# Characterization of Covert Channels in DNS

Hamad Binsalleeh †‡, A. Mert Kara †‡, Amr Youssef ‡, and Mourad Debbabi †‡
† National Cyber Forensics and Training Alliance Canada
‡ Computer Security Laboratory, Concordia University, Montreal, Canada
{h_binsal, ab_kara, youssef, debbabi}@ciise.concordia.ca

***Abstract*—Malware families utilize different protocols to establish their covert communication networks. It is also the case that sometimes they utilize protocols which are least expected to be used for transferring data, e.g., Domain Name System (DNS). Even though the DNS protocol is designed to be a translation service between domain names and IP addresses, it leaves some open doors to establish covert channels in DNS, which is widely known as DNS tunneling. In this paper, we characterize the malicious payload distribution channels in DNS. Our proposed solution characterizes these channels based on the DNS query and response messages patterns. We performed an extensive analysis of malware datasets for one year. Our experiments indicate that our system can successfully determine different patterns of the DNS traffic of malware families.**

***Keywords*—DNS Tunneling, Payload Distribution, Malware**

## I. Introduction

Attackers are known to use different protocols to hide their activities under the radar by tunneling the communication through existing protocols. Such tunneling can effectively defeat traditional firewalls and intrusion detection systems (IDSs). Initially, attackers start exploiting the Internet Relay Chat (IRC) channels to operate and control their activities [8]. Then, they took advantage of newer protocols (e.g., instant messaging, P2P, HTTP) which largely outdated the use of IRC channels [3]. Also, HTTP and P2P protocols are abused by malicious activities such as Zeus [5] (HTTP-based), and Storm [12] (P2P-based). Recently, DNS comes into play for such an abuse due to its wide availability. DNS is a query and response protocol, which responds to each query with the corresponding pre-defined resource record. Architectural flaws in the DNS protocol attracts botnets to abuse the system for different malicious activities [4], [9], [10], [11].

In 2004, Dan Kaminsky [13] demonstrated the feasibility to bypass restricted networks that allow all DNS traffic, such as commercial WiFi hotspots. In this context, DNS is used as a carrier for other protocols by embedding outbound and inbound traffic into query and response messages respectively. Since then, DNS tunneling has been used to design several application tools [14], which operate covert channels through the public DNS infrastructure. Moreover, these tunnels can be established by using free DNS providers, which are already known to be abused for different types of malicious activities [2]. In RSA 2012, Skoudis [20] mentioned an information theft case, which is carried out by a malware family using the DNS protocol to exfiltrate information. For instance, botmasters use DNS query and response packets to carry out malicious instructions and payload updates to individual bots. Recently, few malware families such as Morto [15], Katusha [3], and Feederbot [11], have been identified using the DNS protocol to hide their communications.

Due to the inherent nature of DNS, the protocol is quite inefficient as a payload distribution channel compared to other oftenly used protocols. However, DNS infrastructures are still have been abused by botnet families. Such examples indicate that attackers are willing to exploit DNS as an attack channel due to its wide availability. Past work on DNS abuses [11] mainly focused on specific botnets, and has not been comprehensively studied as compared to e.g., P2P botnets [12].

Malicious networks abuse DNS protocol to distribute attack payload by different behaviors. For instance, Morto uses DNS queries to transfer only single attack payload [15]. However, Feederbot [11] exchanges many parts of attack payload information with the infected machine. In this paper, we characterize the malware families that are using DNS protocol to transfer malicious payloads. The proposed method determines and distinguishes between different query and response patterns. It takes a set of DNS query and response messages for a specific domain name, and then determines the DNS traffic exchange pattern. We use our method to highlight the fundamental characteristics of a payload distribution channel. As for the evaluation of our method, we utilize an extensive malware dynamic analysis reports, which contain detailed behavioral actions conducted by malware samples including network communications. The main contributions of this paper can be summarized as follows:

- We introduce a technique to determine channel patterns and discuss the effectiveness of each pattern in distributing payload distribution over DNS. We found that most of the malware instances are using a specific pattern that can blend within the daily network traffic.
- Evaluation of the proposed method with a 1-year malware dataset covering Jan.-Dec. 2012.

