2015

Detection of DNS Based Covert Channels

Stephen Sheridan
Technological University Dublin, stephen.sheridan@tudublin.ie

Anthony Keane
Technological University Dublin, anthony.keane@tudublin.ie

Recommended Citation

Sheridan, S., Keane, A. (2015). In Proceedings of the 14th European Conference on Cyber Warfare and Security (ECCWS), University of Hertfordshire, Hatfield, UK.

Detection of DNS Based Covert Channels

Mr Stephen Sheridan, Dr Anthony Keane
Institute of Technology Blanchardstown, Dublin, Ireland
stephen.sheridan@itb.ie
anthony.keane@itb.ie

Abstract: Information theft or data exfiltration, whether personal or corporate, is now a lucrative mainstay of cybercrime activity. Recent security reports have suggested that while information, such as credit card data is still a prime target, other data such as corporate secrets, employee files and intellectual property are increasingly sought after on the black market. Malicious actors that are intent on exfiltrating valuable data, usually employ some form of Advanced Persistent Threat (APT) in order to exfiltrate large amounts of data over a long period of time with a high degree of covertness. Botnet's are prime examples of APTs that are usually established on targeted systems through malware or exploit kits that leverage system vulnerabilities. Once established, Botnets rely on covert command and control (C&C) communications with a central server, this allows a malicious actor to keep track of compromised systems and to send out instructions for compromised systems to do their biding. Covert channels provide an ideal mechanism for data exfiltration and the exchange of command and control messages that are essential to a Botnets effectiveness. Our work focuses on one particular form of covert channel that enables communication of hidden messages over normal Domain Name Server (DNS) network traffic. Covert channels based on DNS traffic are of particular interest, as DNS requests are an essential part of most Internet traffic and as a result are rarely filtered or blocked by firewalls. As part of our work we have created a test bed system that uses a covert DNS channel to exfiltrate data from a compromised host. Using this system we have carried out network traffic analysis that uses baseline comparisons as a means to fingerprint covert DNS activity. Even though detection of covert DNS activity is relatively straightforward, there is anecdotal evidence to suggest that most organisations do not filter or pay enough attention to DNS traffic and are therefore susceptible to data exfiltration attacks once a host on their network has been compromised. Our work shows that freely available covert DNS tools have particular traffic signatures that can be detected in order to mitigate data exfiltration and C&C traffic.

Keywords: Data exfiltration, covert channels, advanced persistent threat (APT), DNS, botnet, command & control (C&C).

1. Introduction

The latest European Union Agency for Network and Information Security "Threat Landscape Report" (ENISA, 2014) refers to 2014 as the year of the data breach. The scale of data breaches experienced in 2014 ranged from the theft of personal information stored in the cloud to massive targeted attacks on large corporations. Targeted attacks that threaten intellectual property, customer data and sensitive government information use sophisticated attack chains, as shown in figure 1, to compromise networks and carry out malicious activities such as data exfiltration. The creators of targeted attacks go to great lengths to ensure that each step in the attack chain remains undetected. However, it is the command and control (C&C) and data exfiltration phases of the attack chain that generate the most "noise" in terms of network traffic and are therefore, of particular interest to security researchers that wish to mitigate such threats.

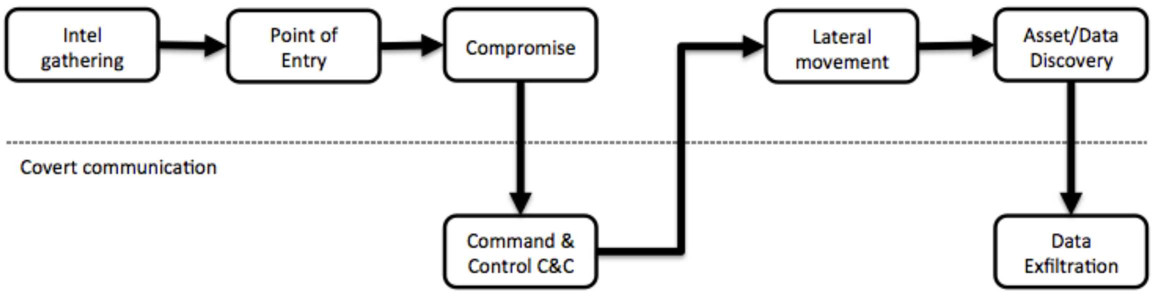


Figure 1: Typical phases of a targeted attack chain

Targeted attacks that include a C&C phase are commonly referred to as Advanced Persistent Threats (APTs). APTs are a category of attack that aggressively pursue and compromise specific targets. APTs often use social engineering techniques such as spearfishing campaigns (Trend Micro, 2012) to gain an entry point into the target network and then attempt to move laterally throughout the network to discover and exfiltrate sensitive assets. C&C communication between the attacker and the compromised host machine is crucial as it allows for:

