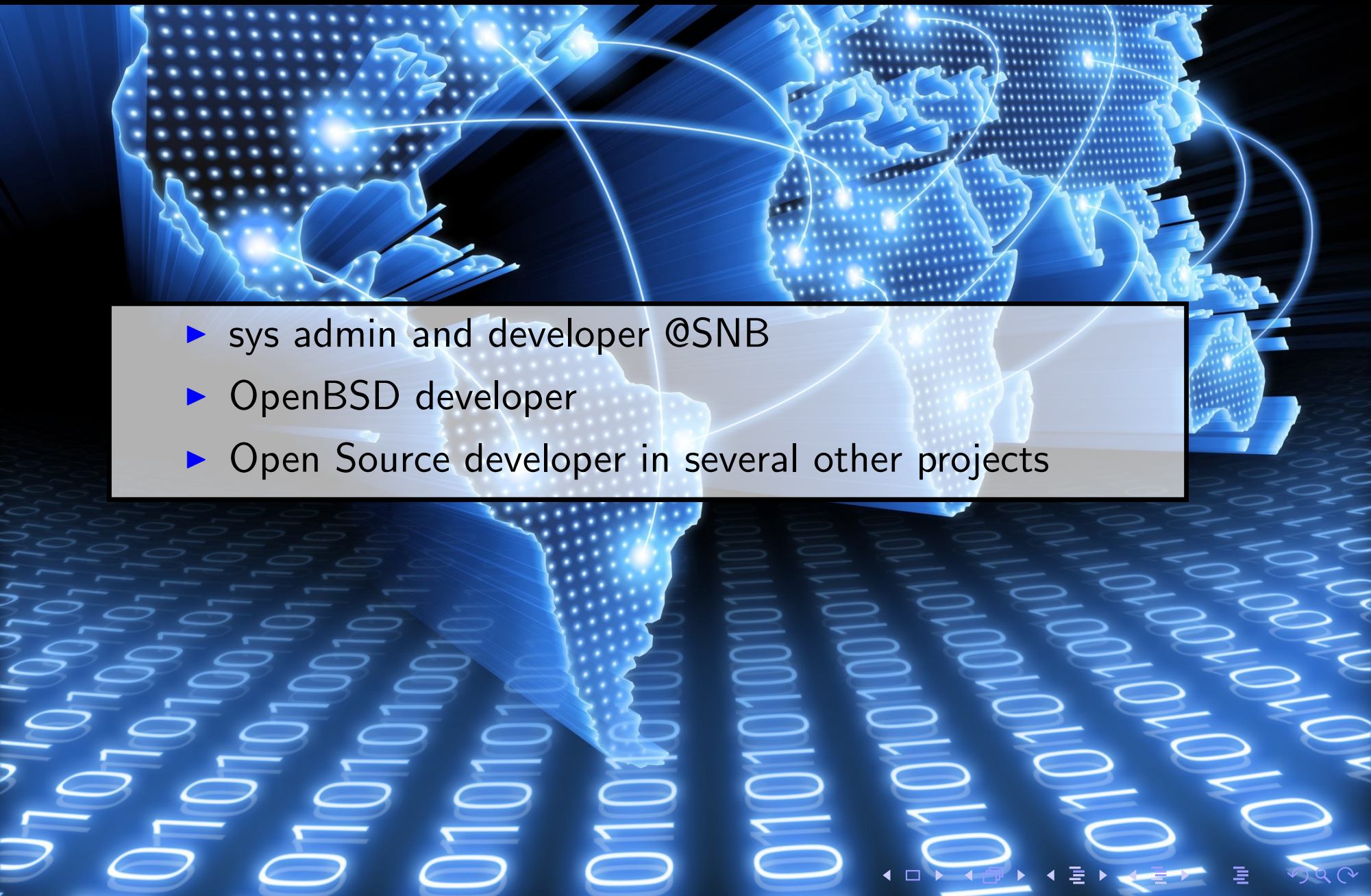
OpenSSH: keep your secrets safe

Giovanni Bechis
<giovanni@openbsd.org>

OpenSourceDay 2015

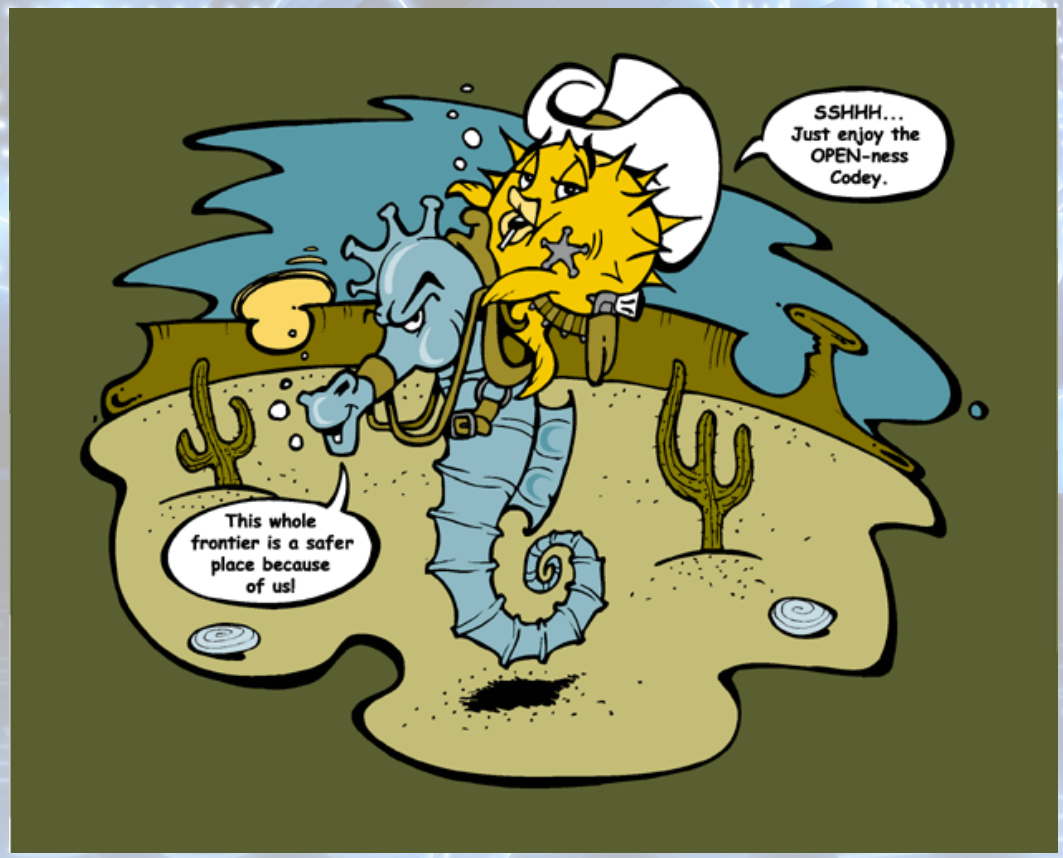
SNB Information Technology & Web Solutions

About Me



- sys admin and developer @SNB
- OpenBSD developer
- Open Source developer in several other projects

What is OpenSSH ?



Free SSH implementation,
used for secure communications and transfer files.

What is OpenSSH ?

- free license
- strong crypto inside (3Des, Blowfish, AES, Arcfour)
- X11 forwarding
- port forwarding (cryptography for plain text protocols)
- strong authentication (Public key, One time password, Kerberos)
- file transfer
- data compression

a bit of history

- 1995
  - Tatu Ylonen releases ssh-1.0.0
  - SSH Communications Security Inc.
- 1999
  - OpenSSH project birth, based on ssh-1 source code
- 2000
  - SSH version 2 protocol has been added to OpenSSH
- 2002
  - SSH added support to Solaris 9 (based on OpenSSH source code)
- 2006
  - SSH version 2 protocol has been defined standard IETF
- 2015
  - Microsoft announces support for ssh protocol in Powershell

SSH protocol

- connection starts on port 22
- client and server determine protocol version to use
- server always have private/public key pair
- public key is sent during connection phase
- client caches server's public key to prevent "man in the middle" attacks

SSH protocol

```
/*
 * Check that the versions match.  In future this might accept
 * several versions and set appropriate flags to handle them.
 */
if (sscanf(server_version_string, "SSH-%d.%d-%[^\n]\n",
    &remote_major, &remote_minor, remote_version) != 3)
    fatal("Bad remote protocol version identification: '%.100s'", buf);
debug("Remote protocol version %d.%d, remote software version %.100s",
    remote_major, remote_minor, remote_version);

active_state->compat = compat_datafellows(remote_version);
mismatch = 0;
```

