Encrypted and Covert DNS Queries for Botnets: Challenges and Countermeasures

Constantinos Patsakis[a,], Fran Casino[a], Vasilios Katos[b]

[a] *Department of Informatics, University Piraeus, 80 Karaoli & Dimitriou str, 18534 Piraeus, Greece*
[b] *Bournemouth University, Poole House P323, Talbot Campus, Fern Barrow, Poole, Dorset, BH12 5BB, UK*

Abstract

There is a continuous increase in the sophistication that modern malware exercise in order to bypass the deployed security mechanisms. A typical approach to evade the identification and potential take down of a botnet command and control server is domain fluxing through the use of Domain Generation Algorithms (DGAs). These algorithms produce a vast amount of domain names that the infected device tries to communicate with to find the C&C server, yet only a small fragment of them is actually registered. This allows the botmaster to pivot the control and make the work of seizing the botnet control rather difficult.

Current state of the art and practice considers that the DNS queries performed by a compromised device are transparent to the network administrator and therefore can be monitored, analysed, and blocked. In this work, we showcase that the latter is a strong assumption as malware could efficiently hide its DNS queries using covert and/or encrypted channels bypassing the detection mechanisms. To this end, we discuss possible mitigation measures based on traffic analysis to address the new challenges that arise from this approach.

Keywords: Malware, Botnets, Domain Generation Algorithm, DNS, Covert Communication

使用DGA规避安全检测后，再与僵尸主机通信，使得解除控制僵尸网络难度增加。

DNS请求对network Administrator 透明

Email addresses: kpatsak@unipi.gr (Constantinos Patsakis), francasino@unipi.gr (Fran Casino), vkatos@bournemouth.ac.uk (Vasilios Katos)

1. Introduction

The amount and sophistication of modern malware are continuously increasing creating a new industry, cybercrime, whose economy is estimated to have a capitalisation of 1.5 trillion dollars [1]. While this industry has many monetisation sources [2] such as spamming [3], ad injection [4], denial of service and phishing to name a few, one of the most crucial aspects is the management of compromised hosts. The critical nature of the latter lies in the fact that this management should allow the adversary to (a) orchestrate further attacks by, e.g. sending his compromised hosts (bots) new commands, (b) prolong the discovery of the attack, and (c) prevent law enforcement agencies from discovering his true identity.

Clearly, a direct communication channel between infected devices and the Command and Control (C&C) server can be efficiently detected and blocked once one detects that a machine has been compromised, by blacklisting a specific IP or domain. To prevent this, malware authors try to use communication channels that disguise the traffic as benign and cannot be easily blocked, e.g. social networks or frequently change the domain names that host the C&C server. In the latter case, which is the focus of this work, the adversary uses a Domain Generation Algorithm (DGA) that generates millions of pseudo-random domain names and the compromised devices try to connect and retrieve new commands, rendering blacklisting methods useless. However, the adversary only registers a handful of them; therefore, they can regularly pivot both the domain as well as the IP of the C&C server without losing the control of the compromised devices. The basic model is illustrated in Figure 1 as there can be several variations. The botmaster has a deterministic pseudo-random generator (PRNG) to create a set of domain names installed in all compromised devices. As a result, these devices would periodically try to resolve these generated domain names. However, the botmaster has registered only a few of them, therefore, only those resolve to an actual machine. To further perplex possible takedown mechanisms, the botmaster uses fast flux to change the IPs that the registered domains resolve to, which may be some of the compromised devices.

之前的方式是一旦发现被控主机就在黑名单中添加：ip或域名进行封禁。

攻击者通过隧道伪装成良性流量避免被检测。

每次机器DGA产生的域名只有部分被周期性用于实际DNS请求。

所有DGA中, 部分域名被僵尸主机实际注册能够解析，同时这些注册域名在解析中就能指向被控机器ip

僵尸主机通过fast flux 可改变已注册域名对应的ip

1.1. Motivation

By design, the widely used DNS protocol is unencrypted and does not allow for authentication, allowing an adversary to initiate a wide range of attacks. The above has led to the introduction of several protocols which offer

DNS 协议设计本身是明文传输&导致很多攻击易发生。

1、实际注册DNS解析的seed domain：fkxxx.com & vvf9xxx.com
2、seed domain 在被控机根据算法生成未注册域名
3、控制端控制注册域名解析到自己服务器获取信息，完成信息泄密 【fast flux 不断修改域名对应ip，防止被封】

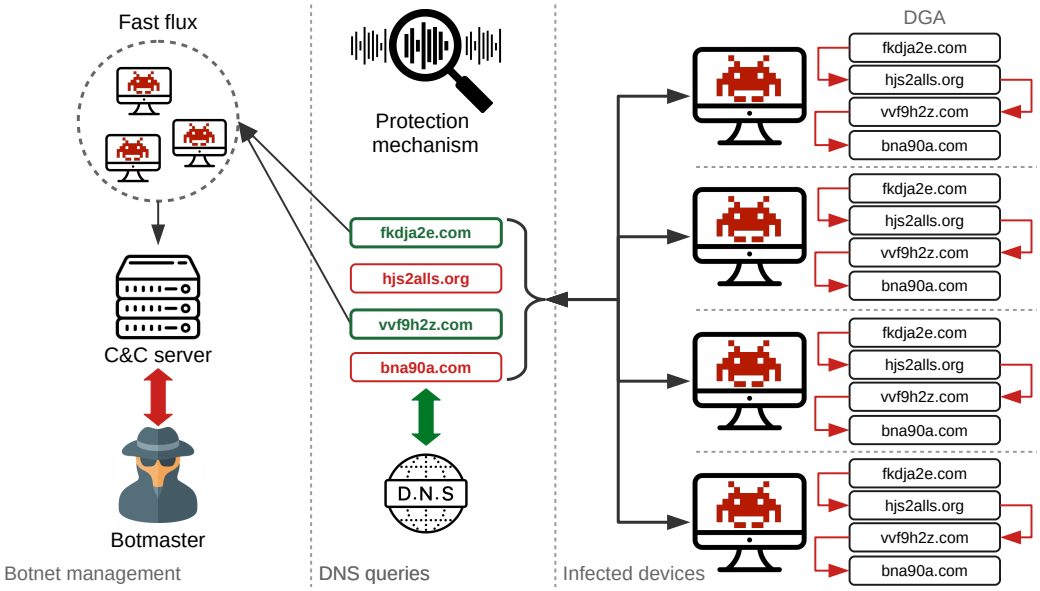


Figure 1: Botnet management with DGAs. Source [5].

confidentiality and authentication of DNS queries and responses including among others DNSCurve, DNSCrypt and the different flavours of DNS over HTTPS/TLS/DTLS. These protocols are actively being deployed by major DNS servers; including Google, Cloudflare, and OpenDNS, while Mozilla is considering making DNS over HTTPS the default option for the DNS queries of the browser[1]. While this may allow for increased privacy of individuals, one has to acknowledge that the fact that DNS queries are unencrypted facilitates both the detection and identification of botnets from the queries that bots make to the C&C server. Therefore, we need to study what happens if a malware decides to use the privacy-preserving DNS services to resolve its C&C server and whether such actions can be detected, as, by definition, current methods would be rendered useless.

1.2. Main contributions

The contribution of this work is twofold. First, we illustrate a critical gap in current state-of-the-art and practice methods for detecting botnets

[1]https://blog.mozilla.org/security/2019/04/09/dns-over-https-policy-requirements-for-resolvers/

当前通过DGA检测botnet的方法缺陷：假设被控机需要使用生成域名连接C&C server，因此在此过程可以阻断DNS请求，阻断之后不断尝试请求且解析失败的机器即为被控机。【默认攻击者dns请求信道不会加密】

that use DGAs. More precisely, the core assumption that researchers and practitioners set to date is that they consider that an infected machine would try to connect to the C&C server using the generated domain names from the DGA and they would be able to intercept these DNS requests. In this regard, a high rate of failed DNS queries from a specific machine implies the presence of an infected machine. Moreover, studying these requests from the perspective of a network administrator one could try to determine whether the request originates from a DGA by, e.g. evaluating the entropy of the domain name. While the hypothesis seems rather straightforward, it is rather strong as they assume that the adversary would use a plaintext channel to resolve the domain name.

