Introduction to SSH

Vance Kochenderfer
Western Pennsylvania Linux Users Group
March 10, 2012

Interactive Login

- Basic syntax
  - `ssh` *`host`*
  - `ssh` *`user@host`*
- Replacement for older utilities
  - telnet
    - Plaintext password sniffing: http://www.youtube.com/watch?v=loiIh9ui26I
  - rlogin

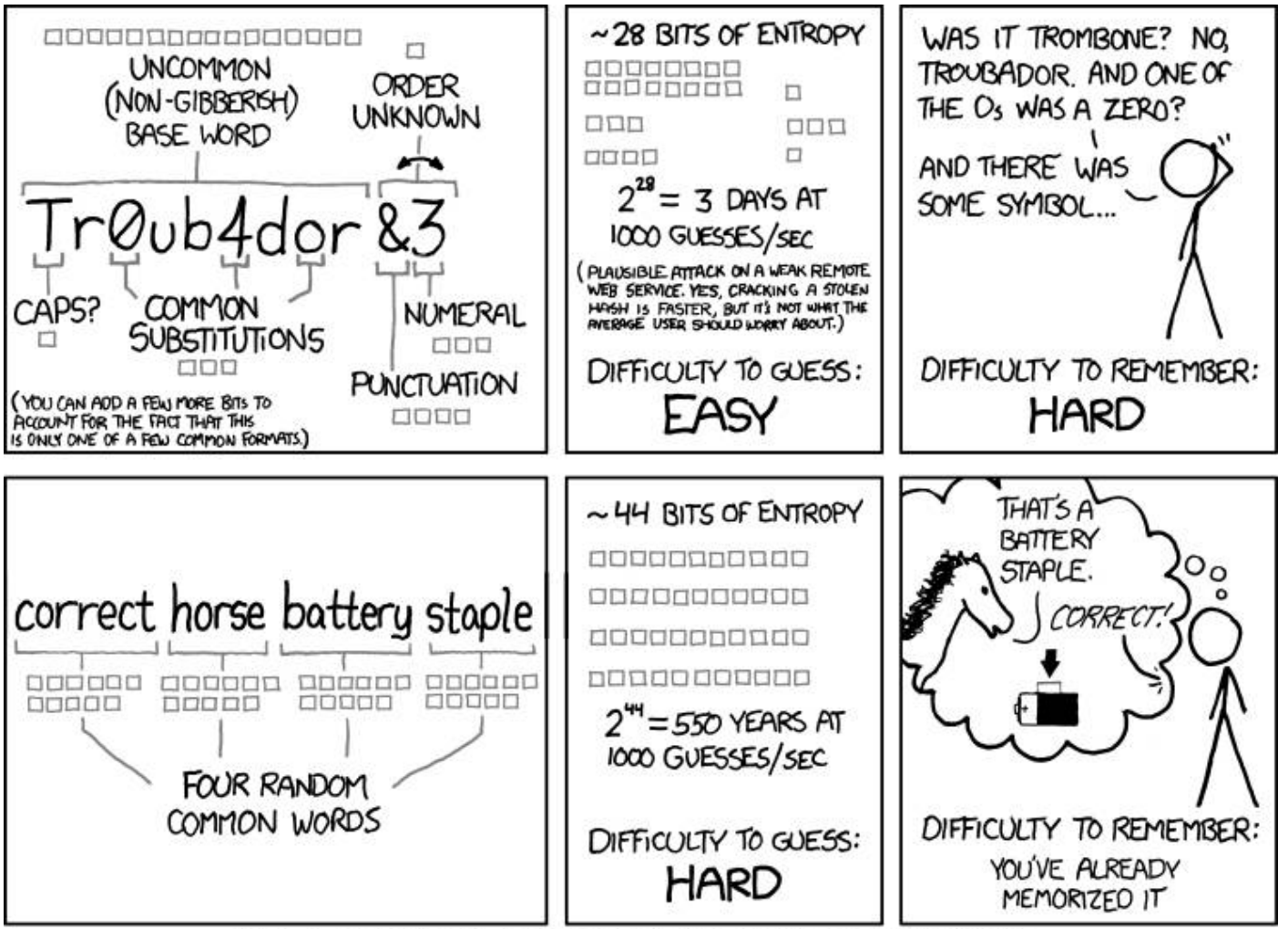
Password Authentication

- Encrypted channel is set up between client and server
- Password is transmitted over the channel
- Risk of password exposure if server is compromised
  - kernel.org: http://lwn.net/Articles/464233/

