SSH Mastery

OpenSSH, PuTTY, Tunnels and Keys

Michael W Lucas
BSDCan 2012
About Me

- Author
- BSD pusher
- irremediable smartass
About You

• How many OpenSSH clients?
• How many PuTTY clients?

• name?
• your goals here?
Contents II

- Port Forwarding
- Host Key Distribution
- Limiting OpenSSH
- OpenSSH VPNs
Security Warning

- SSH is a tool
- Tools can be used for good or evil
- SSH can help you save your company
- SSH can help you destroy your company
- MWL is not responsible for reasonable or unreasonable damages caused by your use/abuse of SSH
SSH Overview

- What is SSH?
- What is OpenSSH?
- SSH Servers
  - OpenSSH – most popular
  - SSH.com -- commercial
- SSH Clients
  - OpenSSH – Unix-like
  - PuTTY -- Windows
SSH Protocol Versions

- SSH-1, original SSH
  - created in 1995 by one guy, Tatu Ylönen, for his own uses
  - can be decrypted by packet sniffers
  - do not use SSH-1
- SSH 1.3, 1.5, 1.99 = SSH-1
- SSH-2, modern SSH
  - only use SSH-2
Encryption 101

- plain text = readable
- ciphertext = unreadable
- algorithm = method for transforming plaintext to ciphertext & back
- key = secret string used as algorithm seed
Encryption Algorithms

- **Symmetric**
  - same method & key used to encrypt & decrypt
  - $A=1, B=2$, etc
  - Fast

- **Asymmetric**
  - different methods to encrypt or decrypt
  - one key for encryption
  - different key for decryption
  - slow
Public Key Encryption

- Asymmetric algorithm
- give one key away
- keep one key secret
- used for SSH, HTTPS, PGP, etc
- Many different asymmetric public key algorithms – RSA, DSA, Blowfish, etc
- Use recommended algorithms
How SSH Uses Encryption

• Public key for initial session setup
• Agree on temporary symmetric secret
• symmetric for most of session
• occasional rekeys
Cool Is Not Secure

• The algorithms used, and the order they are tried in, are chosen for a reason

• Do NOT change them
Configuration Files

• all in /etc/ssh

• ssh_config – host-wide client config

• ssh_host_*_key.pub – private keys

• ssh_host_*_key – public keys

• sshd_config – server config
The OpenSSH Server

- Included by default in any server OS at this conference
- Also available for Windows, via Cygwin, sshforwindows, etc.
Testing sshd

- /etc/ssh/sshd_config
- /usr/sbin/sshd -f sshd_config_test -p 222
  - test alternate configuration
- /usr/sbin/sshd -f sshd_config_test -p 222 -ddd
  - run in foreground
  - one connection only
  - useful for weird debugging
Config File Syntax