In light of recent trends in malware and the continuous use of encryption [6], this work investigates the challenges that arise when the adversary uses encrypted channels to perform DNS queries. While this has not been discussed in the literature nor reported by CERTs, we consider necessary to proactively discuss and study the feasibility of such an approach. Therefore, we study the possibility of botnets using protocols like DNSCurve and DNSCrypt, and mechanisms such as DNS over HTTPS/TLS to communicate with their C&C server using connections with whitelisted domains over standard whitelisted protocols. Moreover, we investigate possible detection mechanisms of such queries and experimentally show that traffic analysis on the exchanged packets can lead to very efficient detection, in terms of both computational overhead and accuracy, for specific DGAs. Therefore, even though DNS queries can be performed over covert channels indicators of compromise (IoCs) can be composed. In fact, we illustrate that using the Hodrick-Prescott filter [7] one can accurately classify them using a small amount of samples.

1.3. Organisation of this work

The rest of this work is structured as follows. In Section 2 we present the related work regarding malware and DGAs. Then, in Section 3 we discuss the usage of covert DNS queries from botnets using DGAs and the challenges that arise for current state of the art mechanisms. In Section 4 we present our experimental setup and the datasets we utilise. Afterwards, in Section 5 we discuss the findings of our extensive experiments using covert DNS queries and the emerging patterns that emerge and showcasing that traffic analysis can efficiently reveal such actions for specific DGAs. Finally, the

article concludes discussing possible countermeasures and summarising our contributions.

2. Related work

攻击者不再将C&C的域名orIP放入malware中，防止识别后失去被控机控制权

Nowadays, a common practice in malware-based campaigns is to use remote servers (i.e. C&C servers), which send instructions/orders to infected devices [8, 9] to perform a DDoS attack for example [10]. The typical mechanism used in the past was to hardcode the IP addresses or possible domains of the C&C servers in the malware, however, from the attacker's perspective, this entailed a set of drawbacks such as losing control of the botnet once the IP or domain of the C&C server was identified. Therefore, several methods to partially or fully decentralise the management of infected devices have been implemented. With peer-to-peer botnets [11], there is no centralised C&C server, but infected hosts act as both client and servers and efficiently pass commands to each infected device. The *Fast Flux* approach imitates Content Distribution Networks (CDNs) by resolving a domain name to multiple IP addresses. This can be achieved by using low TTL which forces DNS to refresh their cache associated with these domains repeatedly. To this end, a subset of the infected nodes may be used as what is called a *flux agent*. A flux agent is a device whose IP is temporarily registered as the host of a domain that the bots are querying, yet they are forwarding the traffic to the C&C server. For more about fast flux networks, the interested reader may refer to [12] and [13].

自控DNS解析服务器通过降低TTL，强制被控服务器定期更新解析内容方便fast flux 控制

fast flux networks

To provide another layer of protection, malware use what we call *Domain Name Generator Algorithms (DGAs)*. To this end, a malware uses a PRNG to create a set of domain names and this approach has become the default methodology [14, 15]. Hence, infected devices check the list of generated domains until they find the C&C server, whose location may also change dynamically. In this regard, blacklisting domains is rendered useless as it implies many practical issues.

DGA域名ip动态变化的情况下，黑名单越来越大，添加找到之前甚至C&C server 域名ip已修改。

In general, DGA detection methods use simple domain name classification according to some features such as entropy, length or lexical characteristics to determine whether a DGA has generated a domain name or not, such as in [16] and [17]. Nevertheless, several classification approaches use additional information such as WHOIS or DNS traffic analysis to detect abnormal behaviours like a high volume of NXDomain responses, which may indicate that a device has been infected [18, 19, 20]. Another technique was proposed

C1

提取DGA特征判断其是否根据其生成恶意域名
有一个DGA方法就要有一个判断？集成学习？
特征1：[16-17]entropy、length、lexical characteristics
特征2：[18-20]Whois、DNS traffic analysis to detect abnormal behavior[eg:high volume of NXdomain respon]

by Yadav et al. in [21], which groups DNS queries originated by a specific client to compute the correlation between distinct requests pointing to the same domains. In a more recent work, Manadhata et al. [22] model the detection problem as a graph inference problem and construct a host-domain graph from proxy logs to classify domains into benign and malicious with a certain probability. Gong et al. [23], perform domain name classification using a clustering algorithm with time-based information as well as IP-domain auxiliary data. Other machine learning approaches can be found in [24] and [19], in which authors use a set of features extracted from network traffic and other well-known domain sources such as Alexa[2] to enhance DGA detection.

domain shadowing: valid domain has be previously hacked

More recently, since domain names generated by DGAs exhibited a high level of randomness facilitating thus their detection, attackers adopted the use of English wordlists, which made it more difficult to discern between valid and DGA-generated domains. Similarly, the concept of *domain shadowing* is also gaining attention, which relies on the use of valid domains that were previously hacked [25]. Other DGAs generate domain names which have high chances of collision with valid domains and/or with other DGA malware [26] so that finding and analysing them becomes more challenging.

DGA检测1：基于LSTM +raw domain的binary classification

There are some techniques to overcome such new generation of DGA algorithms such as the work proposed in [27], where authors use a short-term memory network (LSTM) to perform binary domain classification using raw domain names as features. Similar classification approaches based in neural networks can be found in [28, 29]. In [30], authors use a generative adversarial network (GAN) to implement a deep learning based DGA which is capable of bypass classical deep learning detectors. Subsequently, such information is used as input feedback to the DGA detectors to enhance their accuracy and robustness.

DGA检测2：通过GAN生成的域名能够骗过常规检测。

In the case of [31], the authors are able to characterize and classify similar DGA-generated domains, generating knowledge about the evolving behaviour of botnets. More recently, Curtin et al. developed the smashword score [32], a new metric that uses n-gram overlapping combined with information provided from WHOIS lookups to determine whether a DGA has generated a domain name or not. For a detailed overview and classification of methods of how malicious domains can be detected, the interested reader may refer to [33]. Moreover, a summary of the main literature families is provided in Table 1. Note that none of the existing methods can

DGA检测3：smashword score -n gram overlapping + WHOIS lookup info

[2]https://www.alexa.com/topsites

analyze encrypted communications. Although some approaches do not use side information (i.e. since they only analyze domain name lexical features) as in [28, 27, 29], the fact that DNS query responses are covert hinders the detection since we cannot check whether it was a true or false positive in real scenarios, making the discovery of novel DGA families even more difficult.

Table 1: Summary of literature families and their main characteristics. NN stands for Neural Networks and RF for random-forest classifiers.

DGA Family	Description	Detection features	Main detection methods	Covert DNS detection
Arithmetic-based	A PRNG generates alphanumerical combinations	Randomness, lexical structure	entropy, lexicographic analysis, DNS response, WHOIS	No
Hash-based	Create an hex representation of a hash	Randomness, lexical structure	entropy, lexicographic analysis, DNS response, WHOIS	No
Wordlist-based	Combination of words extracted from a dictionary	Lexical structure and n-gram analysis	Classifiers (NN, RF)	No
Permutation-based	Permutations over valid or word-based domains	Lexical structure, n-gram analysis, DNS response, pattern analysis	Classifiers (NN, RF)	No

域名分析：基于词汇&语句方法

With *DNS tunnelling*, the malware tries to exploit the features of DNS protocol since DNS traffic is allowed unrestricted by most firewall installation. To this end, data are encoded as DNS queries and responses to avoid detection and bypass the installed security measures. Two well-known examples are Feederbot [34] and Morto[3] which used the TXT resource record to deliver their encrypted commands to the infected hosts. Nevertheless, DNS is also exploited by botnets to amplify the bandwidth of their attacks [35, 36].

