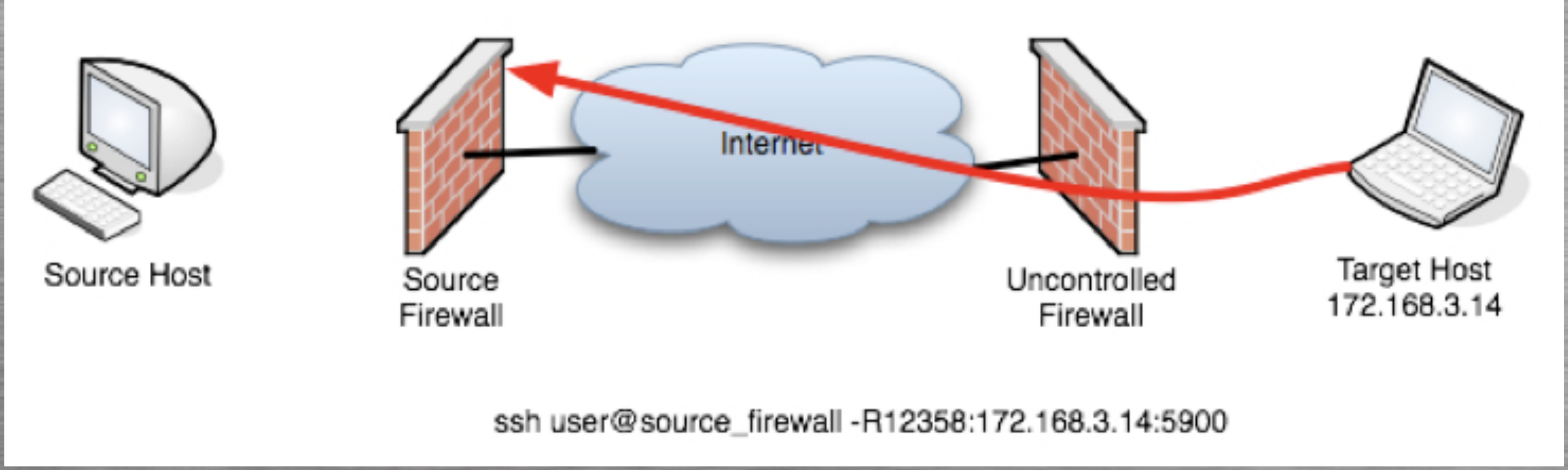
Solution



Solution

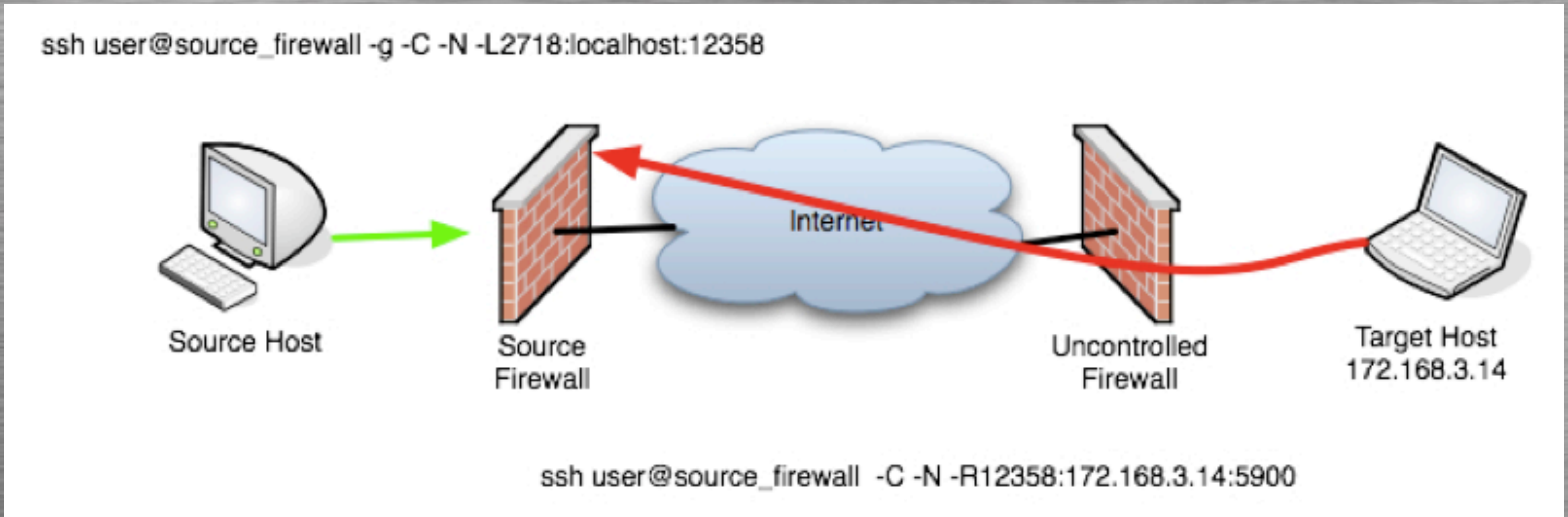

Source Host
Source Firewall
Internet
Uncontrolled Firewall
Target Host 172.168.3.14

Solution



```
