Is Anyone Out There? Monitoring DNS for Misuse

GIAC (GCIA) Gold Certification

Author: Kaleb Fornero, f_kaleb_40@hotmail.com
Advisor: Christopher Walker (MS-ISS, CISSP)

Accepted: *12/29/2016*

Abstract

In the early 1980's, a system was born by which millions of users would unlock the untold amounts of computer information located around the world. The creation of the Domain Name System (DNS) not only allowed for the traversal of the Internet with user-friendly URLs, but also created a means of misuse, a means of deception. This paper will outline the way in which DNS may be abused for command and control channels as well as data exfiltration by deconstructing deceptive packets and outlining the anomalies within them. With this analytical information, the development of active network monitoring rules will be provided to detect these irregularities and identify DNS exploitation.

1. Introduction

Most end users leverage the Internet daily, knowing nothing more than their favorite Uniform Resource Locator (URL) which magically takes them to the website they desire. Despite the sophistication of today's computers, these machines, by design, do not understand human numbers or letter characters, they only understand binary code. They do not communicate via names or .com addresses but rather route information based on IP addresses. So how does the end user's URL get translated into the machine language necessary to traverse the far reaches of cyberspace? This is accomplished by a 20th-century technology known as the Domain Name System (DNS), and it is the very system behind how the Internet functions.

Industry analysts can uncover the anomalies which may lie within a DNS packet by first understand a baseline of how DNS operates. The analysis of unusual DNS packets is used to find examples of potential covert channels and data exfiltration. Finally, the development of monitoring rules are designed not only to detect the specific traffic anomalies covered in this paper, but to verify other unknown instances of DNS misuse.

2. Deconstructing DNS

In a simplistic way, DNS is nothing more than a hierarchical system used to resolve human language into one a computer can understand. This is similar to "a phone book for the Internet. If you know a person's name but don't know their telephone number, you can simply look it up in a phone book. DNS provides this same service to the Internet" (Gonyea, 2010). For example, it can take a URL input by the user and determines the associated IP address for the site. Without this system in place, users would only be able to visit the web pages for which they knew the IP addresses, for example, http://216.58.192.174 instead of http://google.com. This would not be very user-friendly or easy to remember.

The process DNS utilizes to turn http://google.com into http://216.58.192.174 is relatively trivial and encompasses five main steps explored in the following sections:

1. The Request

2. Root Server Query
3. Top Level Domain (TLD) Server Query
4. Domain's Name Server Query
5. The Response

2.1. The Request

When a user wants to visit Google.com, it is a seemingly simple request. The URL is input into the browser, and after a second or two, the page loads. In actually, this query is much different and a little more complicated than experienced by the user. When attempting to refine the URL into an IP address "the first place your computer looks is its local DNS cache, which stores information that your computer has recently retrieved" (Gonyea, 2010). While the storage time of this information is limited and varies depending on user activity as well as other factors, this function prevents repeated queries for the same domain going out the DNS servers if a user continuously revisits a site. This request is analogous to looking at a cellphone's call history for a number recently dialed versus looking for the number in the phone book over and over. Thus this process allows the user experience a seamless translation of information while lightening the traffic being resolved by the DNS servers.

In a situation where the computer's DNS cache does not contain the desired URL information, the request is sent out to the Internet. The request is most often sent to a "recursive resolver, which can be operated by your Internet Service Provider (ISP), your wireless carrier, or a third party provider" (Verisign). These devices often maintain private DNS caches which record the IP address to URL conversions for a large number of the ISP's customers. For more frequently visited websites, this is as far as the query goes since an entry is likely to have been stored in the recursive resolver's DNS cache. In these cases, the response is sent to the end user, a process covered in greater detail in section 2.5. In cases where the recursive resolver does not have the IP address cached, it "knows which other DNS servers it needs to ask to answer your original query" (Verisign). In these instances, the request is forwarded to a root server and is the next step in determining the IP address for the requested URL.

2.2. Root Server Query

There are thirteen core root servers (A through M) spread across the world with thousands of load balanced servers supporting them. Together, these servers maintain information on various top level domains (TLD), and are a critical first step in resolving a user-entered URL to an IP address. The DNS root servers can be seen “as a kind of telephone switchboard for DNS” (Gonyea, 2010). While these servers do not contain the IP addresses being queried, they work to funnel the requests to devices that do.

2.3. Top Level Domain (TLD) Query

There are thirteen DNS root servers which identify the correct Top Level Domain (TLD) needed to forward a request. Once a TLD server receives a request, it first verifies that the desired top level domain is one on which it is authoritative, and if so, returns the IP address for the Domain’s Name Server. While thirteen DNS root servers may seem excessive for such a simple task, computers cannot be aware of the domains under each TLD due to the shear amount and ever changing nature of information. Ensuring every computer and smart device around the world has a complete listing of domains under each TLD would simply be a nightmare to scale and maintain. Even if this were possible, the sheer number of domains under each TLD is immense and would take up a considerable amount of space, especially on devices where storage space is critical (e.g. a cell phone).

There are no longer seven TLDs in the world anymore. Currently, there are over 1,500 top level domains (ICANN, 2016). These range far from the standard ".com" and ".org" names most users are familiar with and now include countries, cities, and product brands with their private TLDs such as ".Microsoft" and ".Sony". This recent explosion of TLDs further necessitates DNS root servers and reinforces their place in the DNS hierarchical system.

2.4. Domain’s Name Server Query

From the TLD servers, the request is sent to the Domain’s Name Server. These “authoritative nameservers are responsible for knowing all the information about a specific domain, which are stored in DNS records” (Gonyea, 2010). Also, known as “A” (or “AAAA” for IPv6), these records contain various kinds of information concerning the domain, including hostnames, subnet masks, and the IP address, for the site. This IP address is the answer that has been sought after; it is the information requested at the start of this process. After this has been achieved, there is only one additional step before the webpage loads for the end user; the response.

2.5. The Response

The information return from the DNS record is known as the response and, as mentioned earlier, contains the IP address of the website. This IP will be the address used by the end user's computer to access the requested website and return its valuable content. When the end user receives the response, the "A" record information is stored in the ISP's recursive resolver and the end user's local DNS cache. The storage of this information allows the IP address to be available for return trips by the user or others leveraging the same ISP. This caching reduces the overall amount of DNS traffic and speeds up the entire process for the end user. Since the Internet is a dynamic place, the information in these caches does not last forever and "all records have a time-to-live value, which is like an expiration date. After a while, the recursive server will need to ask for a new copy of the record to make sure the information doesn't become out-of-date" (Gonyea, 2010). This is what allows out dated information to be discarded and helps to ensure users are always able to access a "live" website.