隧道提取DNS协议特征，将需要传输的信息加密成DNS请求和响应的办法逃避检测。

It is apparent that in the literature researchers always assume that the DNS query information is plaintext or that they can collect it and analyse it. We consider that the latter is a strong assumption as DNS traffic can be encrypted or tunnelled to hide its existence. However, as recently illustrated [5], DGAs can be extended to Resource Identifier Generation Algorithms

[3] https://www.symantec.com/connect/blogs/morto-worm-sets-dns-record

(RIGA) which allows the use of other protocols beyond DNS and allow the orchestration to exploit other more decentralised protocols.

3. Covert DNS queries

In what follows we introduce our threat model and detail how an adversary would try to armour a malware.

3.1. Threat model

模型强假设：已有被控机器，控制主机不直接与被控主机传输消息。
1）将所有DNS请求加密
2）通过其他协议传输
3）绕过防火墙检测
基于上述步骤，C&C 能够与被控机通过其他协议tunnel(eg: DOH) 传输加密在DNS请求记录中的信息。

In our model, we assume that an adversary has managed to compromise a device in the internal network of an organisation. The adversary wants to issue commands to the compromised host (bot) without using a direct communication channel with him; therefore, she opts for the use of a DGA. Furthermore, the adversary wants to hide the existence of this mechanism by encrypting all the DNS queries and tunnel them through another protocol that is allowed from the firewall policy of the host network. To this end, we assume that the adversary has full control of the compromised host and of some other hosts outside the host network that she wants to contact. Therefore, we assume that the adversary cannot drop packets of other hosts nor change the network policies of the host network. Moreover, we assume that any unencrypted traffic will be collected by the host network and the use of any unauthorised protocol or connection with unauthorised host; both internal and external would be blocked.

3.2. Armouring a malware

As discussed, many malware use DGAs in their attempt to hide the actual C&C server. However, their continuous DNS queries may trigger alerts to installed security mechanisms as they will detect a high amount of failed DNS queries to usually random looking domains from specific hosts. This is due to the very nature of the DNS protocol which does not provide data security. Apparently, the lack of these mechanisms exposes users to many threats such as eavesdropping and a set of man-in-the-middle attacks (e.g. DNS data manipulation). Nevertheless, in this scenario, it allows network administrators to monitor the domains that their hosts are trying to connect to.

While there are several solutions to counter such threats, with the most dominant one being DNSSEC [37], we are only interested in protocols that offer confidentiality. The reason behind this choice is that in order to armour

the malware to bypass many network security mechanisms we aim to encrypt the DNS queries and/or tunnel them via other whitelisted protocols. The concept is that if the DNS queries and their responses are encrypted, then no one else could determine which are the requested domains and whether they exist. In this regard, we limit our scope to specific protocols and schemes which are deployed by legitimate DNS servers and offer confidentiality when performing DNS queries. The list is rather limited and consists of the following options:

1. **DNSCurve:** This protocol was proposed by Bernstein [38] and addresses the confidentiality, integrity and availability of DNS. The protocol uses strong yet very fast encryption, however, despite its efficacy it has not been widely deployed, with the most widely used DNS server supporting it being OpenDNS[4].
2. **DNSCrypt:** Denis and Fu introduced this protocol to authenticate communication between a DNS client and a DNS resolver [39]. Currently, it is the most widely used encrypted DNS protocol as many well-known DNS servers provide it, including among others OpenDNS and Yandex.
3. **DNS over HTTPS:** This protocol offers DNS resolution over an encrypted HTTPS connection to provide end-to-end authenticated DNS lookups. Well-known DNS providers such as Google and CleanBrowsing are already compliant with this protocol.
4. **DNS over TLS:** As the name suggests, this protocol runs DNS transactions over TLS (see IETF RFCs 7858[5] and 8310[6]). Therefore, DNS queries sent to the resolver are performed in an encrypted channel, enhancing security and privacy. This solution is implemented by some DNS providers such as Cloudflare and CleanBrowsing.
5. **DNS over Datagram Transport Layer Security (DTLS):** This is an experimental protocol to offer confidentiality for DNS queries [7] using UDP to improve performance and addresses packet loss and reordering that may happen during packet delivery which are traditional problems of DTLS.

[4] `https://umbrella.cisco.com/blog/2010/02/23/opendns-dnscurve/`
[5] https://tools.ietf.org/html/rfc7858
[6] https://tools.ietf.org/html/rfc8310
[7] `http://www.rfc-editor.org/info/rfc8094`

Evidently, the first two options imply some constraints in their adoption from malware as they imply the usage of "exotic" protocols that in, e.g. corporate/monitored environments would be blocked from the network firewall. It should be noticed that similar to HTTPS, DNSCrypt also operates in port 443. While the two protocols differ, if the firewall does not correctly identify the protocol, DNSCrypt could tunnel DNS queries and responses. However, in the case of the latter two methods, the DNS queries and responses are masqueraded under the veil of the widely used HTTPS/TLS protocols with legitimate domains. Therefore, the DNS queries can be performed in an entirely covert way as the traffic does not trigger any alert in the security mechanisms, but it is tunnelled as normal HTTPS/TLS traffic. DNS over DTLS transfers datagrams over TLS, using UDP and port 853. The protocol is quite efficient, however, it is in experimental stage and not actively used by any major provider to be evaluated.

While the use of DNSCurve and DNSCrypt is subject to firewall policy constraints, in all the cases above the host manages to efficiently resolve the IP of a domain name using strong encryption primitives. Practically, the network administrator will have to analyse packets with encrypted traffic. The interested reader may refer to [40] for a more detailed comparison of DNSSEC and DNSCurve.

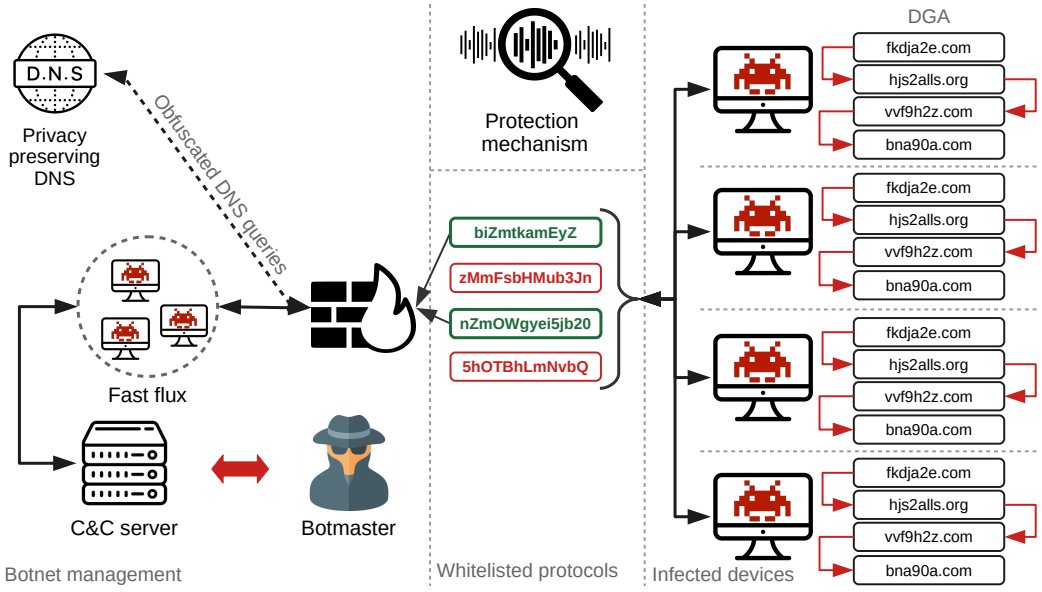


Figure 2: The armored malware.