ssh user@source_firewall -R12358:172.168.3.14:5900
```

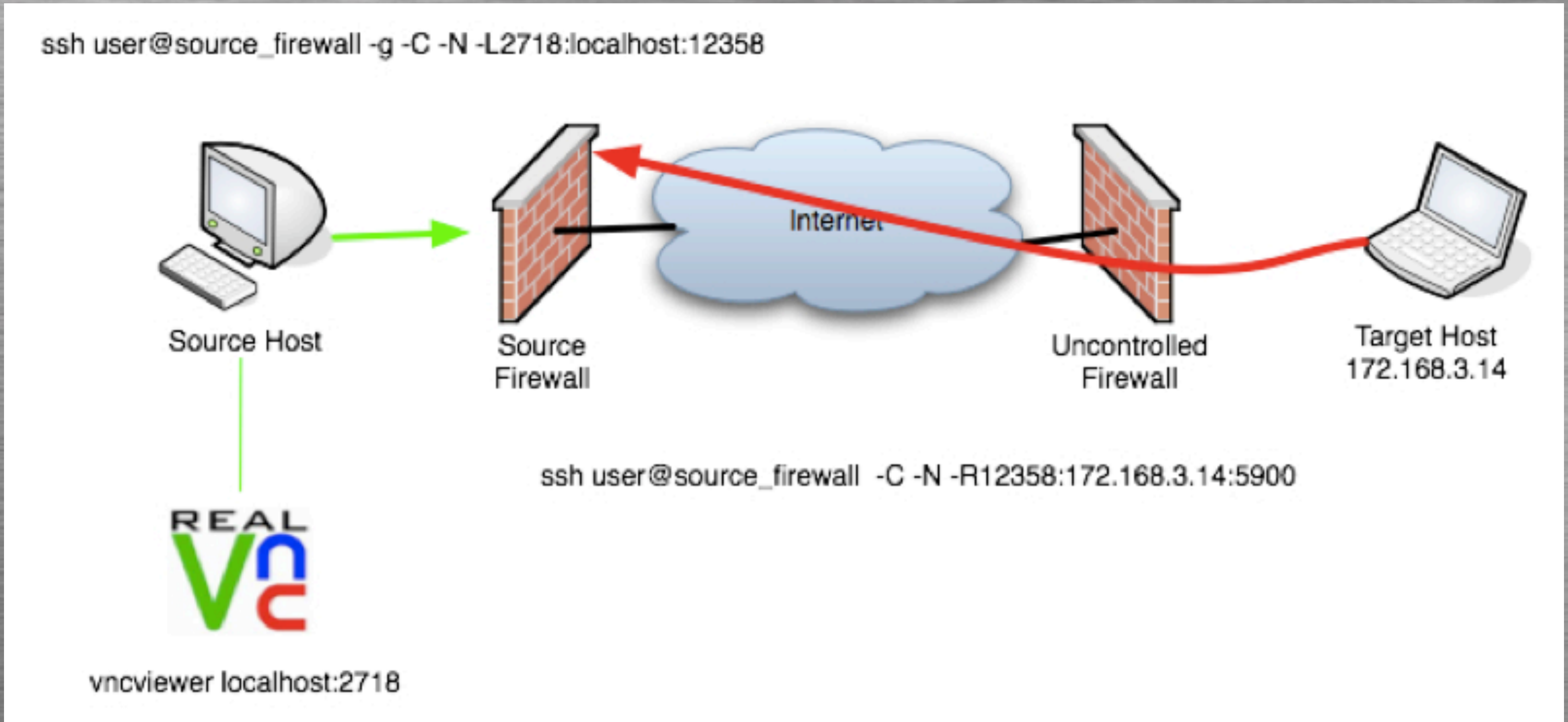
Solution



```
ssh user@source_firewall -g -C -N -L2718:localhost:12358
```

```
ssh user@source_firewall  -C -N -R12358:172.168.3.14:5900
```

