Botnet Detection with DNS Monitoring

Christopher Will
Advisor: Oliver Gasser
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: christopher.will@tum.de

ABSTRACT

Botnets are today the universal tool for malicious activities in the Internet. They can send out spam messages, host fairly redundant malicious webpages, perform DDoS attacks and do much more. Of course, researchers have therefore been trying to effectively find and shut down botnets as quickly as possible. The Domain Name System has become an important part of such botnets, for both the botmaster and the defender. It enables botmasters to either hide their content servers via fast-flux, but also offers a good possibility to communicate with the Command and Control server in the background with the help of Domain Generation Algorithms. This paper gives an overview on recent developments in the field of detecting botnets with the help of the Domain Name System and evaluates the different solutions in terms of required input, practicability, efficiency and privacy.

Keywords

Botnet Detection, Domain Name System

1. INTRODUCTION

Within the past years, the Internet has become a vital part of the lives of billions of people. Today, there are many services that are crucial for everyday life and an outage of servers and services can cause great damage, both monetarily and in terms of reputation. Distributed Denial of Service (DDoS) attacks initiated by botnets are often responsible for such an outage disabling a server to answer to any legitimate traffic because of too many queries by the attacker. But botnets are capable of other malicious activities as well: Their infrastructure can be used to redundantly host content which can range from spam pages that aim to sell some replicated products over phishing sites to pages distributing malware or sending spam or phishing mails. Many large botnets have been identified or even taken down in the past, like Torpig [16], Conficker [11] or Mega-D [12]. But nevertheless, as the papers presented here suggest, there are still a lot of botnets in the Internet that might not have even been detected yet.

A very important part of the botnet is the communication structure between the bot and the botmaster. One way to maintain the communication and receive instructions is the usage of one or a few so-called Command and Control servers (C&C server). This is especially important for the first time after a new host was infected. The question is how to get the address of that server: If an IP is hardcoded, it can easily be identified and blacklisted or taken down; the whole botnet would be rendered useless. The Domain Name System (DNS) is a much better way to initialize the communication because the mapping from a domain to one or more IPs can be done dynamically. This mechanism, however, is still easy to stop because now the domain name can be blocked, which is easily and with little effort achievable. The solution found by malware developers are Domain Generation Algorithms (DGA). The bot malware is able to pseudo-randomly generate a high number of domain names that are all with very high probability not already registered. The botmaster only registers very few of these domain names. Nevertheless, all generated domain names are sent to the DNS server so that the bot gets to know the current IP of the C&C server. This complicates the countermeasures to be taken significantly: Even if a botnet malware is reverse engineered and the possible domain names are known in advance, it is possible for a botmaster to modify the DGA very fast or use obfuscation techniques to cover the DGA [1]. Moreover, even if both the currently active domain name as well as the current C&C IP are blacklisted, the malware is able to reconnect to the botmaster (using another domain and another IP). This can also not be prevented by just registering all generated domains, because they are simply too many. Therefore, other detection techniques have to be applied to discover botnet communication between the zombies and the C&C server.

In addition to the usage of DGAs, which is also called domain-fluxing, some botnets use the so-called fast-flux technique to host pages redundantly. A botmaster can never be sure about how long each bot is online. Therefore, he uses once again the DNS mapping to rapidly change the returned IP addresses so that, e.g., phishing pages keep staying accessible. By monitoring the DNS replies, it is also possible to find such botnets. However, it is important to note that the Round-Robin DNS method and Content Distribution Networks (CDNs), which are used for legitimate load balancing of large web sites, also periodically change the returned IPs for a DNS query. That legitimate traffic has to be distinguished from the illegitimate bot traffic.

This paper provides the technical background to understand the mechanisms to detect botnets with the help of DNS traffic. Then, it gives examples of frameworks that aim to do that as research has been very active in that area in the past few years. In total, one paper dealing with fast-fluxing as well as four papers presenting different systems to detect DGA-based botnet malware are examined. Additionally, the

doi: 10.2313/NET-2014-08-1_04

approaches are compared and analysed with respect to required input, practicability, efficiency and privacy.

In the following Section, we will give a technical background of modern-day DNS usage as well as the structure of a botnet. Section 3 deals with the detection of botnets employing the fast-flux technique to host content, Section 4 gives an overview on different approaches to detect DGA botnets by scanning DNS traffic. Section 5 concludes with a short summary and final thoughts.

2. TECHNICAL BACKGROUND

In this Section, we describe how modern and large web services as well as botnets use the DNS for load balancing and redundancy. Additionally, a short description of today's botnet structures is given.