- Confirmation of system breach and frequent beacon messages
- Information gathering about breached network and hosts machines
- Communication with malware within the compromised network
- Forwarding instructions to download second stage malware
- Data exfiltration

C&C traffic plays a pivotal role in the success of an APT. Therefore, it is in the best interests of an attacker to mask C&C traffic in order to remain undetected. Analysis of known ATPs such as Auora, Zeus and PoisinIvy has shown that ATP creators can use many different forms of C&C ranging from the use of IRC Channels, HTTP/HTTPS Web Traffic (Gu, Zhang & Lee, 2008) and even social media channels such as Twitter (Burghouwt, Spruit & Sips, 2011). Advances in C&C traffic detection have forced APT creators to move towards more covert channels of communication that can sometimes be obfuscated and encrypted. This makes it very difficult for security researchers to distinguish between malicious and benign network traffic.

Our work focuses on a form of covert channel based on the Domain Name Service (DNS) that is responsible for resolving network domain names into IP addresses. Almost all Internet traffic depends on DNS and it is this widespread reliance on DNS that makes it a prime target for use as a conduit for C&C and data exfiltration communications. This paper will focus on the techniques used to embed malicious traffic within DNS packets, it will detail our experimental setup using a freely available tool for covert DNS communication called Iodine (Ekman, 2006) and will identify patterns within Iodine generated DNS traffic that can be used for APT detection. The remainder of this paper will be organised as follows. Section 2 will outline the characteristics of covert channels and will give a brief overview of the DNS protocol. Section 3 will describe our experimental setup using the Iodine DNS Tunnel tool and will outline how it hides data within DNS queries. Section 4 will outline the methods used to profile malicious DNS traffic generated by the Iodine DNS Tunnel software. In section 5 we will discuss our conclusions along with related and future work.

2. Covert channels and DNS

Dietrich *et al*. (2011) have analysed C&C traffic form Malware software known as FeederBot. This ananlysis found that FeederBot did not seem to use any obvious form of communication. By reverse engineering the malware sample the authors found that its C&C communications were hidden in what seemed to be regular DNS network traffic. The work carried out by Dietrich *et al*. and other researchers, is direct evidence that the creators of APTs are developing increasingly sophisticated covert channels over the DNS protocol to send and receive C&C communications and exfiltrate sensitive data. The next two sections of this paper will take a deeper look at the concepts behind covert channels and the DNS protocol in order to demonstrate how this can be achieved.

2.1 Covert channels

The idea of sending a hidden message can be dated back to around 440 BC when Grecians enscribed messages on a wooden tablet, then covered it with wax upon which an innocent covering message was written. There are many examples of covert communication throughout history but one of the first contemporary discussions of covert channels in the context of monolithic software systems is given by Lampson (1973) who defines covert channels as “those not intended for information transfer at all”. The goal is not always, like in the case of encryption, to conceal data, but to conceal the very fact that a communication channel exists.

Simply put, a covert channel is an effective mechanism for sending and receiving information data between hosts without alerting any firewalls or Intrusion Detection Systems (IDS) on the network. The technique derives its stealthy nature by virtue of the fact that it sends traffic through ports that are left open on most firewalls. In addition, the technique can bypass an IDS by appearing to be an innocuous network packet carrying ordinary information when in fact it is concealing its actual data. Covert channels can be used to:

- Steal data
- Evade detection
- Install, spread and control malware
- Bypass government restrictions that limit freedom of expression
- Circumvent pay-walls to access the Internet

Modern Internet communication protocols provide an almost infinite number of ways in which data can be hidden or embed whithin seemingly normal network traffic. This added with the widespread availability of highspeed bandwidth makes hijacking exsitng network protocols an entirely viable option when it comes to exfiltrating large amounts of sensitive data. Fisk *et al.* (2002) put forward some interesting bandwidth numbers based on a large site that had over 500 millions packets of traffic each day. Their research states that based on the assumption that a malicious insider could manipulate 1 bit per packet of data, the site would loose 26 GB of data annually. If the manipulation increased to 8 bits per packet the data loss would rise to 4 GB daily. These numbers are alarming, especially considering that modern covert channels have capacity well in excess of 8 bits per packet.