Solution

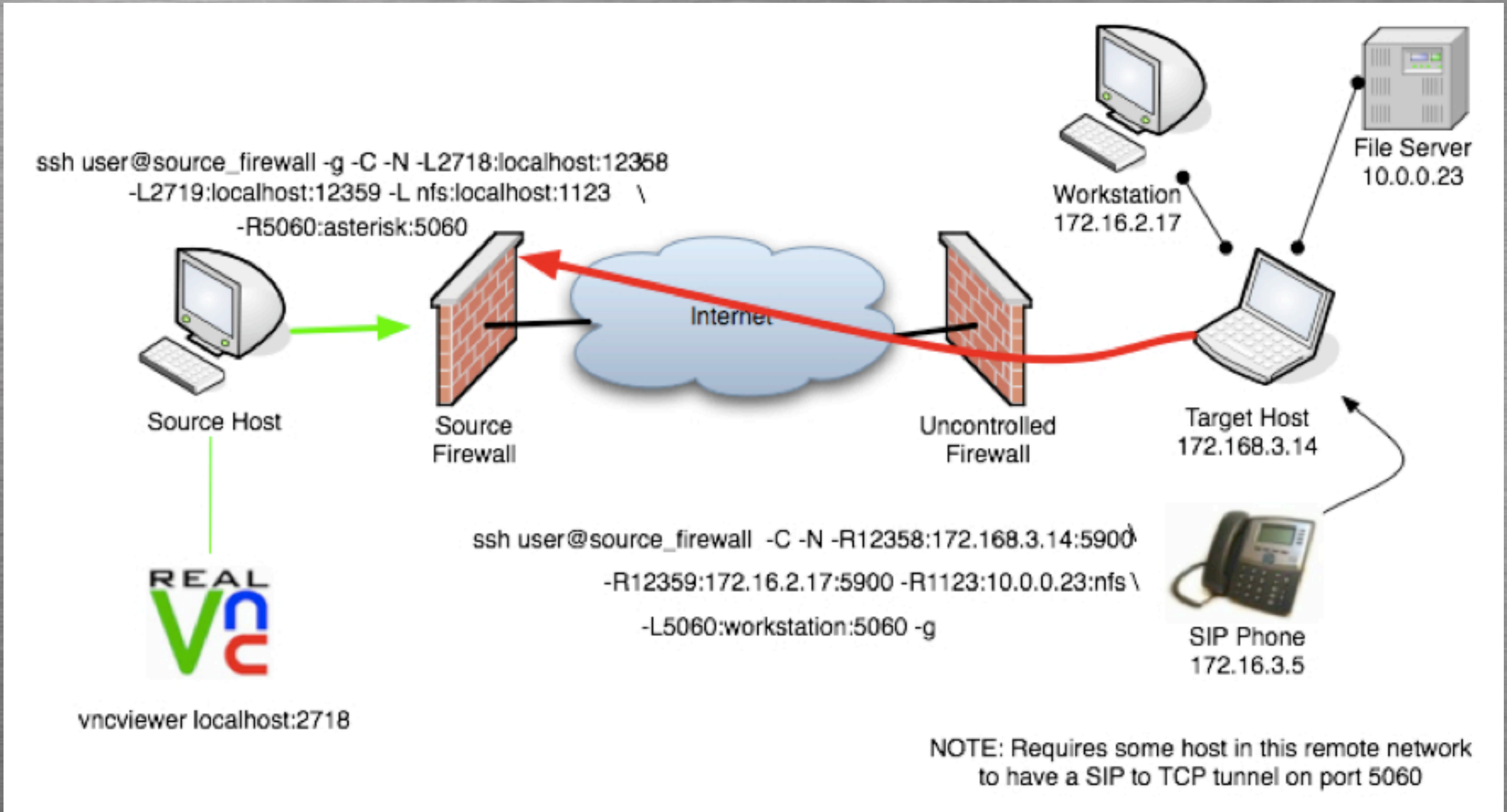


```
ssh user@source_firewall -g -C -N -L2718:localhost:12358
```

```
ssh user@source_firewall  -C -N -R12358:172.168.3.14:5900
```

```
vncviewer localhost:2718
```

