Security and OpenSSH

Stephen Dum

Stephen.dum@frontier.com

Overview

Focus – ssh 2 from the linux users perspective

- Security
- Why ssh?
- Analysis of ssh usage scenarios
- Cool things to do with ssh
- Summary

Why do you use ssh

Security

How secure are the machines you use?

Is your desktop machine secure?

Are the machines you need to access secure?

Are the machines physically secure?

Security ...

How secure is it to type in a password everytime to connect to another machine?

How many people have root access to the machines you need to access?

To your desktop machine?

What can superusers do to circumvent your security?

Security ...

- Trojan horses – modify standard utilities
- Intercept network communication
- Trusted machines

“Trust is inherently Social”

- Bruce Schneider

Features

Security

- Authentication – user identity, server identity, client identity
- Encryption – network traffic encrypted
- Integrity -- data not modified in transit, no man in the middle attacks, replay attacks

Facilities

- Convenience
- Remote login
- File copy between machines
- Remote command execution
- Local access to remote services (imap, smtp, http, …
- Easy access, no need to retype password for each command.

Facilities

- X11 display forwarding
- Run a specific command on login
- Launch cronjobs – with some care

Asymmetric or public key encryption

You generate a private key and a public key with ssh-keygen. Private key is usually protected with a passphrase.

Anyone can encrypt a message with the public key, but you must know the private key (and passphrase) to decrypt the message.

These keys are used to authenticate login attempts on machines. “impossible to crack” – which means computationally infeasible.

SSH Encryption

Authentication uses asymmetric or public key encryption.

Network traffic uses symmetric encryption with a 'random' generated key.

SSH Setup

- Every machine you want to access needs your public key
- Keys in ~/.ssh (on linux) directory must be protected with mode 600 or 700.
- Private key needed on your 'desktop' machine
- Passphrase – remembered, not stored on computer
- Never create key pairs with no passphrase. Ok, never say never, except for a few very special situations

Assumptions

Your 'desktop' is trusted

Scenario 1

Create a simple connection to a machine

```
ssh foo.corp.com
xterm -e ssh foo.corp.com
gnome-terminal -e “ssh foo.corp.com”
```

Prerequisites in Scenario 1

- The ~/.ssh directories must not be readable or writable by anyone else
- Your public key must be on the remote machine foo.corp.com and listed in authorized_keys file
- private key is only needed on your local machine

Connection process in Scenario 1

- Exchange pleasantries, ssh version, capabilities, host key, session key in plain text.
- Negotiate common supported encryption algorithms
- Client generates and sends symmetric encryption key encrypted with host and session key to server

Connection process in Scenario 1

- Client and Server negotiate common supported protocols, and supported keys.
- Server sends challenge encrypted with user's public key
- You're prompted for your passphrase to decrypt the challenge
- User is authenticated
- Communication proceeds.

Security issues

For normal 'other' users of remote machine

- Your public keys are protected, unreadable
- Passphrase – safe other than with a vulcan mind meld, key loggers, spy cams, inquisitive eyes
- Private key is not transmitted to the destination machine, likewise for passphrase

Security against extreme measures

Superuser (on remote machine)

- Can't access your private key from remote machine,
- Can access your public key
- Can't access your passphrase

Local machine 'trusted' right!

ssh-agent

- Runs on your desktop
- Remembers your passphrases for you
- Set this up once when you login or first need it.
- Agent communicates on a ipc connection, it doesn't know who or where a request comes from.

When ssh needs to authenticate a key, it asks the agent to decrypt the message for it.

ssh-agent

Setup adds 2 environment variables

- SSH_AGENT_PID=2926
- SSH_AUTH_SOCK=/run/user/1000/keyring-exg3Ld/ssh

Socket is in a directory with 700 permissions so only you can access it.

ssh-agent security

- Access to agent only allowed by 'you', others can easily deduce the name of the socket, but can't access it. (except superuser).
- Openssh agent stores passphrases in plain text in memory – again others can't access (except root).
- Root could easily deduce the name of the socket and use it as to authenticate a connection (with only minor effort).
- Agent is on your desktop - A trusted machine, *right*!