2.1 Domain Name System

The main concept of the Domain Name System (DNS) can be found in the RFCs 1034 [13] and 1035 [14]. In a nutshell, the DNS provides the mapping service between a domain name and the corresponding IP. If a client wants to access a web page, it sends a query to its local DNS resolver requesting the IP address of the domain name. If no cached DNS response is found there, the resolver asks its pre-defined DNS server for the IP. If it also finds no cached entry for the name, it initiates a cascade of requests to the required zone servers according to the domain name. Once the IP or IPs are found, they are returned to the client, including a TTL number indicating how long that DNS answer should be cached without another lookup. If no mapping is found, a corresponding message is sent to the client as well.

Modern web services with a lot of traffic rely on different techniques involving the DNS to balance the load and mitigate possible DoS attacks [8]. A simple method to distribute the load is the round-robin DNS (RRDNS) method: The DNS server loops through a list of possible servers and returns a different IP for each different request until the beginning of the list is reached again. However, this approach is only suited for rough load balancing [3].

The second, more sophisticated method is the application of Content Distribution Networks (CDNs) [10]. The content distributor is normally a third company which is only responsible for intelligently providing and distributing content so that the load is balanced. To achieve that, the content servers are distributed around the world and the corresponding authoritative DNS server belongs to the CDN. Once it receives a query, it takes several parameters into account, including the availability of resources of the CDN and the distance between requesting client and the possible servers, calculates the currently best IP match and sends it back to the client [8]. To ensure an ongoing balancing, the TTL is low so that a client has to query the DNS server again quickly. This also means that it is not uncommon that the same client gets different IPs – maybe even different locations around the world – each time it queries a DNS lookup.

2.2 Botnets

Beside the botnets with one or more central C&C servers, there are also P2P-based botnets [4]. However, this paper focuses on the ones with C&C servers as P2P botnets have a different communication structure. After a new machine is infected, it attempts to contact a C&C server to get instructions on what to do next. Only then, the new bot becomes an active part of the botnet. If no connection can be established, the bot can practically not be used and is worthless. Different methods have emerged and maybe even combined to make sure that a connection is established. Just like ordinary web site traffic, one option is exploiting the DNS service to establish a reliable connection with the help of Domain Generation Algorithms (DGAs). Bots using DGAs generate pseudo-randomly domain names and send all of them to the DNS server. Most of these queries will fail as the botmaster only registers very few of these domains to map to the C&C IP. This provides a robust communication technique. Nevertheless, other techniques have also been used already, for example with the help of social networks: Kartaltepe et al. report that e.g., Twitter was used as command structure by using an account that posts Base64-encoded messages for the bots [9]. Figure 1 shows the layout of a sample C&C-based botnet. One part of it (Bot 1) could serve as content hoster, some other part could perform a DDoS attack at some server. In the background, the C&C server tells the bots what to do. Of course, it is also possible that the whole botnet is executing the same order.

The DNS techniques explained in Section 2.1 are employed by legal web services with the aim of load balancing. A botmaster who wants to host some content is mainly interested in availability as he has to expect that each bot could drop out at any time. This is where the so-called fast-flux method comes into play [8]: In such a system, the bots act as proxies redirecting traffic to some central content server. Additionally, the TTL of a DNS query is once again very low and each time a new DNS query is started, a different set of IPs of bots is returned. This means that if one bot fails or is taken down, the whole network is not affected. Moreover, this quick change of IPs makes it harder to blacklist a small set of hosts and adds redundancy to the system.

3. FAST-FLUX DETECTION

Botnets can be found by analysing either DNS queries and responses sent by oneself or by just listening and analysing generated DNS traffic by others, depending on the situation. If a botnet uses the fast-flux system to reliably provide some content with a malicious background, it can be found by active queries. The following Section discusses that detection technique in detail. In that scenario, the DNS traffic between some client attempting to get to the hosted content and the DNS server providing the IPs for the different proxy bots is interesting.

