计算机网络基础知识

摘要

计算机网络在当代社会中扮演着越来越重要的角色。但据我观察，我身边的多数人只会使用计算机网络却不知道计算机网络是怎样组织和怎样工作的，有一次我问我父亲是否知道互联网是如何传输信息的，父亲的回答是“电线里面由高低电平代表的 0 和 1”，诚然这种说法的确是对的，但我父亲接着说，你网上看什么，电信部门知道的一清二楚，所以不要乱看一些违法的信息，我不禁苦笑。另一次是我教同学使用 VPN 访问 Youtube，同学看了几个视频之后，紧张得对我说，我不敢看这些了，看多了会被抓的，说罢就删除了 VPN 软件。

在中文论坛上我也会看到与我在现实中观察到的相似论调，仿佛强权或者有关部门是无所不能的。但事实并非如此，我们还是可以既保护隐私又去看我们想看的东西，所以我认为很有必要从计算机网络和密码学方面进行分析，来论证一下我们浏览的东西是否真的会被电信部门知道的一清二楚。于是便有了这篇文章。

在接下来的内容中，将首先介绍计算机网络的组成和工作原理，接下来是 GFW 的封锁策略，之后从密码学的角度分析绕过 GFW 的方法及其安全性，最后以举例的形式向读者展示，并得出结论。

本文使用 LaTeX 完成。

Part I

计算机网络基础设施

我童年的时候还不知道互联网为何物，小学三年级有同学开始讲上网的时候，我脑海里浮现出的之前看过的小说《夏洛的网》，以及由六边形组成的蜘蛛网形状，后来在同学的启蒙下才知道上网就可以聊 QQ，再到初中一年级，大地震之后无所事事的我们开始在网络上分享陈冠希老师的摄影作品，不由感叹网络真是个好东西。直至上大学学了计算机相关的课程，才恍然醒悟，我最初那个蜘蛛网形状才最接近互联网真正的形态。

事实上，互联网的结构远比蜘蛛网要复杂，但是它的基本组成却只有三种：传输介质、交换机和终端。读者对各种传输介质应该不会陌生，从运营商宣传中的各种不同带宽的光纤，到从前很常用的网线，再到现在都用的 Wifi 和 4G 都属于传输介质。如果把传输介质比作干线铁路，那么交换机就是枢纽车站，试想一位旅客准备从湖南小城市岳阳前往甘肃小城市天水，那么他必须在郑州站换乘，同样的，如果读者想用自己家的电脑访问百度，我想读者家里肯定没有直达百度公司机房的网线，所以这个访问请求会经过数个交换机，最终到达百度公司的机房。终端的英文是 terminal，互联网的信息由终端发送，最后会到达令一个终端，终端又分为两种类型: 提供内容或服务的终端（服务器，Server 或 Master）和接受内容或服务的终端（客户端，Local 或 Slave）。终端不仅仅是电脑和手机，地铁站里的售票机，路边的共享单车，甚至是随处可见的交通信号灯都有可能符合终端的定义。

当无数的终端和交换机通过传输介质连接在一起时，就构成了互联网。所以我们称这三样东西为互联网的基础设施。

Part II

TCP/IP 协议和应用层协议

聪明的读者可能已经有疑惑了：互联网上的设备那么多，凭什么从一个终端发送的信息可以正确地找到它想找的那个终端呢？为了回答这个问题，我们不得不隆重介绍 20 世纪最伟大的发明之一：TCP/IP 协议。

其实读者提出的这个问题在互联网发明之初就已经有了，要找到目的地的终端，首先我们需要给所有的终端进行编码，于是 IP 地址这个概念便自然而然得产生了。目前主流的 IP 地址编码方式是 IPv4，IPv4 是一个 32 比特的数字，为了方便记忆，通常使用点分十进制的方式书写，读者在设置自己家的路由器时肯定很熟悉这一串数字：“192.168.0.1”，类似这种写法的就是 IPv4 地址。IP 地址由互联网服务商（电信、联通等）分发给用户。由于 IPv4 地址最多只能编码 2^{32} 个互联网终端，随着共享单车和需要联网的智能电器数目增多，IPv4 的资源接近枯竭，新的编码标准 IPv6 开始被提出来，IPv6 长度达到 128 比特，使用冒分十六进制助记，写法类似于“ABCD:EF01:2345:6789:ABCD:EF01:2345:6789”。目前 IPv4 和 IPv6 地址是并存的，仍以 IPv4 的地址居多。