A simple connection with agent

Run ssh-agent, use ssh-add to add your keys and passphrases.

gnome environment launches a agent for you, with all your login session processes children of the agent. Once keys are added, they are remembered in a password protected vault.

KDE maintains its own password protected vault, not related to agent. The web provides a number of reasonably convenient solutions to this.

There is a script 'keychain' available to do this if all else fails.

A simple connection with agent

Ssh foo.corp.com

Differences in connection process from Scenario 1

- Server sends challenge encrypted with user's public key
- Client passes it to agent to decrypt, and sends result back to server for verification

Agent Connection security

- User didn't have to enter passphrase again
- Private key remained on client
- Agent security applies
- Rest of security same as w/o agent

Tunneling

If your 'corporate' network is isolated behind a firewall, but with a ssh server on the firewall ssh can make a connection through the firewall machine to hosts in the private network

ssh fw.corp.com ssh foo.corp.com

Oops! The remote executed ssh we launched on fw doesn't have access to your agent. So you will need to type your passphrase again – boo!.

Tunneling continued

```
ssh -A fw.corp.com ssh foo.corp.com
```

- The -A tells ssh to setup things so the ssh running on fw provides a 'agent' which just presents a tunnel back to the desktop ssh agent.
- Now the 'ssh foo' proceeds using your desktop agent to authenticate you.

Tunnel security

The new element here is the tunneling of a ssh-agent back to your desktop.

User level - As before, agent sockets for ssh are in a protected directory so no problem.

Tunnel Security superuser

Superuser – root on fw could access the agent socket to authenticate you. If you forward the agent it had better be to a trusted machine. Otherwise, root could use the agent to make a connection to any other machine you can access.

Root doesn't get access to your passphrase or private key, but for the duration of the connection, root could hijack your credentials to make another connection.

A more secure tunnel

```
ssh -N -L 2220:foo.corp.com:22 fw.corp.com&
```

```
ssh -p 2220 localhost
```

This also creates a tunneled connection to foo, without creating an agent on the intermediary machine – but ties up a port.

Which allows

```
scp -P 2220 localhost:somefile .
```

But not

```
scp -P 2220 localhost:somefile bar.corp.com:somefile
```

Tunneling socks5

- ssh -N -D 8181 fw.corp.com
- The '-D 8181' has ssh creating a socks server on fw.corp.com tunneled to localhost port 8181

```
function FindProxyForURL(url,host) {
    if (isPlainHostName(host)) {
        if (host == "corp1" ||
         host == "corp3") {
          //alert("proxy host");
          return "SOCKS5 localhost:8181";
      } else {
          return "DIRECT";
        }
    }
    if ( dnsDomainIs(host, ".sw") ||
       dnsDomainIs(host, ".sw.corp") ||
       dnsDomainIs(host, ".sw.corp.com")) {
      //alert("proxy domain");
        return "SOCKS5 localhost:8181";
    } else {
        return "DIRECT";
    }
}
```

Summary

- Security - communications encrypted -- 'impossible to crack' – computationally infeasible or ok short term, but can crack it with time.
- Integrity – counter measures to prevent hijacking connection.
- Enter your passphrase once, agent allows it to be once per desktop login
- Only run agent on your desktop

ssh

For more information:

SSH The Secure Shell, The Definitive Guide,
by Barrett & Silverman, O'reilly

man pages for ssh, scp, sftp, slogin

http://www.openssh.com

Stephen.dum@frontier.com

Please Enter Your Password...

"cabbage"

Sorry, the password must be more than 8 characters.

"boiled cabbage"

Sorry, the password must contain 1 numerical character.

"1 boiled cabbage"

Sorry, the password cannot have blank spaces.

"50bloodyboiledcabbages"

Sorry, the password must contain at least one upper case character.

"50BLOODYboiledcabbages"

Sorry, the password cannot use more than one upper case character consecutively.

"50BloodyBoiledCabbagesShovedUpYourArse,IfYouDon'tGiveMeAccessnow”

Sorry, the password cannot contain punctuation.

