ubutnu 16.04+ 搭建Shadowsocks

前言

本教程旨在提供简明的Ubuntu 16.04下安装服务器端Shadowsocks。不同于Ubuntu 16.04之前的教程，本文抛弃initd，转而使用Ubuntu 16.04支持的Systemd管理Shadowsocks的启动与停止，显得更为便捷。优化部分包括BBR、TCP Fast Open以及吞吐量优化。

网上有很多快速搭建ss的方法，**但是都是从第三方作者的github上pull下来进行安装的**，考虑到安全因素，本文提供了更为纯净的搭建方式。

本教程仅适用于**Ubuntu 16.04及之后的版本**，基于Python 3，支持IPv6。

WARNNING：搭建SS有风险，不管是什么平台，被检测到都会被ban！

一、服务端配置shadowsocks

Let‘s start!

服务器的选择

首先我们需要一台服务器，现在市面上有很多云服务器。

阿里云： https://cn.aliyun.com/ （新加坡或香港节点）——看脸，容易被警告

vultr：https://www.vultr.com/ （新加坡或日本节点）

安装pip

本教程使用Python 3为载体，因Python 3对应的包管理器pip3并未预装，首先安装pip3：

```
apt-get update
apt install python3-pip
```

安装Shadowsocks

因Shadowsocks作者不再维护pip中的Shadowsocks（定格在了2.8.2），我们使用下面的命令来安装最新版的Shadowsocks：

```
pip3 install https://github.com/shadowsocks/shadowsocks/archive/master.zip
```

如果使用pip3安装失败，则采用pip进行安装：

```
apt-get install python-pip
pip install setuptools
pip install shadowsocks
apt-get install python-m2crypto #安装加密方式
```

安装完成后可以使用下面这个命令查看Shadowsocks版本：

```
ssserver --version
```

创建配置文件

创建Shadowsocks配置文件所在文件夹：

```
mkdir /etc/shadowsocks
nano /etc/shadowsocks/config.json
```

复制粘贴如下内容（注意修改密码）：

```
{
"server":"::",
"server_port":8388, #可以自定义
"local_address": "127.0.0.1",
"local_port":1080,
"password":"你的密码",
"timeout":300,
"method":"aes-256-cfb",
"fast_open": false
}
```

然后按Ctrl + O保存文件，Ctrl + X退出。

来测试下Shadowsocks能不能正常工作了：

```
ssserver -c /etc/shadowsocks/config.json
```

在Shadowsocks客户端添加服务器，如果你使用的是我提供的那个配置文件的话，地址填写你的IPv4地址或IPv6地址，端口号为8388，加密方法为aes-256-cfb，密码为你设置的密码。然后设置客户端使用全局/PAC模式，浏览器登录Google试试应该能直接打开了。
测试完毕，按Ctrl + C关闭Shadowsocks。

配置Systemd管理Shadowsocks

新建Shadowsocks管理文件

```
nano /etc/systemd/system/shadowsocks-server.service
```

复制粘贴：

```
[Unit]
Description=Shadowsocks Server
After=network.target

[Service]
ExecStart=/usr/local/bin/ssserver -c /etc/shadowsocks/config.json
Restart=on-abort

```

```
[Install]
WantedBy=multi-user.target
```

Ctrl + O保存文件，Ctrl + X退出。

启动Shadowsocks：

```
systemctl start shadowsocks-server
```

设置开机自启（可选）：

```
systemctl enable shadowsocks-server
```

至此，Shadowsock服务器端的基本配置已经全部完成了！

二、服务端的优化（以下内容可选）

这部分属于进阶操作，在你使用Shadowsocks时感觉到延迟较大，或吞吐量较低时，可以考虑对服务器端进行优化。

BBR

BBR是Google最新开发的TCP拥塞控制算法，目前有着较好的带宽提升效果，甚至不比老牌的锐速差。

升级Linux内核

BBR在Linux kernel 4.9引入。首先检查服务器kernel版本：

```
uname -r
```

如果其显示版本在4.9.0之下，则需要升级Linux内核，否则请忽略下文。

更新包管理器：

```
sudo apt-get update
```

查看可用的Linux内核版本：

```
apt-cache showpkg linux-image
```

找到一个你想要升级的Linux内核版本，如“linux-image-4.10.0-22-generic”：

```
apt install linux-image-4.10.0-22-generic
```

等待安装完成后重启服务器：

```
reboot
```

删除老的Linux内核：

```
purge-old-kernels
```

开启BBR

运行

```
lsmod | grep bbr
```

如果结果中没有tcp_bbr，则先运行：

```
modprobe tcp_bbr
echo "tcp_bbr" >> /etc/modules-load.d/modules.conf
```