A fast-flux system aims at redundantly providing content that is mostly illegal – e.g., pages selling replicas, phishing pages, malware distributing sites – by exploiting DNS capabilities [8]. In this scenario, the bots act as proxy servers that redirect any client request to a server in the background actually hosting the content. A user is directed to these pages, e.g., by spam mails. As a single bot is much more likely to fail and suddenly be unresponsive to the botmaster, it is important for him that he can quickly react on such an event and provide a client with functional proxies as fast as possible. This is achieved with the fast-flux technique. The

doi: 10.2313/NET-2014-08-1_04

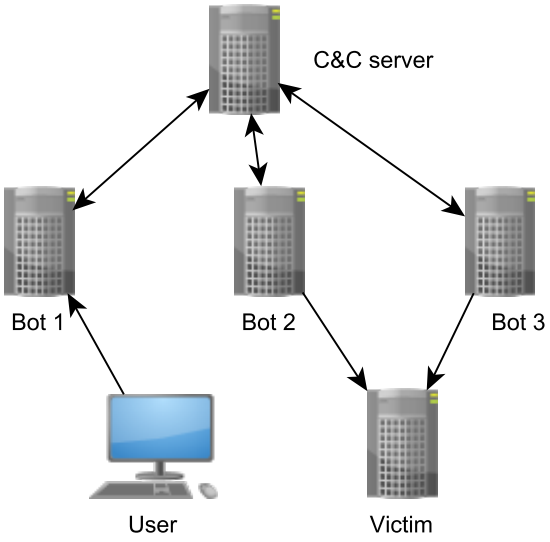

Figure 1: Sample layout of a botnet with a central C&C server in the background.

DNS responses of a domain name with a fast-flux network in the background have a very low TTL (for example, 600 seconds). Additionally, after each new lookup, a new set of IPs different to the one before is returned. This way, it is not a big problem if one or even more proxies fail as there are enough fallback IP addresses.

This technique has similarities with both the round-robin system and CDNs. As fast-fluxing is mainly used for illegal activities and the other ways for legitimate services, it is important to be able to distinguish between these two groups. As the RRDNS method usually employs longer TTLs, a distinction from it can be achieved relatively easily by examining only DNS responses with a low TTL [8].

In general, the distinction between a fast-flux network and a CDN is is more difficult. For the sake of load balancing, the TTL is low there as well. This means that other parameters need to be used as well. Holz et al. suggest to have a closer look at three different values: The number of unique A records (IPs) returned in a series of queries n_A, the number of different Autonomous System Numbers (ASN) n_{ASN} which specify the different network operators and ultimately the different locations of the servers whose IPs are used and the number of nameserver records (NS) in one lookup n_{NS} [8].

In a next step, they tried to calculate a "Flux-Score" for a domain name by defining a function that is a linear combination of the mentioned parameters with some weights w_1 to w_3:

$$f(x) = w_1 \cdot n_A + w_2 \cdot n_{ASN} + w_3 \cdot n_{NS}$$

With the help of a set of benign and fast-flux domains and a 10-fold cross-validation, they came to the following allocation of the weights: $w_1 = 1.32$, $w_2 = 18.54$, $w_3 = 0$. If $f(x) > 142.38$, then a domain is classified as fast-flux, if $f(x) \leq 142.38$, it is considered a benign domain. Using these values and a set of 128 fast-flux domains and 5803 benign domains, 99.98% of all domains were correctly classified.

The allocation of the weights clearly shows that the most distinctive feature of fast-flux networks is that the bots are spread on a large area, possibly around the world. The reason for that is of course that a botmaster cannot choose which machines get infected and takes all bots he can get around the whole world. As CDNs make also use of a lot of different IPs, the number of returned A records is a much weaker indicator of a botnet in the background. Lastly, the amount of different nameserver records plays no role at all and can be completely disregarded. Empirical studies showed then that fast-flux domains can have several thousand different A records and even hundreds of different ASNs [8].

Once a domain using fast-fluxing is found, several steps can be taken to mitigate its harmfulness or even to take down the underlying botnet [8]. A simple method would be to blacklist the affected domains, e.g., by a firewall. By employing a database of fast-flux domains, the authors state that spam filters can be improved as all mails containing domains in the database could be considered spam right away. In order to further analyse the botnet and eventually take it down, a cooperation with the Internet Service Provider (ISP) is required. The bots acting as botnets could be sent an identifiable request while they are watched to identify the content hosting server in the background that could also be the C&C server of the botnet. However, the proxies could use DGAs to complicate the identification and termination of the hosting server in the background. We will look at this method in the next Section.

In conclusion, fast-fluxing botnets can be identified and separated from legitimate DNS load balancing techniques very well and with a low rate of failure. However, as Holz et al. also mention, botmasters might attempt to clone the behavior of a CDN. But as the availability of bots is never guaranteed and as the botmaster will always choose availability to make money over hiding the botnet, botnets will in our opinion never look so similar to CDNs that they cannot be detected anymore (even though an adjustment of the weights of the Flux-Score might be necessary now and then).

