SPOC Tunnel BASH/ZSH Function

This document includes two main sections.

1. The steps needed to run `sshuttle` with domain-specific-dns (which forwards only requests for spoc.charterlab.com to the spoc-jumphost).
2. A function that can be added to your zsh/bash profile that allows a simple, one-step fingerprint authentication for the `sshuttle` command. This is accomplished by using the 1Password CLI utiliity "op" to read the appropriate passwords and securely pass them to the `sshuttle` utlity. I have marked the steps for this method as "OPTIONAL".

Requirements

You will need to set up the following:

- Sudo Privileges
- Install Homebrew
- Install SSHuttle

- Install sshpass
- Add your SPOC password to the MacOS Keychain
- Set up custom DNS Resolver for the `spoc.charterlab.com` domain

Set up Sudo privileges

You will need to modify /etc/sudoers if you have not already so you can run commands as a privileged user.

The configuration below is OPTIONAL:

```
# Cmnd alias specification
Cmnd_Alias CUSTOM = /usr/local/bin/sshuttle, /usr/bin/pgrep, /usr/bin/pkill, /

# User specification
    # root and users in group wheel can run anything on any machine as any use
root        ALL = (ALL) ALL
    # users in group admin can run any command from any machine as any user
    # users from group admin can also run any command from the CUSTOM alias or
%admin      ALL = (ALL) ALL : ALL = (ALL) NOPASSWD: CUSTOM
```

Install Homebrew (MacOS) package manager

- Homebrew Installation Instructions
- Installation Script:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
```

Install SSHuttle utility from Homebrew

```
brew install sshuttle
```

Install SSHPass utility from Homebrew

This utility allows you to forward a password to the `ssh` command. This example uses an environment variable that in turn uses a password manager to securely pipe password information to `sshuttle`.

WARNING: There are methods to invoke `sshpass` that **ARE NOT** secure. As such, the utility is not available directly from homebrew. I use a custom tap and formula hosted on my personal github account.

```
brew tap ajanis/custombrew
brew install ajanis/custombrew/sshpass
```

Storing and Retrieving your password securely using the MacOS Keychain

You can store your SPOC VPN password in your MacOS Keychain. Even if you use a password manager, securely storing a password in the MacOS Keychain will give you a secure method of including the password in a script or CLI utility.

Adding your password to the MacOS Keychain

```
security add-generic-password -s "SPOC VPN" -a "${USER}" -w
```

Retrieving your password from the MacOS Keychain

```
security find-generic-password -s "SPOC VPN" -a "${USER}" -w
```

Modify your sshuttle command to send DNS requests for ONLY 'spoc.charterlab.com' to the sshuttle jumphost.

This will prevent your system from trying other nameservers for `*.spoc.charterlab.com` domains.

(Credit to Josh Hurtado for these instructions)

Create a domain-specific-DNS configuration for the SPOC Lab

Run the following commands as a priviliged / root user on the MacOS machine:

Create the directory /etc/resolver

```
sudo mkdir /etc/resolver
sudo chmod 0774 /etc/resolver
```

Create the resolver file for SPOC at /etc/resolver/spoc.charterlab.com

```
sudo echo 'search spoc.charterlab.com spoc.local nameserver 172.22.73.19' > /e
```

Verify the new resolver for spoc.charterlab.com is present with the following command

(You will have to scroll down a bit to find the correct resolver. An example is provided of the expected output)

```
sudo scutil --dns
```

Example Output:

```
resolver #8
  domain   : spoc.charterlab.com
  search domain[0] : spoc.charterlab.com
  search domain[1] : spoc.local
  search domain[2] : nameserver
  search domain[3] : 172.22.73.19
  flags    : Request A records, Request AAAA records
  reach    : 0x00000000 (Not Reachable)
```

Create files for `allow` and `deny` that will be used by `sshuttle`

The `--dns` flag sends *all* requests to the **sshuttle jumphost**. You will need to modify your `sshuttle` command so that *ONLY* requests for `spoc.charterlab.com` will be handled by the **sshuttle jumphost**.

Create the sshuttle ALLOW file by running the following command

```
echo << EOF >> ~/.spoc.allow.txt
44.0.0.0/8
10.240.12.0/22
10.244.28.0/22
10.240.40.0/22
10.240.64.0/23
10.240.72.0/22
10.240.76.0/22
#Optical's polatis
10.252.254.197/32
10.252.254.9/24
10.252.255.0/24
172.22.32.0/24
172.22.73.31/32
172.22.73.70
172.22.73.99
172.22.73.19
172.22.73.27/32
172.22.73.164/32
#172.22.73.0/24
#172.22.72.0/22
172.23.62.0/24
172.30.124.128/26
172.22.73.128/25
172.22.73.126
172.22.73.127
172.23.35.32/27
35.135.193.64/26
35.135.193.0/24
2600:6ce6:4410::/48
2605:1c00:50f2::/48
2600:6ce7:0:5::/64
2600:6cec:1c0:7::/64
#2605:1c00:50f2:2800::/64
2605:1c00:50f2:280e::/64
2605:1c00:50f2:280e::6100/64
2605:1c00:50f2:2800:172:22:73:100/128
2605:1c00:50f2:2800:172:22:73:164/128
2605:1c00:50f2:2800:172:22:73:31/128
EOF
```

Create the sshuttle DENY file by running the following command