1）已注册的域名fkdxxxxx.com & vvxxxx.com。
2） 通过DGA生成大量随机域名。
3）通过白名单协议如https的tunnel传输加密后的DNS请求[白名单加密后的请求不会被检测机制检出]。
4）过防火墙后到控制端DNS解析服务器解析-->获得传出数据。
5）返回的DNS相应中包含被控命令。

4. Experimental setup

实验目标：提出指标判断1）botnet生成的域名是否通过白名单协议隧道与C&C 服务器交互信息。

The experimental evaluation aims to explore whether is it possible to construct good quality Indicators of Compromise to determine whether encrypted network traffic corresponds to the one generated by a botnet that using a covert channel for requesting the possible domain name of its C&C server, as analysed in the previous section. The quality of an IoC in this paper adopts Bianco's pyramid of pain approach [41]. That is, we aim for creating higher IoCs on the pain scale, such as Network Artefacts or Tools, as the lower IoCs cannot be considered reliable due to the encryption layer. To this end, we assume that the network administrator would perform traffic analysis on the packets on the intercepted HTTPS/TLS traffic. In our experiments, we do not consider DNSCrypt and DNSCurve as the protocols can be easily detected and blocked by a firewall. The reader should note that by following this approach, the adversary manages to have the security and privacy guarantees of standard TLS for her queries. Therefore, it is practically impossible to perform any attack unless there are implementation issues, at the trade-off of linear complexity to the length of exchanged information for using standard TLS communication.

DNSCrypt 和 DNSCurve 的隧道会被防火墙轻易检测

To investigate the capabilities and approaches of traffic analysis in the present problem domain, synthetic datasets were constructed. These datasets consist of covert DNS queries to registered and non-registered domains. For the former, we have used the Alexa top 1000 domains, while for the latter we use non registered domain names generated from ten different DGAs. More precisely, we have used the DGAs presented in Table 2 which have been published by Abakumov [8]. In columns Min, Max, Average and Stdev we refer to the corresponding measures for the domain length of each dataset. Unique values indicate whether all possible lengths of domain names are represented in each dataset. For instance, in Conflicker all the possible values (16-8+1=9) are represented in the dataset, however, this is not the case for, e.g. Alexa as we have 22 unique values in the dataset and there could be 25 unique values, showing that domains of 3 specific lengths do not exist in the dataset.

构造的流量数据：
1、已注册域名：alexa前1000；
2、未注册域名：从10个DGAs中生成。

From each DGA we have kept the first 1000 domains. This procedure creates a rich dataset of 11.000 domains of existing and non-existing domains. Finally, we collect the generated traffic and try to correlate the results to

[8] https://github.com/andrewaeva/DGA

determine whether a device performs covert DNS queries.

Dataset	Min	Max	Average	Stdev	Unique values
Alexa	4	28	11.349	3.237	22
Conflicker	8	16	11.755	1.983	9
CryptoLocker	15	21	17.783	1.424	7
GOZ	20	35	28.241	2.431	16
Matsnu	28	40	30.527	2.038	13
new GOZ	26	32	29.885	1.087	7
Pushdo	11	11	11.000	0.000	1
Ramdo	20	20	20.000	0.000	1
Rovnix	24	38	26.794	2.622	15
Tinba	16	16	16.000	0.000	1
Zeus	26	32	29.878	1.038	7

Table 2: Datasets used in the experiments and their statistical properties in terms of domain length characteristics.

Our methodology for synthesizing our datasets is rather straightforward. For each of the datasets mentioned above, we perform DNS queries over HTTPS and TLS and intercept the generated traffic using the well-known `tcpdump` tool. To facilitate the process and to allow for further validation of the results, we use `pydig`[9] to perform all the covert DNS queries. Then, we parse the generated `pcap` file and perform feature extraction on each captured packet. Using `tshark`; the command line version of `Wireshark`, and Linux command line tools, we filter the traffic to extract the necessary packages from the `pcap` files. The features that we extract from each packet are the source IP, the target IP, the size of each packet, the protocol and the info that `tshark` produces. The latter are some metadata for each package that facilitate the filtering process. Based on these features we investigate whether they can be used independently and in combination to determine whether a machine performs covert DNS queries. To allow the testing and validation of our claims and methodology we have uploaded all the needed scripts and sample datasets in GitHub[10] in the form of `pcap` files.

[9] `https://github.com/shuque/pydig`

[10] `https://github.com/kpatsakis/covert_dns_queries`

TLS：transport layer security

1、pydig: 模拟DOH请求(隐藏后的DNS请求)。
2、 tcpdump：产生模拟流量的pcap文件，包含所有的流量包。
3、tshark：过滤所有pcap文件中DOH相关的packet.
4、特征提取：parse pcap file and perform feature extraction on each packet.
5、特征：sourceIP、targetIP、size of each packet、protocol、tshark info

The aforementioned steps create 22 datasets: 11 for DNS over TLS and 11 for DNS over HTTPS. We argue that the packages of the host should not be considered as their content may greatly vary depending on the client-side implementation. Therefore we limit our results to the packages received from the DNS server over HTTPS/TLS. From these packages, we omit the packages which are the initiation or the termination of TLS as they do not contain actual information for the DNS query but other information such as parameter and cipher negotiation. Therefore, the rest of the packages must contain the actual response for the query to the DNS server. As such, the latter coincides with the experimental results.

The basic concept behind this choice of features is that the response of the DNS server in terms of length for NXdomains will differ significantly from the response for existing domains. In the latter case, the response should contain information like the IP address etc. making it longer in principle. The specifications of DNS in RFC 1035 [42] for responses to DNS queries justify this intuitive result. To facilitate the reader, we provide the output of some tools, for instance, see Figure 3 and 4 which illustrate the output of two DNS queries using `dig`. Clearly, the response of an existing domain contains "different" information in terms of both quality and quantity. Therefore, the research question is whether this differentiation can be observed in the intercepted encrypted traffic.

5. Discussion

In the following paragraphs we discuss our findings of our experimental results. To this end, we first discuss the experimental results using traffic analysis and then we present how one can efficiently perform bot attribution.

5.1. Traffic analysis

On a macroscopic level, examining the size of the responses in each case we notice that some interesting patterns emerge, see Figures 7a-7k and 8a-8k. It should be noted that due to its implementation when using DNS over TLS, each response consists of two packets with the first one having a length of 97 bytes. Evidently, DGAs which produce domains with static length, end up having server responses with static length as well, see Figures 7f, 8f, 7g,8g, 7i and 8i. Similarly, DGAs which produce domains with little variance on their length, see 7b and 8b, have responses with little variance in their length. Clearly, the Alexa dataset presents the most significant variance

```
; <<>> DiG 9.11.3-1ubuntu1.2-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18417
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.google.com. IN A

;; ANSWER SECTION:
www.google.com. 238 IN A 216.58.205.164

;; Query time: 44 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu Oct 25 01:08:12 EEST 2018
;; MSG SIZE  rcvd: 59
```

Figure 3: The DNS response when querying for Google.com.

```
; <<>> DiG 9.11.3-1ubuntu1.2-Ubuntu <<>> rejuxlip.ru
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 64200
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;rejuxlip.ru. IN A

;; Query time: 2475 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu Oct 25 01:10:58 EEST 2018
;; MSG SIZE  rcvd: 40
```

Figure 4: The DNS response when querying for rejuxlip.ru.

in the response lengths as it has the most extensive diversity in terms of both lengths of domains and the response is expected to contain several values for the IPs of the hosts. In Table 3 we provide an overview of the statistical properties of the responses in each case. More precisely, columns Min, Max, Average and Stdev refer to the corresponding measures for the packet size of each dataset. Unique values indicate whether all possible sizes of packets are represented in each dataset. For instance, in Pushdo for DNS over HTTPS, while the range is from 564 to 570, only two values are found in our experiments.

