Secure Linux to Linux Shares

STL!/unix/usr/group

Saint Louis Unix Users Group

https://www.sluug.org/

Stan Reichardt

2019-08-14

WHO

Anybody here using ssh?

Anybody here using sshfs?

Secure Linux to Linux Shares

Secure Linux to Linux Shares

(Using **sshfs** instead of SAMBA)

WHAT

Mostly about using **sshfs** to securely share directories of files between Linux machines, instead of sharing using SAMBA.

Demonstration will show useage for both:

- Command Line Interface (CLI) and
- Graphical User Interface (GUI)

WHAT

Using **sshfs** instead of SAMBA.

WHY

Using SAMBA **may be needed.**

- Family and friends may be using MS Windows.
- Work environment.

Using SAMBA **may not be needed**.

- All my machines are running Linux.

Using SAMBA **may be broken out of the box**.

- "*Just works out of the box*" may not be the experience.
- This is what I have been experiencing for about last two years.
- Known to be a wide spread useability issue.

Using SAMBA **may be difficult to fix or set up**.

- An exercise in fixing something others may need.
- An exercise in fixing something I do not need.
- An exercise in fixing something I may never need.
- An exercise in fixing something I do not want.

WHAT

```
user@example:~$ whatis ssh
ssh (1)          - OpenSSH SSH
```

`client (remote login program)`

WHAT

```
FUSERMOUNT(1) Filesystem in Userspace (FUSE)              FUSERMOUNT(1)

NAME
       fusermount - unmount FUSE filesystems
...
...
More information about fusermount and the FUSE project can be found at
 http://fuse.sourceforge.net/    which then went to   https://github.com/libfuse/libfuse

> The reference implementation of the Linux FUSE (Filesystem in Userspace) interface
>
> About
>
> FUSE (Filesystem in Userspace) is an interface for userspace programs
> to export a filesystem to the Linux kernel. The FUSE project consists
> of two components: the fuse kernel module (maintained in the regular
> kernel repositories) and the libfuse userspace library (maintained in
> this repository). libfuse provides the reference implementation for
> communicating with the FUSE kernel module.
>
> A FUSE file system is typically implemented as a standalone application
> that links with libfuse. libfuse provides functions to mount the file
> system, unmount it, read requests from the kernel, and send responses
> back. libfuse offers two APIs: a "high-level", synchronous API, and a
> "low-level" asynchronous API. In both cases, incoming requests from
> the kernel are passed to the main program using callbacks. When using
> the high-level API, the callbacks may work with file names and paths
> instead of inodes, and processing of a request finishes when the callback
> function returns. When using the low-level API, the callbacks must work
> with inodes and responses must be sent explicitly using a separate set
> of API functions.
```

WHAT

LIBFUSE

- Written in C
- Operating system Unix Unix-like Type File system driver
- License GPL for kernel part,
- LGPL for Libfuse,
- Simplified BSD on FreeBSD,
- ISC license on OpenBSD

Website github.com/libfuse/libfuse

WHAT

SSHFS
https://en.wikipedia.org/wiki/SSHFS

In computing, SSHFS (SSH Filesystem) is a filesystem client to mount and interact with directories and files located on a remote server or workstation over a normal ssh connection. The client interacts with the remote file system via the SSH File Transfer Protocol (SFTP), a network protocol providing file access, file transfer, and file management functionality over any reliable data stream that was designed
as an extension of the Secure Shell protocol (SSH) version 2.0.

The current implementation of SSHFS using FUSE is a rewrite of an earlier
version. The rewrite was done by Miklos Szeredi, who also wrote FUSE.

WHAT

Filesystem in Userspace
https://en.wikipedia.org/wiki/Filesystem_in_Userspace

Filesystem in Userspace (FUSE) is a software interface for Unix and Unix-like computer operating systems that lets non-privileged users create their own file systems without editing kernel code. This is achieved by running file system code in user space while the FUSE module provides only a "bridge" to the actual kernel interfaces.

FUSE is available for Linux, FreeBSD, OpenBSD, NetBSD (as puffs), OpenSolaris, Minix 3, Android and macOS.[2]

FUSE is free software originally released under the terms of the GNU General Public License and the GNU Lesser General Public License.

WHERE

“**sshfs** is a filesystem based on the SSH file transfer protocol.”

“It is used on a client system, i.e. you need to install **sshfs** package on your local computer/laptop powered by CentOS/RHEL/Ubuntu/Debian/Arch/LinuxMint Linux. No need to install anything on server (server1.somewhere.org).”

“You only need an openssh server installed on server side.”