2.2 DNS overview

The domain name system (DNS) acts as the telephone directory of the Internet. DNS is responsible for resolving human readable and memorable domain names into IP addresses. According to RFC 1035, a domain name can consist of the ASCII characters a-Z, 0-9, and dashes (Mockapetris 1987).

2.2.1 Fully qualified domain name space

Each ASCII character in a domain name is stored as 8 bits and is referred to as an octet. A fully qualified domain name (FQDN) is expressed in terms of a sequence of labels. Each label is represented as a one octet length field followed by that number of octets. Since every domain name ends with the null label of the root, a domain name is terminated by a length byte of zero. Figure 2 shows how the domain name www.itb.ie is represented as a FQDN.

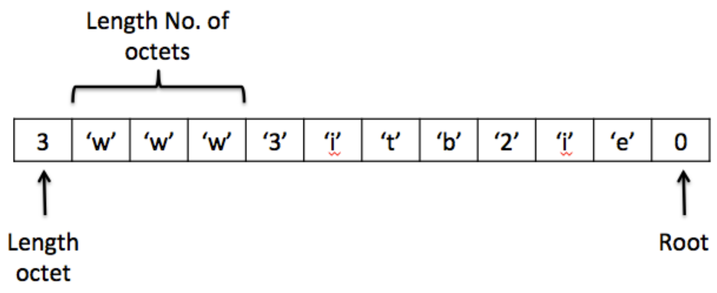


Figure 2: Fully qualified domain name structure.

The maximum size for any FQDN name is 255 octets including the length octets and the root. Therefore, in terms of ASCII characters, the maximum size of a FQDN is 253 octets given that two of the 255 octets are accounted for by the leftmost length octet and the root.

2.2.2 Domain name resolution, a quick overview

When a user types www.mydomain.com into a browser, the browser communicates with a domain name resolver to translate the given URL into an IP address. In order to make the translation, the resolver queries what is effectively a distributed database of DNS records stored on servers located around the globe. The following steps represent a simplified view of the resolution process and assume that a local DNS server does not have the IP address of www.mydomain.com stored in cache.

1. The client requests an **A record,** which represents an IP address, from the local DNS server for www.mydomain.com.

2. The local DNS server receives the requests and it forwards it to one of the 13 root DNS servers.
3. The root DNS server doesn't know anything about www.mydomain.com, but will reply with a referral to the top level DNS servers. In this case the Global Top Level Domain (GTLD) responsible for the .com domain.
4. The local DNS server requests the address resolution from the top level DNS server responsible for the .com domain.
5. The top level DNS server will reply with a referral to the second level DNS server that in this case is the server responsible for the .com domain.
6. The local DNS server requests the address resolution from the second level DNS server (.com server).
7. Because in this case the second level DNS server is authoritative, it will reply with the IP address of the host.
8. The local DNS server replies to the client with the answer.

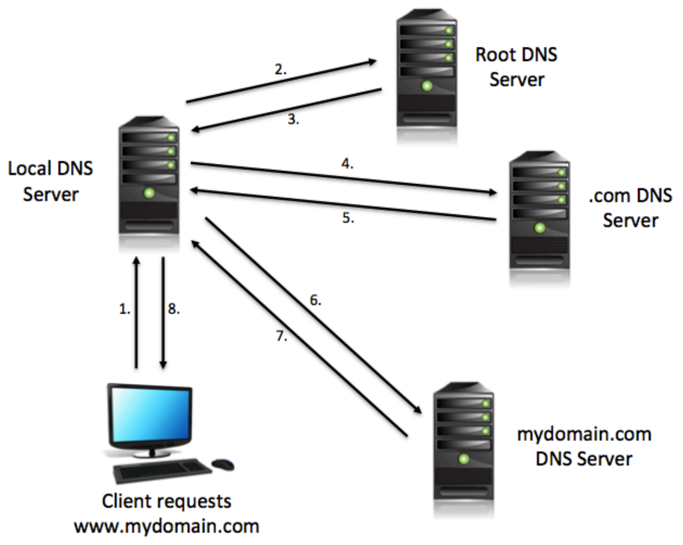


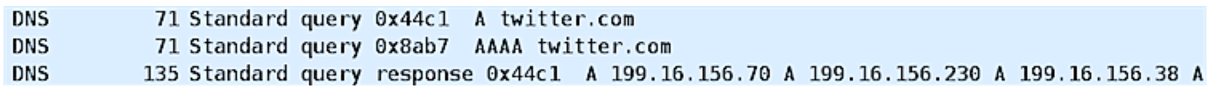
Figure 3: Domain name resolution.