Solution



```
ssh user@source_firewall -g -C -N -L2718:localhost:12358
        -L2719:localhost:12359 -L nfs:localhost:1123    \
                -R5060:asterisk:5060
```

```
ssh user@source_firewall  -C -N -R12358:172.168.3.14:5900\
        -R12359:172.16.2.17:5900 -R1123:10.0.0.23:nfs \
            -L5060:workstation:5060 -g
```

```
vncviewer localhost:2718
```

NOTE: Requires some host in this remote network
to have a SIP to TCP tunnel on port 5060

Problem

- You would like to give users SSH access or use the previous examples in production but need to control/limit their use and abuse.

Solution

- Configure limitations on your ssh daemon and/or user config.
- Constrain port forwarding with PermitOpen configuration option (per server, or user).
- Doing just port forwarding... use ‘no-pty’ option in authorized_keys (this is per key).
- Use forced commands instead of giving shells (works for all kinds of things, not just subversion).

Secured Shell Server

- In sshd_config you can lock things down with the following options:

```
PermitRootLogin no
StrictModes yes
PasswordAuthentication no
PermitEmptyPasswords no
AllowTcpForwarding no
AllowX11Forarding no
UsePrivilegeSeparation yes
Compression yes
UseDNS yes
```

Secured Shell Server

- Don’t forget to remove setuid from passwd(1)
  - chmod -s `whereis passwd`
- User creation should include setting up an encrypted RSA/DSA key and set their login password to ‘garbage’ of length at least 15 characters.

Bonus Problem

- You have a server far away who has a crypto card/accelerator that has locked up and isn’t responding to new SSH sessions?

Solution

Solution

- Change the default cipher in the ssh client to one that the crypto card doesn’t support!

Solution

- Change the default cipher in the ssh client to one that the crypto card doesn't support!
- For example the VPN1411 HiFn Crypto accelerator doesn't support the blowfish cipher.

Solution

- Change the default cipher in the ssh client to one that the crypto card doesn’t support!
- For example the VPN1411 HiFn Crypto accelerator doesn’t support the blowfish cipher.
- Therefore...

Solution

- Change the default cipher in the ssh client to one that the crypto card doesn’t support!
- For example the VPN1411 HiFn Crypto accelerator doesn’t support the blowfish cipher.
- Therefore...
  - ssh -c blowfish user@host

Key Sizes

- Longer key lengths provide ‘better’ security at the cost of decreased performance but don’t go crazy.
- SSH Keys are for authentication only, once authenticated a Diffie-Hellman key exchange is used to generate session keys which can/are re-key’d after specified intervals or traffic use.
- Avoid unencrypted (ie. no/blank password) keys, use an ssh-agent to handle credential management (ie. type the password once per ‘login’ and forget about it).
- Don’t ignore ‘known host key has changed’ messages as they are your last line of defense against MITM attacks. Seriously...

But wait there’s more!

- Ad-hoc VPN using SSH and tunnel devices
  - see ‘ssh -w’ option.
- If you can get any type of traffic out of a network you can tunnel over it.
  - Defense; rate-limit DNS, ICMP and UDP.
- chroot’d sftp server (OpenSSH 4.7+)
- Per user/key SSHD restrictions.
- Per user/key TCP Forwarding restrictions
  - See PermitOnly config option.
- SSH signature visualization makes it easy to recognize keys.
- Use the command channel to add tunnels to already active sessions.

man pages

- The OpenSSH man pages are fantastic... use them. The following 3 man pages were all I needed to reference for this talk.
  - ssh(1) - if it can be done with the client it is here
  - sshd_config(5) - server specific configuration
  - ssh_config(5) - user specific configuration

OpenBSD 4.4 Pre-orders Available!

OpenBSD4.4
MULTI-PLATFORM ULTRA SECURE OPERATING SYSTEM
CD1: i386
CD2: macppc • amd64
track 2 is audio track
CD3: sparc64 • source code
CD can boot on all architectures.
Other architectures only available via FTP

=

Supporting OpenBSD means supporting OpenSSH.

Questions?

The network is down... about 3 stories down.

http://www.youtube.com/watch?v=nGtWYuJ5f64

* Note: Contains language which may offend some.