The rest of the paper is organized as follows. Related work is reviewed in Section II. Our system is described in Section III. Section IV demonstrates the effectiveness of our proposed approach via an experiment on 1-year malware dataset. We give discussions and limitations of our work in Section V, and Section VI provides the concluding remarks.

## II. Related Work

The use of DNS as a communication medium for payload distribution is relatively new and research activities on this topic are limited. Although, these studies are scattered, they can be roughly grouped under three categories: feasibility of using DNS in malicious activities, detection of malicious channels in the DNS protocol, and detection of DNS tunnels.

*Feasibility of using DNS for Malicious Activities:* Xu et al. [23] introduce a resilient mechanism for bots to create covert chan-

978-1-4799-3223-8/14/$31.00 ©2014 IEEE

nels through DNS for command and control communications. They design a stealthy C&C, that supports two different modes. The *Codeword mode* creates a uni-directional communication channel that pulls the attack payload. The *tunneled mode* creates a bi-directional communication channel between bots and the command and control server. The authors also mention some techniques to increase the stealthiness of these channels to make the communication channel virtually undetectable from the host perspective. In fact, their proposed methods are already used by some malware families such as Feederbot and Morto [11], [15]. Similarly, Raman et al. [18] propose a network penetration technique that uses DNS tunneling to infiltrate a secure network to deliver an attack payload. Their technique is based on establishing a tunnel by an exploit code that generates DNS queries. There are several studies on building a covert channel using DNS query and response packets [7], [14], [16], [21]. They discuss the possibility of sending and receiving data through DNS query response packets as well as the performance analysis of existing DNS tunneling tools.

*Malicious Channels in DNS Protocol:* Dietrich et al. [11] are the first to discuss the existence of botnets, which tunnel the command and control channels through DNS. They discovered a malware family Feederbot that exfiltrates data within DNS query sub-domain labels, and infiltrates the attack payloads in DNS response packets. Their detection method introduces the extraction of several features from the response data. While their work showed promising results, it is limited to the detection of aggressive DNS tunnels for command and control channels. Malware families are using more resilient methods for receiving the attack payloads through DNS rather than DNS tunneling [15]. Also, their work focused on the assumption that there will be a certain degree of traffic while some families use the DNS to receive a very limited amount of payload such as the Morto family [15]. Moreover, malware might not receive Base32 or Base64 encoded payload, rather clear text in `TXT` records.

*Detection of DNS Tunnels:* There are some proposed methods for detecting DNS tunneling within a network by using the n-gram analysis [6], [17]. They presented promising results in terms of detecting the tunnels. However, malicious payload distribution channels often do not have extensive upstream data; thus they do not show this characteristic feature of DNS tunneling tools. Therefore, any string based analysis on the queries might not reveal enough differences between regular and malicious queries to detect these channels.

## III. System Description

Our system monitors DNS queries and responses that are conducted by malware samples and characterize payload distribution channels established within DNS messages. As shown in Figure 1, the system consists of one main module, which we label as the payload distribution characterization module. Initially, the system pre-processes the malware behavioral reports and extracts DNS messages that have `TXT` resource record activities. Then, it aggregates the DNS query and response messages of a given domain name. After the pre-processing phase, the collected messages are fed to the query and response pattern analysis module, which provides the payload distribution channels characteristics. In the following section, we provide details on how to characterize payload distribution channels.

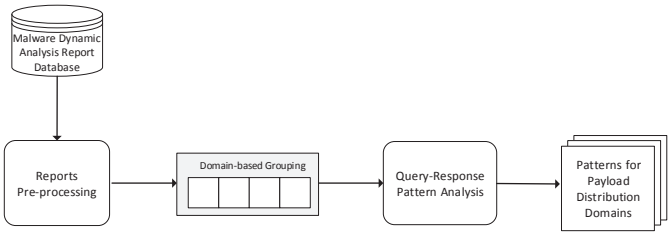



Fig. 1: System overview.

