4 - Ubuntu 16.04 + SSR翻墙
jianshu.com 更新2018年8月20日

4 - Ubuntu 16.04 + SSR翻墙

Miliimoulins 已关注

2018.04.22 12:44* 字数 1819 阅读 14073 评论 16 喜欢 13

身为一名程序媛翻墙是必须的，mac和windows下都有现成的shadowsocks客户端，直接配置就好，Linux用戶。本文提供一套可行的linux翻墙方案，当然前提是你要有SS账号提供商提供的服务器地

> 这是一个系列：
> 1、Ubuntu系统安装
> 2、Ubuntu 16.04 + nvidia + cuda9.1
> 3、Ubuntu 基本工作环境 + 深度学习环境配置
> 4、Ubuntu 16.04 + SSR翻墙

配置SSR客户端

> Shadowsocks 与 Shadowsocksr 区别：
> SS是原版，SSR是原版基础上衍生出来的第三方版本，兼容原版协议，比原版多了一些伪装功
> SSR 主要特点是增加了一些人性化功能，比如服务器连接统计、连接管理、协议转换、多重

1、下载ssr客户端

```
git clone https://github.com/ssrbackup/shadowsocksr
```

2、配置ss文件

进入刚刚clone下来的文件夹，有一个文件叫 `config.json` / `user-config.json` ，这是配置文件的模板

```
cp config.json /etc/shadowsocks.json
```

然后对这个文件进行配置：

```
sudo gedit /etc/shadowsocks.json
```

用下面替换该文件中的内容（具体的服务器地址，端口，密码，加密方式，协议插件，混淆插件

```
{
    "server": "0.0.0.0",
    "server_ipv6": "::",
    "server_port": 80890,
    "local_address": "127.0.0.1",
    "local_port": 1080,

    "password": " ",
    "method": "chacha20",
    "protocol": "auth_sha1_v4",
    "protocol_param": "",
    "obfs": "http_simple",
    "obfs_param": "",
    "speed_limit_per_con": 0,
    "speed_limit_per_user": 0,

    "additional_ports" : {}, // only works under multi-user mode
    "additional_ports_only" : false, // only works under multi-user mode
    "timeout": 120,
    "udp_timeout": 60,
    "dns_ipv6": false,
    "connect_verbose_info": 0,
    "redirect": "",
    "fast_open": false
}
```

主要用到的配置是下面的这几个选项：

```
"server": "0.0.0.0",  # 服务器地址  对于ss：ping+域名，得到地址。
"server_port": 80890,    # 端口
"password": " ",    # 密码
"method": "chacha20",      # 加密方式
"protocol": "auth_sha1_v4",  # 协议插件
"obfs": "http_simple",    # 混淆插件
```

3、启动ssr客户端

```
python ~/shadowsocksr/shadowsocks/local.py -c /etc/shadowsocks.json
```

注：如果出错：

```
IPv6 support
Traceback (most recent call last):
  File "local.py", line 81, in <module>
    main()
  File "local.py", line 43, in main
    config = shell.get_config(True)
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/shell.py", line 299, in
    check_config(config, is_local)
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/shell.py", line 129, in
    encrypt.try_cipher(config['password'], config['method'])
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/encrypt.py", line 46, in
    Encryptor(key, method)
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/encrypt.py", line 90, in
    random_string(self._method_info[1]))
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/encrypt.py", line 119, i
    return m[2](method, key, iv, op)
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/crypto/sodium.py", line
    load_libsodium()
  File "/home/miliimoulins/shadowsocksr/shadowsocks/../shadowsocks/crypto/sodium.py", line
    raise Exception('libsodium not found')
Exception: libsodium not found
```

因为当前Python版本为系统自带python 2.7，缺很多东西。将python默认版本改为anaconda python
版本的方法:

```
export PATH=~/anaconda3/bin:$PATH
source ~/.bashrc
```

【如果直接修改python默认版本为anaconda3： `sudo gedit ~/.bashrc` 在最后一行添加： `export P`
然后再执行： `python ~/shadowsocksr/shadowsocks/local.py -c /etc/shadowsocks.json` ，成功：

```
IPv6 support
2018-04-20 15:53:35 INFO     util.py:85 loading libsodium from /home/miliimoulins/anaconda3
2018-04-20 15:53:35 INFO     local.py:50 local start with protocol[auth_sha1_v4] password [
2018-04-20 15:53:35 INFO     local.py:54 starting local at 127.0.0.1:1080
2018-04-20 15:53:35 INFO     asyncdns.py:324 dns server: [('127.0.0.1', 53)]
```

4、转换HTTP代理

Shadowsocks默认是用Socks5协议的，对于Terminal的get,wget等走http协议的地方是无能为力的，
基于Polipo的。

```
sudo apt-get install polipo      # 安装Polipo
sudo gedit /etc/polipo/config    # 修改配置文件
```