仅仅解决了编码，还是无法回答上面的问题。为了在浩如烟海的互联网终端中找到自己想要的，需要交换机的助力。在交换机上加入一些小小的缓存区，存放转发路由表，在互联网拓扑结构中实现迪杰斯特拉（Dijkstra）算法，互联网上的信息就可以找到正确找到目的地。以上算法不是本文主要讨论的部分，读者如果感兴趣可以自行百度。总之读者只要记住，有一种方法可以根据 IPv4 地址，用有限的时间就可以找到到达该目的地的最短路由。

解决了找路的问题，并非一切就大功告成。因为电信号是不稳定的，大家是打网络游戏的时候经常报怨的可能就是延迟和丢包率，造成丢包的原因就是网络中电信号的不稳定。如果仅仅使用 IP 协议发送和接收信息，会无法处理丢包。TCP 协议的发明就是为了解决丢包而发明的。

TCP 的工作过程十分简单，可以用下面的一段对话来表示。

S：“我要说十句话。”

L：“说吧”

S：“好的”

S：（一句话）

L：“收到了”

……

S：“不说了”

L：“好的，我也不说了”

S：“好的”

上述的对话分为建立过程，传输过程和结束过程。建立过程需要三次握手，传输过程中发送一个包都要等对方的回复，如果一段时间没有收到，就需要重复发送。结束过程也需要握手。TCP 协议保证了即使在恶劣的环境中，互联网也能传输完整的信息。只有极少数的互联网程序没有采用 TCP 协议，而是采用简化版的协议 UDP，此处不做介绍。

有了 TCP 这个工具，各种互联网应用就可以自由发挥了，互联网应用们通过规定上述对话片段中那十句话的格式来实现。读者最熟悉的互联网应用大概就是网页吧，大家在浏览器上打入”baidu.com” 后，浏览器会自动补全成”https://www.baidu.com”，表明使用 HTTPS 协议请求百度。现今的网页多使用 HTTPS 协议，最初网页是通过 HTTP 协议来实现的，如果要发送 HTTP 协议，上述对话就会变成这样。

S：“我要说一句话。”

L：“说吧”

S：“好的”

S：“HTTP 协议，请求，谷歌”

L：“收到了”

L：“HTTP 协议，回复，404”

S：“收到了”

S：“不说了”

L：“好的，我也不说了”

S：“好的”

也许读者也会经常遇到 404，但读者可能不知道 404 如同航空业中的 “Mayday” 一样，是 HTTP 协议中规定的一个暗号，意思是 “服务器无法找到被请求的页面”。

HTTP 协议是传输的是明文，确实会被电信部门看得一清二楚。HTTPS 协议的提出就是为了解决安全性问题。在讲 HTTPS 协议之前我想先普及两个密码学基本概念：对称加密和非对称加密，对称加密是指双方共同使用一个密钥，这个密钥可以加密也可以解密，而非对称加密是指存在公钥和私钥两种密钥，公钥用来加密，私钥用来解密，公钥可以大方的公布出去，但私钥必须妥善保存。

在 HTTPS 协议中，服务器将自己的公钥发给客户端，双方通过非对称加密的形式协商一个对称加密算法和密钥，之后传输信息时就用这个协商好的加密算法和密钥，这样的话，传输的信息只有客户端和服务器可以解密，中间的交换机即使拦截了信息也无法解密。

以上加密过程存在两个 bug：首先是服务器的私钥是否已经别人被破解，这个问题通过证书机构来解决，读者可能在访问网站时会看到 “证书已过期” 的警告，证书机构是向服务器提供密钥，并评估安全性的机构，所以读者在进行比较隐私的活动时最好不要去访问 “证书已过期” 的网站。第二个 bug 是网站是否将自己的私钥交给了有关部门，这个问题至今无法解决，所以大家尽量要去知名度高的网站，不要在钓鱼网站上泄露自己的信用卡密码。