然后：

```
sysctl -p
```

保存生效。运行：

```
sysctl net.ipv4.tcp_available_congestion_control
sysctl net.ipv4.tcp_congestion_control
```

若均有bbr，则开启BBR成功。

优化吞吐量

新建配置文件：

```
nano /etc/sysctl.d/local.conf
```

复制粘贴：

```
# max open files
fs.file-max = 51200
```

```
# max read buffer
net.core.rmem_max = 67108864
# max write buffer
net.core.wmem_max = 67108864
# default read buffer
net.core.rmem_default = 65536
# default write buffer
net.core.wmem_default = 65536
# max processor input queue
net.core.netdev_max_backlog = 4096
# max backlog
net.core.somaxconn = 4096

# resist SYN flood attacks
net.ipv4.tcp_syncookies = 1
# reuse timewait sockets when safe
net.ipv4.tcp_tw_reuse = 1
# turn off fast timewait sockets recycling
net.ipv4.tcp_tw_recycle = 0
# short FIN timeout
net.ipv4.tcp_fin_timeout = 30
# short keepalive time
net.ipv4.tcp_keepalive_time = 1200
# outbound port range
net.ipv4.ip_local_port_range = 10000 65000
# max SYN backlog
net.ipv4.tcp_max_syn_backlog = 4096
# max timewait sockets held by system simultaneously
net.ipv4.tcp_max_tw_buckets = 5000
# turn on TCP Fast Open on both client and server side
net.ipv4.tcp_fastopen = 3
# TCP receive buffer
net.ipv4.tcp_rmem = 4096 87380 67108864
# TCP write buffer
net.ipv4.tcp_wmem = 4096 65536 67108864
# turn on path MTU discovery
net.ipv4.tcp_mtu_probing = 1

net.ipv4.tcp_congestion_control = bbr
```

运行：

```
sysctl --system
```

编辑之前的shadowsocks-server.service文件：

```
nano /etc/systemd/system/shadowsocks-server.service
```

在**ExecStart**前插入一行，内容为：

```
ExecStartPre=/bin/sh -c 'ulimit -n 51200'
```

即修改后的shadowsocks-server.service内容为：

```
[Unit]
Description=Shadowsocks Server
After=network.target

[Service]
ExecStartPre=/bin/sh -c 'ulimit -n 51200'
ExecStart=/usr/local/bin/ssserver -c /etc/shadowsocks/config.json
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

重载shadowsocks-server.service：

```
systemctl daemon-reload
```

重启Shadowsocks：

```
systemctl restart shadowsocks-server
```

开启TCP Fast Open

TCP Fast Open可以降低Shadowsocks服务器和客户端的延迟。实际上在上一步已经开启了TCP Fast Open，现在只需要在Shadowsocks配置中启用TCP Fast Open。

编辑config.json：

```
nano /etc/shadowsocks/config.json
```

将fast_open的值由false修改为true。Ctrl + O保存文件，Ctrl + X退出。

重启Shadowsocks：

```
systemctl restart shadowsocks-server
```

注意：TCP Fast Open同时需要客户端的支持，即客户端Linux内核版本为3.7.1及以上；你可以在Shadowsocks客户端中启用TCP Fast Open。

至此，Shadowsock服务器端的优化已经全部完成了！

三、客户端配置shadowsocks

本章主要讲解如何在ubuntu16.04+的环境下配置shadowsocks客户端。

安装客户端

Windows、Mac、IOS、Android平台只需安装对应平台的shadowsocks，有手就行的操作，这里不再赘述。接下来我们主要讲解如何在ubuntu16.04+的环境下配置shadowsocks客户端。

首先，我们需要安装shadowsocks：

```
apt-get install python3-pip
pip3 install https://github.com/shadowsocks/shadowsocks/archive/master.zip
apt-get install python-m2crypto #安装加密方式
```

启动客户端

安装好后，在本地我们要用到sslocal ，终端输入sslocal --help 可以查看帮助，通过帮助提示我们知道各个参数怎么配置，比如 sslocal -c 后面加上我们的json配置文件，或者像下面这样直接命令参数写上运行。比如：

```
sslocal -s 11.22.33.44 -p 50003 -k "123456" -l 1080 -t 600 -m aes-256-cfb
```

-s表示服务IP, -p指的是服务端的端口， -l是本地端口默认是1080, -k 是密码（要加""，-t 超时默认300, -m 是加密方法默认aes-256-cfb，为了方便我推荐直接用sslcoal -c 配置文件路径 这样的方式，简单好用。

我们可以在 /etc/shadowsocks/ 下新建个文件config.json。内容是这样：

