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

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？

2) 如何建立 DNS Tunnel 绕过认证系统？

3) 是否能够通过 DNS Tunnel 进行网络通信，以及通信质量？

【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】

一、 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。

1. 实验原理分析

1.1. DNS 的运行机制

域名系统(Domain Name System，DNS)是一种实现主机名和 IP 地址之间的映射的层次化的分布式数据库系统[1]。针对用户的 DNS 查询，在做 DNS 查询时，如果查询的域名在 DNS 服务器本机的 cache 中找不到，DNS 系统去互联网查询，通过迭代和递归查询请求的方式，最终返回结果响应。这个过程中，虽然我们没有直接连到外网，因为网关不会转发我们的 IP 数据包出去，但是局域网上的 DNS 服务器帮我们做了中转。

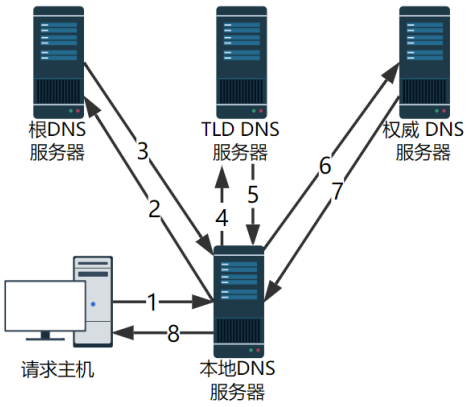

Figure 1 DNS 查询

由于目前一般的防火墙以及网络都不会拦截 DNS 数据包，所以，如果我们在互联网上有一台定制的 DNS 服务器，我们就能利用 DNS 来交换数据包。我们可以基于 DNS 协议，建立 DNS Tunnel 来绕过 Web Portal，穿越网关计费系统，在客户端 Client 和服务端 Server 之间进行数据传输。

DNS 查询的例子如下(端口 4430 用来登陆 web 管理界面，端口 53 用于域名解析)：

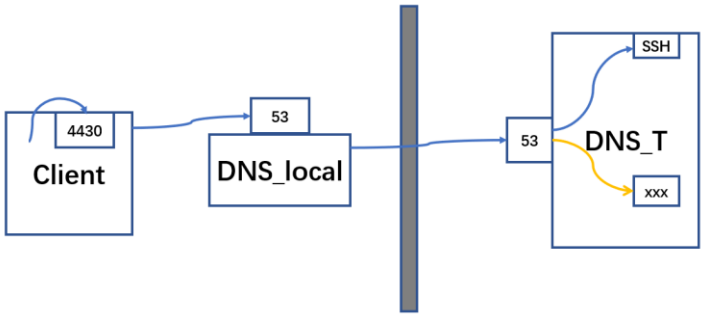

Figure 2 DNS 查询例子

我们连上无密码 WiFi 热点后，就可以使用局域网中的 DNS 服务器 DNS_local，向它的 53 端口发送 DNS 请求，如 abc.def.edu，假设 DNS_local 找不到这个域名，它就会向根 DNS 服务器请求，根服务器发现.edu 的域名后，交给.edu 域名服务器解析，之后又交给 def.edu 的域名服务器，如果存在 abc.def.edu 这条 A 记录，就会返回 abc.def.edu 对应的 IP 地址。反之，不存在，会返回找不到该域名的 IP 地址。

我们可以在 DNS 服务器 DNS_T 上添加一条 A 记录，如： ns.def.edu 222.222.222.222, 再添加一条 NS 记录：def.edu NS ns.def.edu, 指定由公网 DNS 服务器 DNS_T 来解析这个域名，这台服务器运行着 DNS tunnel 的 Server 端，不会返回 abc.def.edu 的 IP 地址，而是将我们的请求转发到设定好的端口，如 SSH 的 22 端口，再该端口返回的数据转发到 53 端口返回给客户端 Client(即我们的电脑或移动端设备)，这样，我们就能利用服务器 DNS_T 的资源了。如果该服务器由 http 或 sock

代理，我们就能利用这个代理穿越网关计费系统来上网了。

1.2. DNS Tunnel 的原理

DNS 隧道技术(DNS Tunnel)，利用 DNS 查询过程来建立隐蔽信道，从而实现数据传输，DNS Tunnel 系统环境如下图所示。

Figure 3 DNS Tunnel 系统环境

图中的符号说明如下：

Client_DNS_tunnel:运行着 DNS Tunnel 软件的客户端 Client，它想绕过防火墙来实现与服务器端 S_DNS_tunnel 进行数据传输。

S_DNS_local:局域网中的 DNS 服务器，负责解析局域网中所有主机的 DNS 请求。

S_DNS_tunnel:运行 DNS Tunnel 软件的服务器端。

数据交换的过程如下：

使用 DNS Tunnel 技术时，需要先在互联网中定制一台服务器 S_DNS_tunnel，这台服务器负责特定域名(例如，”xxx.com”)的解析。局域网中的客户端主机 Client_DNS_tunnel 受到防火墙的拦截，无法直接连到外网，但是可以进行 DNS 查询。