3. Experimental setup using the Iodine DNS tunnel

DNS stands out amongst most protocols as a candidate for covert communication as it is one of the most relied upon components of the Internet and it is largely ignored in terms of security policies and firewall rules. This makes DNS a convenient medium for attackers to exploit and create covert communication channels for their nefarious purposes. A number of DNS tunneling tools currently exist that allow for covert communication over DNS, some are more flexible and stable then others. According to tests carried out by Aiello *et al.* (2012), Iodine shows linear behaviour in various network configurations, has a lower overhead then other DNS tunnel software and is probably the best tool for general purpose DNS tunnelling.

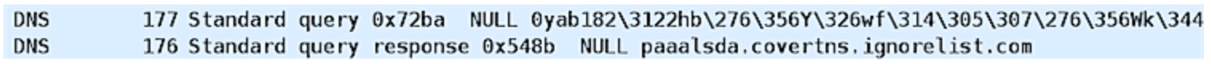
3.1 Payload encapsulation

Iodine, developed by Erik Ekman, allows the tunneling of IPv4 data through DNS. Iodine works by taking advantage of unused domain name space for payload storage that is usually encoded using Base32 or Base64. As can be seen in figure 4 (a) and (b), this specially encoded domain name is then formed into a DNS request that is sent to a controlled domain name server which in turn forwards it to a server running the Iodined server software. The Iodined server software then strips out the encoded portion of the domain name and replies back with a similarly encoded DNS response. Iodine splits IP packets into several DNS packets and sends them separately. IP packets are resembled at the endpoint (in a way similar to IP fragmentation). It is highly portable, working on Linux, MacOS X, FreeBSD, NetBSD, OpenBSD and Windows. According to its creator, it allows up to 1 Mbit/s downstream bandwidth, while upstream is limited. Iodine uses NULL RDATA, which allows every DNS reply to contain over a kilobyte of compressed payload data. Iodine is highly configurable and in cases where NULL RDATA is not available, due to a particular server implementation, it can fall back on other DNS record types such as TXT, Service Record (SRV), MX, CNAME and A records.



```
DNS         71 Standard query 0x44c1  A twitter.com
DNS         71 Standard query 0x8ab7  AAAA twitter.com
DNS        135 Standard query response 0x44c1  A 199.16.156.70 A 199.16.156.230 A 199.16.156.38 A
```

Figure 4 (a): Standard DNS A record request and response (IPV4/IPV6).



```
DNS        177 Standard query 0x72ba  NULL 0yab182\3122hb\276\356Y\326wf\314\305\307\276\356Wk\344
DNS        176 Standard query response 0x548b  NULL paaalsda.covertns.ignorelist.com
```

Figure 4 (b): Iodine TX and RX over DNS Request and Response using NULL Record.

3.2 DNS setup

Iodine requires control over a real domain like mooo.com, and a server with a public IP address to run the Iodine server. A subdomain of the domain name under control, say cns.mooo.com, needs to be delegated to the iodined server. For the purposes of our experiments we used the services of FreeDNS (FreeDNS, 2014) to setup a domain and subdomain similar to that shown in table 1.

Domain name	Record	Address
c.mooo.com	A	192.168.1.10
cns.mooo.com	NS	c.mooo.com

Table 1: Domain name server setup for Iodine DNS Tunnel. **Note**: address for A record should be the external IP address of server running Iodined server software.