Password Authentication (2)

- Good password management
  - Never (almost) reuse passwords
  - Use a password/passphrase with sufficient length

Password Authentication (3)



Password Authentication (4)

- Good password management
  - Never (almost) reuse passwords
  - Use a password/passphrase with sufficient length
    - randwords script
    - or, a tool like KeePass, LastPass, or Password Safe
  - If necessary, write them down and keep them in your wallet:
    http://www.schneier.com/blog/archives/2005/06/write_down_your.html

Public Key Authentication

- Symmetric vs. asymmetric ciphers
  - Symmetric (aka shared secret): sender uses a key to encrypt, receiver uses same key to decrypt
  - Asymmetric: sender uses one key (public) to encrypt, receiver uses a different key (private) to decrypt
    - Public and private keys are mathematically related, but figuring out the private key is computationally hard
    - OK for everyone to know the public key, but the private key must be protected

Public Key Authentication (2)

- Setting up
  - Generate private/public key pair: ssh-keygen
  - Set a passphrase for private key - see prior notes on password management
  - Copy public key to `~/.ssh/authorized_keys` on target host (can use `ssh-copy-id` *`user@host`*)
- Security advantages
  - Even if server is compromised, attacker cannot impersonate you
    - But anyone who obtains your private key and passphrase can

Host Keys

- You've proven who *you* are, but how do you know the server is who *it* claims to be?
- Key fingerprint is displayed on first connection - key stored in `~/.ssh/known_hosts`
- Can verify fingerprint out-of-band, or via DNS
  - `dig -t SSHFP` *`host`*
  - `ssh -o "VerifyHostKeyDNS ask"` *`user@host`*
- Dire warning given if key changes

Running a Single Command

- Syntax
- `ssh` *`user@host command`*
  - `ssh root@example.com reboot`
- Command must be quoted if more than one word
- Standard input is sent over the connection
  - `cat foo | ssh` *`user@host`* `'cat > foo'` would copy file *foo* to host

Implementations

- OpenSSH is the standard version (client and server) included on Linux, *BSD, Mac OS X
- PuTTY is a popular client for Windows (also works on Linux)
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/
- Make sure whatever you use supports the SSHv2 protocol; SSHv1 has security flaws

File Transfer with scp

- Replacement for rcp
- Basic syntax: `scp` *`local_path remote_path`*
  - `scp` *`file user@host:directory/`*
  - `scp` *`file user@host:/absolute/path/`*
  - `scp` *`file user@host:directory/newfile`*
- `scp` *`remote_path local_path`* also works

File Transfer with scp (2)

- Or even: scp *`remote_path remote_path`*
  - `scp` *`user1@host1:directory/file user2@host2:directory/file`*
  - File directly copied from *host1* to *host2* if possible!
- Useful options
  - `-r` does recursive copy like cp
  - `-p` preserves timestamps and file mode like cp, but not ownership
  - `-C` uses gzip compression
  - `-l limit` does rate-limiting; *limit* in Kbit/s

File Transfer with sftp

- Replacement for ftp
- Basic syntax: `sftp` *`user@host`*
  - can also specify remote directory to start in: `sftp` *`user@host`*`:`*`directory/`*
- Commands resemble ftp - try `help`
  - `get` *`remote-path [local-path]`*
  - `put` *`local-path [remote-path]`*
    - `mget` and `mput` are just aliases for `get` and `put`
  - `ls` *`[path]`*
  - `lls` *`[ls-options [path]]`*

Disadvantages of scp/sftp

1. Cannot resume a partial transfer
2. File ownership is not preserved
3. Using glob characters (wildcards) like * can be ambiguous or insecure

- rsync can solve #1, and #2 with -o option
- tar can solve #2 and #3

Other File Transfer Options

- Linux
  - Dolphin (KDE), Nautilus (GNOME), gFTP, mc
- Windows
  - Filezilla, WinSCP
- ...and many more