至此最早说的有关部门是否会把我们浏览的东西知道的一清二楚的问题已经可以回答了，在 HTTPS 协议流行之后肯定是不行的。

不知道读者有没有注意到一个问题，我们之前说的终端地址都是类似于 “192.168.0.1” 这样的地址，怎么在 HTTP 协议中就变成了 “www.baidu.com”，难道电脑会自动翻译嘛？不是的。最初的时候我们访问的网址都是 IP 地址，但是随着门户网站的发展，各大公司对自己的一串数字怎么看都怎么不顺眼，于是它们合作建设了一种叫做 DNS 的服务器，提供域名解析服务。这样，他们可以给自己起个好听的名字，当用户访问时，这个域名先被送到 DNS 中解释成 IP 地址，再依照这个 IP 地址去进行 HTTP 访问。现在门户网站还利用 DNS 进行分布式部署，例如在北京就将 “www.baidu.com” 解释成北京的百度机房，在厦门就将这个域名解释成厦门的百度机房，可以让用户得到更低的延迟。

以上就是计算机网络中最基本的一些协议，有了这些协议，我们就可以去访问网页。但还有更多的协议没有被介绍，例如进行语音通信的 telnet 协议，提供文件传输服务的 ftp 协议，读者可以自行探索。

Part III

GFW 封锁策略

Great Fire Wall 并非墙的官方名称，官方对于墙的态度始终很暧昧，至今没有任何政府的声音承认墙的存在。据说 GFW 的名称来源于华尔街日报的一篇报导，并最终在西方媒体中形成共识，再进口到了我们国家的。但 GFW 在一直发展和进步，墙的高度在一步步加高，这却是不争的事实。

2013 年之前，用明文传输的 HTTP 协议还在大行其道，GFW 对大部分网站的策略是封锁关键字，只要发现网页中有国家领导人姓名或者一些丑闻就会由电信部门的交换机冒充客户端强行结束当前的 TCP 连接。对其它罪大恶极的网站，例如谷歌、脸书、推特和法轮功媒体，以及中华民国政府机关网站采用封锁 IP 的方式。

2013 年之后，HTTPS 协议开始流行，封锁关键字失去了作用，GFW 开始关照一些比较中立的网站，例如维基百科和部分西方媒体的中文网。至迟从 2015 年开始，GFW 开始封锁日本语和英语网站，日经、卫报、美联社和路透社相继沦陷。今年

GFW 开始封锁更为中立的高雅创作网站 AO3。

从 GFW 在 2013 年之后对 HTTPS 的应激反应也可以看得出，它自 2013 年开始已经不知道我们浏览的内容了，也无法再利用关键字屏蔽，所以只能扩大封锁范围。

目前 GFW 的封锁策略主要有两种：封锁 IP 和污染 DNS。

封锁 IP 是指国内的交换机过滤掉黑名单内的 IP 地址。

污染 DNS 是指在中国的 DNS 服务器中加入一些错误的答案，例如将“www.google.com”对应到一个法国政府的网站，不仅可以封锁谷歌，还可以引导墙内的网民对法国政府网站进行 DDoS 攻击，一举两得。

Part IV

VPN 和混淆

VPN 指虚拟专用网络。VPN 的原理非常简单，就是 A 通过 B 去和 C 交流，我最近在看《神话》，其中有一段是小川通过高月来和玉漱公主传情，在这个场景中，高月就像是一台 VPN 服务器。最初 VPN 的设计并非是为了翻墙的，而是某些大公司为了方便在外地出差的员工能连到自己公司的内网来处理公务，所以它虽然能够保证信息的安全，但是特征特别明显。在墙 2013 年加高的最初几个月，大家发现可以使用 VPN 协议翻墙，于是便有了使用 VPN 来指代翻墙工具的传统，即使现在原版 VPN 协议已经完全无法在墙内使用。

虽然 VPN 协议已经被封杀，但 VPN 的原理却被大家广泛利用，因为这简单的通过 B 去访问 C，却是 GFW 无法解决的硬伤。要想彻底阻断这种翻墙方式，必须阻断所有中国的出境流量，而在全球化的今天，封杀网络无异于自断财路。

聪明的中国程序员和国际友人们开发了许多用于翻墙的仿 VPN 协议，其中最为成功的一款软件是 Shadowsocks，加密算法在部署时预设，只是在建立连接的时候交换密钥，最大程度得隐藏自己的特征。据说目前 GFW 无法精确识别 Shadowsocks 的流量，有论文自称使用人工智能算法可以达到 95% 的准确率。我猜这也是为什么有些翻墙服务会在国庆和六四前后被封禁 IP，而在节日之后解封的原因，因为害怕误伤%5 的友军。