As the returned IP addresses for the same host within a longer period of time have to be analysed, the identification requires to actively send DNS queries. This might require resources and could be detected by the botnet. Other systems to identify fast-flux networks have been proposed as well, which are largely based on the parameters presented here [2]. Lastly, it should of course be noted that not all botnets are used to host content, but do something different, like performing a DDoS attack. If that is the case, fast-fluxing detection techniques cannot be applied.

4. DGA DETECTION

In this Section, we present and evaluate several approaches that target identifying the communication between bots and the C&C server while using Domain Generation Algorithms (DGAs) and the Domain Name System (DNS). First, the general structure of the different frameworks is explained,

then, the different approaches are presented.

4.1 General Structure

The initial assumption is always the same: Next to all the benign DNS queries, there might always be traffic from a botnet malware that uses a DGA to get the IP address of the C&C server. This means that the malware creates pseudo-randomly a large amount of domain names. These domains have then of course a certain structure (mostly, they are longer than legitimate domains and consist of a sequence of letters and numbers). All of these domains are then requested at a DNS server. The botmaster registers only a small set of these domains for his C&C servers so that most domain resolutions will fail, only a few will reveal the IP address of the C&C server. This opens up possibilities to detect this rather unusual DNS traffic. The frameworks examined always take similar steps to find that traffic. They are shown in Figure 2 and described in the following.

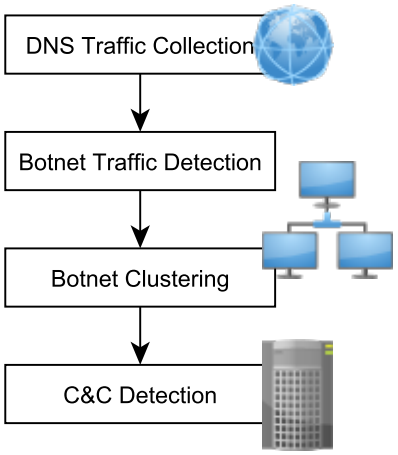

Figure 2: General structure of frameworks detecting DGA-based botnets with DNS monitoring.

DNS Traffic Collection
First of all, the detection framework needs a DNS traffic input. This traffic can be previously collected from a DNS server, but it may also be live data while the system is currently attached to and scanning traffic at a DNS server. It is common to first test the system offline with some recorded input and then evaluating if it is possible to let it run live. This mass collection and analysis of data raises privacy questions, of course. Some frameworks deal with this issue, some others do not. The topic will be mentioned again at each single evaluation.

Botnet Traffic Detection
The next step is the most important one: The identification of the botnet DNS traffic. Naturally, the most part of the DNS queries are benign and completely uninteresting for us. Therefore, some algorithms and/or learning techniques have to be applied to filter out all the benign queries so that only the botnet traffic remains. There are a lot of different methods to identify the botnet queries. The most common one is the exploitation of the fact that botnet malware using a DGA sends out a lot of queries that result in a reply stating that the domain does not exist. As such a reply is relatively uncommon as typos of users etc. are rare compared to all the correct DNS queries, this method poses a very good starting point for further, more detailed filtering.

Botnet Clustering
Normally, a lot of bot traffic of different botnets is found in the previous step. In order to better analyse them, the bots are in some frameworks further grouped according to certain parameters. This provides overview because it is common that different malware uses different DGAs that have different features and can be separated. This way, it is possible to find out which bots might belong to the same botnet and if some bots belong to a previously detected botnet or are part of a yet unknown botnet (or have at least an unknown DGA).

C&C Detection
After having identified the different botnets, the last step is to get to know the IP address of the heart of the botnet, the C&C server. As the bots are known by now, they simply have to be watched to see which of their DGA-generated DNS requests are actually successful.

This is only a rough outline of the different frameworks. There might be more steps involved, like the training of the framework or anonymization, sometimes less. The different approaches are now presented.

4.2 Pleiades

The Pleiades system by Antonakakis et al. is a very sophisticated framework which uses the "non-existing" resolution replies from a DNS server (NXDomains) to detect new classes of DGAs [1]. It consists of two different modules, the *DGA Discovery* module and the *DGA Classification and C&C Detection* module. The first one is responsible for detecting and clustering botnet queries with the help of NXDomains. The latter one is responsible for generating a model that helps finding out whether a correctly resolved DNS request by a previously identified bot was also generated by the DGA or is legitimate traffic to find the C&C server.

The *DGA Discovery module* is given all NXDomain traffic collected within a specific time period at a DNS server. These domains are then clustered in two different ways. The first way starts by computing different statistical features of the observed domains. These features are n-gram features (the frequency distribution of n-grams for different n's), entropy-based features and structural domain features (like length or number of unique TLDs). Altogether, 33 different values are computed. With the help of the X-Means clustering algorithm [15], the different domains are grouped.