### A. Query and Response Patterns

DNS protocol is based on query and response messages to manage domain name systems. A query from any client can be formed to retrieve different information from a name server, which will respond accordingly. By observing the communication between client and server, we can model the relation between query and response messages. The query and response relations can be used to distinguish between different behaviors of payload distribution. When we observe any payload distribution activity, we have three parameters that are used to establish the channels in DNS. These parameters are the second-level domain, sub-domain, and `TXT` record. The second-level domain is used to carry out payload distribution activity. The sub-domain is used to transfer any information from the client to the server. The `TXT` record is the response information from the server to the client. During any session, the client and the server agree on a specific domain name to be used for the payload distribution activity. Therefore, the second level domain parameter is determined before any session. Now, we are left with two parameters that are used to form the communication channel. Based on the nature of the payload distribution channel, these two parameters have different behaviors. The aim of query and response pattern analysis module is to differentiate between different behaviors of payload distribution. To achieve this goal, we analyze the exchange behavior of query and response messages. This module is built based on two observations:

*Observation 1:* Payload distribution channels through DNS are forced to transfer small quantities of information with each DNS message because DNS response packets are limited to 512 bytes of characters if Extension mechanisms for DNS (EDNS) is not used [22].

*Observation 2:* Transferring more information through DNS protocol, it results in a high rate of DNS queries and responses between the client and the name server [21].

Figure 2 shows different payload distribution scenarios that are considered as the main four possible behaviors out of the two payload distribution parameters as sub-domains and `TXT` records. Figure 2a explains how the client is changing the sub-domains to send data to the name server, which will respond by the corresponding `TXT` records for each sub-domain; (Many-to-Many relation). Figure 2b shows how the client is changing the sub-domains to update the name server about its status, and the server is replying with the same `TXT` record for all possible sub-domains; (Many-to-Single relation). Figure 2c explains how the client is sending the same sub-domain that is

answered by several TXT records from the server; (Single-to-Many relation). This case rarely occurs within a small period of time because these responses are cached by caching resolvers for a period of time (TTL). Figure 2d shows how the client is sending the same sub-domain that is answered by only one TXT record from the server; (Single-to-Single relation). In general, malware families retrieve updates from a malicious infrastructure in three different approaches: full payload updates (Many-to-Many relation) , periodical updates (Many-to-Single relation), or one-time update (Single-to-Single relation).

*Definition 1:* The query-response pattern model is a tuple $G = \langle \{d \cup T \cup S\}, E \rangle$, where:

- $d$ is domain name node,
- S = $\{s_1, s_2, \cdots, s_m\}$ is a finite set of sub-domain nodes,
- T = $\{t_1, t_2, \cdots, t_k\}$ is a finite set of TXT record nodes,
- $E \subseteq (D \times S) \cup (S \times T)$ is a finite set of pairs of distinct nodes, called edges.

We model the query and response relationship for each domain using a directed graph as captured by Definition 1. For each vertex $v$ in $G$, we define two functions: the in-degree of $v$, which is denoted by $inDegree(v)$, returns the number of *entering* edges to the node $v$: $inDegree(v) = |\{u \in V \mid (u, v) \in E\}|$, and the out-degree of $v$, which is denoted by $outD(v)$, returns the number of *leaving* edges from the node $v$: $outDegree(v) = |\{u \in V \mid (v, u) \in E\}|$.

As shown in Figure 2, the query and response relationship patterns share some properties as given by Property 1.