如果有人习惯了翻墙，那么在国庆和六四前后不能翻墙的日子就会像毒瘾发作一样难受。有没有办法避免的敏感的日子被查封呢？这个办法是有的，那就是混淆。

混淆纯粹就是因应 GFW 加高而发展出来的技术，它的核心思想是，既然 GFW 严查 Shadowsocks 流量，甚至在特殊的日子里将无法分类的流量也归类到 Shadowsocks 中，那么最安全的方法就是将我的流量伪装成其它类型的正常流量。当前流行的混淆方式叫做 V2ray。它将 Shadowsocks 流量伪装成 HTTP，使用 HTTP 协议的格式来传输翻墙流量。经我个人试验，使用 VPN+ 混淆，我成功躲过了 2019 年国庆的 VPN 封杀浪潮。

虽然未来墙会越来越高，但我相信我们的翻墙技术也会越来越成熟。

Part V

云计算

读者肯定会问，我怎么去找一个国外的计算机帮我转发流量呢？为了解决这个问题，我要提一个大家应该都听过的概念——云计算。云计算最开始也不是为了翻墙而设计的，而是随着互联网上汇聚的计算资源、存储资源、数据资源逐渐增加而产生的一种新的互联网服务类别。当一个公司有空闲的计算资源的时候，它当然愿意将自己的计算能力卖给别人赚钱。

读者可以把云计算理解成为你可以花钱从互联网的某处购买一台电脑，你可以通过网络向这台电脑发送指令，这台电脑处理完任务之后再将结果发送回来。那么，如果我们向云计算终端发送的指令

是诸如“访问 Youtube”之类，这个云计算终端充当的角色恰恰就是之前 VPN 中提到的中间人。

Part VI

For Example

读者可能已经有些不耐烦了：你倒是上干货啊。放心，干货马上就来。我在这里分享一下我部署 Shadowsocks+V2ray 环境的经历，抛砖引玉。

要完成一个仿 VPN 的模式，就必须要找到一个靠谱的 B。我建议大家尽量选择国外云服务厂商的产品[1]，因为我有朋友因为使用阿里云香港节点访问 Youtube 而收到过阿里云的邮件警告。我的服务器是在 vultr 上购买的。