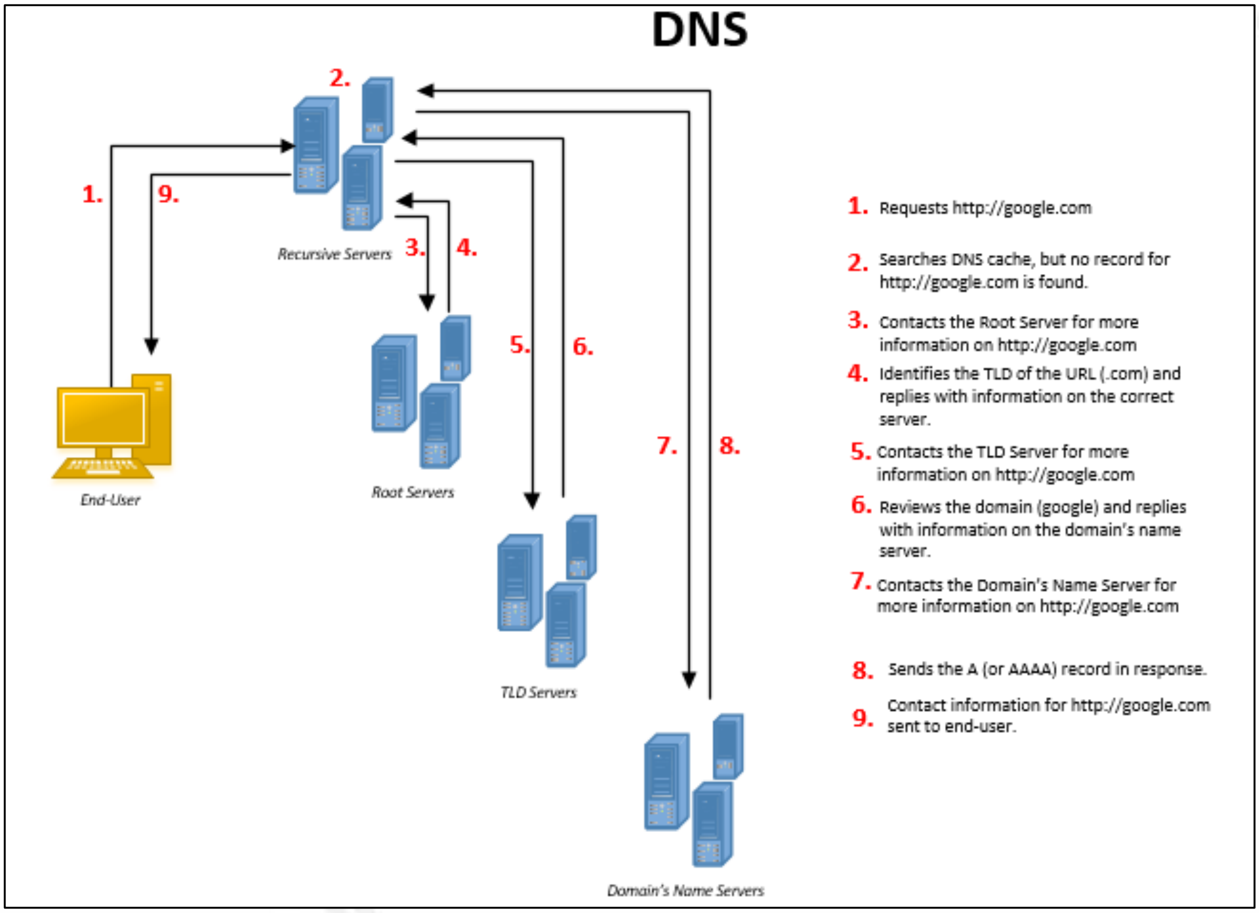
There are multiple steps involved in resolving a user's URL (e.g. http://google.com) to an IP address (e.g. http://216.58.192.174) that is routable on the Internet. While this process does seem complex and sluggish, it actually occurs incredibly fast in real-time. Due to the load balancing of servers and the geographic positioning of critical resources, a typical DNS query is responded to in a few milliseconds.

2.6. Summarizing DNS

Below are three images which demonstrate the steps involved in translating a URL to an IP address leveraging DNS. The first image (Figure 1) shows a high-level overview of the DNS resolution steps, and the second (Figure 2) and final (Figure 3) images illustrate Wireshark's interpretation of a standard DNS query and a typical DNS response, respectively.

Figure 1: High-Level Overview of DNS Resolution Process

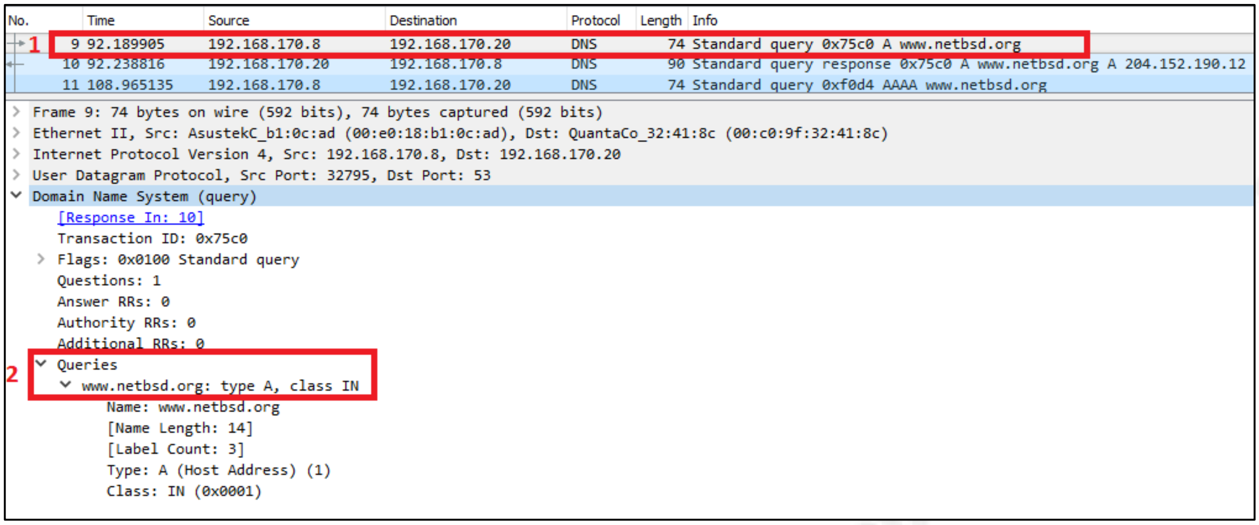
This figure traces an end user's DNS request through the resolution process which provides an IP address for the desired website.