- Boring option-then-value syntax

```plaintext
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```
Network & Protocol Options

Port 22
AddressFamily any (inet | inet6)
ListenAddress 0.0.0.0 | ::

Protocol 2 – no excuses for your servers!
Banner & motd

- Banners appear before auth, but might not work for all clients & can interfere with automation
  
  Banner /etc/ssh/ssh-banner

- motd always displays, after auth
  
  PrintMotd yes
Verify clients against DNS

UseDNS yes

- makes sure forward & reverse DNS match
- subject to DNS attacks
- IPv6
- Conclusion: don't bother
Restricting Access by User or Group

- Processed in order listed in config file
- first-match basis
- \{Deny, Allow\} Users – user list
- \{Deny, Allow\} Groups – group list
Restrict by User or Group II

- Demo system:
  - wheel: mwlucas
  - staff: mwlucas, pkdick, jgballard
  - support: pkdick, mwlucas
  - billing: jgballard
Deny Billing People

- **OK:**
  
  DenyUsers jgballard

- **Better:**
  
  DenyGroup billing
Allow only admins

• Presence of an Allow* option tells ssッド to deny logins by default

AllowGroups wheel
Deny one user in group

- Users and groups distributed via LDAP. One admin is forbidden access to this server.

DenyUsers pkdick
AllowGroups support
Automation

- rsync user from one machine

  AllowUsers backup@192.0.2.0/25
  AllowGroups support, wheel

- List hosts by network or hostname, but beware DNS
Wildcards

- ? matches exactly one character
- * matches zero or more characters

- *.blackhelicopters.org – any host
- ??????.blackhelicopters.org – matches sloth & wrath, not envy or gluttony.
Wildcards in Networks

- 192.0.2.1? - 192.0.2.10 through 192.0.2.19
- 192.0.2.* - any host in 192.0.2.0/24
- 192.0.2.0/24 – by netmask

Separate multiple entries with commas.
Negation

- `!*.`.blackhelicopters.org – everything that's not under this domain.
- Excludes blackhelicopters.org itself
- Best with exclusions
  - `!lust.blackhelicopters.org,*.`.blackhelicopters.org
- djm describes as "a little fiddly"
Conditional Configuration

- Match by user, group, network, etc
- Example, X11 forwarding

Match User mwlucas

X11 Forwarding Yes
More User Matches

Match Group wheel
  X11Forwarding yes

Match User mwlucas,jgballard
  X11Forwarding yes
Match by Host

Match Address 192.0.2.0/29, 192.0.2.64/27
  X11Forwarding yes

Match Host *.blackhelicopters.org
  X11Forwarding yes
Multiple Matches

Match Address 192.0.2.8 User mwlucas
  X11Forwarding yes
Permitted Matches

• Can only match on certain items
• see sshd_config(5) for full list
• In short, can change auth methods, chroot, access, key locations, maximums, etc.
• Cannot change things like UsePAM, ChallengeResponseAuthentication, etc.
Placing Matches

• All configuration that follows a Match belongs to that Match, until next Match or EOF.

• Place Matches at end
Sample Matches

X11Forwarding no
PasswordAuthentication no
...
Match Group wheel
    X11Forwarding yes
Match Address 192.0.2.0/29, 192.0.2.128/27
    PasswordAuthentication yes
Root SSH Access

- Do not allow logging in as root
- Use sudo, pfexec, other tools
Chrooting Users

- Useful for Web servers, other multi-user servers with individual cells

- Must populate chroot (varies by OS)
  - set permissions on chroot
  - create home dir for imprisoned user
  - create device nodes
  - install shell
Permissions & Directory

• chroot directory owned by root, just like system home dir

• User's $HOME from /etc/passwd relative to jail. If $HOME is /home/pkdick, and chroot is /prison/, directory is /prison/home/pkdick

• $HOME owned by user, contains dotfiles, etc

• static-linked shell
Device Nodes

- Varies by OS, devfs or MAKEDEV
- expect /dev/urandom, /dev/null, /dev/stderr, /dev/stdin, /dev/stdout, /dev/tty, /dev/zero
Assign chroot

- Specify user's root directory as the Chroot Directory. Dumps everyone together in one chroot.

  ChrootDirectory /prison

- %h = user's home directory in /etc/passwd. Locks user into their own directory

  ChrootDirectory %h
More chroot

- `%u` expands to username. Lots of unique users in shared chroot area.

```plaintext
ChrootDirectory /prison/home/%u
```
Choosing users

ChrootDirectory none

...  

Match Group billing

   ChrootDirectory /prison/billing

• If most users chrooted, reverse & allow wheel shell
Protecting sshd

- Hail Mary Cloud
- privilege separation
- packet filter, TCP wrappers
- disable passwords, allow only keys
- change port?
Verifying Server Keys

- Long strings of text
- Many users dismiss verifying keys as impossible
- Is entirely possible, you can make it easier
- Automated distribution is best
Get the Server Fingerprint