当 Client_DNS_tunnel 想要传递隐蔽信息”password123abc”给 S_DNS_tunnel 时，Client_DNS_tunnel 可以发出 DNS 查询请求”password123abc.xxx.com”，由于这个域名的特殊性，局域网的 DNS 服务器 S_DNS_local 查询不到，只能联系根域名服务器，再一级一级地查询，这个 DNS 请求最终传给 S_DNS_tunnel。

S_DNS_tunnel 可以解析域名得到传输的隐蔽信息。同时，S_DNS_tunnel 可以在 DNS 响应报文中封装隐蔽信息，发回给客户端 Client_DNS_tunnel，这样就实现了数据包的交换。

1.3. Web Portal 的原理

Portal 认证，也称为 Web 认证，是一种重要的 WiFi 网络认证技术，它的好处在于，网络本身不加密，也不需要客户端的认证部署，而是要求终端在 Web 页面进行认证后才能上网。用户在连接到不加密的 WiFi 网络后，当用户访问互联网资源时，会弹出 Web 页面要求用户输入用户名和密码，认证通过后才能上网。

Portal 认证系统主要由认证客户端、交换机、DHCP 服务器、Portal 服务器、AAA 服务器、Bas 设备组成[2]。portal 认证系统的组成示意图如下：

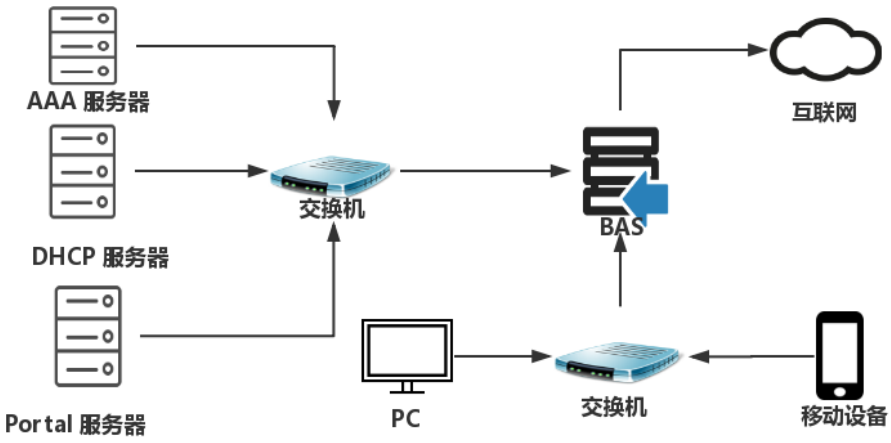

Figure 4 Portal 系统组成

Portal 认证系统各组件的功能如下：

1. 交换机：负责将认证网段中的认证客户端所发出的 HTTP 请求重定向到 Portal 服务器。
2. BAS(Broadband Access Server)：宽带接入服务器，一种设置在网络汇聚层的用户接入服务设备，可以实现用户的汇聚、认证、计费、IP 地址管理等服务。
3. AAA 服务器：负责对客户端 Client 进行认证和计时。
4. DHCP 服务器：自动给内部认证网络的用户分配 IP 地址。
5. Portal 服务器：接受 Portal 客户端的认证请求，并提供 Web 认证页面。
6. 移动设备：通过浏览器来获取网络使用权。

Portal 网络认证的一般流程如下：

1. 客户端设备 Client 连接到不加密的 WiFi 后，Client 向 DHCP 服务器索要一个合法 IP 地址。
2. 得到 IP 地址后，BAS 根据客户端的 IP、MAC 地址、端口号等信息建立 ACL 服务策略，使得客户端只能访问特定类型的服务器，如 Portal 服务器、DNS 服务器和某些内部服务器。
3. 用户在浏览器输入 url 后，客户端会自动跳转访问 Portal 服务器，Portal 服务器通过 BAS 获取用户信息，若用户未经过认证，则提供认证页面。
4. 用户在认证页面输入账号及密码并提交后，账号信息会提交给 BAS，BAS 将验证用户信息，并把信息发送给 AAA 服务器。
5. AAA 服务器认证后，返回结果给 BAS，若认证通过，BAS 将修改用户的 ACL 策略，允

许用户访问互联网。

2. 实验工具与环境

2.1. 实验环境介绍

2.1.1. 服务端

操作系统	Ubuntu 16.04
网络带宽	1Mbps
外网地址	119.29.145.13
Iodine	0.7.0
Shadowsocks	2.8.1

2.1.2. 客户端

操作系统	Window10
TAP-Windows	9.9.2
Iodine	0.7.0
Shadowsocks	4.0.10

2.2. 实验工具介绍

2.2.1. Iodine

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。

2.2.2. Shadowsocks

Shadowsocks 的运行原理与其他代理工具基本相同，使用特定的中转服务器完成数据传输。

在服务器端部署完成后，用户需要按照指定的密码、加密方式和端口，使用客户端软件与其连接。在成功连接到服务器后，客户端会在本机上构建一个本地 Socks5 代理（或 VPN、透明代理）。浏览网络时，网络流量会被分到本地 Socks5 代理，客户端将其加密之后发送到服务器，服务器以同样的加密方式将流量回传给客户端，以此实现代理上网。