1. Requests http://google.com
2. Searches DNS cache, but no record for http://google.com is found.
3. Contacts the Root Server for more information on http://google.com
4. Identifies the TLD of the URL (.com) and replies with information on the correct server.
5. Contacts the TLD Server for more information on http://google.com
6. Reviews the domain (google) and replies with information on the domain's name server.
7. Contacts the Domain's Name Server for more information on http://google.com
8. Sends the A (or AAAA) record in response.
9. Contact information for http://google.com sent to end-user.

Figure 2: Wireshark's construction of a DNS query

When a user enters the URL http://www.netbsd.org into their internet browser, a packet (Box 1) is sent across the network to retrieve the associated IP address. The specifics of the request can be seen by inspecting the packet details (Box 2).

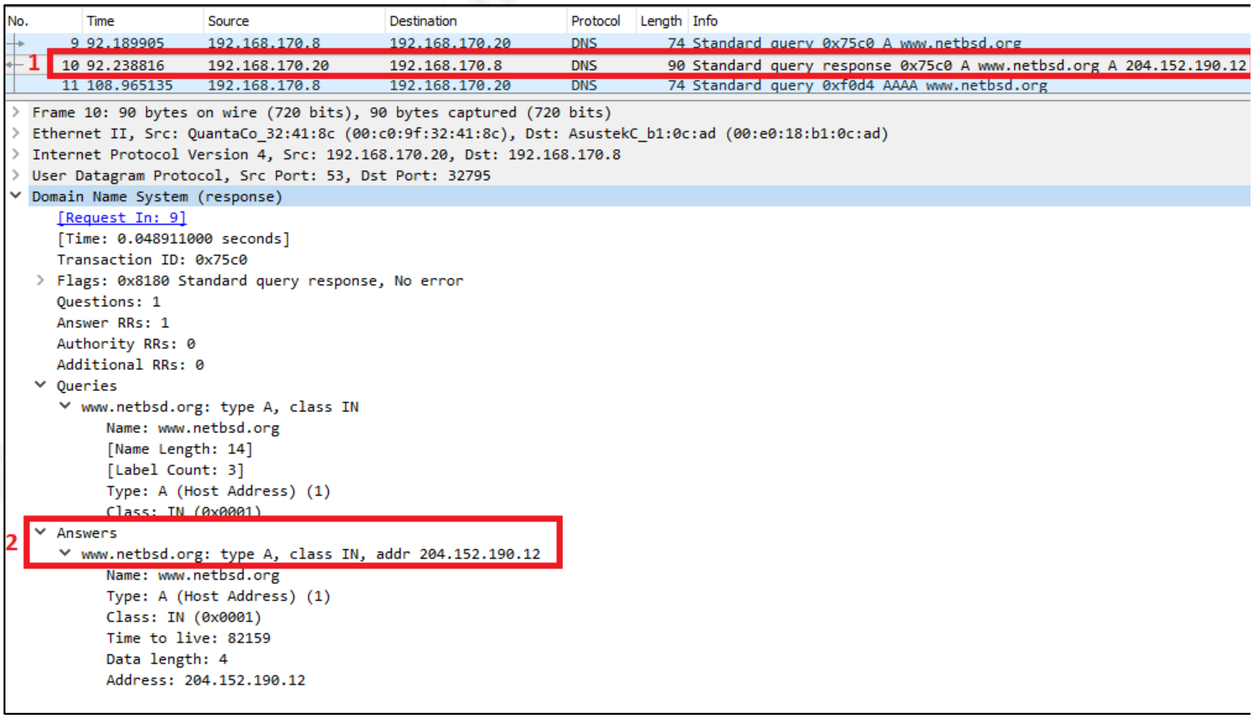


```
No.   Time        Source           Destination      Protocol Length Info
    9 92.189905   192.168.170.8    192.168.170.20   DNS         74 Standard query 0x75c0 A www.netbsd.org
   10 92.238816   192.168.170.20   192.168.170.8    DNS         90 Standard query response 0x75c0 A www.netbsd.org A 204.152.190.12
   11 108.965135  192.168.170.8    192.168.170.20   DNS         74 Standard query 0xf0d4 AAAA www.netbsd.org

> Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: AsustekC_b1:0c:ad (00:e0:18:b1:0c:ad), Dst: QuantaCo_32:41:8c (00:c0:9f:32:41:8c)
> Internet Protocol Version 4, Src: 192.168.170.8, Dst: 192.168.170.20
> User Datagram Protocol, Src Port: 32795, Dst Port: 53
v Domain Name System (query)
    [Response In: 10]
    Transaction ID: 0x75c0
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.netbsd.org: type A, class IN
        Name: www.netbsd.org
        [Name Length: 14]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

PCAP file distributed in the SANS 504 Intrusion Detection In-Depth class

Figure 3: Wireshark's interpretation of a DNS response

When the Domain's Name Server responds to the user's request with the desired IP address, a packet (Box 1) is sent across the network providing the information. The specifics of the response can be seen by inspecting the packet details (Box 2).



```
No.   Time        Source           Destination      Protocol Length Info
    9 92.189905   192.168.170.8    192.168.170.20   DNS         74 Standard query 0x75c0 A www.netbsd.org
   10 92.238816   192.168.170.20   192.168.170.8    DNS         90 Standard query response 0x75c0 A www.netbsd.org A 204.152.190.12
   11 108.965135  192.168.170.8    192.168.170.20   DNS         74 Standard query 0xf0d4 AAAA www.netbsd.org