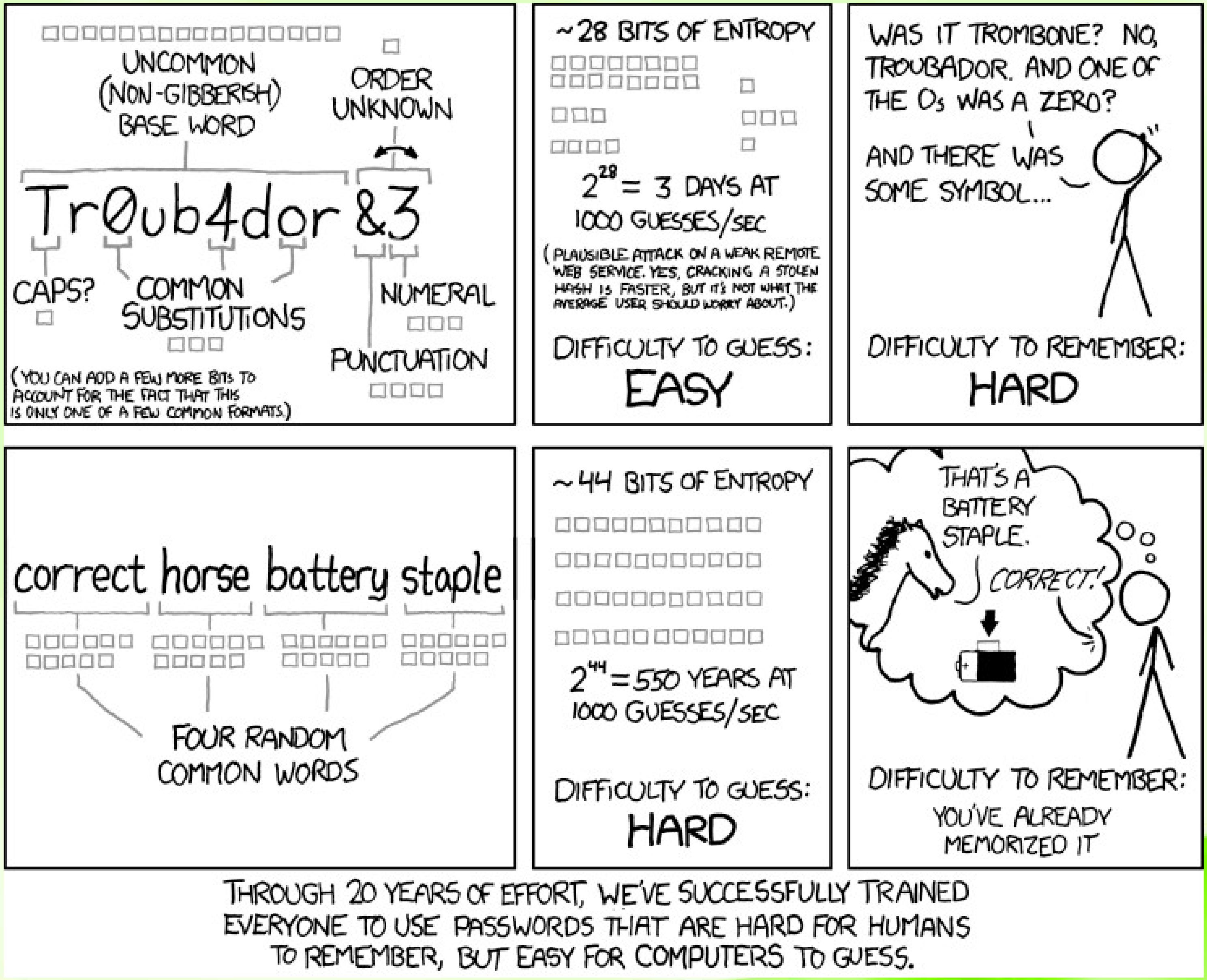
“ReallyPissedOff50BloodyBoiledCabbagesShovedUpYourArseIfYouDontGiveMeAccessnow”

Sorry, that password is already in use.

Passwords and Encryption

- People still use old unix 8 char DES encryption algorithms, circa 1970 (25 encryption passes)
- Machines a bit faster now, cracking that is nearly trivial.
- Gnu crypt now supports a number of algorithms, including sha-512 and a programmable number of passes default 5000.

http://xkcd.com/936


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~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
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
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.