It is clear that the results mentioned above indicate that even though the DNS queries are performed covertly via legitimate DNS servers, traffic analysis can efficiently determine the queries on existing and NXDomains with almost 100% accuracy if the DGA generates domains of static length. In this regard, one could develop a lightweight approach to constructing IoCs. The corresponding rule would look for packages to privacy-preserving DNS servers like Google, Cloudflare etc. over HTTPS and then look for the 97 vs X pattern or the stable value pattern on the received packages to determine whether a device has been compromised and tries to perform DNS queries over a covert channel.

However, a thorough and complete detection requires providing attribution information which in this case it would be threat intelligence on the type of infection and bot attempting to communicate with its C&C or botmaster. Detecting NXDomain responses is expected to be performed at an early stage of the intrusion detection process to allow the security controls to block the connection or respond in the prescribed security policy manner. The incident response process would be considered complete if it would be capable of providing evidence of the bot that has successfully compromised the system. As such, the aim of the incident response exercise is twofold: a) identify quickly that a bot has infected the system; b) identify the particular bot or family.

Currently, in an unencrypted, standard DNS setting, the computed IoC describing a bot infection would be a large number of NXDomain responses within a particular time frame. We have described above how to replicate this IoC under TLS and HTTPS communication. In the following section, we present an approach for constructing higher quality IoCs capable of identifying the underlying bot.

	Dataset	Min	Max	Average	Stdev	Unique values
HTTPS	alexa	476	704	541.610	27.646	109
	Conflicker	470	599	565.162	20.612	50
	CryptoLocker	568	592	580.045	6.586	25
	GOZ	574	594	585.218	3.710	21
	Matsnu	595	611	601.425	3.617	17
	new GOZ	580	605	594.347	6.797	26
	Pushdo	564	570	567.806	2.890	2
	Ramdo	574	580	577.708	2.915	2
	Rovnix	591	610	596.900	3.979	20
	Tinba	521	589	585.445	6.410	6
	Zeus	580	605	591.737	6.414	26
TLS	alexa	134	619	220.951	80.746	235
	Conflicker	121	354	214.670	58.580	37
	CryptoLocker	189	205	196.532	5.484	14
	GOZ	195	209	202.217	2.484	15
	Matsnu	214	224	216.566	2.046	11
	new GOZ	201	218	211.854	5.281	14
	Pushdo	185	185	185.000	0.000	1
	Ramdo	196	196	196.000	0.000	1
	Rovnix	210	223	212.759	2.684	14
	Tinba	145	202	201.483	5.405	2
	Zeus	202	218	211.520	5.245	13

Table 3: Statistical properties. Note that for TLS we omit the packages with length 97 bytes as this appears in all interactions.

5.2. Bot attribution

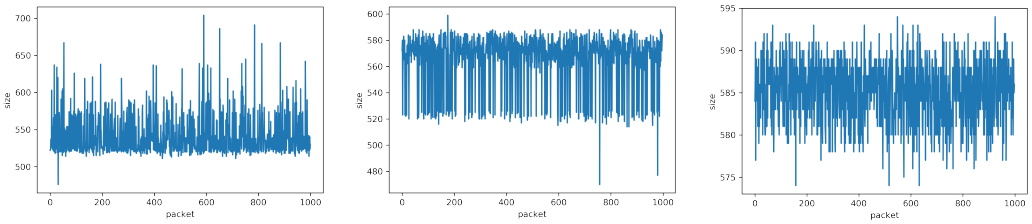
The approach for constructing bot IoCs is based on the following assumptions:

- the bot uses a fixed DGA strategy to generate the domains,
- the domains are generated sequentially,
- the process is repeated until an existing domain is received.

These assumptions allow us to consider the network traffic generated by the bot as a time series variable. This is also the first significant distinction

between the Alexa dataset and all other bots we considered in our experiments. In what follows, we present only the results for the case of DNS over HTTPS traffic, yet similar results apply for DNS over TLS.

Figure 5 shows a sample of the datasets, more precisely Alexa, Conficker and Goz, plotted over time. More precisely, the figure illustrates the packet size (axis Y) for each dataset, if we consider that one performed a covert DNS query for each of the domains of the corresponding dataset sequentially (axis X). It can be observed that the variables display a significant amount of noise.



(a) Alexa (b) Conficker (c) Goz

Figure 5: Time plots of Alexa, Conficker and Goz

In the proposed approach we attempt to apply filtering in order to remove the noise and identify any trend. This approach is commonplace in the economic analysis of time series, and we explore its applicability to the current problem domain. More specifically, in the context of time series analysis the objective is to identify a trend so that this can be used to make predictions of the future values of a given variable. This can be attempted if one would establish that there is correlation between the immediate past observations (lags) for any given point in the series. If this is the case, this knowledge can be extracted from the series in a form of a trend, thus breaking down the series (or *signal*) to a linear combination of a *trend* and a *cycle*, with the latter capturing any periodicity. Once these trends are identified, we attempt to see whether they are distinct for each malware and different from benign traffic. Moreover the examined datasets can be considered to be time series (although the x axis does not explicitly capture time) as the DNS packets generated from a DGA follow a sequential order over time.

The approach of time series analysis is as follows. Assuming that the variable is composed of a cycle (which is true as we have DNS requests and responses) and a trend, we consider the Hodrick-Prescott (HP) filter [7] to

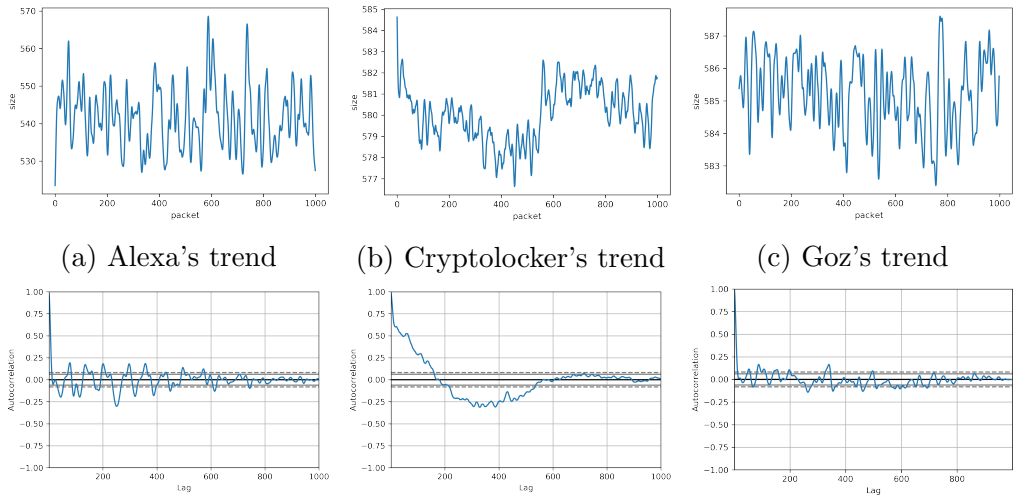
be a good candidate. The Hodrick-Prescott filter separates a time-series y_t into a trend τ_t, a cyclical component ζ_t and an error component ϵ_t such that:

$$y_t = \tau_t + \zeta_t + \epsilon_t$$

The components are determined by minimising the following quadratic loss function:

$$\min_{\tau_t}(\sum_{t=1}^{T}(y_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1}[(\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1})]^2)$$

with the multiplier λ specifying the penalty offered by the second term. Figure 6 shows the trends of the different variables representing the bots following the application of the HP filter. In this instance, we picked Alexa, Cryptolocker and Goz. Below each plot, there is also an autocorrelation plot showing the autocorrelation of the respective variable. Unsurprisingly Alexa's autocorrelation (Figure 6d) drops quickly as this is not a time series and each observation (domain) is independent of the previous ones. Cryptolocker is quite distinct from Alexa, but Goz shows some similarity.