```
mkdir /etc/shadowsocks/
touch config.json
nano /etc/shadowsocks/config.json
```

将以下内容复制到config.json中：

```
{
"server":"你的服务器IP",
"server_port":8388,#服务端端口号
"local_port":1080,
"password":"你的密码",
"timeout":600,
"method":"aes-256-cfb"
}
```

（1）直接启动：

```
sslocal -c /etc/shadowsocks/config.json
```

（2）以后台守护进程运行：

```
sslocal -c /etc/shadowsocks/config.json -d start
```

以上就是SS客户端的搭建了，这个时候我们发现上网时并不可以翻墙，原因是需要将sock5代理映射为http代理。代理的软件很多，我选择了推荐度比较高的privoxy，下面是privoxy的配置。

安装privoxy

首先，我们要安装privoxy：

```
apt install privoxy
```

接下来进行一些配置，打开 /etc/privoxy/config：

```
nano /etc/privoxy/config
```

找到其中的4.1节，看一下有没有一句listen-address localhost:8118的代码，如果被注释了，取消注释。因为版本不一样这句的状态可能会不一样。 然后再

将 localhost 改成 127.0.0.1，如图所示：

```
#        listen-address  192.168.0.1:8118
#
#      Suppose you are running Privoxy on an IPv6-capable machine and
#      you want it to listen on the IPv6 address of the loopback
#      device:
#
#        listen-address [::1]:8118
#
listen-address  127.0.0.1:8118
#
#  4.2. toggle
#  ============
```

接着找到5.2节，在本节末尾加入代码：

```
forward-socks5 / 127.0.0.1:1080 .
```

结果如图所示：

```
#
#      If you also want to be able to reach servers in your local
#      network by using their names, you will need additional
#      exceptions that look like this:
#
#       forward           localhost/     .
forward-socks5 / 127.0.0.1:1080 .
#
#
#  5.3. forwarded-connect-retries
#  ==============================
#
```

启动privoxy

执行以下命令启动privoxy：

```
service privoxy start
```

配置系统proxy

编辑：

```
nano /etc/profile
```

加入以下内容：

```
export http_proxy=http://127.0.0.1:8118/
export https_proxy=http://127.0.0.1:8118/
```

```
export HTTP_PROXY=http://127.0.0.1:8118/
export HTTPS_PROXY=http://127.0.0.1:8118/
```

让环境变量生效：

```
source /etc/profile
```

重启privoxy服务：

```
service privoxy restart
```

测试

输入以下指令，如果有正常返回，则Terminal已经可以连接外网：

```
wget www.google.com
```

四、为浏览器添加代理

上一章节的步骤实现了terminal与外网的连接，但是你会发现浏览器还不能连接外网。

安装插件

我们需要给chromium安装SwitchyOmega插件，但是没有代理之前是不能从谷歌商店安装这个插件的，但是我们可以从Github上直接下载最新版 https://github.com/FelisCatus/SwitchyOmega/releases/ ，然后浏览器地址打开chrome://extensions/，将下载的插件托进去安装。（如果提示安装失败，就把文件后缀改为.zip，解压后使用”加载已解压的扩展程序“进行安装）

设置代理地址

安装好插件会自动跳到设置选项，有提示你可以跳过。左边新建情景模式-选择代理服务器-比如命名为SS，其他默认之后创建。之后在代理协议选择SOCKS5，地址为127.0.0.1,端口默认1080 。然后保存立即应用选项。

五、为SS设置PAC模式

实际上，第四章的配置为全局模式，换句话说，无论我们访问什么网站，都走的是代理，本章将讲解如何为代理设置PAC模式。
要实现PAC模式，我们就需要两样东西：1.国内能访问的域名 2.pac文件生成器。

获取GFWlist

无需走代理的域名可从github上下载：https://github.com/gfwlist/gfwlist
将gfwlist.txt下载到相应路径，比如我将其放置在 /home/×××/vpn/ 下

安装genpac

```
apt-get update
pip install genpac
```

生成pac

进入gfwlist所在路径，然后执行：

```
genpac --pac-proxy "SOCKS5 127.0.0.1:1080" --output="autoproxy.pac" --gfwlist-local gfwlist.txt
```

若无报错，可在当前路径下生成名为 autoproxy.pac 的文件。

配置浏览器

以SwitchySharp为例，选择相应的情景模式，将原有的手动配置切换为自动配置，导入相应的pac文件。

至此，所有准备工作都已完成，愉快地网上冲浪吧！