```bash
# ssh-keygen -lf ssh_host_rsa_key.pub

2048
/etc/ssh/ssh_host_rsa_key.pub (RSA)
```

- Capture all keys to file
- Can also use ssh-keyscan, requires you verify all keys yourself
Make Keys Available

- Must get fingerprints to users
- access must be easy & secure
- easiest: secure Web site
- don't use email or unencrypted public site

Later: how to do this for your users
Verifying Clients

- Both OpenSSH client & PuTTY present host key fingerprint for verification upon first connection
Changed Host Keys

- User gets a warning upon connection that the key has changed. Possibilities:
  - Sysadmin oops!
  - Client is wrong. Desktop security? Corrupt cache?
  - Server upgrade? Get new fingerprint
  - round-robin DNS?
  - Intruder controls server

- DO NOT CONNECT UNTIL YOU KNOW WHY
SSH Clients

- How many PuTTY users in the room?
- How many OpenSSH client users in the room?
Debugging OpenSSH Client

- `ssh -v hostname`
- increase number of `-vs` for more detailed debugging
- actually read the output
ssh Configuration

- `/etc/ssh/ssh_config` – global
- `~/.ssh/ssh_config` – individual
- Documented in `ssh_config(5)`
- Use alternate with `-f filename`
- All config options work in both
- Can use patterns just like `sshd`
Per-Server Configuration

Host *.blackhelicopters.org
Port 2222

- Matches
  ssh avarice.blackhelicopters.org

- does not match
  ssh avarice

- Can also use IP, netmask, patterns
Changing Username

• on command line
  
  $ ssh jerkface@server.customer.com
  
  $ ssh -l jerkface server.customer.com

• In config file
  
  Host server.customer.com server

  User jerkface
Changing Port

• On command line
  
  $ ssh -p 2222 gluttony

• In config file
  
  Host gluttony
    Port 2222
Options on Command Line

- Anything in ssh(1) can be specified on command line with \(-o\)
  
  \$ ssh -o BindAddress=192.0.2.5 gluttony

- You can use multiple \(-o\)

- Use the config file
Updating Host Key Cache

- Keys cached in $HOME/.ssh/known_hosts
- Update policy option: StrictHostKeyChecking
- Only update by hand? Set to yes.
- Ask user to verify, then add? Set to ask.
Hashing known_hosts

- Hash hostnames in known_hosts, so intruder doesn't know your network
  HashKnownHosts yes
- Use ssh-keygen -H to hash unhashed entries
PuTTY Client

- Windows SSH, telnet, serial, rlogin <cough> client
- Download from http://www.chiark.greenend.org.uk/~sgtatham/putty/
- Not by the OpenSSH paranoids, still pretty good
- Download the full installer
Saving PuTTY Defaults

- Example: set default username
- Beneath "Connection," select "Data."
- In "auto-login," put username
- Save as Default Settings
Saving PuTTY Sessions

• Add server hostname, protocol, port, etc.
• Enter session name
• click Save
• Can also save other settings, such as X11 forwarding, as sessions, e.g., "dns1-x11"
• Saved defaults not propagate to saved sessions!
PuTTY Management

• Upper left hand corner drop-down menu.
• Useful tricks:
  • Duplicate Session
  • Saved Sessions
  • New Sessions
  • Change Settings
PuTTY Configuration

- In Windows Registry, under `HKEY_CURRENT_USER\Software\SimonTatham`
- Can copy from machine to machine
- Can distribute valid configs via Active Directory
Debugging PuTTY

- Event Log, in upper left drop-down menu
- serious debugging, use Session Log.
  - Before opening new session, go to Session -> Logging
  - Choose log type. I usually use All session output.
  - Give directory and name for debug file
Copy Files over SSH

- FTP predates TCP/IP. It’s an appalling protocol.
- Apps like rsync travel over SSH
- Two SSH-based protocols, SFTP and SCP
  - SCP: rcp with SSH backend. Basically unmaintained
  - SFTP: newer copy program, maintained
SCP

- copies individual files
  `$ scp source-host:file dest-host:file`
- Copy data1 to host server1:
  `$ scp data1 server1:`
- Without the colon, I securely copy file data1 to local file server1. Probably not right.
SCP II

- Copy remote file to local:
  $ scp data1:server1 .

- Change filename
  $ scp data1 server1: data2

- Change location:
  $ scp data1 server1: /tmp/
SCP III

- Change usernames
  
  ```bash
  $ scp data1 jerkface@server1:
  ```

- Recursive scp
  
  ```bash
  $ scp -rp /home/mwluucas server1:
  ```
SFTP

- More modern, interactive
- Looks awfully like FTP