The protocol version is determined based on banner

SSH protocol

- SSH-1.5 $\Rightarrow$ ssh version 1
- SSH-1.99 $\Rightarrow$ ssh version 1 and 2
- SSH-2.0 $\Rightarrow$ ssh version 2

SSH version 1

- do not use it !!
- version 1 is the original protocol version as released by Tatu Ylonen
- modified between 1995 and 1997
- final version is 1.5
- it has never become a standard
- monolithic structure

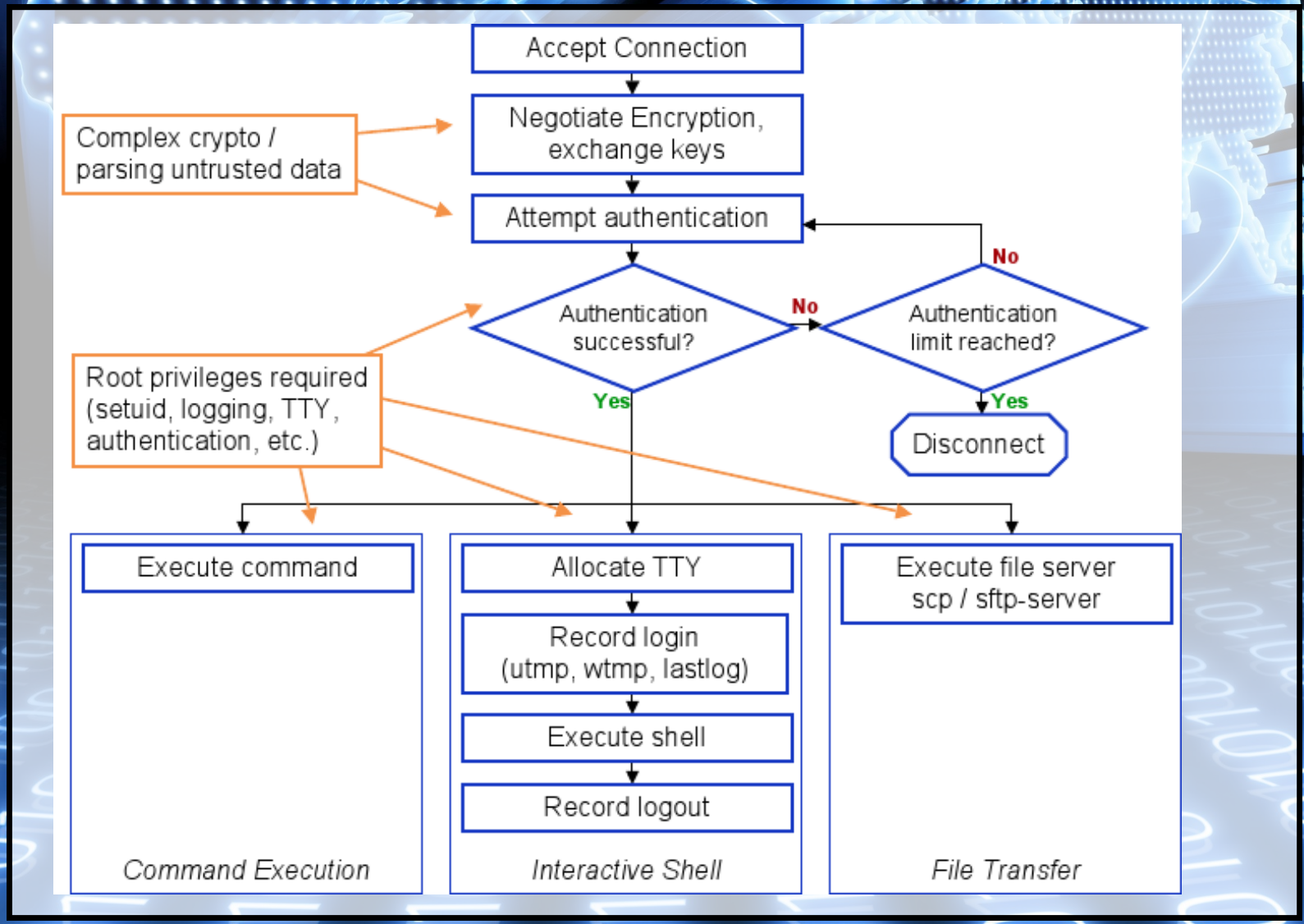
SSH version 2, modular structure

- transport protocol
  - manages cryptography, compression and integrity
  - provides "services"
- authentication protocol
  - permits the authentication of the client
  - supports many authentication methods
    - Password
    - Public key
    - Challenge-response
    - Host based
- connection protocol
  - interactive logins
  - command execution
  - port forwarding
  - X11 forwarding