> Frame 10: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
> Ethernet II, Src: QuantaCo_32:41:8c (00:c0:9f:32:41:8c), Dst: AsustekC_b1:0c:ad (00:e0:18:b1:0c:ad)
> Internet Protocol Version 4, Src: 192.168.170.20, Dst: 192.168.170.8
> User Datagram Protocol, Src Port: 53, Dst Port: 32795
v Domain Name System (response)
    [Request In: 9]
    [Time: 0.048911000 seconds]
    Transaction ID: 0x75c0
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.netbsd.org: type A, class IN
        Name: www.netbsd.org
        [Name Length: 14]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    v www.netbsd.org: type A, class IN, addr 204.152.190.12
        Name: www.netbsd.org
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 82159
        Data length: 4
        Address: 204.152.190.12
```

PCAP file distributed in the SANS 504 Intrusion Detection In-Depth class

In order to understand how a system such as DNS could be misused, one needs to have a fundamental understanding of how the process works and what typical query and response activity looks like on a network. Without this understanding there is a much higher probability the monitoring rules discussed later in this paper will not be as impactful and the meaning behind their development misplaced.

3. Improving the Security of DNS

In the following sections, an exploration of the multitude of ways in which DNS can be misused is provided. For each of these methods, various monitoring options will be proposed as a way to detect this type of abuse in a given network. While these detective measures are critical to alerting investigators, they are only reactive in nature. Thus, for them to trigger, in a true-positive capacity, the DNS system must already have been misused. The goal of this section will be to discuss the main flaws in DNS and methods in which to improve upon them.

The main flaw with DNS is that it trusts the information it receives from the Internet. This type of trust was appropriate years ago, when this system was created, but the Internet of today is a very different place. When DNS "trusts" the information it gets from the Internet, this is in reference to the DNS response. As mentioned in the previous section, after a variety of servers around the world are queried, an A or AAAA record is returned to the inquiring computer. This device assumes the response is correct and comes from an authoritative source. While this is true in most situations, this process does present an opportunity for an attacker to “poison” a DNS cache by injecting false information.

A DNS security tool, DNSSEC, works to resolve this flaw by validating the authenticity of the response received with a public key (known as a DNSKEY) and a private key (known as an RRSIG). For this validation to work correctly, “each answer to a DNS lookup will contain an RRSIG DNS record, in addition to the record types that were requested. The RRSIG record is a digital signature of the answer DNS resource record set” (Mice & Men). With this information, the DNSSEC enabled server can decrypt the response using the public key and the same algorithm used to calculate the original DNS record hash. If the newly calculated hash matches the original attached to the record, the DNS server recognizes the response as authoritative and records the information in its cache. If the hashes do not match, the information is discarded.

Implementing DNSSEC will increase the integrity of the DNS records received on the network; however, there is still one potential flaw in its design. In order for DNSSEC to operate as intended, a public key must be received. Furthermore, the key itself must be maintained by a trusted source that is digitally signed. While this does present more of a challenge to an attacker, there is the opportunity to falsely sign a public key. This can be done to either cause calculated hashes for a particular domain to mismatch the original hash, or to modify the DNS record from the authoritative source so that its hash matches the newly calculated “evil” hash. In the first situation a person may attempt to have the hashes mismatch to ensure all queried records are discarded, thus preventing a DNSSEC enabled server from resolving the domain. In the second situation an attacker may attempt to have the DNS record’s hash match their new “evil” hash to allow false information to be added to the device's cache.

DNSSEC is a tremendous step forward to ensure the information in a DNS cache is correct and from the authoritative source. With that, there is still a need to monitor a company's DNS logs for misuse. This monitoring is still required because for DNSSEC to be successful, all DNS sites must adopt it. There are a few innovative sites which have implemented this technology, but this is only a subsection of the entire DNS infrastructure. Until the time when all domains have implemented this technology, DNS will continue to operate on a trust basis with the Internet.

4. Finding Anomalies in DNS

DNS is a system necessary for the Internet to operate as expected. Due to this, DNS packets are typically not blocked or even examined by security personnel as often as others. This limited review presents an opportunity for an individual with malicious intentions to leverage this service in a way it was not intended.

In this section, two separate aspects of DNS misuse will be examined: using DNS in covert channels, and the exfiltration of data through DNS packets. There are countless examples and unique variants of each of these abuse tactics, but for the purpose of this paper, only a few examples of each will be covered before developing robust monitoring rules to detect this type of network activity.

4.1. Covert Channels

Discreet, secretive, and often unnoticed, covert channels are a common fear among security personnel around the world. These types of communication channels are robust and have a range of uses. For example, "DNS covert channels can be used to bypass a Wi-Fi paywall to avoid paying a service fee, or to run an unapproved application from a work computer… Cyberattackers can use a DNS covert channel in a more dishonest way, such as a communications channel between a computer they have compromised and a computer they operate" (Piscitello, 2016). If this was not disturbing enough, the use of covert channels is on the rise.

The main reason to use a covert channel is to avoid detection and remain in a victim's network as long as possible. However, this is easier said than done. For instance, "from the point of view of a botmaster, a trade-off between C&C communication visibility and the bot inherent need to communicate arises. On the one hand, bots must communicate with their C&C instance to receive instructions and transmit data such as stolen credentials" (Dietrichyz). This communication is needed, but dangerous for the attacker as, it makes their activity vulnerable to detection via well-crafted monitoring rules.

4.1.1. Packet Analysis of a Covert Channel Used for Command and Control Activity

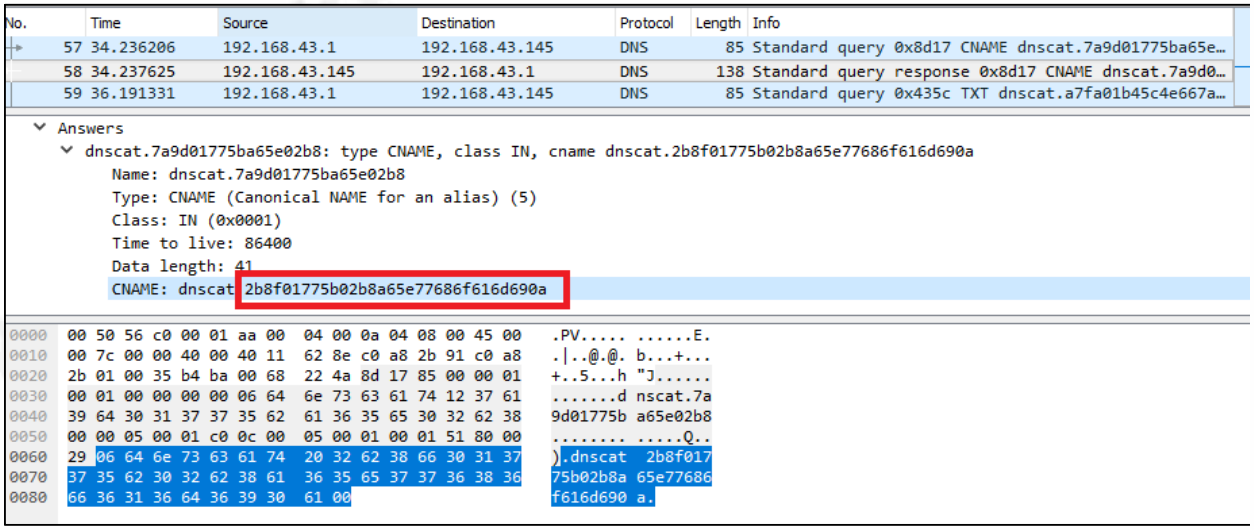
In this example, an exploration of one of the better-known pieces of malware which can leverage DNS for command and control activity, dnscat2, will be provided. At its core, there are two main components to dnscat2: the client, and the server. The client is the code installed on the target computer which sends the "special" DNS packets. The server belongs to the attacker and listens for incoming connections from the client(s).

What makes dnscat2 unique is the fact that it routes DNS traffic through the computer’s specified DNS servers. There's nothing alarming with this since the traffic is being routed just like a regular DNS request. Where things take a turn for the worse is when the query is forwarded to the Domain’s Name Server, which is under the control of the attacker. To add even another layer of obscurity to the mix, the server itself acts differently depending on the traffic it receives. For example, if it receives traffic for one of the client domains, it can try to set up a connection with the client. On the other hand, if it receives other traffic, it can ignore it or be set-up to forward the requests upstream, so the requests can be answered as expected. Using a PCAP file distributed in the SANS 504 Intrusion Detection In-Depth class, multiple examples of command and control traffic transmitted via DNS will be examined.

The first example (Figure 4) shows the attacker issuing the “whoami” command through a DNS query response. It should be noted this is not readily apparent in Wireshark, but after converting the Hex to its ASCII representation (Figure 5), one can see the attackers command.

Figure 4: Typical dnscat2 Response

The red outlined section in this Wireshark image represents the attacker’s “whoami” command being sent to the client system.

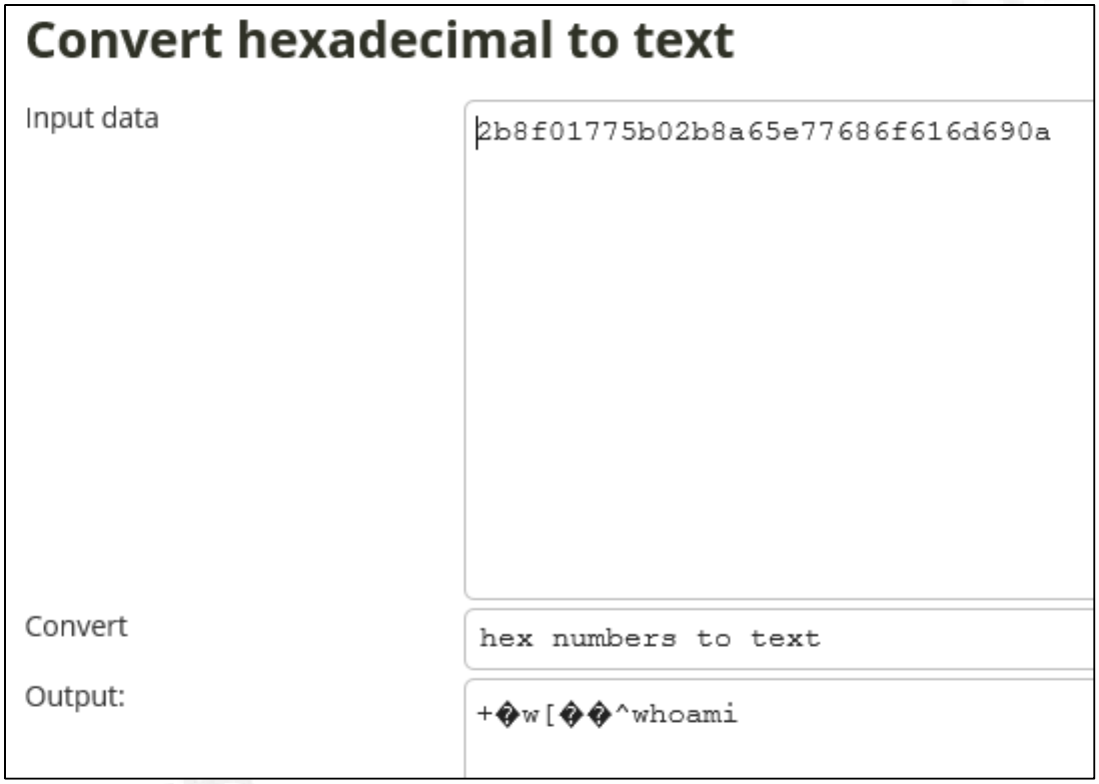