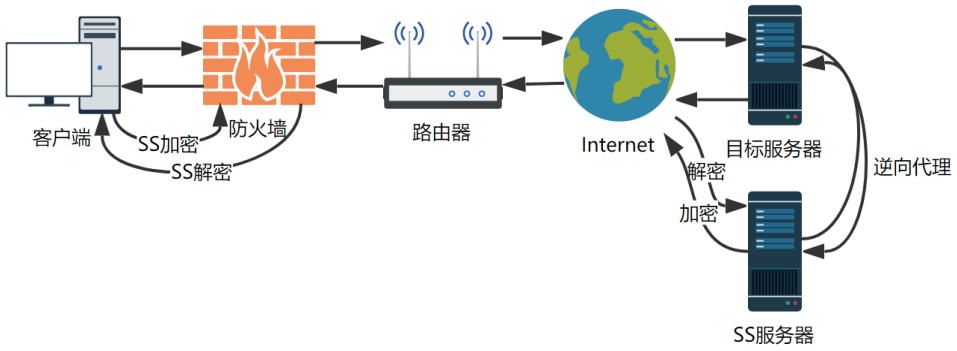

Figure 5 Shadowsocks 原理

通过 Shadowsocks 代理我们可以将数据加密后发出到指定服务器，然后服务器再去访问我们需要的服务器。而不是直接访问目标服务器，这样可以避开防火墙的数据过滤，同时也可以指定将需要的流量发送到指定的服务器。

3. 搭建 DNS Tunnel

3.1. 服务器端配置

3.1.1. 安装必要软件

首先安装 iodine 和 shadowsocks，iodine 可以直接使用 apt-get 安装，shadowsocks 使用 pip 安装。

```
#更新
sudo apt-get update
#安装iodine
sudo apt-get install iodine
#安装python和pip
sudo apt-get install python-gevent python-pip
#安装shadowsocks
sudo pip install shadowsocks
```

3.1.2. 配置 DNS

我们配置 DNS，此处我们使用拥有的域名 chonor.cn 配置 DNS 解析。

新增两条解析，如下：

dns	A	默认	119.29.145.13	-	600	2018-06-30 00:18:35	修改 暂停 删除
wangan	NS	默认	dns.chonor.cn.	-	600	2018-06-30 00:17:44	修改 暂停 删除

Figure 6 DNS 解析配置

两条解析意义如下：

1. 将 dns.chonor.cn 解析到 119.29.145.13 上，A 类型是用来指定域名的 IPv4 地址。
2. 将 wangan.chonor.cn 域名交给 dns.chonor.cn 解析。NS 类型是域名服务器记录，将子域名交给

其他 DNS 服务商解析时类型需要改成 NS，此处就是将 wangan.chonor.cn 交给我们伪造 DNS 服务器解析。

此时当本机查询 wangan.chonor.cn 时，经过根域名服务器和顶级域名服务器等一系列 DNS 服务器后，到达腾讯云的 dns 时。会得到 wangan.chonor.cn 需要 dns.chonor.cn 这个 DNS 服务器来获取 IP 的信息，之后发现 dns.chonor.cn 的 A 记录，得到 dns.chonor.cn 这个 DNS 服务器的 IP，向之后就会向我们伪装 DNS 服务器发起查询 wangan.chonor.cn 的 IP，此时我们就可以通过这个过程进行通信，整个 DNS 查询到我们伪造的 DNS 服务器过程如下：

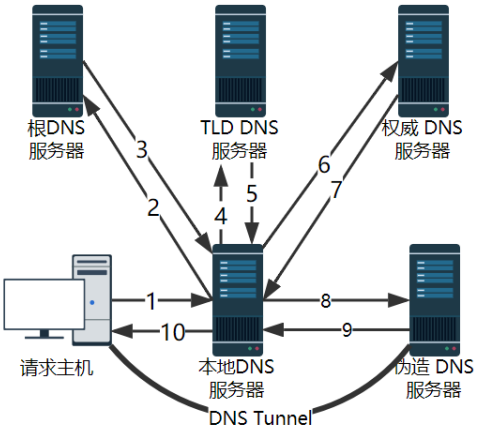

Figure 7 DNS 隧道建立过程

3.1.3. 配置 iodine 服务端

配置命令如下：

```
nohup iodined -f -P 123456 192.168.2.1 wangan.chonor.cn &
# -f 开启前台显示
# -P 连接密码
# 192.168.2.1 内网网段
# wangan.chonor.cn 监听wangan.chonor.cn的DNS查询请求
# nohup [] &  后台持续运行
```