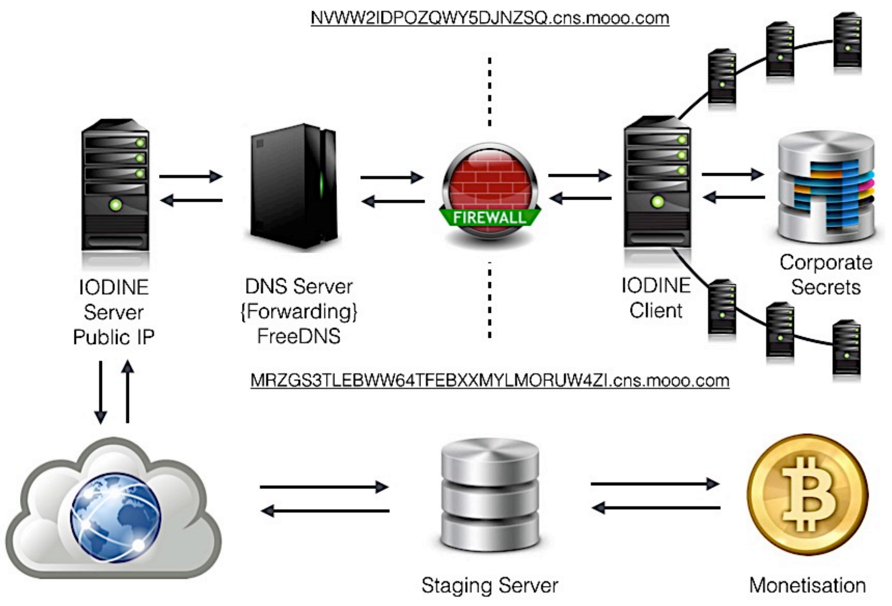

NVWW2IDPOZQWY5DJNZSQ.cns.mooo.com
FIREWALL
IODINE Server Public IP
DNS Server {Forwarding} FreeDNS
IODINE Client
Corporate Secrets
MRZGS3TLEBWW64TFEBXXMYLMORUW4ZI.cns.mooo.com
Staging Server
Monetisation

Figure 5: Typical Iodine setup that could be used for data exfiltration. The top half of the setup (from client to server) reflects our own experimental test bed.

Our experimental setup shown in figure 5 replicates a typical scenario in which Iodine could be used for data exfiltration. Using this setup, it is assumed that the Iodine client software has already been installed on a host machine within an imaginary corporate network. Since the focus of our work is on covert channel detection over DNS it really doesn't matter how our host machine became infected in the first instance. To make the experimental scenario more realistic we have installed a popular firewall called Snort between the Iodine client and server. For the purposes of our experiments we loaded the Snort firewall with the latest set of community rules that include specific alerts and signatures for malware and suspicious DNS activity. As mentioned previously we used the FreeDNS service to setup forwarding DNS records so that our Iodine client could communicate with the Iodine server. The Iodine server was run on an Ubuntu virtual machine hosted on the Okeanos project

(Okeanos, 2011). The Okeanos Infrastructure as a Service (IaaS) project allows for up to two free virtual machines that are assigned external IP addresses. The rest of the infrastructure shown in figure 5 (staging server and monetisation) is merely included to illustrate what happens to sensitive data once it has been successfully exfiltrated.

4. Profiling Iodine DNS tunnel traffic

The experimental system outlined in section 3 was used to profile the Iodine generated traffic. This was achieved by running a number of experiments to generate network traffic that was captured using the WireShark packet analyser.

4.1 Baseline analysis

In order to better understand the nature of DNS traffic generated by the Iodine Tunnel software we first captured baseline network traffic that was comprised of normal network activities such browsing the web and viewing YouTube videos. Our hypothesis was that we should see a correlation between the frequency and amount of HTTP and DNS packets.

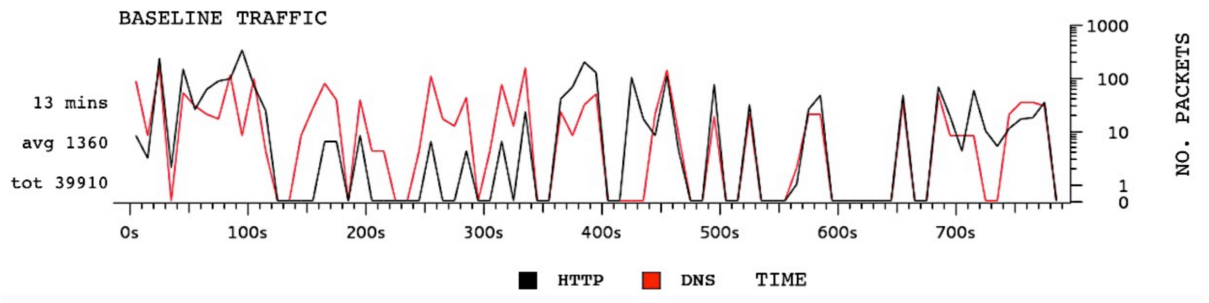


Figure 6: Baseline traffic capture of 39910 network packets captured over 13 minutes with an average packet size of 1360 bytes.

As can be seen from figure 6, our graphs x-axis represents time in seconds and the y-axis represents the number of packets over time on a logarithmic scale. When HTTP and DNS activity are overlaid we see a strong correlation between them in terms of frequency and number of packets. This stands to reason as DNS requests are formed when a user generates HTTP requests by browsing the web.