(a) Alexa's trend (b) Cryptolocker's trend (c) Goz's trend

(d) Alexa's trend autocorrelation (e) Cryptolocker's trend autocorrelation (f) Goz's trend autocorrelation

Figure 6: Trends and autocorrelations

Autocorrelation is an indication of the number of previous observations (lags) affecting the current observation. It was established that for all bots a

minimum of 4 lags is sufficient to reconstruct the whole series. By running an autoregressive moving average (ARMA) estimation over all series, we obtain the coefficients summarised in Table 4.

	constant	**lag(-1)**	**lag(-2)**	**lag(-3)**	**lag(-4)**
Alexa	3.418966 {0.469880} [0.000]	3.088595 {0.029306} [0.000]	-3.624877 {0.082269} [0.000]	1.913501 {0.082175} [0.000]	-0.383532 {0.029205} [0.000]
Conficker	2.785166 {4.716197e-01} [0.000]	3.107565 {0.029506} [0.000]	-3.657214 {0.083259} [0.000]	1.920612 {8.340822e-02} [0.000]	-0.3758890 {2.967494e-02} [0.000]
Cryptolocker	2.843495 {5.729647e-01} [0.000]	2.880937 {0.028428} [0.000]	-3.301188 {0.075057} [0.000]	1.851547 {7.451386e-02} [0.000]	-4.361981e-01 {2.778673e-02} [0.000]
Goz	3.497740 {5.019608e-01} [0.000]	3.104442 {0.029296} [0.000]	-3.665087 {0.082379} [0.000]	1.944414 {8.240207e-02} [0.000]	-3.897455e-01 {2.932082e-02} [0.000]
Matsnu	3.111954e+00 {4.716860e-01} [0.000]	3.152710 {0.028967} [0.000]	-3.77899 {0.08182} [0.000]	2.032400e+00 {8.174793e-02} [0.000]	-4.112938e-01 {2.888826e-02} [0.000]
NewGoz	2.243034 {0.522166} [0.000]	3.168183 {0.029030} [0.000]	-3.822154 {0.082280} [0.000]	2.070548 {8.274406e-02} [0.000]	-4.203520e-01 {2.957008e-02} [0.000]
Pushdo	3.645356e+00 {5.041236e-01} [0.000]	3.110475 {0.029258} [0.000]	-3.685589 {0.082007} [0.000]	1.962390 {8.174939e-02} [0.000]	-3.936961e-01 {2.896300e-02} [0.000]
Ramdo	3.359594 {4.817741e-01} [0.000]	3.164252 { 0.028482} [0.000]	-3.842610 {0.080164} [0.000]	2.126389 {8.033351e-02} [0.000]	-4.538455e-01 {2.865658e-02} [0.000]
Rovnix	4.380914 {5.473323e-01} [0.000]	3.092777 { 0.029101} [0.000]	-3.652384 {0.081594} [0.000]	1.954086 {8.163828e-02} [0.000]	-4.018180e-01 {2.914092e-02} [0.000]
Tinba	0.878688 {0.304202} [0.000]	3.380042 {0.025792} [0.000]	-4.351385 {0.073868} [0.000]	2.533331 {7.300452e-02} [0.000]	-5.634898e-01 {2.479662e-02 [0.000]
Zeus	6.110055e+00 {7.452222e-01} [0.000]	2.773122 {0.029171} [0.000]	-3.025082 {0.076807} [0.000]	1.637556 {7.675001e-02} [0.000]	-3.959203e-01 {2.908530e-02} [0.000]

Table 4: ARMA estimation for all bots - Std. Error in curly brackets, Probability in square brackets

To utilise ARMA coefficients as IoCs, we need to investigate whether each tuple can uniquely identify the bot. This is examined by testing whether statistical differences between the different IoCs (coefficients) exist. In addition,

we need to test whether subsets of each series can produce the same IoCs as well as the minimum number of (consecutive) observations needed to produce them.

The investigation of the existence of statistical differences was performed as follows. A single variable was produced by stacking all variables and adding a second variable to label the origin of the data and observations. That is, the first 1000 observations were from Alexa (with the label ID=0) the following observations were from Conficker (ID=1), then Cryptolocker (ID=2) and so forth, in accordance with the order displayed in Table 4. We then ran an analysis of variance (ANOVA) with Duncan's test to perform the clustering. The results are shown in Table 5. From the tests, 10 distinct groups emerged, where all bots were successfully separated except Goz and Tinba which were grouped together.

	Sum of squares	**df**	**Mean square**	**F**	**Sig.**
Between Groups	2986937.080	10	298693.708	2310.923	0.000
Within Groups	1419585.793	10983	129.253		
Total	4406522.873	10993			

Table 5: ANOVA and clustering results

Finally, by performing t-tests, we measured that in most cases it requires between 20 to 30 observations to construct the IoCs (ARMA coefficients). From a practical perspective, a network administrator by observing the encrypted network traffic and capturing the lengths of requests and responses, would be able to detect the bot infection, identify the bot; if this is already in the IoC database. Moreover, she may escalate by taking action before the bot completes the communication with the botmaster - that is, if the communication needs more than 30 DNS lookups.

Similar results were obtained with the analysis of the TLS traffic. It is worth mentioning that the distinction between Alexa and non-Alexa (i.e. bot) traffic was stronger, but the distinction between the bots is less apparent. In other words, it is possible to identify the anomalous activity and flag the botnet infection, but attributing to the particular bot would be less effective.

Evidently, in both cases, we follow the same steps; therefore, it is needed to investigate the complexity that the proposed traffic analysis implied with regard to the length of exchanged information. While most steps are linear, e.g. collecting the data, applying the Hodrick-Prescott filter etc., the ANOVA