配置后如下

```
Opened dns0
Setting IP of dns0 to 192.168.2.1
Setting MTU of dns0 to 1130
Opened IPv4 UDP socket
Listening to dns for domain wangan.chonor.cn
```

Figure 8 iodine 服务端配置结果

此时说明开始监听 wangan.chonor.cn 的 DNS 查询请求。

3.1.4. 配置 shadowsocks

新建一个 config.json 文件

```
vi config.json
```

config.json 内容如下：

```
{
  "server":"::",
  "server_port":8166,
  "local_port":1080,
  "password":"123456",
  "timeout":600,
  "method":"aes-256-cfb"
}
//server :: 监听所有IP地址
//server_port 8166 shadowsocks 端口为8166
//local_port 1080 本地端口为1080
//passoword 123456 密码为123456
//timeout 600 超时时间 600s
//method aes-256-cfb 加密方式为aes-256-cfb
```

启动 shadowsocks

```
/usr/local/bin/ssserver -c ~/config.json -d start
```

3.2. 客户端配置

客户端只需，安装配置 iodine，iodine 的 window 客户端需要使用到 OpenVpn 的 TAP-Window，TAP-Windos 是虚拟网卡。所以需要先安装 TAP-Windos，此处我们从 OpenVpn 的官网下载 TAP-Windos-9.9.2。此处安装过程省略。

之后我们从 iodine 官方下载最新的 iodine 客户端，解压后使用管理员权限的 CMD 打开。运行命令如下：

```
iodine -P 123456  -f  119.29.145.13  wangan.chonor.cn
# -f 开启前台显示
# -P 连接密码
# 119.29.145.13 服务器地址
# wangan.chonor.cn 需要查询的域名
```

配置后效果如下：

```
E:\网安\iodine-0.7.0-windows\64bit>iodine -P 123456 -f 119.29.145.13  wangan.chonor.cn
Opening device 以太网 6
Opened IPv4 UDP socket
Opened IPv4 UDP socket
Opened IPv4 UDP socket
Sending DNS queries for wangan.chonor.cn to 119.29.145.13
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Enabling interface '以太网 6'
Setting IP of interface '以太网 6' to 192.168.2.2 (can take a few seconds)...

Server tunnel IP is 192.168.2.1
Testing raw UDP data to the server (skip with -r)
Server is at 10.104.175.209, 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.
```

将wangan.chonor.cn 的DNS查询发送给服务器 119.29.145.13

服务器隧道的IP地址是 192.168.2.1

Figure 9 iodine 客户端配置

此时我们可以看到我们分配到 IP 为 192.168.2.2，服务器端的隧道 IP 为 192.168.2.1，因为 iodine 需使用虚拟网卡，此时我们可以看到虚拟网卡名称为‘以太网 6’。

3.3. 连通性测试

我们尝试 Ping 通服务器的 IP

```
PS C:\Users\Chonor> ping 192.168.2.1

正在 Ping 192.168.2.1 具有 32 字节的数据:
来自 192.168.2.1 的回复: 字节=32 时间=20ms TTL=64
来自 192.168.2.1 的回复: 字节=32 时间=21ms TTL=64
来自 192.168.2.1 的回复: 字节=32 时间=20ms TTL=64
来自 192.168.2.1 的回复: 字节=32 时间=21ms TTL=64

192.168.2.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 20ms, 最长 = 21ms, 平均 = 20ms
```

Figure 10 ping 服务器隧道 IP

此时说明我们的 DNS Tunnel 正常，此时我们抓包查看信息，分别对虚拟适配器‘以太网 6’和实际连接外网的网卡抓包。

Time	Source	Destination	Protocol	Length	Info
0.000000	192.168.2.2	192.168.2.1	ICMP	74	Echo (ping) request id=0x0001, seq=1370/23045, ttl=128 (reply in 2)
0.008882	192.168.2.1	192.168.2.2	ICMP	74	Echo (ping) reply id=0x0001, seq=1370/23045, ttl=64 (request in 1)
1.008815	192.168.2.2	192.168.2.1	ICMP	74	Echo (ping) request id=0x0001, seq=1371/23301, ttl=128 (reply in 4)
1.017554	192.168.2.1	192.168.2.2	ICMP	74	Echo (ping) reply id=0x0001, seq=1371/23301, ttl=64 (request in 3)
2.023398	192.168.2.2	192.168.2.1	ICMP	74	Echo (ping) request id=0x0001, seq=1372/23557, ttl=128 (reply in 6)
2.032433	192.168.2.1	192.168.2.2	ICMP	74	Echo (ping) reply id=0x0001, seq=1372/23557, ttl=64 (request in 5)
3.032390	192.168.2.2	192.168.2.1	ICMP	74	Echo (ping) request id=0x0001, seq=1373/23813, ttl=128 (reply in 8)
3.041276	192.168.2.1	192.168.2.2	ICMP	74	Echo (ping) reply id=0x0001, seq=1373/23813, ttl=64 (request in 7)