```
$ sftp server1
sftp> put data1
sftp> get data2
sftp> lcd /tmp
sftp> cd /var/db/postgres
```
Per-Host Configuration

- Both read ssh_config
- ssh command-line options don't always map to scp/sftp, e.g., use -P to change port
Windows SCP/SFTP

- Command-line apps like pscp.
- Use WinSCP for GUI app
- Free for personal use, restrictions to redistribute
- Transparently switches between SFTP and SCP protocols depending on what server supports
- Looks like any other Windows app
WinSCP tips

• Import PuTTY key cache: Saved Sessions -> Tools->Import.

• Turn off SSHv1: select SSH, set Preferred SSH protocol version to 2. Select Stored Sessions, then Save defaults...

• Defaults do not propagate to saved sessions

• Explorer-style window: Preferences, choose Explorer.
Configuring SCP/SFTP server

- For scp, `scp(1)` must be in default system `$PATH`.
- SFTP server bundled with sshd, activated with `sshd_config`
  
  `Subsystem sftp /usr/libexec/sftp-server`

- Disabling only removes obvious file copy methods. If you're really concerned, chroot sftp users.
SFTP-Only Users

Match Group sftponly
ChrootDirectory %h
ForceCommand internal-sftp
AllowTcpForwarding no
SSH Key Auth

- Passwords are a weak point in security
- Humans make really bad passwords
- One-time auth (OPIE) annoying
- Two-factor auth annoying and introduces additional points of failure
- Give each user a keypair, encrypted with a passphrase
Passphrase

- Text string used to encrypt private key
- If private key is stolen, useless without passphrase
- Make passphrase too long to guess by brute force, too complex to guess, too long to shoulder-surf.
- Numbers, words, letters, symbols and space.
Tr0ub4dor & 3

~28 bits of entropy

$2^{28} = 3$ days at 1000 guesses/sec
(Plausible attack on a weak remote web service. Yes, cracking a stolen hash is faster, but it's not what the average user should worry about.)

Difficulty to guess: EASY

Difficulty to remember: HARD

Correct horse battery staple

~44 bits of entropy

$2^{44} = 530$ years at 1000 guesses/sec

Difficulty to guess: HARD

Difficulty to remember: You've already memorized it

Through 20 years of effort, we've successfully trained everyone to use passwords that are hard for humans to remember, but easy for computers to guess.

http://xkcd.com/936/
Good Passphrases

- Not a cliche, saying, or media catchphrase
- My passphrase from 1999:
  - "Come closer, my darling child, but not too close, for I, too, cannot be trusted."
  - It's a mingling of two different translations of Lautreamont's *Maldoror* (1868).
  - I can still remember it, you'd have a hard time guessing it.
  - I am not recommending you read the book.
- My current passphrase is longer & more obscure
Why Kill Passwords?

- Simple two-factor auth (passphrase & file)
- SSH-breaking clouds (Hail Mary)
- Shuts up smart SSH scanners
SSH Agents

- Typing passphrases is more annoying than typing passwords
- SSH agent takes the key file, accepts your passphrase, and stores decrypted private key in memory (never to disk)
- When you SSH to a host, SSH client asks agent for passphrase
- Type passphrase once, use it all day
Agent Risks

- Lock Your Desktop!
- Multiuser Machines
- Sysadmins
Install Public Key on Server

- $HOME/.ssh/authorized_keys
- Should be readable by everyone – it's public
- Should not be writable by anyone but you
- Use SCP/SFTP, not copy & paste
- ssh-copy-id
Create Keypair with OpenSSH

$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/mwlucas/.ssh/id_rsa):

Enter passphrase (empty for no passphrase): ...

Enter same passphrase again: ...

Your identification has been saved in /home/mwlucas/.ssh/id_rsa.

Your public key has been saved in /home/mwlucas/.ssh/id_rsa.pub.

The key fingerprint is: ...
Using SSH Key for Auth

client$ ssh sloth

Enter passphrase for key '/home/mwlucas/.ssh/id_rsa': ...

sloth$
OpenSSH Agent

- Varies by desktop GUI, might Just Work
- Command-line:
  $ ssh-agent /bin/tcsh
  $ ssh-add
- XDM: use openssh-askpass
- startx: use command-line before starting GUI (WindowMaker), or maybe just ssh-add (cwm)
PuTTY User Auth Keys

- Use PuTTYgen, included with full install
- Very standard Windows GUI; start, click "Generate"
- 1024 bits is minimum, unless you're logging into a VAX
- Save generated key.
- Select Conversions -> Export OpenSSH Key.
Using Auth Keys w/PuTTY

- For first attempt, use key without agent
- On left side of PuTTY, select Connection -> SSH -> Auth. Give full path to private key file.
- Install key on server.
- Log in.
- Should be asked for passphrase.
- Do not save this session
PuTTY Agent: Pageant

- Select Add Key, browse to your key, select, enter passphrase
- Enter passphrase again. Eventually you'll get it right.
- SSH to your server
- PuTTY enable/disable agent: Connection -> SSH -> Auth, "Attempt Authentication using Pageant" checkbox
Pageant at Startup

- Add Pageant shortcut to Startup menu
- Edit Target field to add full path to private key.

"C:\Program Files\PuTTY\pageant.exe"
"C:\Users\mwlucas\keys\work.ppk"
Key File Management

- One key per client machine
- Back up private keys to offline media
Disabling Passwords in sshd

- `/etc/ssh/sshd_config`