SSH versions

- weak integrity checks in ssh 1.x crc
  - packets can be spoofed
  - lot of complex tricks to detect attacks
  - attacks cannot be prevented, only checked
- man in the middle attacks are easier with ssh 1.x
  - the problem is before public key exchange phase
  - key exchange with D-H in ssh 2.x removes this problem if publick keys are used
- SSH 2.x is recommended because:
  - a lot more secure
  - the protocol is an IETF standard
  - extensible protocol
  - but it has more per packet overhead

SSH anatomy


Accept Connection
Negotiate Encryption, exchange keys
Attempt authentication
Complex crypto / parsing untrusted data
Authentication successful?
No
Authentication limit reached?
No
Yes
Yes
Disconnect
Root privileges required (setuid, logging, TTY, authentication, etc.)
Execute command
Allocate TTY
Record login (utmp, wtmp, lastlog)
Execute shell
Record logout
Execute file server scp / sftp-server
Command Execution
Interactive Shell
File Transfer

SSH code security

- code audit
- input validation
  - no buffer overflows
  - no memory leaks
- no unsecure functions used
  - all unsecure functions has been removed, and the secure counterpart has been used instead
  - strcpy $\Rightarrow$ strlcpy
  - atoi $\Rightarrow$ strtonum
- sandboxing techniques are used
  - setrlimit(2)
  - pledge(2)
  - other sandboxing techniques in portable version
- complex code removal
  - openssl ASN.1 parser has been replaced by a minimal parser

SSH code security

- process separation
  - monitor process runs as uid 0
  - slave process chroots in /var/empty
  - slave process is executed as dedicated user in pre-auth or as logged-in user in post-auth phase
- changes to the protocol
  - protocol compression is activated only in post-auth phase to minimize the effects of possible bugs in zlib
- sshd double exec is used to better use mitigation techniques available in some operating systems (OpenBSD, recent Windows versions, Linux + patches, ...)

OpenSSH configuration

Server configuration file is /etc/ssh/sshd_config

- Port 22
- ListenAddress 0.0.0.0
- Protocol 2
- UsePrivilegeSeparation sandbox
- StrictModes yes
- PermitRootLogin prohibit-password

OpenSSH configuration

Some other useful feautures

- certificate and S/Key authentication
- chroot
- X11 forwarding
- port forwarding
- socks proxy
- persistent connections
- visual fingerprints
- vpn

Using certificates

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:uuNH/ECxy0i1T19EHXovdbfs3e134b4Yv++PpB1dPtc giovanni@ssh-test.org
The key's randomart image is:
+---[RSA 2048]----+
|              ..o.|
|         o     o .|
|        . +   o .+|
|       . + .   +.=|
|      . =S+ . ..oo|
|       ..* . . .=*|
|       .. o   .+oE|
|       ... .  ++==|
|      .oo    ..+*%|
+----[SHA256]-----+

$ ssh-add
Enter passphrase for /home/giovanni/.ssh/id_rsa:
Identity added: /home/giovanni/.ssh/id_rsa (/home/giovanni/.ssh/id_rsa)
$
```

sftp chroot

In some situations you should not permit a user to explore the whole filesystem