```
No.   Time        Source          Destination     Protocol  Length  Info
   57 34.236206   192.168.43.1    192.168.43.145  DNS           85  Standard query 0x8d17 CNAME dnscat.7a9d01775ba65e…
   58 34.237625   192.168.43.145  192.168.43.1    DNS          138  Standard query response 0x8d17 CNAME dnscat.7a9d0…
   59 36.191331   192.168.43.1    192.168.43.145  DNS           85  Standard query 0x435c TXT dnscat.a7fa01b45c4e667a…

  Answers
    dnscat.7a9d01775ba65e02b8: type CNAME, class IN, cname dnscat.2b8f01775b02b8a65e77686f616d690a
      Name: dnscat.7a9d01775ba65e02b8
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
      Time to live: 86400
      Data length: 41
      CNAME: dnscat 2b8f01775b02b8a65e77686f616d690a

0000  00 50 56 c0 00 01 aa 00  04 00 0a 04 08 00 45 00   .PV..... ......E.
0010  00 7c 00 00 40 00 40 11  62 8e c0 a8 2b 91 c0 a8   .|..@.@. b...+...
0020  2b 01 00 35 b4 ba 00 68  22 4a 8d 17 85 00 00 01   +..5...h "J......
0030  00 01 00 00 00 00 06 64  6e 73 63 61 74 12 37 61   .......d nscat.7a
0040  39 64 30 31 37 37 35 62  61 36 35 65 30 32 62 38   9d01775b a65e02b8
0050  00 00 05 00 01 c0 0c 00  05 00 01 00 01 51 80 00   ........ .....Q..
0060  29 06 64 6e 73 63 61 74  20 32 62 38 66 30 31 37   ).dnscat  2b8f017
0070  37 35 62 30 32 62 38 61  36 35 65 37 37 36 38 36   75b02b8a 65e77686
0080  66 36 31 36 64 36 39 30  61 00                     f616d690 a.
```

PCAP file distributed in the SANS 504 Intrusion Detection In-Depth class

Figure 5: "whoami" HEX to ASCII Conversion

The "whoami" command becomes apparent after converting the hexadecimal representation from Figure 4 to ASCII text.



Convert hexadecimal to text

Input data