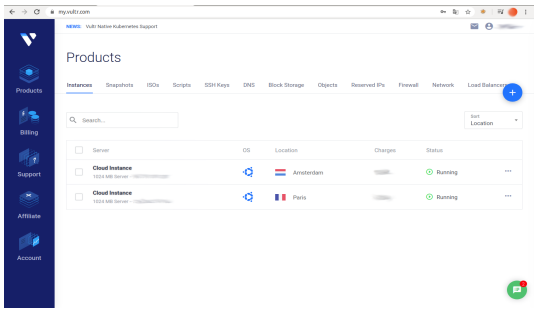

图 1: 我在 vultr 上租借的服务器列表，位置分别在法国和荷兰

支付方式而言，国内银行发行的信用卡和支付宝微信在目前看来是安全的，我在网上看到有人说可以购买国外的虚拟信用卡来支付，但没有实践。

购买服务器时，操作系统尽量选 Ubuntu 的最新版本，因为 Ubuntu 系统的防火墙设置对普通用户最友好。

购买完成后，点击服务器信息，可以看到 root 密码等信息。

有了服务器就可以去部署软件了，先从服务器端说起。首先必须通过 SSH 工具登录，linux 系统

[1]推荐 vultr、digitalocean、谷歌云、亚马逊云

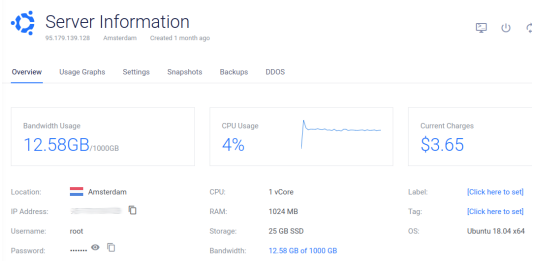

图 2: 服务器信息栏，里面可以找到 Password

可以直接使用 ssh 命令，Windows/Android/iOS 推荐使用 Termius APP。

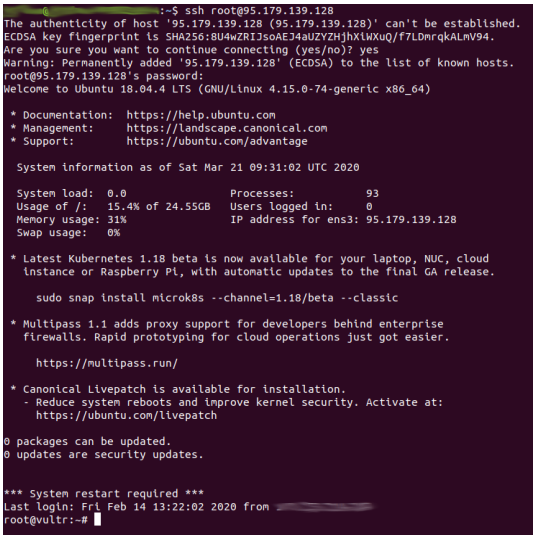

图 3: 使用 ssh 命令登录购买的服务器

以 Linux 系统为例，输入图示的命令，随后输入之前查找到的 root 密码，就可以登录到远程终端。我们需要准备 Shadowsocks 和 V2ray 两个软件，这两个软件都可以在 Github 中找到。我会在外部链接中附上相关软件在 Github 上的页面。Github 上的 shadowsocks 有很多版本，有一些需要自己安装，有一些已经集成到了 Ubuntu 的包管理系统中，在这里推荐 shadowsocks-libev，它可以将 shadowsocks 集成到 Linux 系统的服务中，不需要自己创建进程。首先讲软件的准备，在终端输入：

```
apt install shadowsocks-libev %在
```

```
    Ubuntu软件库中安装shadowsocks-
    libev软件
wget https://github.com/shadowsocks/
    v2ray-plugin/releases/download/v1
    .3.0/v2ray-plugin-linux-arm64-v1
    .3.0.tar.gz %下载v2ray
tar -xvf v2ray-plugin-linux-amd64-v1
    .3.0.tar.gz %解压
mv v2ray-plugin_linux-amd64 /usr/bin
    /v2ray-plugin %移动v2ray到bin文件
    夹中
```

至此我们已经准备好了软件，接下来需要修改配置文件，输入：

```
vim /etc/shadowsocks-libev/config.
    json %使用文本编辑器vim打开配置文
    件
```

将配置文件修改为：

```
{
"server": "服务器IP地址",
"server_port": 服务器端口,
"password": "mypassword",
"timeout": 300,
"method": "aes-128-gcm"
}
```

文件中的 method 指的是加密算法，这里推荐使用 aes-128-gcm。

之后需要在服务配置中设置开启 v2ray 插件功能，在终端输入：

```
vim /etc/default/shadowsocks-libev %
    使用文本编辑器vim打开服务配置
```

将

```
DAEMON_ARGS="-u"
```

改为

```
DAEMON_ARGS="--plugin␣v2ray-plugin␣
    --plugin-opts␣\"server\""
```

最后只需要开启服务即可，命令是：

```
systemctl start shadowsocks-libev
```

为了确认服务开启成功，可以再输入：

```
systemctl status shadowsocks-libev
```

查看服务运行状态。

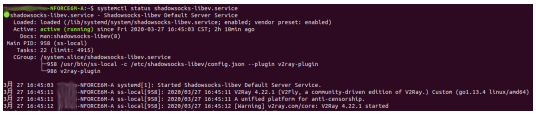

图 4: 服务正常运行时，会看到绿色的 active(running) 文字

配置好了服务器端，我们还需要配置客户端，但遗憾的是，目前四大主流操作系统：Windows、Linux、Android 和 iOS，目前我只掌握了前三个操作系统的解决方案。

1 Windows

点击此处下载 Shadowsocks 的客户端，再点击此处下载 V2ray 客户端，之后将 V2ray 解压后的二进制文件放入 Shadowsocks 的目录中，输入设置好的 IP 地址、加密算法和插件程序即可。

2 Linux

与服务器端设置过程相似。

3 Android

点击此处下载 Shadowsocks 的 Android 客户端，再点击此处下载 V2ray 的 Android 客户端，之后打开 Shadowsocks APP，输入之前设置好的

IP 地址、密码和加密算法，并在插件程序中选择 V2ray。

以上是我设置自己的服务器的过程，该服务器从 2019 年 9 月 23 号开始运行以来抵挡住了多次 GFW 加高的过程，可以初步断定 Shadowsocks+V2ray 的配置在目前来说是安全的。

Part VII

外部链接

github 中的 Shadowsoks 项目主页，内有详实的使用说明和对软件的介绍及讨论。