```
echo << EOF >> ~/.spoc.deny.txt
#corp
142.136.0.0/16
142.136.235.173
22.0.0.0/8
33.0.0.0/8
#10.151.0.0/16
#SPOC
#10.240.72.137
35.135.192.78/32
#172.23.62.20
172.23.62.21
172.22.73.17
#172.22.73.13
#2605:1c00:50f2:2800:172:22:73:17/128
#2605:1c00:50f2:2800:172:22:73:13/128
#2605:1c00:50f2:2800:172:22:73:18/128
#2605:1c00:50f2:280b:172:23:62:222/128
EOF
```

Running SSHuttle from your terminal

```
sshuttle -v -r $USER@35.135.192.78:3022 -s ~/.spoc.allow.txt -X ~/.spoc.deny.t
```

SSHUTTLE HELPER FUNCTION - Easily start and stop the `sshuttle` process in the background and securely inject your password from MacOS Keychain

NOTE: This simple function is just a wrapper for the `sshuttle` command that uses the CLI component of the MacOS Keychain to securely inject your SPOC password without having to manually enter them every time.

The function also maintains a log of the current session for debugging, and will keep you connected to the SPOC network even if you are inactive for awhile.

You can modify the function to invoke `sshuttle` with any arguments you wish. And of course, you can still run the `sshuttle` command from the CLI.

ALSO NOTE: The script has several basic checks to ensure you have all of the following:

- The SPOCUSER variable set
- The `sshpass` utility installed.
- Your SPOC Password added to your MacOS Keychain (The script will prompt you to create it if not present).

Creating the Helper Script

```
echo << EOF >> ~/.spoc.zsh
#!/bin/bash

spoctunnel () {

# ADD TO PROFILE
# Add the following uncommented line to your shell profile
# [[ -f ~/.spoc.zsh ]] && source ~/.spoc.zsh


colorRed="\033[31m"
colorGreen="\033[32m"
colorYellow="\033[33m"
colorBlue="\033[34m"
colorDefault="\033[0m"

SSHUTTLESTATE=$1
LOGFILE="$HOME/.sshuttle.log"

# SET SPOCUSER TO YOUR SPOC ACCOUNT NAME
SPOCUSER="ajanis"
if [ -z "$SPOCUSER" ]; then
    echo -e "
    ${colorRed}No User set for SPOC SSH Connection defined.

    Set the ${colorYellow}'SPOCUSER' ${colorRed}variable to your
    ${colorYellow}SPOC Username ${colorRed}in the helper script

    ${colorDefault}"
    return
    fi

# INSTALL SSHPASS
if [ ! -x $(which sshpass) ]; then
    echo -e "${colorRed}

    You need to install the 'sshpass' tool via Homebrew.
    Assuming you have homebrew installed, run the following commands:

    ${colorYellow}brew tap ajanis/custombrew
    brew install ajanis/custombrew/sshpass

    ${colorDefault}"
    return
fi

# Password storage/retrieval mechanism
# Support for 1password and MAC OS KeyChains
# Example command for CLI access provided below

# 1Password CLI
#SPOCPASSWD_1PASSWD="$(op read op://Charter/charterlab-spoc/password)"

# MAC OS Keychain
# MAC OS Keychain
SPOCPASSWD_KEYCHAIN="$(security find-generic-password -s 'SPOC VPN' -a ${USER}
if [ -z $SPOCPASSWD_KEYCHAIN ]; then
        echo -e "

        ${colorRed}No SPOC Password found in your MacOS Keychain!