*Property 1:* $inDegree(d) = 0$, $outDegree(t_k) = 0$, $outDegree(d) = inDegree(s_m)$, and $outDegree(s_m) = inDegree(t_k)$

In order to distinguish between the patterns shown in Figure 2, we define the normalized distance measure between two non-zero integer values $i_1, i_2$ as formulated in Equation 1.

$$Dist(i_1, i_2) = \frac{|i_1 - i_2|}{max(i_1, i_2)} \quad (1)$$

Since the query and response patterns can form complex relationships, we limited our system to recognize only one pattern for each domain. Therefore, we have to choose a candidate node to represent the set. As matter of fact, the system selects the *most commonly used node degree* inside the targeted set. Let $t \in T$, and $s \in S$, each query and response pattern can be recognized by the following properties:

*Property 2:* Given that $inDegree(t)$ is the common value in $inDegree(T)$, then Many-to-Many pattern holds when, $Dist(outDegree(d), inDegree(t)) \geq 0.5$, and $Dist(|S|, |T|) < 0.5$.

*Property 3:* Given that $inDegree(t)$ is the common value in $inDegree(T)$, then Many-to-Single pattern holds when, $Dist(outDegree(d), inDegree(t)) < 0.5$, and $Dist(|S|, |T|) \geq 0.5$.

*Property 4:* Given that $inDegree(t)$ is the common value in $inDegree(T)$ and $outDegree(s)$ is the common value in $outDegree(S)$ then Single-to-Many pattern holds when, $Dist(outDegree(s), inDegree(t)) \geq 0.5$, and $Dist(|S|, |T|) \geq 0.5$.

*Property 5:* Given that $inDegree(t)$ is the common value in $inDegree(T)$ and $outDegree(s)$ is the common value in $outDegree(S)$, then Single-to-Single pattern holds when, $Dist(outDegree(s), inDegree(t)) < 0.5$ , $Dist(outDegree(d), inDegree(t)) < 0.5$ , and $Dist(|S|, |T|) < 0.5$.

**Algorithm 1:** EXTRACTQUERYRESPONSEPATTERN

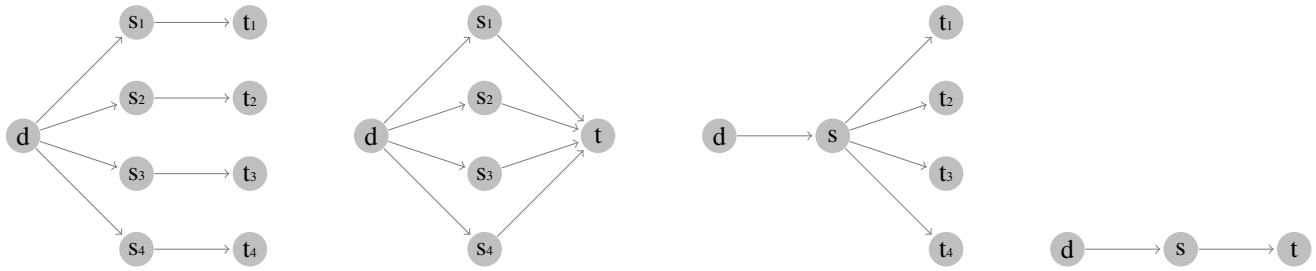
**Input**: A domain name $d$, set of sub-domains $S = \langle s_1, s_2, \ldots, s_n \rangle$, set of TXT records $T = \langle t_1, t_2, \ldots, t_m \rangle$
**Output**: Query and Response pattern mode, $\{Many\_Many, Many\_Single, Single\_Single\}$

1 $D \leftarrow getSnapshoptFrom\_pDNS(w)$
2 **foreach** *Domain* $d$ **do**
3 $G \leftarrow Create\_Relation\_Graph(d, S, T)$
4 $SubDomain\_Degree \leftarrow Common(outD(S))$
5 $TXT\_Degree \leftarrow Common(inD(T))$
6 $Domain\_Degree \leftarrow Common(outD(D))$
7 $SubDomains\_Counter \leftarrow |S|$
8 $TXT\_Counter \leftarrow |T|$
9 $Pattern\_Mode = None$
10 **if** *Property 5* **then**
11 $Pattern\_Mode = Single\_Single$
12 **if** *Property 2* **then**
13 $Pattern\_Mode = Many\_Many$
14 **if** *Property 3* **then**
15 $Pattern\_Mode = Many\_Single$
16 **if** *Property 4* **then**
17 $Pattern\_Mode = Single\_Many$
18 **return** $Pattern\_Mode$