（a）虚拟适配器“以太网 6”抓包

No.	Time	Source	Destination	Protocol	Length	Info
2	0.917355	192.168.1.211	119.29.145.13	DNS	166	Standard query 0x15c9 NULL 0ueba82\3122db\276\356Y\
3	0.936433	119.29.145.13	192.168.1.211	DNS	163	Standard query response 0xf79a NULL pabagm4a.wangan
4	0.945963	192.168.1.211	119.29.145.13	DNS	96	Standard query 0x33f8 NULL pabigm4i.wangan.chonor.c
5	0.963856	119.29.145.13	192.168.1.211	DNS	169	Standard query response 0x15c9 NULL 0ueba82\3122db\
6	1.932303	192.168.1.211	119.29.145.13	DNS	166	Standard query 0x5227 NULL 0yfbb82\3122db\276\356Y\
7	1.952686	119.29.145.13	192.168.1.211	DNS	162	Standard query response 0x33f8 NULL pabigm4i.wangan
8	1.962034	192.168.1.211	119.29.145.13	DNS	96	Standard query 0x7056 NULL pabqgm4q.wangan.chonor.c
9	1.980351	119.29.145.13	192.168.1.211	DNS	169	Standard query response 0x5227 NULL 0yfbb82\3122db\
12	2.952285	192.168.1.211	119.29.145.13	DNS	168	Standard query 0x8e85 NULL 02gbc82\3122db\276\356Y\
13	2.971456	119.29.145.13	192.168.1.211	DNS	163	Standard query response 0x7056 NULL pabqgm4q.wangan
14	2.977030	192.168.1.211	119.29.145.13	DNS	96	Standard query 0xacb4 NULL pabygm4y.wangan.chonor.c
15	2.995246	119.29.145.13	192.168.1.211	DNS	171	Standard query response 0x8e85 NULL 02gbc82\3122db\
19	3.968205	192.168.1.211	119.29.145.13	DNS	168	Standard query 0xcae3 NULL 0ahbd82\3122db\276\356Y\
20	3.988829	119.29.145.13	192.168.1.211	DNS	163	Standard query response 0xacb4 NULL pabygm4y.wangan
21	3.997750	192.168.1.211	119.29.145.13	DNS	96	Standard query 0xe912 NULL paaagm5a.wangan.chonor.c

（b）实际网卡抓包

Figure 11 ping 服务器隧道 IP 抓包

此时我们可以看到我们 Ping 应该为 ICMP 包，在虚拟适配器中为的确是 ICMP 包，但是此时我们对实际网卡抓包同服务器之间 Ping 已经变成了我们同 DNS 服务器之间的 DNS 包，说明此时实际流量已经伪装成 DNS 包发出。

从虚拟适配器‘以太网 6’中我们可以看到此时通信的 ip 是隧道两端服务器和客户端分配的内网 ip。而我们实际的通信 ip 为我们网卡在路由器下分配到的内网 IP 和我们服务器公网 IP。

3.4. Shadowsocks 代理上网

我们如果需要使用 DNS Tunnel 可以使用 ip route 增加路由规则，但是这样我们要访问的所有地址都需要加到路由规则中，所以我们可以使用 shadowsock 代理上网，直接在 DNS Tunnel 中增加 Shadowsocks 代理。

