**Proxy、SSH 和 VPN 的区别**

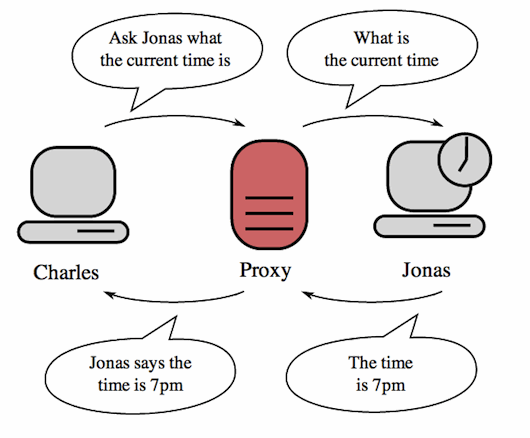
*作者：*[YOUNG](http://jingpin.org/author/youngeblog/)*| 发布：*

虽然[翻墙工具](http://jingpin.org/100-anti-censorthip-tools/)有成百上千个，但是如果把它们分类，绝大部分都是 Proxy、SSH 和 VPN 这三类。

那么，Proxy、SSH 和 VPN 之间有什么区别呢？谁的安全性最高呢？

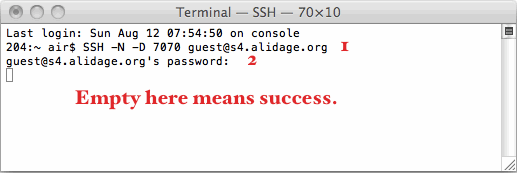
如果把传输的信息看作一个包裹，那么我们就可以用三个简单的比喻来做比较。

**1、Proxy**



Proxy (代理)，可以看作是一个快递员，负责将你的包裹送达收件人。

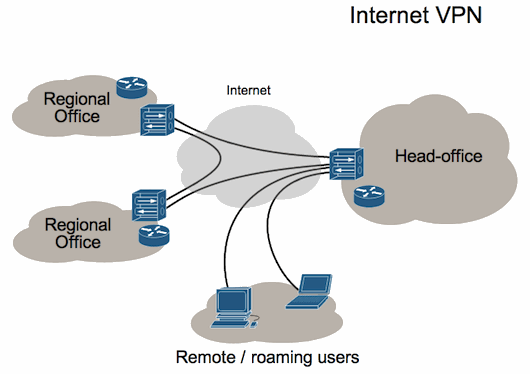
**2、SSH**



SSH （Secure Shell, 安全外壳协议）是一种加密渠道，本身并不能翻墙，需要结合代理服务器才能够翻墙。所以，我们通常所说的用来翻墙的 SSH，都是指 SSH 代理。

也就是说，SSH 代理 ＝ 代理 + SSH，可以看作是加密了的代理，就好比把包裹装在保险箱里面再交给快递员。

**3、VPN**



图片来源： http://en.wikipedia.org/wiki/Virtual\_private\_network

而使用 VPN（Virtual Private Network，虚拟私人网络），就好比不通过快递员，你亲自把包裹交给收件人所在公司的前台，再由前台转交给收件人。

从以上三个比喻可以看出，VPN 的安全性要高于 SSH、而 SSH 又高于 Proxy，因为相对来讲，包裹中途被拦截的几率要比目的地的高，特别是当大家都是在同一个服务器上搭建的时候，但是大多数情况下，这三类翻墙工具都是不在同一个服务器上的，所以安全性也是相对的。

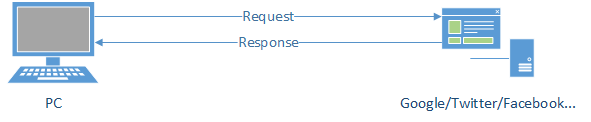
另外，Proxy 和 SSH 的作用是局部的，只适用于你指定的应用程序（例如浏览器），而 VPN 的作用是全局的，适用于你的整台设备，不管是浏览器、邮件客户端、iTunes 或者其他任何的连接到互联网的应用程序。

还有，通过 Proxy 和 SSH 翻墙的时候，需要设置网络的代理地址，而 VPN 则不需要。

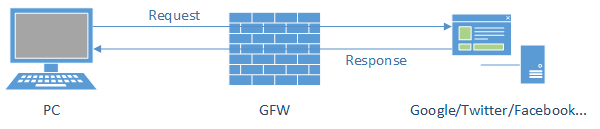
写给非专业人士看的 Shadowsocks 简介

这个文章来源于一个朋友在科学上网的过程中，搞不清楚 Shadowsocks 的配置问题，在这里我想按照我对 Shadowsocks 的理解简单梳理一下，以便一些非专业人士也能了解

long long ago…

在很久很久以前，我们访问各种网站都是简单而直接的，用户的请求通过互联网发送到服务提供方，服务提供方直接将信息反馈给用户  


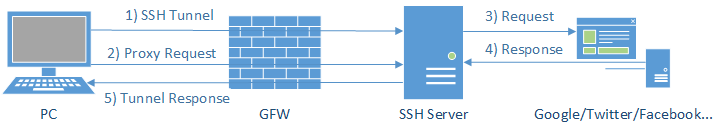
when evil comes

然后有一天，[GFW](http://zh.wikipedia.org/wiki/金盾工程) 就出现了，他像一个收过路费的强盗一样夹在了在用户和服务之间，每当用户需要获取信息，都经过了 GFW，GFW将它不喜欢的内容统统过滤掉，于是客户当触发 GFW 的过滤规则的时候，就会收到 Connection Reset 这样的响应内容，而无法接收到正常的内容  


ssh tunnel

聪明的人们想到了利用境外服务器代理的方法来绕过 GFW 的过滤，其中包含了各种HTTP代理服务、Socks服务、VPN服务… 其中以 ssh tunnel 的方法比较有代表性

1) 首先用户和境外服务器基于 ssh 建立起一条加密的通道  
2-3) 用户通过建立起的隧道进行代理，通过 ssh server 向真实的服务发起请求  
4-5) 服务通过 ssh server，再通过创建好的隧道返回给用户



由于 ssh 本身就是基于 RSA 加密技术，所以 GFW 无法从数据传输的过程中的加密数据内容进行关键词分析，避免了被重置链接的问题，但由于创建隧道和数据传输的过程中，ssh 本身的特征是明显的，所以 GFW 一度通过分析连接的特征进行干扰，导致 ssh  
存在被定向进行干扰的问题

shadowsocks

于是 [clowwindy](https://github.com/clowwindy/shadowsocks) 同学分享并开源了他的[解决方案](http://www.v2ex.com/t/32777)

简单理解的话，shadowsocks 是将原来 ssh 创建的 Socks5 协议拆开成 server 端和 client 端，所以下面这个原理图基本上和利用 ssh tunnel 大致类似

1、6) 客户端发出的请求基于 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或局域网的其他机器，不经过 GFW，所以解决了上面被 GFW 通过特征分析进行干扰的问题  
2、5) 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 的时候是常规的TCP包，没有明显的特征码而且 GFW 也无法对通讯数据进行解密  
3、4) ss-server 将收到的加密数据进行解密，还原原来的请求，再发送到用户需要访问的服务，获取响应原路返回  