```
2b8f01775b02b8a65e77686f616d690a
```

Convert: hex numbers to text

Output:

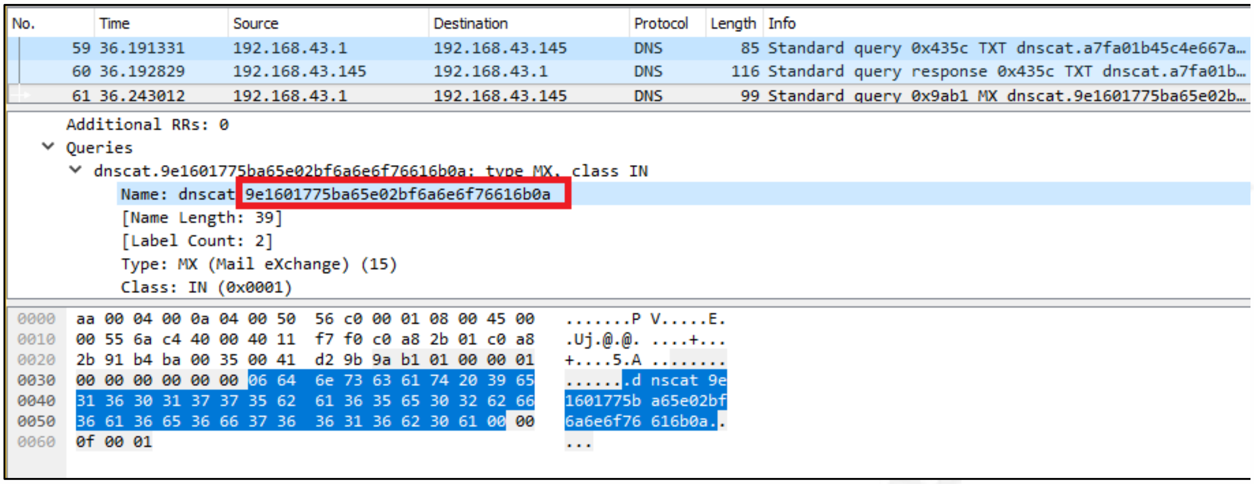
```
+�w[��^whoami
```

Source: http://www.unit-conversion.info/texttools/hexadecimal/

The second example (Figure 6) demonstrates the victim's computer returning its name in a subsequent DNS query. This can be seen after converting the Hex to its ASCII representation (Figure 7).

Figure 6: Typical dnscat2 Query

The red outlined section in this Wireshark image represents the client returning the information requested through the attacker's "whoami" command.

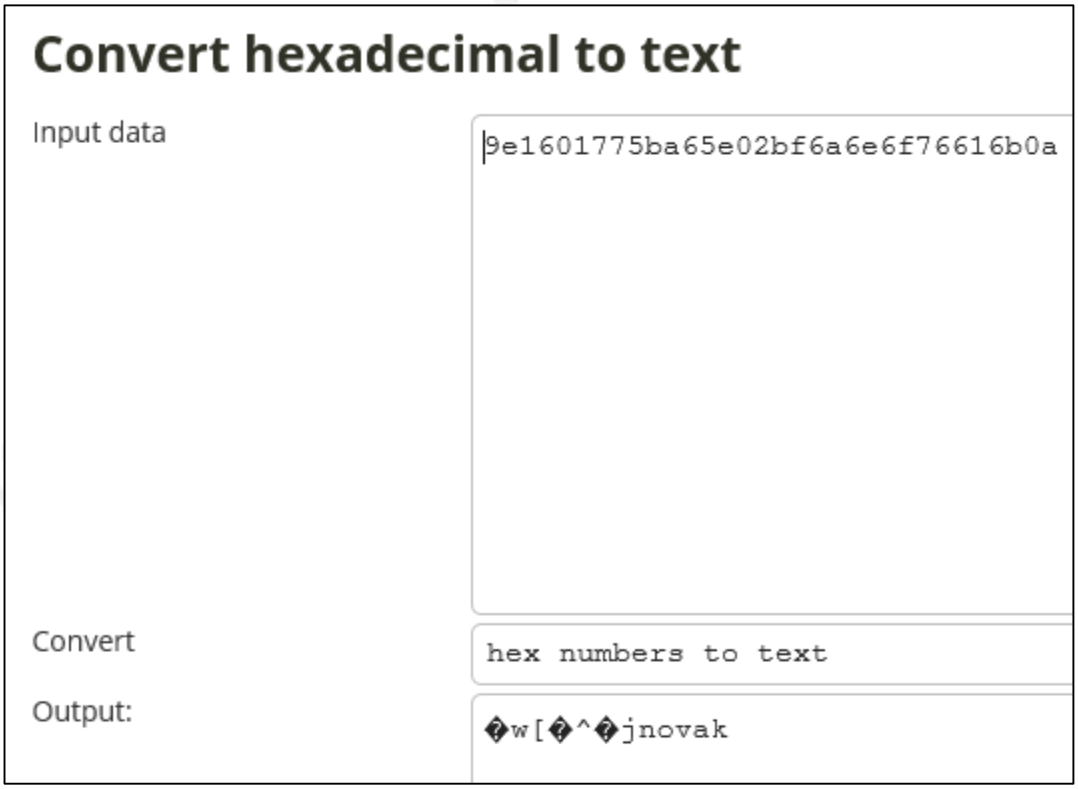


```
No.  Time       Source          Destination     Protocol Length Info
  59 36.191331  192.168.43.1    192.168.43.145  DNS          85 Standard query 0x435c TXT dnscat.a7fa01b45c4e667a…
  60 36.192829  192.168.43.145  192.168.43.1    DNS         116 Standard query response 0x435c TXT dnscat.a7fa01b…
  61 36.243012  192.168.43.1    192.168.43.145  DNS          99 Standard query 0x9ab1 MX dnscat.9e1601775ba65e02b…

    Additional RRs: 0
  ⌄ Queries
    ⌄ dnscat.9e1601775ba65e02bf6a6e6f76616b0a: type MX, class IN
        Name: dnscat 9e1601775ba65e02bf6a6e6f76616b0a
        [Name Length: 39]
        [Label Count: 2]
        Type: MX (Mail eXchange) (15)
        Class: IN (0x0001)