此处使用 Shadowsock 的 windows 客户端，配置如下：

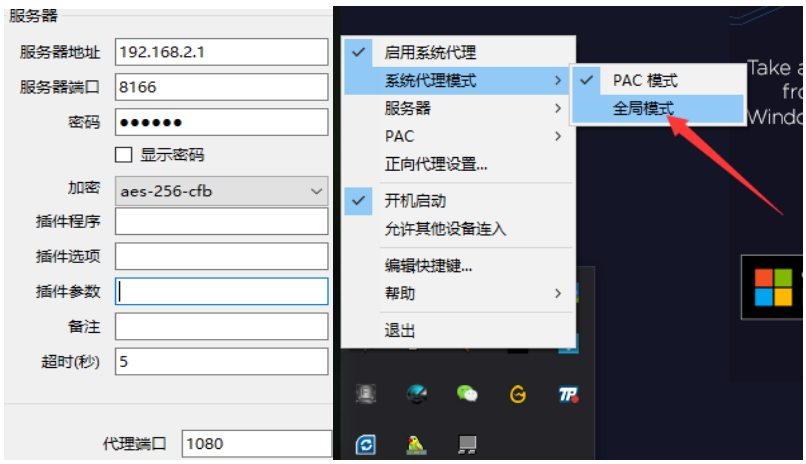

Figure 12 shadowsocks 服务器配置

此时我们打开百度抓包测试 Shadowsocks 是否正常，是否使用了 DNS Tunnel。

1	0.000000	192.168.2.1	192.168.2.2	TCP	66 8166 → 60908 [SYN, ACK] Seq=0 Ack=1 Win=21800 Len=0 MSS=1090
2	0.000073	192.168.2.2	192.168.2.1	TCP	54 60908 → 8166 [ACK] Seq=1 Ack=1 Win=68 Len=0
3	0.000255	192.168.2.2	192.168.2.1	TCP	827 60908 → 8166 [PSH, ACK] Seq=1 Ack=1 Win=68 Len=773
4	0.007969	192.168.2.1	192.168.2.2	TCP	54 8166 → 60830 [ACK] Seq=1 Ack=1 Win=179 Len=0
5	0.027991	192.168.2.2	192.168.2.1	TCP	284 60940 → 8166 [PSH, ACK] Seq=1 Ack=1 Win=68 Len=230
6	0.043871	192.168.2.1	192.168.2.2	TCP	54 8166 → 60827 [ACK] Seq=1 Ack=1 Win=179 Len=0
7	0.091685	192.168.2.1	192.168.2.2	TCP	66 8166 → 60942 [ACK] Seq=1 Ack=1 Win=188 Len=0 SLE=412 SRE=1502
8	0.100259	192.168.2.1	192.168.2.2	TCP	54 8166 → 60870 [ACK] Seq=1 Ack=1 Win=188 Len=0
9	0.108824	192.168.2.1	192.168.2.2	TCP	1144 8166 → 60934 [ACK] Seq=1 Ack=1 Win=205 Len=1090
10	0.117757	192.168.2.1	192.168.2.2	TCP	1144 8166 → 60934 [ACK] Seq=1091 Ack=1 Win=205 Len=1090

（a） 虚拟适配器“以太网 6”抓包

1	0.000000	119.29.145.13	192.168.1.211	DNS	150 Standard query response 0x89e0 NULL paaibqcy.wangan.chonor.cn NULL paaibqcy.wangan.chonc
2	0.000880	192.168.1.211	119.29.145.13	DNS	153 Standard query 0xa80f NULL 0ecbk82\3122db\276\356Y\326Ak\304\341\307\276\346WE\354\275sc
3	0.028990	119.29.145.13	192.168.1.211	DNS	156 Standard query response 0xa80f NULL 0ecbk82\3122db\276\356Y\326Ak\304\341\307\276\346WE\
4	0.029728	192.168.1.211	119.29.145.13	DNS	153 Standard query 0xc63e NULL 0icb182\3122db\276\356Y\326Ak\304\341\327\276\346WE\354\274sc
5	0.037868	119.29.145.13	192.168.1.211	DNS	208 Standard query response 0xc63e NULL 0icb182\3122db\276\356Y\326Ak\304\341\327\276\346WE\
6	0.038465	192.168.1.211	119.29.145.13	DNS	165 Standard query 0xe46d NULL 0mdbm82\3122db\276\356Y\326gk\314\341\303\276\346WE\344\275\3
7	0.046401	119.29.145.13	192.168.1.211	DNS	232 Standard query response 0xe46d NULL 0mdbm82\3122db\276\356Y\326gk\314\341\303\276\346WE\
8	0.047383	192.168.1.211	119.29.145.13	DNS	154 Standard query 0x029c NULL 0qebn82\3122db\276\356Y\326Ak\304\341\323\276\346WE\354\274\3
9	0.055616	119.29.145.13	192.168.1.211	DNS	531 Standard query response 0x029c NULL 0qebn82\3122db\276\356Y\326Ak\304\341\323\276\346WE\
10	0.056726	192.168.1.211	119.29.145.13	DNS	323 Standard query 0x20cb NULL 0ufao82\276COm\363\372aaaacuad\336\334w\342acag\372\311yk\276
11	0.064765	119.29.145.13	192.168.1.211	DNS	377 Standard query response 0x20cb NULL 0ufao82\276COm\363\372aaaacuad\336\334w\342acag\372\

（b） 实际网卡抓包

Figure 13 DNS Tunnel 内建 shadowsocks 代理 抓包

此时我们和服务之间的通信全部交由 Shadowsocks 代理，所有的通信全部 TCP 协议通信端口变成了我们之前设置的 8166，但是此时我们实际网卡中还是显示的还是 DNS 包，说明 Shadowsocks 代理走 DNS Tunnel 是实际可行的。