        ${colorGreen}Please enter your SPOC password when prompted to securely

        ${colorDefault}
        "
        security add-generic-password -a ${USER} -s 'SPOC VPN' -w
            fi


# Set SPOC password to MacOS Keychain Password result
SPOCPASSWD="${SPOCPASSWD_KEYCHAIN}"

# SSHuttle option menu
case $SSHUTTLESTATE in
    start)
      if ! pgrep -f sshuttle; then
      echo > $LOGFILE
      echo -e "${colorGreen}Starting SSHuttle connection to SPOC Jumphost
      ${colorDefault}"
      SSHPASS=${SPOCPASSWD} \
      bash -c "sshpass -e sshuttle -v -r $SPOCUSER@35.135.192.78:3022 \
      -s ~/.spoc.allow.txt \
      -X ~/.spoc.deny.txt \
      --ns-hosts 172.22.73.19 \
      --to-ns 172.22.73.19" >>$LOGFILE 2>&1 &
      fi
      ;;
    start_1pw)
      if ! pgrep -f sshuttle; then
      echo > $LOGFILE
      SSHPASS=${SPOCPASSWD} \
      bash -c "sshpass -e sshuttle -v -r $SPOCUSER@35.135.192.78:3022 \
      -s ~/.spoc.allow.txt \
      -X ~/.spoc.deny.txt \
      --ns-hosts 172.22.73.19 \
      --to-ns 172.22.73.19" >>$LOGFILE 2>&1 &
      fi
      ;;
    start_keychain)
      if ! pgrep -f sshuttle; then
      echo > $LOGFILE
      SSHPASS=${SPOCPASSWD} \
      bash -c "sshpass -e sshuttle -v -r $SPOCUSER@35.135.192.78:3022 \
      -s ~/.spoc.allow.txt \
      -X ~/.spoc.deny.txt \
      --ns-hosts 172.22.73.19 \
      --to-ns 172.22.73.19" >>$LOGFILE 2>&1 &
      fi
      ;;
      ;;
    stop)
      if pgrep -f sshuttle; then
      echo -e "${colorGreen}Killing SSHuttle connection to SPOC
      ${colorDefault}"
      sudo pkill -f sshuttle >>$LOGFILE 2>&1
      fi
      ;;
    tail)
      tail -F $LOGFILE
      ;;
    cat)
      cat $LOGFILE
      ;;
    *)
      echo -e "$0 (start|stop|tail|cat|start_1pw|start_keychain)
      start:          | Starts sshuttle using -s ~/.spoc.allow.txt and -X ~/.s
      stop:           | Shuts down the sshuttle application
      tail:           | Tails the sshuttle process log file at ~/.sshuttle.log
      cat:            | Displays the entire file at ~/.sshuttle.log
      start_1pw:      | Same as start + Uses 1password CLI for password retrie
      start_keychain: | Same as start + Uses MacOS Keychain for password retri
      ;;
esac
}

EOF
```

Include ~/.spoc.rc Helper Script just created in your SHELL profile / rcfile

This example uses .zshrc, but you can substitute the rcfile for your $SHELL of choice

```
cat << EOF >> .zshrc
[[ -f ~/.spoc.zsh ]] && source ~/.spoc.zsh
EOF
```

Runing the SSHuttle Helper Script

- Open a new terminal window or reinstantiate your shell with exec `$SHELL`
- Run `spoctunnel` to see help

```
❯ spoctunnel
spoctunnel (start|stop|tail|cat|start_1pw|start_keychain)
      start:          | Starts sshuttle using -s ~/.spoc.allow.txt and -X ~/.s
      stop:           | Shuts down the sshuttle application
      tail:           | Tails the sshuttle process log file at ~/.sshuttle.log
      cat:            | Displays the entire file at ~/.sshuttle.log
      start_1pw:      | Same as start + Uses 1password CLI for password retrie
      start_keychain: | Same as start + Uses MacOS Keychain for password retri
```

- Run `spoctunnel start` to start the `sshuttle` application

```
❯ spoctunnel start
Starting SSHuttle connection to SPOC Jumphost

[4] 83000
```

- You will be prompted for your system/sudo password or fingerprint by 1Password or MacOS Keychain (*unless you have configured passwordless sudo*)

- Run `spoctunnel tail` to view logs

```
❯ spoctunnel tail
c : Connected to server.
fw: setting up.
fw: >> pfctl -s Interfaces -i lo -v
fw: >> pfctl -s all
fw: >> pfctl -a sshuttle6-12300 -f /dev/stdin
fw: >> pfctl -E
fw: >> pfctl -s Interfaces -i lo -v
fw: >> pfctl -s all
fw: >> pfctl -a sshuttle-12300 -f /dev/stdin
fw: >> pfctl -E
c : Accept TCP: 10.153.3.239:52481 -> 44.230.79.122:443.
 s: SW 4:44.230.79.122:443: uwrite: got EPIPE
c : Accept TCP: 10.153.3.239:52484 -> 44.230.79.122:443.
c : Accept TCP: 10.153.3.239:52486 -> 172.22.73.99:443.
c : Accept TCP: 10.153.3.239:52487 -> 172.22.73.99:443.
```

- Run `spoctunnel stop` to shut down the `sshuttle` application

```
❯ spoctunnel stop
83000
83004
83029
83036
Killing SSHuttle connection to SPOC

[4]  + 83000 terminated  SSHPASS=${SPOCPASSWD} bash -c  >> $LOGFILE 2>&1
```