Algorithm 1 shows an overview of query and response pattern recognition in four steps. Step 1 (Line 1) takes a snapshot from the dynamic analysis reports database for a pre-defined window of time. This step produces a set of query and response messages for each domain that appears within the targeted window. Step 2 (Line 3) process every domain name by constructing the relation graph between sub-domains and TXT records. Step 3 (Lines 4-6) calculates the out-degree vector for all sub-domains, in-degree vector for all TXT records, and the out-degree of the domain. From these vectors, we get the commonly used degree, which is considered as a strong representative for the relation between all nodes. Step 4 (Lines 7-9) counts the distinct values of sub-domains and TXT records. Step 5 (Lines 10-17) determines the pattern mode based on the properties of each pattern, which can be applied in arbitrary order (Property 2-5).

## IV. EXPERIMENTAL RESULTS

In this section, we explain the experimental results of our system. We evaluate our system with a malware dynamic analysis reports, which consists of malware DNS traffic. The results show that the traffic patterns are deeply correlated with the characteristics of payload distribution channels. During our experiments, we test our system with malware dataset for one year. Dynamic analysis reports are preprocessed to extract




(a) Many-to-Many (b) Many-to-Single (c) Single-to-Many (d) Single-to-Single

Fig. 2: Examples of the Query and Response Exchange Patterns

DNS traffic, which has `TXT` records. After the preprocessing, these DNS packets are grouped based on domain names. Then, these domains are queued up for query and response pattern analysis. After the analysis, each domain is labeled with the corresponding pattern (Section III-A).

*Datasets:* We observe malware samples provided by a major security vendor over one-year period. We receive the malware feed on a daily basis and then analyze each sample in a controlled environment to generate dynamic behavioral analysis reports. In our analysis, we just consider malware samples that conduct activities using the `TXT` resource record for the DNS protocol. Table I shows some of the statistics about the malware feed recorded between January 2012 and December 2012.

| | | |
|---|---|---|
| Malware dataset | Period | 1-year |
| | No. of samples | 15 Million |
| | No. of samples with `TXT` activities | 18 Thousand |

TABLE I: Dataset statistics.

*Query and Response Patterns:* In the first step of our system, we determine the query and response patterns on the captured DNS traffic. To evaluate the effectiveness of each pattern to carry out payload distribution channels, Figure 3 compares the average distinct DNS messages with the number of malware instances per pattern as well as the number of observed domains. Many-to-Many pattern can be considered as the best candidate for distributing large volume of data while it was probed by small number of malware instances during the year of 2012. This extensive payload retrieval scheme would easily alert the IDS [19]. On the other hand, Single-to-Single pattern allows to carry small volume of data while maintaining a low network footprint. By observing our malware samples, we found most of the malware instances used this pattern to retrieve the attack payloads as on-time update. Because each of these instances send a single query to receive the attack payload, their queries can easily blend in the daily network traffic. Compared to other patterns, Single-to-Single is the best candidate to establish a fully resilient channel in DNS. Single-to-Many pattern requires to update the zone file to distribute different resource data for the same query. This is technically difficult to maintain because of the caching behavior of name servers. As we see in Figure 3, there is no single malware instance using this pattern. Although Many-to-Single pattern has a single response to the different queries like Single-to-Single pattern, it creates a large number of queries, that can be also noticed by IDSs.