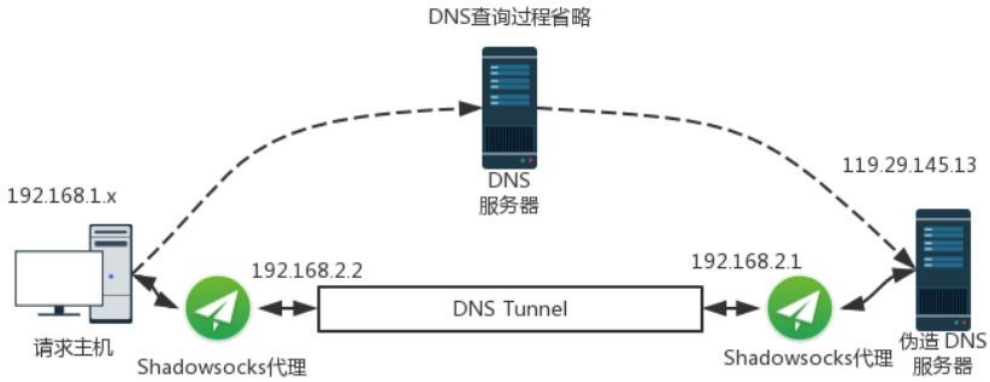

Figure 14 DNS Tunnel 最终拓扑

4. 测试 DNS Tunnel 免费上网

4.1. 上网环境

首先，我们在全家 Family Mart 连接到免费 WiFi，浏览器会弹出以下 Web 认证页面，此时我们还不能上网。

Figure 15　全家 wifi 登录界面

4.2. 测试免费上网

此时我们使用，之前的配置方式进行配置，配置过程略，之前一致。

Figure 16 DNS Tunnel 连接 WiFi

此时测试网络连通性

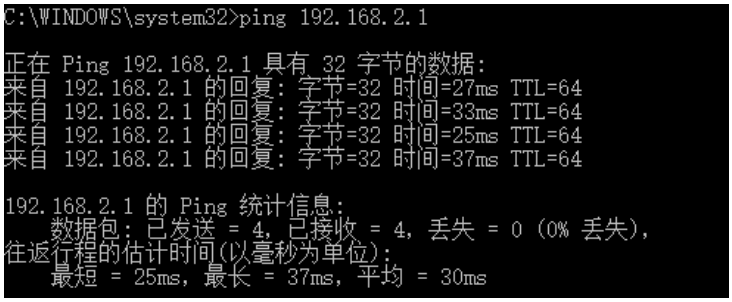

Figure 17 ping 服务器隧道 IP

此时，我们就顺利绕过了网关计费系统，连上了 Family Mart 的 WiFi。

4.3. DNS Tunnel 通信质量

在完成 DNS Tunnel 的配置后，我们已经验证其是能够穿越网关计费系统上网。以及检验 DNS Tunnel 的通信质量。为此，我们在一个测试网速的网站 http://www.speedtest.net/ 上进行网速测试。

由于 DNS 查询基于 UDP,其实就是用 UDP 模拟 TCP，因此效率很低，网速较慢，测试的平均网速只有 0.26Mbps，网速很慢且延迟很大达到 68ms，勉强能够登录 QQ 和微信，在火狐浏览器进行百度搜索时延迟较长，响应速度较。

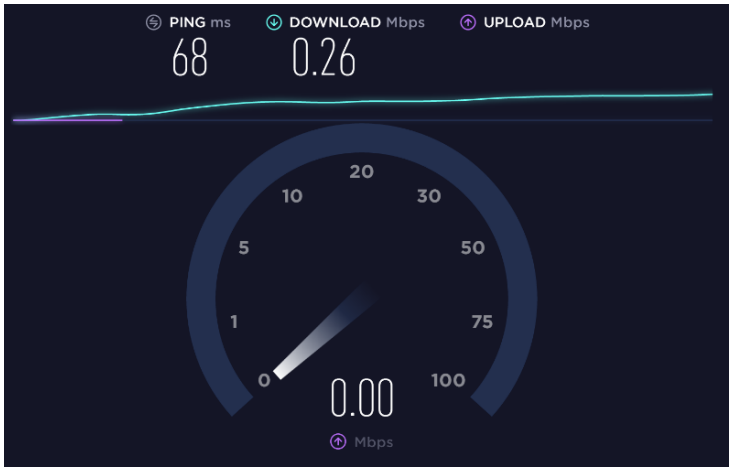

Figure 18 网速测试-DNS Tunnel

为了比较不利用 DNS Tunnel 穿越网关计费系统连接到 Family Mart 免费 WiFi 的数据交换的速度，我们选择通过 Web/Portal 认证的方式连上了 WiFi，同样在网站 http://www.speedtest.net/ 进行网速测试，具体结果如下：

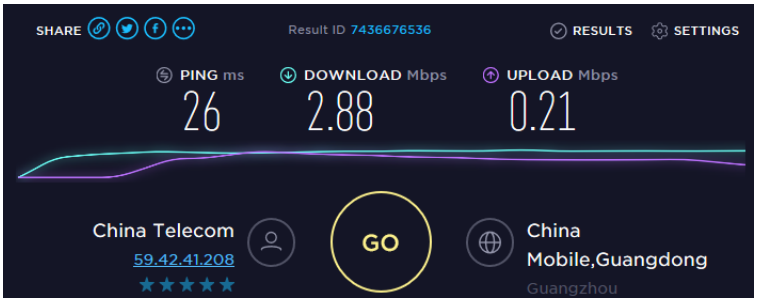

Figure 19 网速测试-Portal 认证

平均的下载速度为 2.88Mbps，是使用 DNS Tunnel 的$\frac{2.88\text{Mbps}}{0.26Mbps} = 11.08$倍，此时的延迟是使用 DNS Tunnel 的$\frac{26\text{ms}}{68\text{ms}} = 38.23\%$，说明了 DNS Tunnel 通信质量确实不乐观，当我们考虑到我们的 VPS 外网出口只有 1Mbps 时，也就是说明此时 DNS Tunnel 最大带宽为 1Mbps，说明此时 DNS Tunnel 的带宽利用率仅为 26%，同时还附带了较大的延迟。