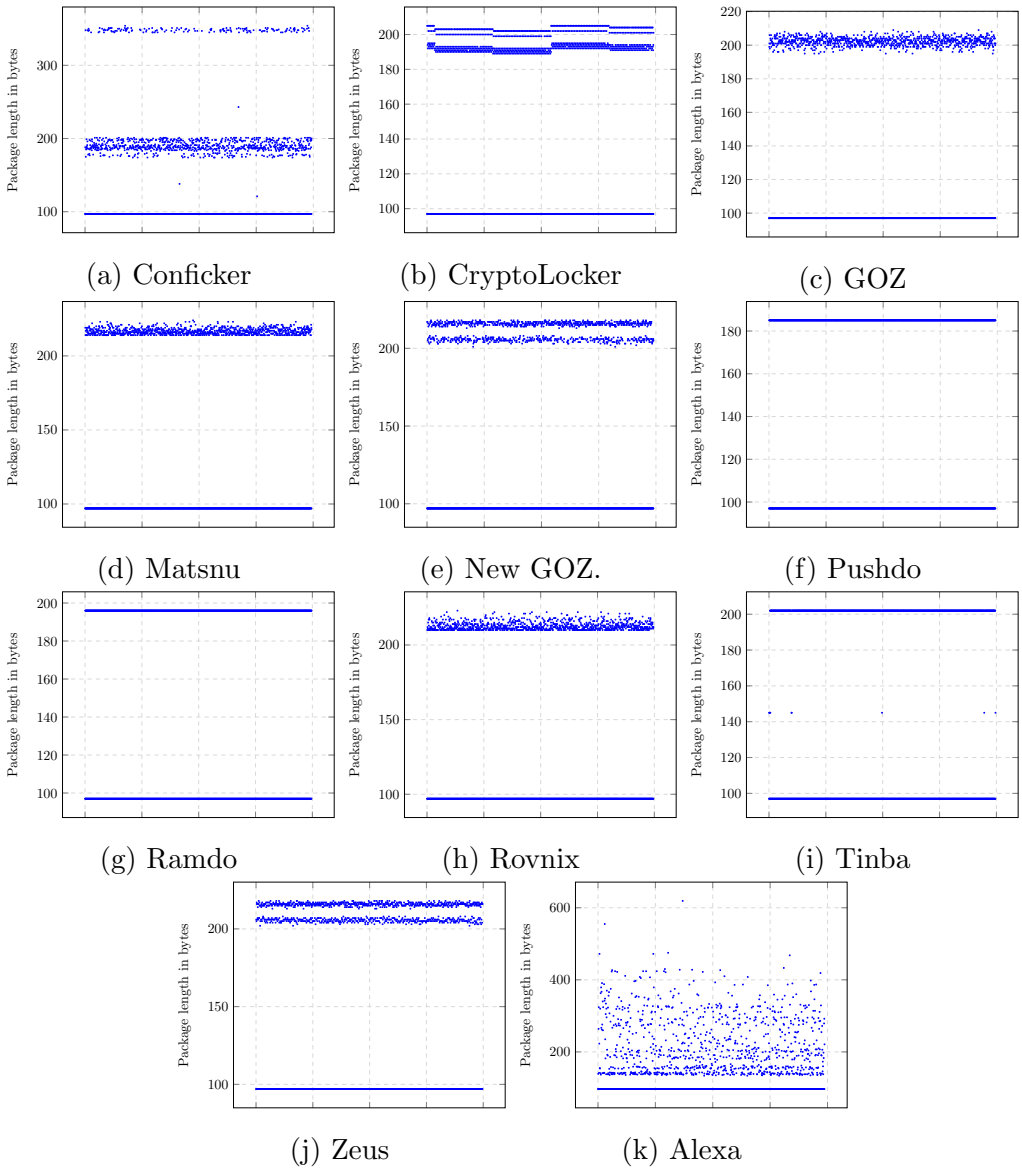


Figure 7: Package size diversity for different datasets (DGAs and Alexa) using DNS over TLS.

method is quadratic. Therefore, the whole process of the proposed traffic analysis has quadratic complexity to the length of exchanged information.

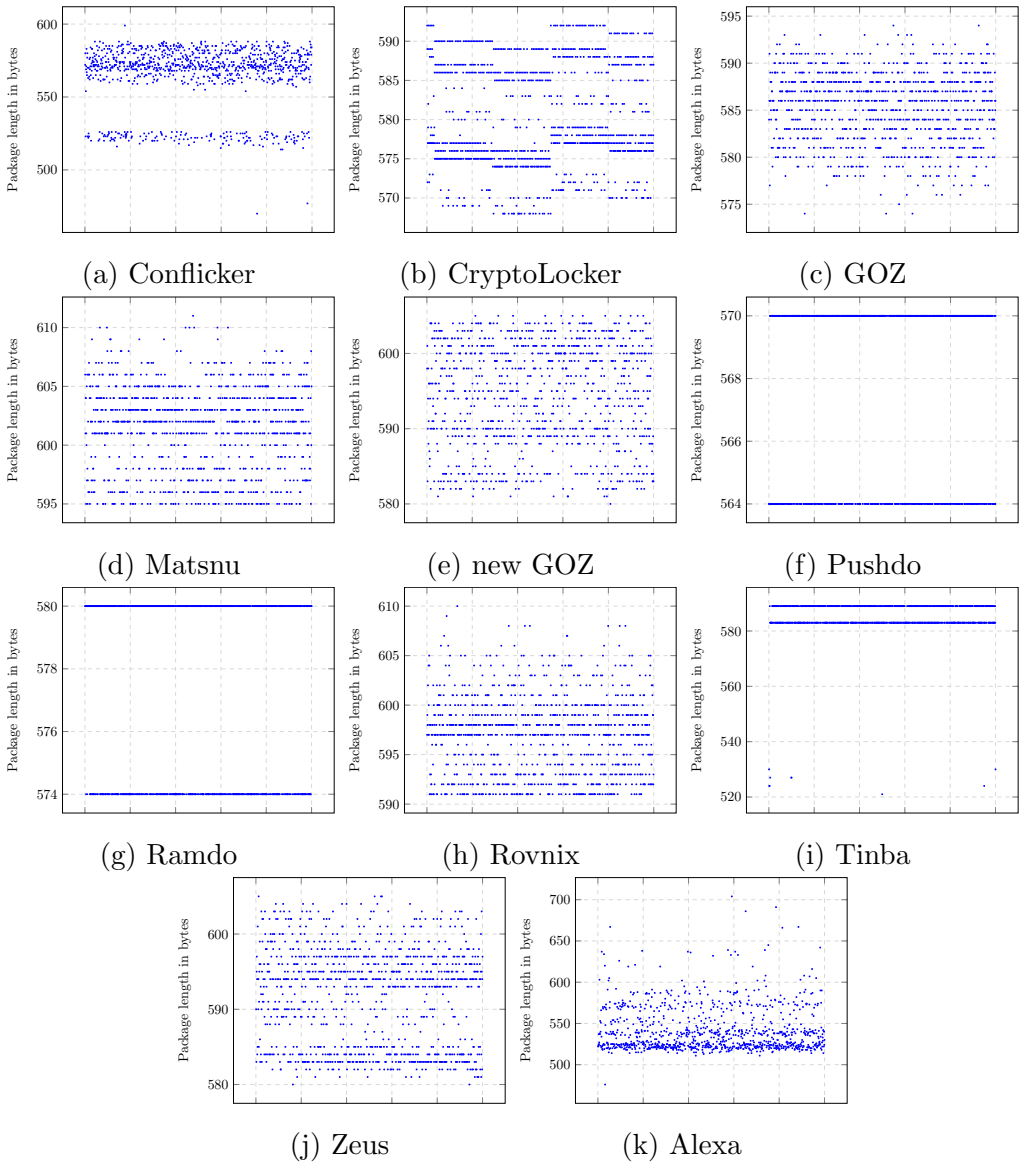


Figure 8: Package size diversity for different datasets (DGAs and Alexa) using DNS over https.

6. Conclusions

The continuous arms race between malware authors and anti-malware has made modern malware to use a wide range of cryptographic and obfuscation

methods. Therefore, modern malware has become far more stealthy making the quest for novel methods to detect and classify malware a big challenge.

Trying to overcome future challenges, we investigate the use of covert channels for making DNS queries. The motivation towards this direction is that DGAs are detected via the amount and rate of NXDomain responses and the use of pattern matching and machine learning to classify the queries based on their entropy. More recently, in order to circumvent the latter detection, some DGAs have started using combinations of words so that the queries do not appear so random. Nevertheless, the amount and rate of NXDomain responses remains the same.

We conjunct that in order to circumvent this limitation, malware might soon resort to encrypted DNS queries. As we discussed in our work, this shift would render many of currently deployed security mechanisms useless as they heavily depend on monitoring the unencrypted traffic with the DNS servers. The currently available methods and protocols that are provided by major DNS servers; including Google, Cloudflare, and OpenDNS, if used properly, may allow infected hosts to perform encrypted DNS queries masqueraded as typical HTTPS traffic with whitelisted domains, bypassing this way all deployed security mechanisms.

To address this threat, we showcase how traffic analysis can be used to provide a lightweight security mechanism and construct IoCs. Using domain names generated from several DGAs we showcase that there are emerging patterns that allow us to identify and classify them accurately. Indeed, despite their covert nature, we illustrate that the Hodrick-Prescott filter can classify them using around 30 samples with very high accuracy and perform bot attribution.

Acknowledgments

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the projects YAKSHA (Grant Agreement no. 780498), CyberSec4Europe (`https://www.cybersec4europe.eu`) (Grant Agreement no. 830929) and the Marie Skłodowska-Curie grant agreement No 778229 (Ideal-Cities).

The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

[1] N. Ismail, Global cybercrime economy generates over $1.5tn, according to new study, https://www.information-age.com/global-cybercrime-economy-generates-over-1-5tn-according-to-new-study-123471631/ (2018).