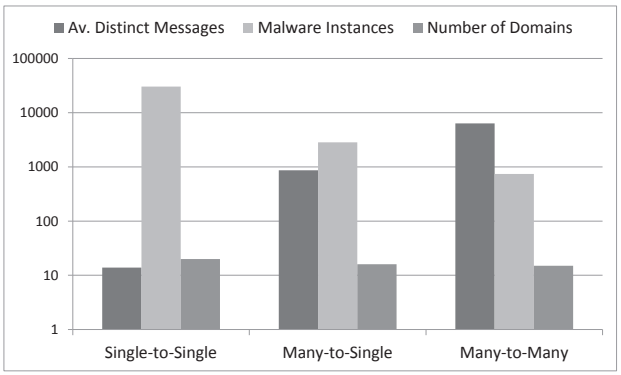



Fig. 3: Average number of query and response messages within one day window.

Even though the number of domains between all of the patterns is very close, there is a distinct variation in the amount of generated traffic as shown in Figure 3. Many-to-Many pattern is generating the most extensive traffic compared with the other patterns. The high volume of data exchange can reveal the name server used as payload provider. In order to hide this name server, botmasters are using it only for the bootstrapping phase to initiate payload distribution channels. Malware samples are heavily using Single-to-Single pattern, which has the least amount of traffic, because it makes them difficult to be detected by traditional defense mechanism.

It is known that malware families are likely to share functionalities by utilizing common modules [1]. We observe the same behavior between different malware families that share the same patterns in our evaluation. Figure 4 shows the query and response pattern distribution across different malware families. The intersection between patterns gives the number of malware families that share the same module. For example, there are nine malware families utilizing Single-to-Single and Many-to-Single pattern. These samples use Single-to-Single transfer pattern to contact with the botmaster and receive a single packet, then use Many-to-Single pattern to check for periodic updates. There are also six malware samples that utilize all these pattern combined to establish a more complex payload distribution channels.

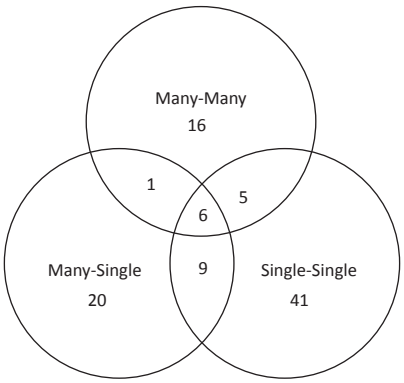



Fig. 4: Query and response pattern distribution for observed malware families.

## V. Discussion

During the analysis in the malware dataset, we observed that some malware families use bootstrapping methods for initiating the payload distribution channels. The initial DNS query is directly made to a rogue name server without traversing the DNS hierarchy. Therefore the malware authors could use any domain, even unresolvable domains. For instance, some malware instances used known domains such as `yahoo.com`. However, this first query is just the initiate the payload distribution channel. In this first query, bot sends some data in sub-domain labels that are used as key derivation parameters [11]. The response for this query comes encoded with Base 64 however not encrypted. In this payload, the bot receives the domain name that will carry out further payload distribution channel. After the bootstrapping process is completed, bots start querying this domain to receive the attack payload in a sequential manner.

Since each malware family can use different domains, it is possible that the our system assigns multiple pattern labels to a single malware family. As explained earlier, some malware families utilize a bootstrapping technique to initiate the payload distribution channel and then use another domain name with different exchange pattern. In this case, our method will give two different labels to the same malware family.

## VI. Conclusions

In this paper, we shed some light on some of the characteristics of malicious payload distribution channels in DNS. We introduced a solution which is able to characterize the payload distribution channels based on the DNS query and response exchange messages. Our system takes a set DNS query and response messages for a specific domain name as an input, and then determines the payload distribution pattern of that particular domain. The evaluation on malware dynamic analysis reports showed that our method can determines payload distribution channel patterns for different malware families.

## References

[1] Spyeye and Zeus malware: Married or living separately? Technical report, Threat post, 2011. http://threatpost.com/spyeye-and-zeus-malware-married-or-living-separately-101411.

