Skip to main content.

BSD Now

A Weekly BSD Podcast - News, Interviews and Tutorials

Subscribe on Youtube « | Subscribe with iTunes « | RSS: MP3 | Video | HD Video | HD Torrent Feed
Navigation: Home | Episodes | About | Live | Shirt | Contact Us | Tutorials

Combining SSH and tmux

2013-11-20

Live demo in BSD Now Episode 012 | Originally written by TJ for bsdnow.tv | Last updated: 2014/03/01

NOTE: the author/maintainer of the tutorial(s) is no longer with the show, so the information below may be outdated or incorrect.

Today I'll be covering two topics in one. SSH and tmux go hand-in-hand in my daily life, so combining them into one tutorial makes a lot of sense. SSH, specifically OpenSSH, is used by

millions of people every day. It's been ported to many operating systems and has become the de facto standard for remote administration of UNIX-like systems. Its roots come from the OpenBSD project, but they also develop a portable version for other OSes. FreeBSD, NetBSD and DragonFlyBSD all provide their own specific SSH documentation. Whatever platform you use it on, you'll want to become familiar with a number of commands and utilities, some of which include:

- ssh - the client command
- sshd - the daemon/server component
- scp - a tool to securely copy files
- sftp - an FTP client/service alternative
- ssh-keygen - a private/public key pair generator

There are a few others, but we'll mostly be focusing on these five. This tutorial will cover basic usage and configuration of an SSH daemon and client, but if you'd like to really learn OpenSSH inside and out, I recommend reading SSH Mastery. It's a very good book and can teach you all the great tricks you'd probably never think of trying. It's possible to create a VPN with SSH and tunnel all your traffic through it. You can run graphical applications through it and much, much more.

SSH

We'll start with the daemon configuration. Usually, the file is located at /etc/ssh/sshd_config. A default configuration is fine for now, but definitely give the man page a read for some extreme

tweaking possibilities. Here is an example of one of mine:

```
Port 2222
ListenAddress 1.2.3.4
AddressFamily inet
Protocol 2
HostKey /etc/ssh/ssh_host_ed25519_key
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 10
PermitRootLogin no
StrictModes yes
MaxAuthTries 2
MACs hmac-sha2-256-etm@openssh.com,hmac-sha1
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes256-ctr
RSAAuthentication no
PubkeyAuthentication yes
AuthorizedKeysFile  .ssh/authorized_keys
RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreRhosts yes
PasswordAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
KerberosAuthentication no
GSSAPIAuthentication no
AllowUsers tj allan kris
AllowAgentForwarding yes
```

```
AllowTcpForwarding yes
X11Forwarding yes
X11DisplayOffset 10
PrintLastLog yes
TCPKeepAlive yes
UsePrivilegeSeparation sandbox
Subsystem   sftp    /usr/libexec/sftp-server
```

Be sure to read the man page before copying and pasting anything from that. You might lock yourself out of the machine otherwise. How you start the server will depend on what platform you're using, but for most of the BSDs, the following will do:

```
# /etc/rc.d/sshd start
```

Now the service will be listening on port 22 (unless you changed it). I'm assuming you've already created a normal user for daily activities. While it's possible to login as root over SSH, it's highly discouraged from a security standpoint. Assuming networking is working on both the server and the client (and you've allowed SSH to pass through any firewalls), you should be able to issue the following command on the client computer:

```
$ ssh username@serverip
```

And be presented with a password prompt. Once you login, you'll be at a shell for that server. To securely transfer files between the client and server, you could do something like:

```
$ scp ~/myfile username@serverip:/tmp
```

This would copy the "myfile" file in your home directory to /tmp on the server. You can reverse the addresses to copy from the server to the client and so on. If you prefer a more interactive approach to swapping files, SFTP might be the right tool for you.

```
$ sftp username@serverip
```

Now you'll have a new SFTP prompt.

```
Connected to bsdnow.tv.
sftp>
```

From here, you can do normal FTP commands to manipulate files: cd, lcd, rm, get, put, pwd, lpwd, ls, lls, etc. See the sftp man page linked above for more details. As of OpenSSH 6.3, you can also resume partially-downloaded files. If you want to **only** allow SFTP, see our tutorial for that.

Before we move on to tmux, there are a couple more things I'd recommend doing for your SSH setup. The first is setting up a public/private key pair instead of using passwords. This makes logging in to remote systems faster and safer. From your remote shell session, let's generate a key to use. I prefer Ed25519 keys, but RSA, DSA and ECDSA are also options. While SSHed into the server, we'll make the ~/.ssh directory.

```
$ mkdir ~/.ssh
```

From the client PC, let's generate a key pair.

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/tj/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tj/.ssh/id_ed25519.
Your public key has been saved in /home/tj/.ssh/id_ed25519.pub.
The key fingerprint is:
67:5d:f7:fa:5f:4c:1d:fb:cb:f8:2f:11:7a:4f:9e:15 tj@bsdnow.tv
The key's randomart image is:
+--[ED25519  256--+
|                 |
|                 |
|              ...|
|           . ..E=|
|        S o .. o=|
|         o  . o++|
|             ..=*|
|              +o=|
|             ..=*|
+-----------------+
```

On the client system, we'll use scp to copy over the public key to the server.