将下面的内容整个替换到文件中并保存：

```
# This file only needs to list configuration variables that deviate
# from the default values. See /usr/share/doc/polipo/examples/config.sample
# and "polipo -v" for variables you can tweak and further information.
logSyslog = false
logFile = "/var/log/polipo/polipo.log"

socksParentProxy = "127.0.0.1:1080"
socksProxyType = socks5

chunkHighMark = 50331648
objectHighMark = 16384

serverMaxSlots = 64
serverSlots = 16
serverSlots1 = 32

proxyAddress = "0.0.0.0"
proxyPort = 8123
```

重启Polipo:

```
/etc/init.d/polipo restart
```

验证代理是否正常工作：

```
export http_proxy="http://127.0.0.1:8123/"
curl www.google.com
```

如果正常，就会返回抓取到的Google网页内容。此时，终端里面可以访问外网了。
另外，在浏览器中输入http://127.0.0.1:8123/便可以进入到Polipo的使用说明和配置界面。

5、配置浏览器

之前参考别人的方法来配置了firefox，it worked，但是到了chrome方法就不适用了。我们可以直接
打开 `system settings->network->network proxy` ：
`Method:Manual`
下面都设置为： `127.0.0.1` 端口： `8123`
点击 `Apply system wide`

Done!!!Cheers!

注意：

每次开机后都要执行： `python ~/shadowsocksr/shadowsocks/local.py -c /etc/shadowsocks.json` 来

有时候不走运的话，误杀该进程（关机 或者关闭当前terminal 或者ctrl z当前python进程）会发生

```
curl: (7) Failed to connect to 127.0.0.1 port 8123: Connection refused
```

Solution：

- 首先一般这种情况下，如果你不是直接关闭的当前terminal，则你的Python进程可能还没有杀掉
个python进程的process number）。

- 然后重新启动ss服务。ps：我建议用**screen**命令来执行，至少不用担心误杀进程： `screen pytho`
`/etc/shadowsocks.json` 。pps：screen模式下， `ctrl+a` ，然后 `d` （detach），或者直接可以关闭当

- 如果还是不能上网： 重启polipo： `/etc/init.d/polipo restart` 。

- 验证： `export http_proxy="http://127.0.0.1:8123/"` ， `curl www.google.com` 。

6、Ubuntu开机后自动运行

现在可以科学上网了，可是每次开机都要手动打开终端输入命令，虽然命令并不长，但是每次都
果不是用的screen命令），还会引发上面一系列问题。

Solution：

- 写个脚本

我们可以在比如 `/home` 下新建个文件叫做 `shadow.sh` ，在里面写上我们启动ssr客户端需要的命令，
不能上，就每次都自动重启一下polipo吧：

```
#! /bin/bash
#shadow.sh
screen python ~/shadowsocksr/shadowsocks/local.py -c /etc/shadowsocks.json
/etc/init.d/polipo restart
```

到终端执行命令 `sh /home/shadow.sh` ，成功的话会有信息输出。你也可以到浏览器去试试。
【如果用的不是screen， 直接python的话：这个时候你虽然输入的少了，可是关了终端还是会掉

至此：开机后打开终端运行： `sh /home/shadow.sh` ，就可以自由上网了。

- 加入开机运行

这里我们需要 在 **`/etc`** 下编辑一个叫 **`rc.local`** 的文件，需要root权限 `sudo gedit /etc/rc.local` 。
[或者：在终端先 `su` 获取root权限（ ps: 关于 `su` 输入密码之后显示su: Authentication failure的解决

```
sudo passwd root    # 输入的命令
[sudo] password for miliimoulins:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

然后再次 `su` 就可以进入root权限了）： `gedit /etc/rc.local`]

编辑：在 `exit 0` 前面添加： `sh /home/shadow.sh` ，

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

sh /home/shadow.sh>/home/d.txt
exit 0
```

然后ctrl+s保存。
这个时候你可以 `reboot` 重启，测试下看看能不能后台自动运行。对于一般的需要开机自启动的对
但是本例不行。可以先去看下我们要它输出的 `d.txt` ，竟然发现是 **`Must be connected to a termi`**

经过一番搜索发现远离linux是找不到 `screen` 这条命令?
参考对于 `sslocal` 执行时的解决办法：需要添加路径，发现 `sslocal` 和 `ssserver` 这两个命令是被存
两个文件移动到 `/bin` 下，就可以了。
但是对于 `screen` 不行。
所以目前还是先在开机后，打开终端输入 `sh /home/shadow.sh` 吧hhhhhh，还是很方便的。
关于如何顺利的开机自启动screen命令，现在先不花时间去研究了，毕竟一行命令已经很方便了。
件。后面有时间弄好了再更新。

Whatever, cheers!

> 参考目录：
>
> 1、https://blog.csdn.net/superbfly/article/details/54950451
>
> 2、https://www.jianshu.com/p/c336fd0bdfbe

Evernote，让记忆永存