```
Subsystem  sftp  internal
Match user giovanni
  ForceCommand internal-sftp
  ChrootDirectory /chroot
```

port forwarding

If a firewall is blocking some services you need to access, you could use a machine that will act as a bridge.

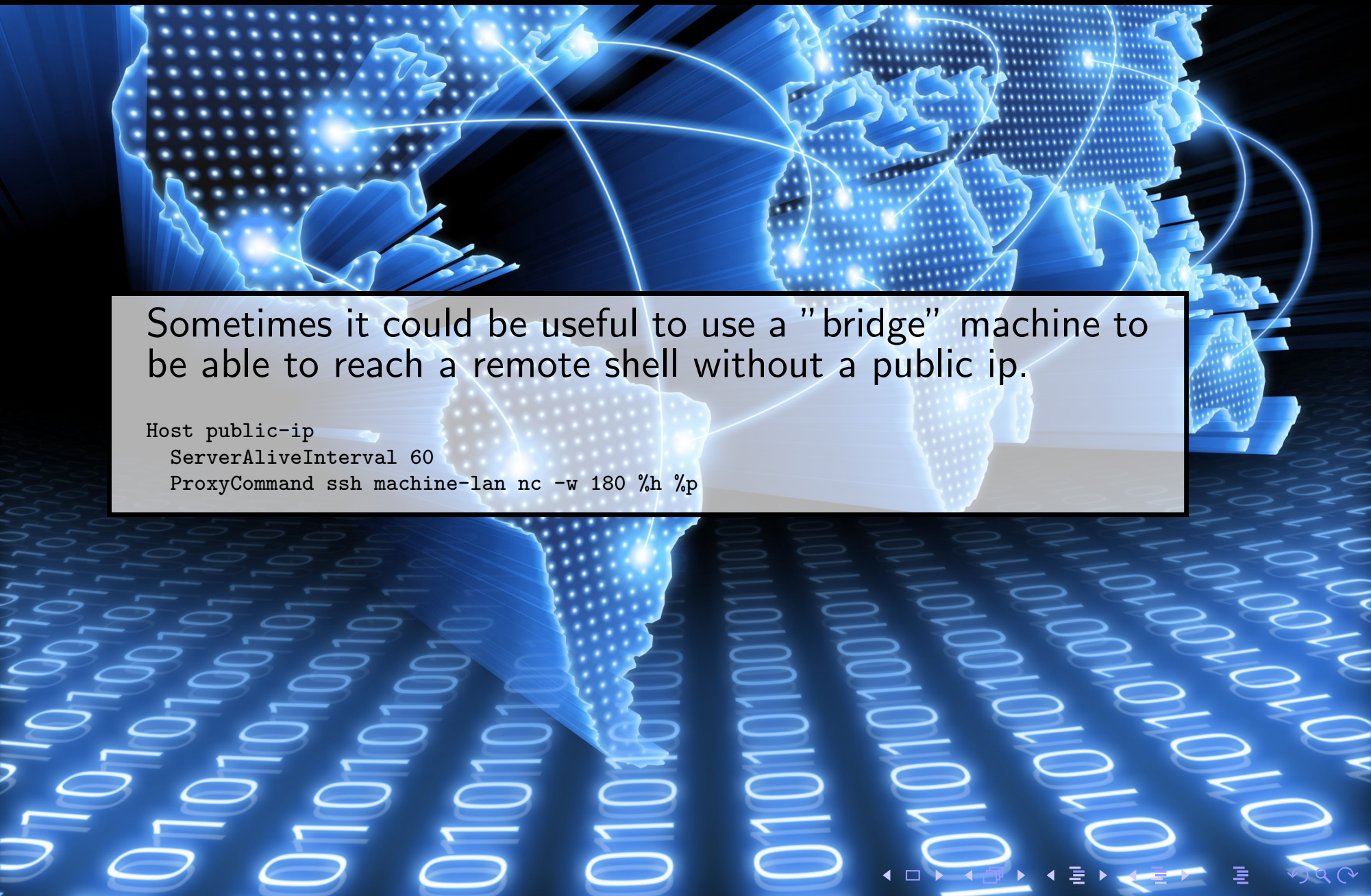
```
ssh -L 9025:mail.example.net:25 shell.example.net
```

dynamic port forwarding

Using "dynamic port forwarding" you can tell sshd to act as a socks proxy. This way you can use Firefox to browse the internet with the public ip address of the remote machine.