  ```
  ChallengeResponseAuthentication no
  PasswordAuthentication no
  PubkeyAuthentication yes
  UsePAM no
  ```
Selectively Allow Passwords

Match Address 192.0.2.0/24

PasswordAuthentication yes
Agent Forwarding

- Servers only allowing login via key, good
- Must copy file from one server to another
- Don't want to copy private key to server
- Solution? Forward agent requests back to desktop
- Forwards requests through $SSH_AUTH_SOCK, back to client.
Agent Forwarding Risks

- Anyone who can access socket can access agent.
- Do you trust root?
- Do you trust machine?
Enable Forwarding

- On server
  
  AllowAgentForwarding yes

- in ssh
  
  ForwardAgent yes

- in PuTTY
  
  - Connection -> Data -> SSH->Auth.
  
  - Under Authentication Parameters.
  
  - Forward Agent check box.
pam_ssh_agent_auth

• auto-auth sudo via your SSH agent

• in sudoers:

  Defaults env_keep += "SSH_AUTH_SOCK",timestamp_timeout=0

• sudo PAM config:

  auth sufficient \
  /usr/local/lib/pam_ssh_agent_auth.so \
  file=~/.ssh/authorized_keys

  auth required pam_deny.so

  account include system

  session required required pam_permit.so
Security Sensitive Topics

- SSH can act as arbitrary wrapper around other protocols
- Network admins love them
- Security managers hate them
- Which one is you?
X11 Forwarding

- Enable on server
  `X11Forwarding yes`

- Enable X11 secure subset on client
  `ForwardX11 yes`

- Enable all of X11 on client
  `ForwardX11Trusted yes`

- Can enable per-host, per-user, etc.
Is X11 Forwarding Working?

- Check \$DISPLAY
  
  $ \texttt{echo ~DISPLAY}$
  
  localhost:10.0

- Any other result = X not going over SSH!
- Test with xterm, xeyes, etc.
PuTTY X11 Forwarding

- Need X server
- Xming – X.org based – on sourceforge
- PuTTY X11 forwarding = X11Trusted
- On by default
- Connection -> SSH -> X11, first box is Enable X11 Forwarding
- Turn it off by default, on as needed
Port Forwarding

- Wrap arbitrary traffic inside SSH
- Drives corporate security admins insane, because users can bypass access controls
- Network and server guys love it, for the same reason
- Obey corporate security policy
Port Forwarding Types

- **Local Port Forwarding**
  - grab a port on local machine
  - attach to SSH server

- **Remote Port Forwarding**
  - grab a port on remote machine
  - attach to SSH client

- **Dynamic Port Forwarding**
  - forward all traffic to server via SOCKS
Privileged Ports

- On Unix-like systems, ports below 1024 can only be bound by root.
- Affects port forwarding as well.
- Can forward to a privileges port, not just from.
- Can forward any port on Windows-like systems.
Local Forwarding

- Attach local port to remote port
- Tunnel insecure protocol over SSH
  
  `$ ssh -L localIP:localport:remoteIP:remoteport host`

- If no IP specified, attach to 127.0.0.1; can skip first colon in that case
- Can set permanently in `ssh_config`

  `LocalForward localIP:localport remoteIP:remoteport`
ssh: tunnel HTTP over SSH

• connect port 80 on localhost to port 80 on server's localhost

• must run as root

  $ sudo ssh -L 80:127.0.0.1:80 mwlucas@www

• Make /etc/hosts entry pointing host at 127.0.0.1

• To set permanently, use ssh_config entry

  Match Host www
  LocalForward localhost:8080 localhost:80
PuTTY: tunnel HTTP over SSH

- Select Connection->SSH->Tunnels
- Set "source port" to 80
- Set Destination to 127.0.0.1:80
- at the bottom, select Local
- To bind network-facing IP locally, select "Local ports accept connections from other hosts"
Remote Port Forwarding

- Attach remote port to local port
- Tunnel insecure protocol over SSH
  
  
  \$ ssh -R localIP:localport:remoteIP:remoteport host

- If no IP specified, attach to 127.0.0.1; can skip first colon in that case
- Can set permanently in ssh_config
  
  RemoteForward localIP:localport remoteIP:remoteport
ssh: remote forward SSH

• connect port 2222 on server's localhost to port 22 on client's localhost

  $ sudo ssh -R 22:127.0.0.1:2222 mwlucas@www

• To set permanently, use ssh_config entry

  Match Host www
  RemoteForward localhost:2222 localhost:22
PuTTY: remote forward SSH

- Select Connection->SSH->Tunnels
- Set "source port" to 2222
- Set Destination to 127.0.0.1:22
- at the bottom, select Remote
- To bind network-facing IP on server, select "Local ports accept connections from other hosts"
Using Remote Forwarding

- Log into server
- SSH to port 2222
- will be connected to client's SSH daemon
- this is why security admins hate it
Dynamic Port Forwarding

- Attach local port to server
- Local port is SOCKS proxy
  
  \$ ssh -D localIP:localport server

- If no IP specified, attach to 127.0.0.1; can skip colon in that case

- Can set permanently in ssh_config

  Host servername
  
  DynamicForward host:port
ssh: dynamic forwarding

- connect port 9999 on server's localhost to port 22 on client's localhost
  
  `$ ssh -D 9999 www`

- To set permanently, use ssh_config entry
  
  `Match Host www`
  
  `RemoteForward workstation:9999`
PuTTY Dynamic Forwarding

- Select Connection->SSH->Tunnels
- Set "source port" to 9999
- Leave Destination blank
- at the bottom, select Dynamic
- To bind network-facing IP on server, select "Local ports accept connections from other hosts"
Testing Dynamic Forwarding

- Configure Web browser to use SOCKS proxy on localhost, port 9999
- Browse out to Internet, bypassing company security policy
- Impact on company security
  - an illicit SOCKS proxy in a secure environment will get you fired with prejudice.
  - Or you can legitimately use dynamic forwarding to access your secure environment.
- Po-tay-to, po-tah-to
Choosing IP Addresses

- Bind to local address, only client or server can use the forwarding
- Bind to network-facing address, everyone can use it.
Host Key Distribution

- Your users cannot be trusted.
- You don't want to be bothered by dumb user questions
- If a user sees a warning, it should be scary
- Distribute pre-verified host keys to client machines solves all this
Gather Host Keys

- build your own known_hosts with all algorithms
  
  `ssh -o HostKeyAlgorithms=ssh-rsa server`
  `ssh -o HostKeyAlgorithms=ssh-dss server`
  `ssh -o HostKeyAlgorithms=ecdsa-sha2-nistp256 server`
OpenSSH Host Key Distribution

- ssh checks `/etc/ssh/ssh_known_hosts` as well as `$HOME/.ssh/known_hosts`
- Automate distribution: rsync, puppet, whatever
- To revoke a key, put string `@revoked` in front of entry. User will see scary warning.
ssh_known_hosts vs known_hosts

- $HOME/.ssh/known_hosts checked before /etc/ssh/ssh_known_hosts
- Best to move known_hosts to known_hosts_personal
- Don't just erase; user might have legitimate keys not on your network
Distributing known_hosts for PuTTY

- `kh2reg.py` part of PuTTY distribution
  
  
  ```
  $ hk2reg.py known_hosts > puttykids.reg
  ```

- install reg script via login script / AD
Limiting SSH

• keywords in authorized_keys can limit actions possible over SSH.
• authorized_keys contains single lines, each the contents of a key.pub file.

ssh-rsa AAAA.......wC9
mwlucas@blackhelicopters.org
Keywords in authorized_keys

- put limiting keywords at beginning of key
- command="/bin/whatever" – this key can only run this command