4.2 Passive tunnel traffic analysis

In our next experiment we established a connection between the Iodine client and server. As DNS is based on UDP, which unlike TCP/IP is a connectionless protocol, the Iodine client and server must continuously poll one another to maintain a connection. This means that even if the DNS tunnel is not actively exfiltrating sensitive data we should still see an elevated level of DNS activity from the frequent polling.

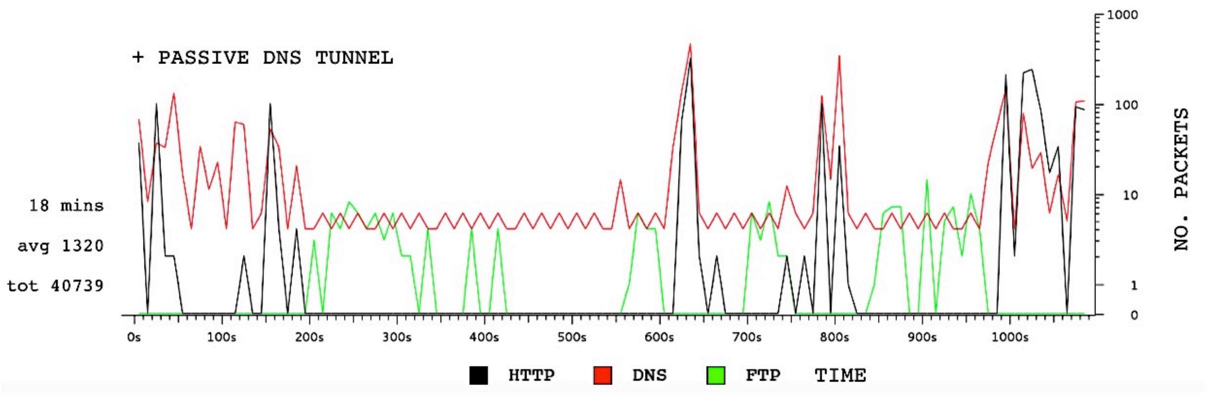


Figure 7: Passive DNS tunnel traffic capture of 40739 network packets captured over 18 minutes with an average packet size of 1320 bytes.

As can be seen from figure 7, even with background FTP traffic included, it is clear that there is an elevated level of DNS activity that for the most part does not seem to correlate with HTTP or FTP traffic. We do witness spikes in DNS activity when there is an associated spike in HTTP activity. However, the DNS activity does not drop back to normal levels once it has served the HTTP requests. The timespan from 200s to 600s should be cause for concern for any network security analyst. While there is a clear increase in background DNS traffic, it is interesting to note there is only a negligible difference in the average packet size of 1360 bytes in figure 6 and 1320 bytes in this experiment.

4.3 Active tunnel traffic analysis (data exfiltration)

The experiment shown in figure 8 represents a typical scenario where a sensitive document with a file size of 98kb is exfiltrated from the system. The file transfer protocol (FTP) was used to send a file out through the tunnel. While this may not be the most clandestine approach it is useful to study in terms of the traffic patterns it produces.

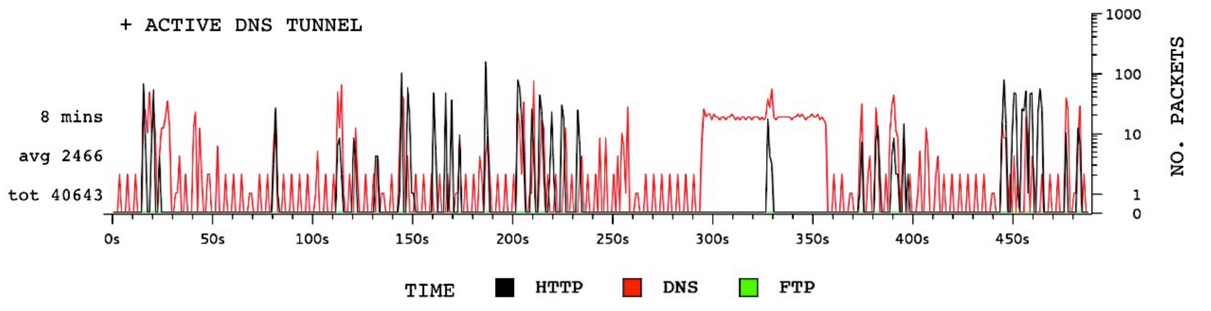


Figure 8: Active DNS tunnel traffic capture of 40643 network packets captured over 8 minutes with an average packet size of 2466 bytes.