```
ssh -D 8080 shell.example.net
```

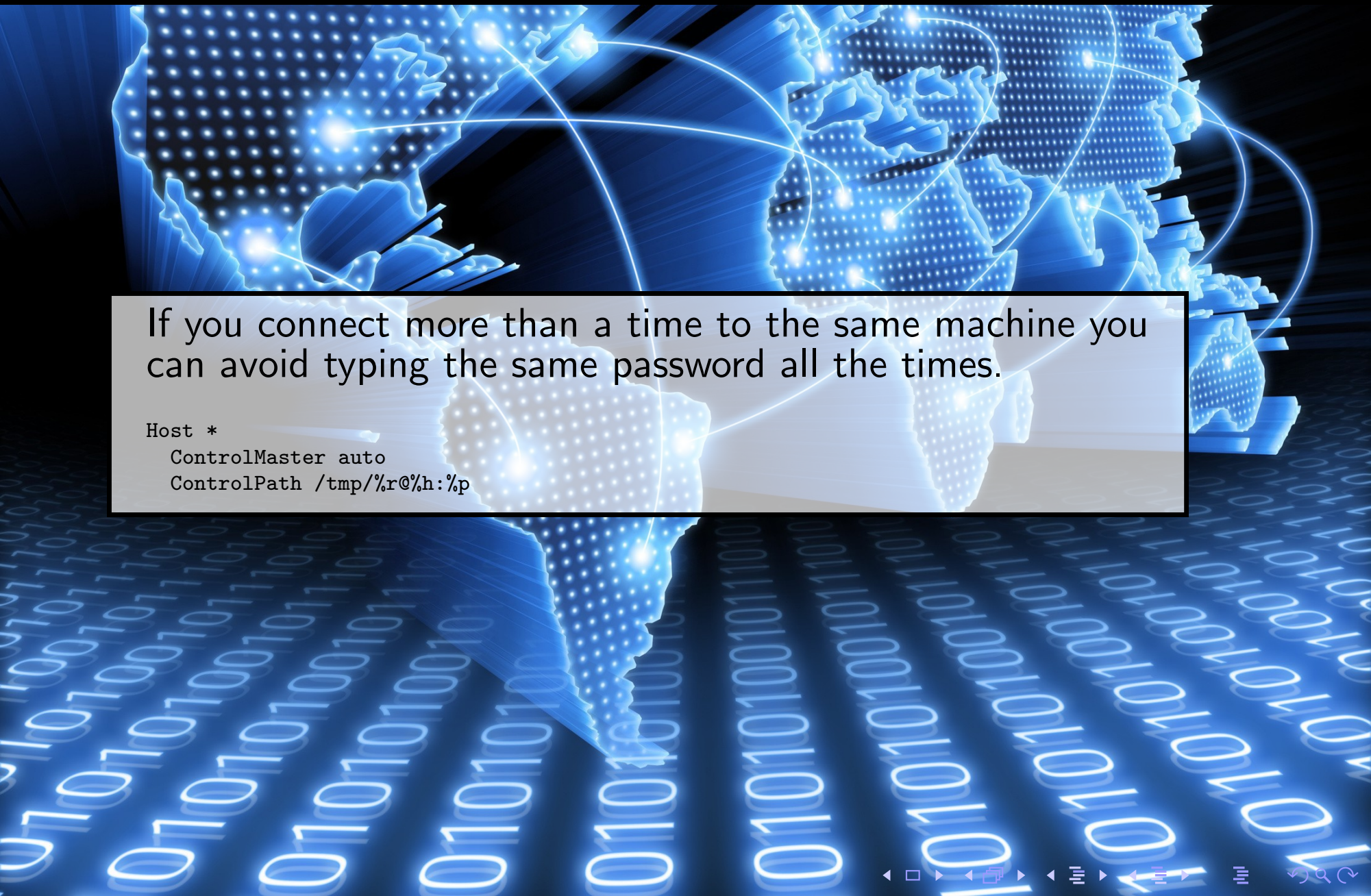
using a "bridge" machine



Sometimes it could be useful to use a "bridge" machine to be able to reach a remote shell without a public ip.