```bash
command="sudo ifconfig tun0 inet 192.0.2.2 netmask 255.255.255.252"
ssh-rsa...
```
Limiting Locations

• Restrict which IP addresses a key can be used from:

from="192.0.2.0/29" ssh-rsa AAAA....
Restrict Forwarding

• Kill various forwardings
  • no-agent-forwarding
  • no-port-forwarding
  • no-X11-forwarding

• Permit certain types of forwarding
  • permitopen="127.0.0.1:25"
Keys for Automated Processes

- rsync, rsnapshot, nagios, etc, can use SSH transport
  $ ssh-keygen -f nagios-key -N ''
- Have process use this key with -i flag:
  $ ssh -i nagios-key server1
Limiting Automated Processes

- That which is not necessary is forbidden

command="dump /home > /backups/`date +s`.dump", from="192.0.2.8", no-agent-forwarding, no-portforwarding, no-X11-forwarding ssh-rsa AAAA......wC9 mwlucas@blackhelicopters.org
Avoiding Root

- Use `sudo(8)` to avoid using root
- Sample `/etc/sudoers` entry

```
automation ALL=NOPASSWD: /bin/dump /home > /backups/`date +s`.dump
```
SSH VPN

- You can use SSH as a VPN
- Varies widely by operating system
- We don't have time to cover all of the options
- Don't do this if you have any other choice
- Sometimes, you have no other choice