0000  aa 00 04 00 0a 04 00 50  56 c0 00 01 08 00 45 00   .......P V.....E.
0010  00 55 6a c4 40 00 40 11  f7 f0 c0 a8 2b 01 c0 a8   .Uj.@.@. ....+...
0020  2b 91 b4 ba 00 35 00 41  d2 9b 9a b1 01 00 00 01   +....5.A ........
0030  00 00 00 00 00 00 06 64  6e 73 63 61 74 20 39 65   .......d nscat 9e
0040  31 36 30 31 37 37 35 62  61 36 35 65 30 32 62 66   1601775b a65e02bf
0050  36 61 36 65 36 66 37 36  36 31 36 62 30 61 00 00   6a6e6f76 616b0a..
0060  0f 00 01                                           ...
```

PCAP file distributed in the SANS 504 Intrusion Detection In-Depth class

Figure 7: Client's Name HEX to ASCII Conversion

The client's response becomes apparent after converting the hexadecimal representation from Figure 6 to ASCII text.



Convert hexadecimal to text

Input data: 9e1601775ba65e02bf6a6e6f76616b0a

Convert: hex numbers to text

Output: �w[�^�jnovak

Source: http://www.unit-conversion.info/texttools/hexadecimal/

As previously mentioned, there is a balancing act between avoiding detection and usability. The images above demonstrate that while an attacker needs to be able to communicate with their infected hosts to distribute their commands, this makes them visible. During this increased visibility is when detection can occur.

4.1.2. Monitoring for Covert Channels Used for Command and Control Activity

Most of the time, to be useful, an infected host must be able to send and receive information. This communication is called command and control activity and is often necessary for malicious code to be updated or for desired information to be sent to unsavory characters. While command and control activity can be the power of malicious code, it is also its weakness because it makes the code more vulnerable and allows it to be detected. This communication activity is the weakness that will be exploited for the creation of monitoring rules.

Monitoring for dnscat2 is easy. Assuming the source code is not altered, dnscat2 is easy to detect since the queries and responses all have “dnscat” prefixed to them. The following is a simple Snort rule that can be used to identify this traffic:

```
Alert udp $HOME_NET and <> any 53 (msg: “dnscat2 Detection”; content: “dnscat.”;
sid:1234566;)
```

While the detection may be simple for this application, it demonstrates a method to establish a DNS tunnel. Building upon this, more complex monitoring rules can be developed to review network traffic for DNS tunneling when “dnscat” is not a prefix to the queries and responses. To do this, one can leverage the frequency and number of DNS requests and responses to detect this type of traffic.

For example, a Security Analyst has researched a network and deemed any DNS query over the size of 100 bytes is “anomalous” and indicative of activity which warrants investigation. In Snort, the following alert can be set up to detect DNS packets that are over the size of 100 bytes:

```
Alert udp $HOME_NET and -> any 53 (msg: “Large DNS Query”; dsize: >100;
sid:1234567;)
```

Using this Snort alert the network can be monitored for any DNS query packets that are 100 bytes or greater. While Snort alerts will not protect the system or defend against an attacker or their botnet, it will allow an investigator to reduce the time between an attack and when remediation actions can begin.

4.2. Data Exfiltration

Data exfiltration is the end game for most attackers, and it is one of the last actions they will perform on a network. This is because removing data from a network can be a noticeable process, making it easier to set up detection rules for this type of activity. An attacker’s goal is to “remain undetected in the network to gain access to the company’s crown jewels or valuable data. This important data include intellectual property, trade secrets, and customer information. In addition, threat actors may also seek other sensitive data such as top-secret documents from government or military institutions” (Trend Micro, 2013). It is to the attacker’s advantage to remain undetected as long as possible. This helps to ensure multiple means of persistence are installed and targeted data is located and sent outside of the network.

When examining potential data exfiltration incidents or setting up a monitoring rule for their detection, it is important to keep in mind that these are reactive controls. A flaw, vulnerability, or oversight in the network has allowed an attacker to complete their mission and he or she has sent potentially sensitive information outside of the network. Once the data is outside, there is often little which can be done to retrieve it. Often the extent of the damage to a company depends on the data itself. To better understand how to monitor for potential data exfiltration via DNS, an example of what this specific activity might look like will be reviewed.

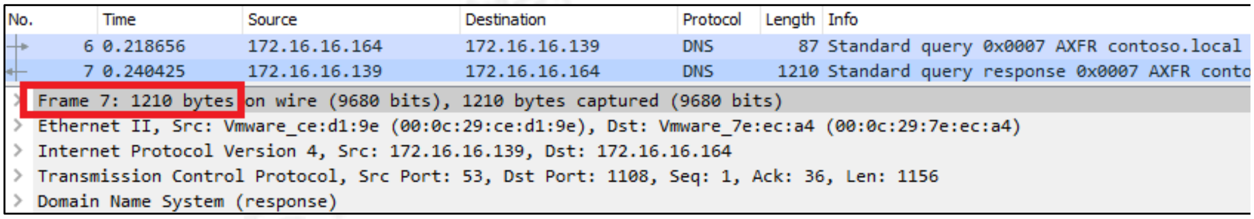
4.2.1. Packet Analysis Showing Data Exfiltration via DNS

In this example, large outbound DNS packets leaving the network will be explored. While this is not always malicious in its intent, it is certainly anomalous and worthy of an investigation. As seen in the "typical" DNS queries and responses in section 2.6, a normal DNS query is about 74 bytes while a standard DNS response is about 100 bytes. These numbers can range in size based on the amount of information being requested or returned, but they will typically stay within this range.

In Figure 8 below, it can be seen that this DNS query response is well over the typical threshold at 1210 bytes. While this is inbound traffic and not necessarily data exfiltration, it can be an indication of the groundwork being set for exfiltration. This response has the potential to contain malicious code, commands, or other unsavory items that have now been delivered to a computer on a given network.

Figure 8: Large DNS Response

The red outlined section shows the total packet size (1210 bytes) of a large DNS response.