同时我们可以看到在正常认证的情况下，下载曲线（绿色）较为稳定，而使用 DNS Tunnel 的下载曲线稳定性较差。

接下来，我们进行文件传输速度比较。

在使用 DNS Tunnel 穿越网关计费系统来连接到 FamilyMart 免费上网的情况下，在传输较大的图片文件 (http://attach.bbs.miui.com/forum/201507/01/101024hpo6hqhvt77dhsqq.jpg ，大小为 2.36M)时，所需理论时间为：$\Delta t = \frac{2.36M*8bits}{0.26Mbps} = 76.62s$，实际传输时间为2 分 21 秒 = 141s。可能原

因是，在传输图片时，带宽并不会全部都用在图片传输上，很多程序也占用着一定的网络带宽，而且在实验过程中发现。

Figure 20 DNS Tunnel-下载图片

接着，我们选择通过 Web/Portal 认证的方式连上了 WiFi，利用相同版本的迅雷软件，下载同一张图片(2.36M)，所需理论时间为：$\Delta t = \frac{2.36M*8bits}{2.88Mbps} = 6.56s$实际传输时间为 9s，平均下载速度是使用 DNS Tunnel 的$\frac{242.99KB/s}{15.51KB/s} = 15.67$倍。

Figure 21 WiFi 认证-下载图片

根据两者的速度曲线，可以看出，使用 DNS Tunnel 上网时，下载同一图片文件的下载速度很慢，而且波动非常剧烈，而不使用 DNS Tunnel 上网时，下载速度很快，而且下载速度很稳定。所以，DNS Tunnel 的通信质量很较差。

二、 实验体会与感想。

本次期末实验，我们小组经过谨慎思考，最终选定了“利用 DNS Tunnel 穿越网关计费系统”这个主题。我们首先对 DNS 运行的机制进行分析，接着在知网上查找相关文献资料，对 DNS Tunnel 和 Portal 认证的原理进行举例分析，明确了使用 WEB/ Portal 认证方式的无密码 WiFi 认证的漏洞所在之处，可以利用 DNS Tunnel 绕过网关计费系统。

在开放的免费 WiFi 中，在未通过 Web 验证的情况下，除指定 IP 外的服务器会被禁止访问，但是当前 WiFi 热点的 DNS 服务是开启的，即输入一个网址，DNS 服务会将请求发给 DNS 服务器请求域名解析服务。

因为我们的 DNS 请求会被转发至外网，所以可以利用这个漏洞来访问外部 IP，将 DNS 解析请求指向支持 DNS Tunnel 的 DNS 服务器，通过支持 DNS Tunnel 的 DNS 服务器解析我们的请求，随后这个 DNS 服务器会将我们的请求结果发回我们使用的主机，若我们的请求中包括访问一些网站的请求，DNS 服务器会将访问的结果发回，从而实现连接外网。

为完成本次实验，我们选定了一个存在这种漏洞的商业 WIFI 环境——全家 FamilyMart 的免费 WiFi。在进行了 DNS 配置、iodine 客户端服配置、ping 服务器隧道 IP、在服务器搭建好 shadowsocks。然后将 shadowsocks 配置为全局代理后，我们就可以绕过认证透过 DNS 隧道进行上网了。

最后，为了比较 DNS Tunnel 的通信质量，为此，我们做了一系列的对照实验。

首先在一个测试网速的网站 http://www.speedtest.net/ 上进行网速测试，发现不利用 DNS Tunnel 穿越网关计费系统连接到 Family Mart 免费 WiFi 的数据交换的速度是使用 DNS Tunnel 的 11.08 倍，说明了 DNS Tunnel 通信质量确实不乐观。

接着，我们进行文件传输速度比较。在传输同一张大小为 2.36M 的图片文件时，通过 Web/Portal 认证的方式连上了 WiFi，利用相同版本的迅雷软件，平均下载速度是使用 DNS Tunnel 的 15.67 倍。

根据两者的速度曲线，使用 DNS Tunnel 上网时，下载同一图片文件的下载速度很慢，而且波动非常剧烈，而不使用 DNS Tunnel 上网时，下载速度很快，而且下载速度很稳定。可见，DNS Tunnel 的通信质量一般。

综上，我们在完成本次期末实验过程中，学习到了很多计算机网络安全的知识。

三、 参考文献。

[1]谷传征,王轶骏,薛质.基于 DNS 协议的隐蔽信道研究[J].信息安全与通信保密,2011(12):81-82+85.

[2]马燕,范植华.Web/Portal 认证技术研究[J].微电子学与计算机,2004, 21(8):76-79.