[2] F-secure threat report H2. Technical report, F-Secure, 2012. www.f-secure.com/static/doc/labs_global/Research/Threat_Report_H2_2012.pdf.

[3] The role of DNS in botnet command & control. Technical report, OpenDNS, 2012. http://info.opendns.com/rs/opendns/images/OpenDNS_SecurityWhitep-aper-DNSRoleInBotnets.pdf.

[4] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: detecting the rise of DGA-based malware. In *USENIX Security Symposium*, Bellevue, WA, USA, Aug. 2012.

[5] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. On the analysis of the Zeus botnet crimeware toolkit. In *Conference on Privacy Security and Trust*, Ottawa, Ontario, Canada, Aug. 2010.

[6] K. Born and D. Gustafson. Detecting DNS tunnels using character frequency analysis. In *Annual Security Conference*, Las Vegas, NV, USA, Apr. 2010.

[7] S. Bromberger. DNS as a covert channel within protected networks. Technical report, National Electronic Sector Cyber Security Organization, 2011. http://energy.gov/sites/prod/files/oeprod/-DocumentsandMedia/DNS_Exfiltration_2011-01-01_v1.1.pdf.

[8] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX SRUTI Workshop*, volume 39, page 44, 2005.

[9] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In *ACM Computer and Communications Security*, pages 211–222, Alexandria, VA, USA, Oct. 2008.

[10] D. Dagon, N. Provos, C. P. Lee, and W. Lee. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Network and Distributed System Security Symposium*, San Diego, California, USA, Feb. 2008.

[11] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. van Steen, and N. Pohlmann. On botnets that use DNS for command and control. In *European Conference on Computer Network Defense*, pages 9–16, Gothenburg, Germany, Sept. 2011.

[12] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *Usenix Workshop on Large-Scale Exploits and Emergent Threats*, San Francisco, CA, USA, Apr. 2008.

[13] D. Kaminsky. Black Ops of DNS. In *Black Hat Briefings*, Las Vegas, NV, USA, July 2004.

[14] A. Merlo, G. Papaleo, S. Veneziano, and M. Aiello. A comparative performance evaluation of DNS tunneling tools. In *Conference on Computational Intelligence in Security for Information Systems*, pages 84–91, Torremolinos-Málaga, Spain, June 2011.

[15] C. Mullaney. Morto worm sets a (DNS) record. Technical report, Symantec, 2011. http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record.

[16] L. Nussbaum, P. Neyron, and O. Richard. On robust covert channels inside DNS. In *Information Security Conference*, volume 297, pages 51–62. Pafos, Cyprus, May 2009.

[17] C. Qia, X. Chenb, C. Xud, J. Shia, and P. Liub. A bigram based real time DNS tunnel detection approach. 17:852–860, May 2013.

[18] D. Raman, B. D. Sutter, B. Coppens, S. Volckaert, K. De, P. D. Bosschere, and E. V. Buggenhout. DNS tunneling for network penetration. In *Conference on Information Security and Cryptology*, volume 7839, pages 55–77. Seoul, Korea, Nov. 2012.

[19] R. Rasmussen and P. Vixie. Surveying the DNS threat landscape. Technical report, Internet Identity, 2013. http://www.internetidentity.com/white-papers/.

[20] E. Skoudis. The six most dangerous new attack techniques and what's coming next? In *RSA Conference (RSA '12)*, San Francisco, CA, USA, Feb. 2012.

[21] T. van Leijenhorst, D. Lowe, and K. Chin. On the viability and performance of DNS tunneling. In *Conference on Information Technology and Applications*, Cairns, Queensland, Australia, June 2008.

[22] P. Vixie. Extension mechanisms for DNS (EDNS0). RFC 2671, Aug. 1999.

[23] K. Xu, P. Butler, S. Saha, and D. Yao. DNS for massive-scale command and control. *IEEE Transactions on Dependable and Secure Computing*, 10(3):143–153, 2013.