```
No.     Time        Source          Destination     Protocol Length Info
      6 0.218656    172.16.16.164   172.16.16.139   DNS        87 Standard query 0x0007 AXFR contoso.local
      7 0.240425    172.16.16.139   172.16.16.164   DNS      1210 Standard query response 0x0007 AXFR conto
> Frame 7: 1210 bytes on wire (9680 bits), 1210 bytes captured (9680 bits)
> Ethernet II, Src: Vmware_ce:d1:9e (00:0c:29:ce:d1:9e), Dst: Vmware_7e:ec:a4 (00:0c:29:7e:ec:a4)
> Internet Protocol Version 4, Src: 172.16.16.139, Dst: 172.16.16.164
> Transmission Control Protocol, Src Port: 53, Dst Port: 1108, Seq: 1, Ack: 36, Len: 1156
> Domain Name System (response)
```

PCAP file distributed in the SANS 504 Intrusion Detection In-Depth class

4.2.2. Monitoring for Data Exfiltration via DNS

Detecting data exfiltration is not the ideal step in which to identify an attacker in a network; it is a reactive stance, and the data has already left the network. However, there is some value that can come from monitoring this activity. The data has been lost, but it can serve as a warning that information is being sent outside out an organization. This alert can be used to mount an investigation and implement mitigation efforts to prevent its continuation.

Monitoring for data exfiltration can be easier given the very nature of information leaving the network. This is because data leaving a network creates a choke point within an organization, an opportunity for the monitoring team. One way to detect data exfiltration is to monitor for abnormally large DNS packets. While these large packets are not necessarily malicious, they do represent anomalous traffic and may be worth investigating. Another way is to monitor for a large amount of DNS traffic over time. This type of monitoring is the better of the two options when looking at DNS traffic because it presents an opportunity to detect significant outbound transfers which occur over time to avoid detection.

Looking at the first method, there are a few items which need to be determined before a Snort rule can be set up. First, what is a large packet on the network? This question may be more applicable in some situations compared to others. When working with a small environment, a "guess and check" methodology can be used to identify this threshold. A Snort rule may be established with an estimated threshold to see what alerts are generated. The more appropriate solution would be to take a sampling of the DNS traffic on a network and perform statistical analysis. Ideally, one would want to gather as much traffic as possible, but because DNS is something “an average user interacts more than 30 times a day” with, this could be a significant amount of traffic depending on the size of the network (Verisign).

Once the data sample is acquired, a statistical standpoint should be considered. First off, is the data normally distributed? What is the mean, median, and mode? What is the standard deviation? These calculation do not need to be performed by hand as there are many pieces of software and websites available to assist in determining this information. The beauty of identifying this information is that once they are calculated, the typical pattern in a network can be determined. The median is the average packet size, but this combined with the standard deviation allows one to calculate packet size ranges to various percentiles. For example, the data residing in the 95th percentile could be determined. Using this percentile would mean the given range accounts for 95% of the DNS traffic and anything above or below that range is the more anomalous 5% of DNS packet sizes. Since DNS packet sizes below the identified threshold are not relevant to the monitoring rule being developed, it is only necessary to focus on the sizes exceeding the range. These large packets would, therefore, account for the upper 2.5% of the DNS packet sizes sampled. Assuming this calculation was completed for a given network, and it was determined that any DNS query or response in the top 2.5% is anomalous traffic, the previous Snort rule monitoring for large DNS queries can be coupled with one looking for large DNS responses:

```
Alert udp $HOME_NET and -> any 53 (msg: “Large DNS Query”; dsize: >100;
sid:1234567;)

Alert udp $HOME_NET and <- any 53 (msg: “Large DNS Response”; dsize: >100;
sid:1234568;)
```

Looking for a single large DNS packet is excellent for detecting script-kiddies or those quickly dumping data from a network. Due to their simplistic nature, these rules are not likely to catch a more determined and patient adversary who is crafting their DNS packets to remain small in an attempt to avoid detection. For DNS transfers over time, a type of calculation should take place where the total amount of data sent from a host, via DNS, is calculated over time and compared to normal transfers identified in section 4.2.2.

Similar to the previous statistical analysis performed, a calculation can be completed to determine the average amount of data sent via DNS for a standard device on a network over a given period of time. To perform this calculation, a Security Analyst will need the DNS data and just like before, the more data, the more accurate the results will be. Since it is desired to analyze the amount of DNS traffic from a single host over time, the next step will be to define that time period. From my experience, most security teams are comfortable with averaging data over a 24-hour period. While a lot that can happen over 24 hours and there are working and non-working hours that can skew the results, it is still an effective starting point. While additional granularity can be added into the calculation (e.g. accounting for weekends and holidays), the easiest place to start the estimation is with the average. To get this, add all of the outbound DNS packets together and divide the results by the number of days in the sample.

While this is only the mean of the data, it is possible to surmise what the average DNS packet transfer rates over 24 hours per host will be on the given network. From here, the same statistical analysis described earlier in this section should be performed to determine a range of typical outbound data and better establish the abnormal threshold to monitor.

5. Conclusion

There is an art form to monitoring rules, a type of trial and error that must take place for analysts to gain the insights and experiences necessary to monitor their environments. Through the multiple examples given for identifying traffic relating to data exfiltration as well as command and control traffic via DNS, monitoring rules built for specific situations are effective, but easily bypassed. Each network is different, and the user activity on that network is as diverse as the company itself.

The more appropriate solution comes from using statistical approaches to determine the routine activity in a network. Alerting based on the frequency and number of DNS queries/responses combined with monitoring for the size of the packets (per packet and over time) are a few ways to provide a dynamic solution for an organization. With these dynamic rules, the alerts are based on unusual activity and not the particular tool. This type of monitoring allows for a more robust monitoring solution that is more diverse than a single tool and will remain relevant longer regardless if the tools used are ones seen before.

DNS is the backbone of the Internet. It is what makes traveling to countless websites and unlocking untold amounts of information possible for the masses. Like most things in life, there are proper ways to use a particular technology, and there are just as many ways to use it in ways it was never intended. This paper explored the high-level topic that is DNS and walked through each of the main steps in this process. A fundamental understanding of a particular technology is imperative to understanding its potential flaws and methods in which it could be exploited. While the ultimate goal of any Security Analyst is prevention, most understand there will be a time where someone accesses the network. In these situations, it is imperative to have the proper monitoring rules in place to alert the appropriate individuals who can investigate and mitigate the situation.

References

Gonyea, C. (2010, August). DNS: Why It's Important & How It Works. Retrieved December 06, 2016, from http://dyn.com/blog/dns-why-its-important-how-it-works/

Verisign. How To Get Online: How the Domain Name System (DNS) Works. Retrieved December 06, 2016, from https://www.verisign.com/en_US/website-presence/online/how-dns-works/index.xhtml

Brain, M., & Crawford, S. (2000, April). How Domain Name Servers Work. Retrieved December 06, 2016, from http://computer.howstuffworks.com/dns.htm

ICANN. (2016, December). List of Top-Level Domains. Retrieved December 7, 2016, from http://data.iana.org/TLD/tlds-alpha-by-domain.txt

Piscitello, D. (2016, December). What Is a DNS Covert Channel? Retrieved December 08, 2016, from https://www.icann.org/news/blog/what-is-a-dns-covert-channel

Dietrichyz, C. J., Rossow, C., Freilingy, F. C., Box, H., Steen, M., Pohlmannz, N. On Botnets that use DNS for Command and Control. Retrieved December 11, 2016, from http://www.few.vu.nl/~herbertb/papers/feederbot_ec2nd11.pdf

Trend Micro. (2013). Data Exfiltration: How Do Threat Actors Steal Your Data? Retrieved December 13, 2016, from http://about-threats.trendmicro.com/cloud-content/us/ent-primers/pdf/how_do_threat_actors_steal_your_data.pdf

Mice & Men. Domain Name System Security Extensions (DNSSEC). Retrieved December 19, 2016, from https://www.menandmice.com/resources/articles/dnssec/