[2] T. Moore, R. Clayton, R. Anderson, The economics of online crime, Journal of Economic Perspectives 23 (3) (2009) 3–20.

[3] J. M. Rao, D. H. Reiley, The economics of spam, Journal of Economic Perspectives 26 (3) (2012) 87–110.

[4] Y. Chen, P. Kintis, M. Antonakakis, Y. Nadji, D. Dagon, M. Farrell, Measuring lower bounds of the financial abuse to online advertisers: A four year case study of the TDSS/TDL4 botnet, Computers & Security 67 (2017) 164 – 180.

[5] C. Patsakis, F. Casino, Hydras and ipfs: a decentralised playground for malware, International Journal of Information Security (2019) 1–13.

[6] F. Casino, K. R. Choo, C. Patsakis, Hedge: Efficient traffic classification of encrypted and compressed packets, IEEE Transactions on Information Forensics and Security 14 (11) (2019) 2916–2926.

[7] R. J. Hodrick, E. C. Prescott, Postwar us business cycles: an empirical investigation, Journal of Money, credit, and Banking (1997) 1–16.

[8] M. Antonakakis, et al., Understanding the mirai botnet, in: 26th USENIX Security Symposium (USENIX Security 17), USENIX Association, Vancouver, BC, 2017, pp. 1093–1110.

[9] Y. Nadji, R. Perdisci, M. Antonakakis, Still beheading hydras: Botnet takedowns then and now, IEEE Transactions on Dependable and Secure Computing 14 (5) (2017) 535–549.

[10] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, Ddos in the iot: Mirai and other botnets, Computer 50 (7) (2017) 80–84.

[11] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, D. Dagon, Peer-to-peer botnets: Overview and case study, HotBots 7 (2007) 1–1.

[12] T. Holz, C. Gorecki, K. Rieck, F. C. Freiling, Measuring and detecting fast-flux service networks, in: Proceedings of the Network and Distributed System Security Symposium, 2008.

[13] O. Katz, R. Perets, G. Matzliach, Digging deeper - an in-depth analysis of a fast flux network, `https://www.akamai.com/us/en/multimedia/documents/white-paper/digging-deeper-in-depth-analysis-of-fast-flux-network.pdf` (2016).

[14] A. K. Sood, S. Zeadally, A taxonomy of domain-generation algorithms, IEEE Security Privacy 14 (4) (2016) 46–53. doi:10.1109/MSP.2016.76.

[15] R. Perdisci, I. Corona, G. Giacinto, Early detection of malicious flux networks via large-scale passive dns traffic analysis, IEEE Transactions on Dependable and Secure Computing 9 (5) (2012) 714–726.

[16] A. J. Aviv, A. Haeberlen, Challenges in experimenting with botnet detection systems, in: Proceedings of the 4th Conference on Cyber Security Experimentation and Test, CSET'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 6–6.

[17] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, S. Ranjan, Detecting algorithmically generated domain-flux attacks with DNS traffic analysis, IEEE/ACM Transactions on Networking 20 (5) (2012) 1663–1677.

[18] Y. Zhou, Q.-S. Li, Q. Miao, K. Yim, DGA-based botnet detection using DNS traffic, J. Internet Serv. Inf. Secur. 3 (2013) 116–123.

[19] N. Jiang, J. Cao, Y. Jin, L. E. Li, Z. Zhang, Identifying suspicious activities through dns failure graph analysis, in: The 18th IEEE International Conference on Network Protocols, 2010, pp. 144–153.

[20] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, D. Dagon, From throw-away traffic to bots: detecting the rise of DGA-based malware, in: Proceedings of the 21st USENIX conference on Security symposium, USENIX Association, 2012, pp. 24–24.

[21] S. Yadav, A. L. N. Reddy, Winning with DNS failures: Strategies for faster botnet detection, in: M. Rajarajan, F. Piper, H. Wang, G. Kesidis (Eds.), Security and Privacy in Communication Networks, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 446–459.

[22] P. K. Manadhata, S. Yadav, P. Rao, W. Horne, Detecting malicious domains via graph inference, in: M. Kutyłowski, J. Vaidya (Eds.), Computer Security - ESORICS 2014, Springer International Publishing, Cham, 2014, pp. 1–18.

[23] Y. Gong, S. Qitian, Z. Zhang, A DGA odyssey PDNS driven DGA analysis, https://pc.nanog.org/static/published/meetings/NANOG71/1444/20171004_Gong_A_Dga_Odyssey__v1.pdf (2017).

[24] G. Zhao, K. Xu, L. Xu, B. Wu, Detecting APT malware infections based on malicious DNS and traffic analysis, IEEE Access 3 (2015) 1132–1142.

[25] D. Liu, Z. Li, K. Du, H. Wang, B. Liu, H. Duan, Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, ACM, New York, NY, USA, 2017, pp. 537–552.

[26] The DGA of pykspa "you skype version is old", https://www.johannesbader.ch/2015/03/the-dga-of-pykspa/ (2015).

[27] D. Tran, H. Mac, V. Tong, H. A. Tran, L. G. Nguyen, A lstm based framework for handling multiclass imbalance in dga botnet detection, Neurocomputing 275 (2018) 2401–2413.

[28] G. Attardi, D. Sartiano, Bidirectional lstm models for dga classification, in: International Symposium on Security in Computing and Communication, Springer, 2018, pp. 687–694.

[29] C. Choudhary, et al., Algorithmically generated domain detection and malware family classification, in: International Symposium on Security in Computing and Communication, Springer, 2018, pp. 640–655.

[30] H. S. Anderson, J. Woodbridge, B. Filar, DeepDGA: Adversarially-tuned domain generation and detection, in: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, AISec '16, ACM, New York, NY, USA, 2016, pp. 13–21.

[31] S. Schiavoni, F. Maggi, L. Cavallaro, S. Zanero, Phoenix: Dga-based botnet tracking and intelligence, in: S. Dietrich (Ed.), Detection of Intrusions and Malware, and Vulnerability Assessment, Springer International Publishing, Cham, 2014, pp. 192–211.

[32] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, A. Mosquera, Detecting DGA domains with recurrent neural networks and side information, arXiv preprint arXiv:1810.02023.

[33] Y. Zhauniarovich, I. Khalil, T. Yu, M. Dacier, A survey on malicious domains detection through DNS data analysis, ACM Computing Surveys 51 (4) (2018) 67:1–67:36.

[34] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. Van Steen, N. Pohlmann, On botnets that use DNS for command and control, in: 2011 seventh european conference on computer network defense, IEEE, 2011, pp. 9–16.

[35] M. Anagnostopoulos, G. Kambourakis, P. Kopanos, G. Louloudakis, S. Gritzalis, Dns amplification attack revisited, Computers & Security 39 (2013) 475–485.

[36] M. Anagnostopoulos, G. Kambourakis, S. Gritzalis, New facets of mobile botnet: architecture and evaluation, International Journal of Information Security 15 (5) (2016) 455–473.

[37] G. Ateniese, S. Mangard, A new approach to DNS security (DNSSEC), in: Proceedings of the 8th ACM conference on Computer and Communications Security, ACM, 2001, pp. 86–95.

[38] D. J. Bernstein, Dnscurve: Usable security for dns, dnscurve. org.

[39] F. Denis, Y. Fu, DNSCrypt (2015).

[40] M. Anagnostopoulos, G. Kambourakis, E. Konstantinou, S. Gritzalis, Dnssec vs. dnscurve: A side-by-side comparison, in: Situational Awareness in Computer Network Defense: Principles, Methods and Applications, IGI Global, 2012, pp. 201–220.

[41] D. Bianco, The pyramid of pain, `http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html` (2014).

[42] P. Mockapetris, Rfc 1035-domain names-implementation and specification, `http://www.ietf.org/rfc/rfc1035.txt` (2004).