WHERE

```
+-----------------------+                    +----------------+
| server1.somewhere.org |     ssh/sshfs      | Client         |
| /home/username        |-----<------>-------| SSHFS          |
+-----------------------+                    | /mnt/server1   |
                                             +----------------+
```

WHERE

Our example:

```
     +------------------------+                    +-------------------+
     | Server                 |     ssh/sshfs      | Client            |
     | bock.sluug.org         |----<------->-------| SSHFS             |
     | /home/stanr            |                    |/home/stan/STANR   |
     +------------------------+                    +-------------------+
```

```
   user@example:~$ mkdir /home/stan/STANR

   user@example:~$ sshfs -C stanr@bock.sluug.org:/home/stanr -o port=2206  ~~????~~
   ...OR
   user@example:~$ sshfs -C stanr@206.197.251.210:/home/stanr ~/STANR -o port=2206
```

Or use an alias

Such as connect_stanr='sshfs -C stanr@206.197.251.210: ~/STANR -o port=2206'

```
   user@example:~$ connect_stanr

   user@example:~$ ls /home/stan/STANR

   user@example:~$ cd /home/stan/STANR

   user@example:~$ ls
```

Prerequisites

Configuring firewall

NEED port access

NEED port access ~ Don't let your firewall block you

NEED port 22 or **whatever** you use for your ssh connections

HOW

- NEED directory
- NEED empty mount point (empty directory)
- NEED to make directory

```
mkdir (1)             - make directories
mkdir (2)             - create a directory
```

```
user@example:~$ mkdir PINK
user@example:~$ mkdir BLUE
user@example:~$ mkdir STANR
user@example:~$ mkdir WWW
```

HOW

NEED client side software

```
user@example:~$ clear
user@example:~$ aptitude show ssh
....
user@example:~$ clear
user@example:~$ sudo apt install ssh
```

HOW

NEED server side software

```
user@example:~$ clear
user@example:~$ aptitude show openssh-server
....

user@example:~$ whatis openssh-server
openssh-server: nothing appropriate.

user@example:~$ sudo apt install openssh-server
```

HOW

```
user@example:~$ whatis sshfs
sshfs (1) - filesystem client based on ssh
```

```
SSHFS: Provides access to a remote filesystem through SSH
```

HOW

```
user@example $ whatis fusermount
Fusermount (1)- unmount FUSE filesystems
```

HOW

Automating the connection

Can be scripted.

Scripting eases complexity.

Aliases work about as well.

Aliases are all that I use.

HOW

```
user@example: $ view ~/.bash_aliases ## (partial)

alias connect_mercury='sshfs -C stan@192.168.1.11: ~/MERCURY'
alias connect_venus='sshfs -C stan@192.168.1.12: ~/VENUS'
alias connect_public='sshfs -C stan@192.168.1.13: ~/Public'
alias connect_earth='sshfs -C stan@192.168.1.13: ~/EARTH'
alias connect_mars='sshfs -C stan@192.168.1.14: ~/MARS'
alias connect_stanr='sshfs -C stanr@206.197.251.210: ~/STANR -o port=2206'
alias connect_www='sshfs -C stanr@206.197.251.210:/srv/www ~/WWW -o port=2206'
#
alias drop_earth='fusermount -u ~/EARTH'
alias drop_mercury='cd; fusermount -u ~/MERCURY'
alias drop_mars='cd; fusermount -u ~/MARS'
alias drop_public='cd; fusermount -u ~/Public'
alias drop_stanr='cd; fusermount -u ~/STANR'
alias drop_venus='cd; fusermount -u ~/VENUS'
alias drop_www='cd; fusermount -u ~/WWW'
```

HOW

https://linuxize.com/post/how-to-use-sshfs-to-mount-remote-directories-over-ssh/

Conclusions

“In this session, you have learned how to use SSHFS to mount a remote directory over SSH. This can be useful when you want to interact with the remote files using your local machine applications.

For a complete list of the sshfs options, type man sshfs [or (p)info] in your terminal.

You may also want to restrict user access to their home directory by setup up an SFTP Chroot Jail environment and change the default SSH port to add an extra layer of security to your server.”

HOW

REFERENCES:

http://fuse.sourceforge.net/

https://en.wikipedia.org/wiki/SSHFS

https://linux.die.net/man/1/sshfs

https://mintcast.org/2019/06/19/mintcast-311-ssh-mp3/

https://www.youtube.com/watch?v=u-X3q6CmsKc (poor sound quality)

https://linuxize.com/post/how-to-use-sshfs-to-mount-remote-directories-over-ssh/

.

DEMONSTRATION

Demonstration

1st with CLI

2nd with GUI

What Are Your Questions?

What are your questions?

Secure Linux to Linux Shares

STL!/unix/usr/group

Saint Louis Unix Users Group

https://www.sluug.org/

Stan Reichardt

2019-08-14