```
$ scp ~/.ssh/id_ed25519.pub username@serverip:.ssh/authorized_keys
```

You can disable password authentication in sshd_config by changing the "PasswordAuthentication yes" line to "PasswordAuthentication no" after doing this step. Be sure to restart the daemon for changes to take effect. At this point you should be able to SSH into the server without entering your password. It's using the key pair we just generated for authentication now.

---

tmux

Ok, say you're SSHed into your server and getting work done, but suddenly your connection drops. What happens to the SSH session? It's gone.

Say you're SSHed into your server and want to leave something running in the background while you disconnect. Not happening.

This is where tmux comes in to save the day. What is tmux, you ask? According to the project page: "It lets you switch easily between several programs in one terminal, detach them (they keep running in the background) and reattach them to a different terminal - and a lot more."

Traditionally, sysadmins used a tool called "screen" to achieve a similar effect. Unfortunately, screen was not released under a free software license. After years of having no other option, Nicholas Marriott decided to write an alternative to screen that was licensed in a more favorable manner. In 2009, the first version of tmux was released.

If you're using OpenBSD, tmux is part of the base system. Theo had a story about that. For the rest of us, we need to install tmux from ports or pkgsrc or however you'd like to get it. Let's SSH into the server, install tmux and fire it up.

```
$ tmux
```

You should see a prompt similar to this:

Anything you do in this tmux session will still be there if your connection drops or you close the terminal. The keybindings for tmux are different than screen's, but there is a workaround for that. You can edit your config file to mimic screen's keybindings if you don't want to relearn anything. I've provided a copy of my ~/.tmux.conf in case you want to do it that way. It also has some other usability improvements. If you'd like to use that one instead of the default, do:

```
$ ftp -o ~/.tmux.conf http://www.bsdnow.tv/patches/tmuxconf
```

If you get disconnected while working in tmux, you can reconnect and run:

```
$ tmux attach
```

And you'll be right back where you were! It's possible to split a terminal into multiple windows, open new tabs, reorder and close tabs and much more. See the keybindings list for a complete list of things you can do.

Latest News

New announcement

2017-05-25

Hi, Mr. Dexter. Also, we understand that Brad Davis thinks there should be more real news....

Two Year Anniversary

2015-08-08

We're quickly approaching our two-year anniversary, which will be on episode 105. To celebrate, we've created a unique t-shirt design, available for purchase until the end of August.

Shirts will be shipped out around September 1st. Most of the proceeds will support the show, and specifically allow us to buy...

New discussion segment

2015-01-17

We're thinking about adding a new segment to the show where we discuss a topic that the listeners suggest. It's meant to be informative like a tutorial, but more of a "free discussion" format. If you have any subjects you want us to explore, or even just a good name...

How did you get into BSD?

2014-11-26

We've got a fun idea for the holidays this year: just like we ask during the interviews, we want to hear how all the viewers and listeners first got into BSD. Email us your story, either written or a video version, and we'll read and play some of them for...

1 | 2 | 3 | 4 | 5 | Next >

Episode 228: The Spectre of Meltdown

2018-01-10

Direct Download:HD VideoMP3 AudioTorrent This episode was brought to you by Headlines Meltdown Spectre Official Site Kernel-memory-leaking Intel processor design flaw forces Linux, Windows redesign Intel’s official response The Register mocks intels response with pithy annotations Intel’s Analysis PDF XKCD Response from FreeBSD FreeBSD's patch WIP Why Raspberry Pi isn’t vulnerable to Spectre or Meltdown Xen mitigation patches Overview of affected FreeBSD Platforms/Architectures Groff's response We’ll...

Episode 227: The long core dump

2018-01-03

Direct Download:HD VideoMP3 AudioTorrent This episode was brought to you by Headlines NetBSD 7.1.1 released The NetBSD Project is pleased to announce NetBSD 7.1.1, the first security/critical update of the NetBSD 7.1 release branch. It represents a selected subset of fixes deemed important for security or stability reasons. Complete source and binaries for NetBSD 7.1.1...

Episode 226: SSL: Santa’s Syscall List

2017-12-27

Direct Download:HD VideoMP3 AudioTorrent This episode was brought to you by Headlines FreeBSD Q3 Status Report 2017 FreeBSD Team Reports FreeBSD Release Engineering Team Ports Collection The FreeBSD Core Team The FreeBSD Foundation Projects FreeBSD CI Kernel Intel 10G iflib Driver Update Intel iWARP Support pNFS Server Plan B Architectures AMD Zen (family 17h) support Userland Programs Updates to GDB Ports FreeBSDDesktop OpenJFX 8 Puppet Documentation Absolute FreeBSD, 3rd Edition Manual Pages Third-Party Projects The nosh Project FreeBSD...

Episode 225: The one true OS

2017-12-20

Direct Download:HD VideoMP3 AudioTorrent This episode was brought to you by Headlines TrueOS stable release 17.12 We are pleased to announce a new release of the 6-month STABLE version of TrueOS! This release cycle focused on lots of cleanup and stabilization of the distinguishing features of TrueOS: OpenRC, boot speed, removable-device...