```
Host public-ip
  ServerAliveInterval 60
  ProxyCommand ssh machine-lan nc -w 180 %h %p
```

persistent connections



If you connect more than a time to the same machine you can avoid typing the same password all the times.

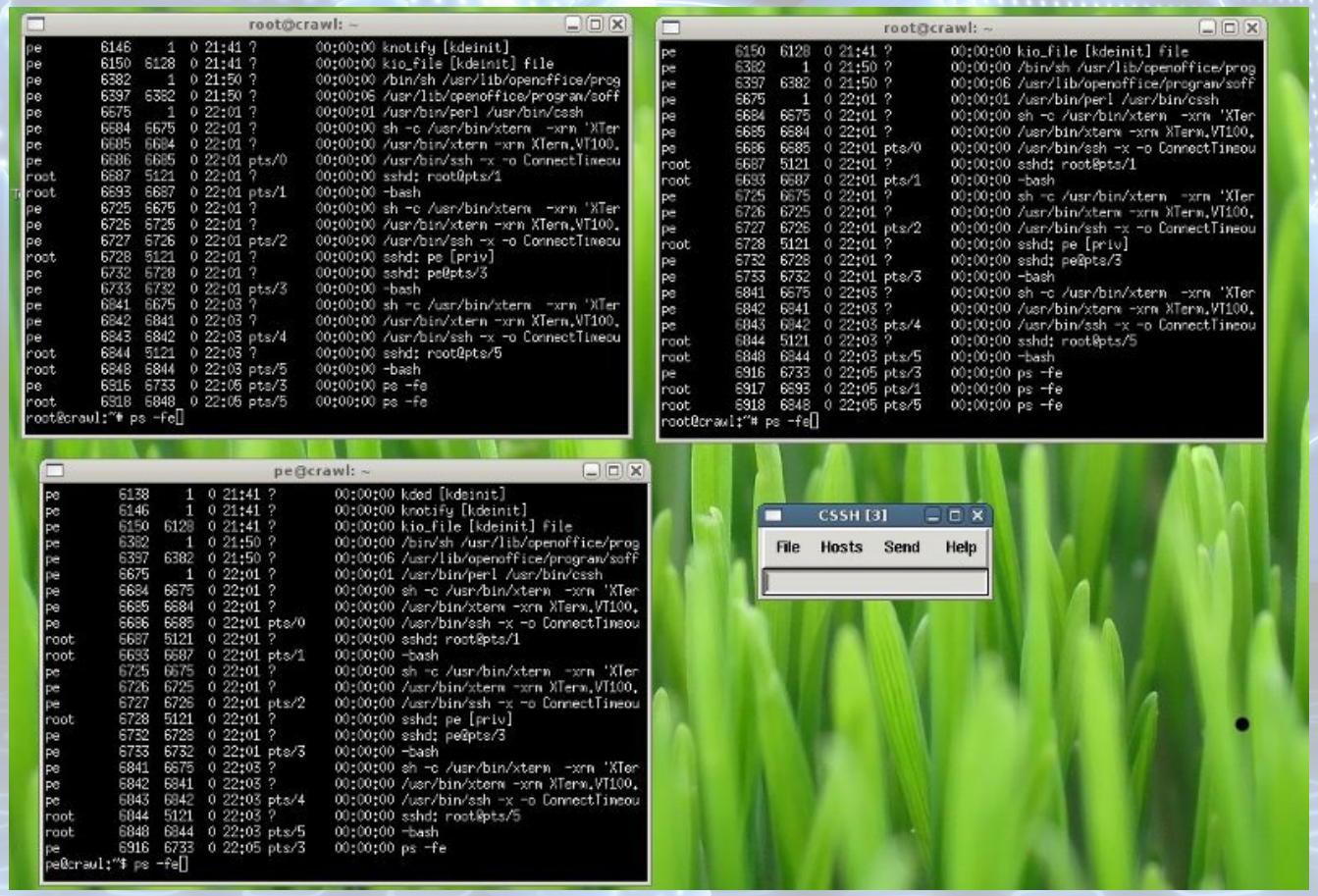
```
Host *
  ControlMaster auto
  ControlPath /tmp/%r@%h:%p
```

visual fingerprints

Using the visualfingerprint parameter in /etc/ssh/ssh_config you a visual fingerprint of the server you connect to will be printed.

```
$ ssh shell.example.net
+---[RSA 2048]----+
|              ..o.|
|         o     o .|
|        . +   o .+|
|       . + .   +.=|
|      . =S+ . ..oo|
|      ..* . . .=*|
|      .. o   .+oE|
|      ... .  ++==|
|     .oo    ..+*%|
+----[SHA256]-----+
```

ClusterSSH



ClusterSSH opens terminal windows with connections to specified hosts and an administration console. Any text typed into the administration console is replicated to all other connected and active windows.


SNB
Information Technology
& Web Solutions
TOP
SECRET