As can be seen from figure 8, there is a clear indication of abnormal DNS traffic from 300s to 360s on the timeline. Once again we see higher then normal background DNS activity and an unmistakable spike in activity for the duration of the file transfer. We also see an increase in the average packet size over the experiment timeframe due to larger DNS packets leaving the network.

If we zoom in on the file transfer over the Iodine tunnel, as shown in figure 9, we can see that the transfer occurs approximately 5 minutes into the traffic capture and lasts for approximately 1.5 minutes.

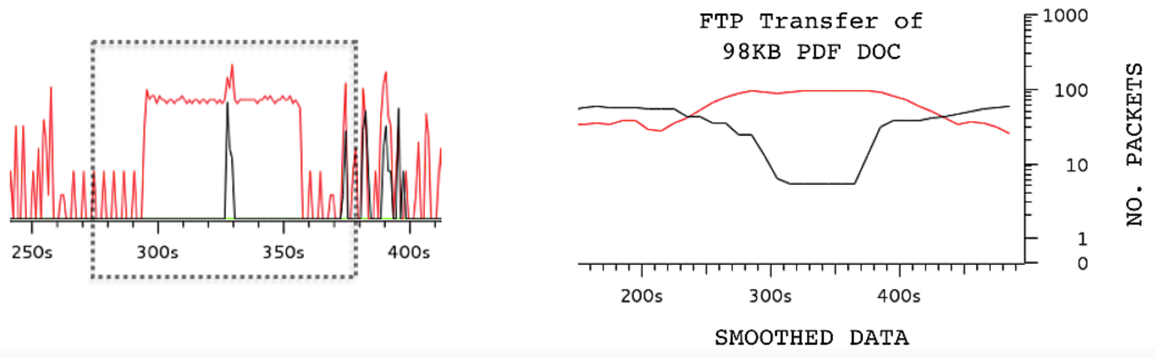


Figure 9: Detailed section showing file transfer.

Even after smoothing out the traffic data to allow for any irregularities, we can still see a clear DNS traffic anomaly where the number of DNS packets increases and the number of HTTP packets decreases during the file transfer.

4.4 Specific firewall rules for Iodine

Given that our traffic analysis tests were conducted without any specific firewall rules to detect Iodine traffic, we decided to include some freely available rules specifically designed to look for Iodine signatures (Chamberland, 2009). Figure 10 shows two Snort rules written by Michel Chamberland at SecurityWire.com.

```
alert udp any any -> any 53 (content:"|01 00 00 01 00 00 00 00 00 01|"; offset:
2; depth: 10; content:"|00 00 29 10 00 00 00 80 00 00 00|";  \
msg: "covert iodine tunnel request"; threshold: type limit, track by_src, count
1, seconds 300; sid: 5619500; rev: 1;)

alert udp any 53 -> any any (content: "|84 00 00 01 00 01 00 00 00 00|"; offset:
2; depth: 10; content:"|00 00 0a 00 01|";  \
msg: "covert iodine tunnel response"; threshold: type limit, track by_src, count
1, seconds 300; sid: 5619501; rev: 1;)
```

Figure 10: Iodine specific Snort Rules.

The two Snort alert rules listed in figure 10 make use of the *content* keyword that allow searching for specific signatures content in the packet payload. Snort uses the Boyer-Moore (Boyer & Moore, 1977) pattern-matching algorithm to check for an exact match to given payload signature. We applied both of these rules to the Snort firewall system and transferred a 98177 byte PDF file across the DNS tunnel using FTP. The file transfer occurred over 10.8 seconds and generated only one alert in the Snort log file as can be seen in figure 11.

```
[**] [1:5619501:1] covert iodine tunnel response [**]
[Priority: 0]
01/29-18:14:17.452827 83.212.127.160:53 -> 10.0.2.15:56166
UDP TTL:64 TOS:0x0 ID:35070 IpLen:20 DgmLen:92
Len: 64
```

Figure 11: iodine generated alert in Snort log file.

5. Conclusions & future work

Our work set out to investigate the detection of DNS based covert channels. We implemented an experimental setup in order to capture baseline and malicious Iodine DNS tunnel traffic for comparison. A firewall with a basic rule set was active during all of our experiments and at no time did it alert or detect any malicious DNS traffic. Our results show that even when the Iodine DNS tunnel is not actively exfiltrating data it still creates a noticeable amount of background DNS packets as a result of its needing to handshake with the Iodined server in order to maintain a connection. This background traffic alone is enough to cause concern and should be easily identifiable by looking at DNS packet frequency. Our deeper analysis of active Iodine traffic shows that exfiltration of sensitive data over the Iodine DNS tunnel, via FTP, results in clearly distinguishable traffic patterns that should raise immediate concern.

It is important to note that the same base line analysis techniques that we have used can provide covert channel developers with valuable information on the behaviour of a network and in particular where it might be possible to further hide malicious DNS traffic amongst other protocols. Butler *et al.* (2011) outline a scheme to distribute malicious DNS packets amongst other network traffic using a poisson distribution resulting in inter-arrival times that follow an exponential distribution. Using an approach like this would camouflage covert traffic even further, making baseline analysis and anomaly detection more difficult.

The Iodine software used in our work generates “new” DNS packets thereby adding to existing network traffic. Born (2010) outlines a system of piggybacking on legitimate traffic instead of creating new network packets. Born shows how it is possible to inject messages and desired recipients into DNS packets that sit at a broker until they can be passively delivered to the appropriate client. The idea of communicating covertly over DNS without creating any new network packets is appealing because existing network traffic characteristics such as number of packets transmitted and packet frequency would not change making baseline analysis far less effective.

As an extension of the work presented in this paper we plan on further investigating more robust DNS covert channel detection techniques that are based on DNS packet frequency and distribution, average packet size and statistical analysis of packet payloads. We also plan to test a number of commercially available firewalls in an out-of-box configuration in order to build a recommended configuration model. Our ultimate goal is to develop a multifaceted detection strategy that will address some of the more sophisticated advances in DNS covert channels described above.

References

Aiello, M., Merlo, A., & Papaleo, G. (2012) Performance assessment and analysis of DNS tunnelling tools. Logic Journal of IGPL, jzr029.

Burghouwt, P., Spruit, M., Sips, H. (2011) Towards detection of botnet communication through social media by monitoring user activity. Information Systems Security, pp. 131–143. Springer

Butler, W. Lampson. (1973) A Note on the Confinement Problem. In communications of the ACM, 16, 10, pp. 613-615.

Dietrich, C.J., Rossow, C., Freiling, F.C., Bos, H., van Steen, M., Pohlmann, N. (2011) "On

Botnets That Use DNS for Command and Control," Seventh European Conference on Computer Network Defense, pp.9,16, 6-7

E. Ekman. (2006) Iodine: IP-over-DNS, IPv4 over DNS tunnel. Available at: http://code.kryo.se/iodine/ [Accessed 25th September 2014].

European Union Agency for Network and Information Security (2014) 'ENISA Threat Landscape 2014'. Available at: https://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/enisa-threat-landscape-2014 [Accessed 10th January 2015].

Free DNS Hosting, Dynamic DNS Hosting, Static DNS Hosting, subdomain and domain hosting. (2001) Available at: http://freedns.afraid.org/ [Accessed October 2014]

G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil. (2002) "Eliminating Steganography in Internet Traffic with Active Wardens", in Proc. Information Hiding, pp.18-35.

Gu, G., Zhang, J., & Lee, W. (2008) BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. Proceedings of the 15th Annual Network and Distributed System Security Symposium.

K. Born. (2011) PSUDP: A Passive Approach to Network-Wide Covert Communication. Black Hat USA.

Mockapetris. P. (1987) Domain names - concepts and facilities. RFC 1034. Available at: http://www.ietf.org/rfc/rfc1034.txt [Accessed 14th of January 2015]

Okeanos, Infrastructure as a service "IaaS". (2011) Available at: https://okeanos.grnet.gr/home/ [Accessed January 2015]

Patrick Butler, Kui Xu, and Danfeng Daphne Yao. (2011). Quantitatively analyzing stealthy communication channels. In Proceedings of the 9th international conference on Applied cryptography and network security (ACNS'11), Javier Lopez and Gene Tsudik (Eds.). Springer-Verlag, Berlin, Heidelberg, 238-254.

R.S. Boyer, J. Strother Moore. (1977) A Fast String Searching Algorithm. Communications of the Association for Computing Machinery, 20(10), 1977, pp. 762-772.

Trend Micro (2011) 'Spear-Phishing Email: Most Favored APT Attack Bait'. Available at: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf [Accessed 2nd December 2014].