Based on the assumption that the same malware creates at least partially the same domain names, the second way is the creation of a bipartite graph consisting of host vertices on the one side and NXDomain vertices on the other side. If a host queried a certain NXDomain, these two vertices are connected with an edge. After some more improvements of the graph, the X-Means clustering is applied here as well, grouping NXDomains that have been requested by a similar set of hosts.

doi: 10.2313/NET-2014-08-1_04

To finalize the clustering phase, the two different sets of clusters are now merged by computing all possible intersections between both sets of clusters. All resulting clusters that are too small (the authors chose a minimum cluster size of 40) are dropped. All other clusters are kept for the next steps. As the authors lay focus on detecting new DGAs, they want to filter out all domain clusters generated by already known DGAs. This means that with the help of the DGA Classifier, it is determined if a certain cluster belongs to a known DGA with a certain probability and if that is the case, these clusters are dropped as well (but the IP addresses are added to a detection report).

The *DGA Classification and C&C Detection module* has two main purposes: It is supposed to determine if a specific set of NXDomains likely belongs to an already known DGA and it is supposed to tell if it is likely that a single active domain query performed by a host that already sent suspicious NXDomain requests also belongs to the botnet communication. The module takes a list of popular legitimate domains, a list of NXDomains generated by bots in a controlled environment and the newly discovered NXDomains from above as input.

The identification and comparison of NXDomain clusters is done with the help of the Alternating Decision Trees learning algorithm. The module compares each input with all known DGAs and returns for the most likely DGA a label with its name together with the probability that this assignment is correct.

The C&C detection works with Hidden Markov Models. One model is used per DGA. Before it can be used, it needs a training with a set of NXDomains generated by the same DGA. In active use, it is then given one successfully resolved domain at a time which was requested by a previously classified bot. The module calculates the probability that this particular domain name was also generated by the classified DGA. If it is above a certain probability threshold, it is branded a candidate C&C domain. The output of this detection module can then be used to maintain an IP blacklist to block bot communication with the C&C server.

Pleiades was tested within a time period of over two years observing 187,600 distinct hosts querying at least one NXDomain and 360,700 distinct NXDomains in total. Among other findings, six DGAs have been observed that do not belong to any known malware. An analysis of the DGA Classifier revealed that its detection rate lies at 99.7% with a false positive rate of 0.1%. The C&C server detection is not that successful: When choosing a false positive rate of 3%, the detection rate is above 91% for five out of six examined botnets. The detection rate for the last one (Boonana) is 27.67%. However, these results are still considerably better than the ones of previously presented systems above.

Altogether, Pleiades is a tool that is technically very advanced. The idea to search for DGA botnets by scanning the NXDomain traffic is implemented very well and the classification works with a very high detection rate. However, as the Boonana botnet shows, the detection of C&C servers – even though it works very well for most botnets – can be tricked: According to Antonakakis et al., Boonana creates pseudo-randomly third-level domains while the used second-level domains are owned by dynamic DNS providers [1]. This usage of two different domain levels obviously confused the Hidden Markov Model. The authors state that in a real-world scenario the detection rate of Boonana can be improved, though. Another consideration the authors make is that a DGA could deliberately produce a number of domain names that will never successfully resolve to a C&C IP. This is especially a problem if the same generator, but with a different seed, is used. That way, the learning process could be harder for the Hidden Markov Model.

There is no particular statement about Pleiades' performance. However, as it was used in a real environment for years, we can assume it is fast enough to handle the real-time DNS traffic. Additionally, no notes about privacy are made. But as no benign traffic is saved on purpose, privacy issues are very small for Pleiades. Nevertheless, anonymization of the data could add more privacy.

4.3 Predentifier

The Predentifier framework by Frosch et al. uses several passively collected DNS features and training sets to identify botnet communication [5]. In order to distinguish between benign traffic and malicious botnet traffic, 14 different features of DNS queries and their content in general are used. By applying the k-Nearest-Neighbor (kNN) technique, the traffic is classified.

Before the system can be used, it requires a training set of labeled benign and malicious domains which serves as reference to distinguish incoming traffic. For each domain, 14 different features are determined and saved. These features are lexical features of the domain (like the number of digits compared to length of domain), DNS answer-based features (like the number of returned IP addresses), IP address-based features (like the number of different ASNs of the IP addresses), zone-based features (like the TTL value) and WHOIS-based features (like the age of a domain). Frosch et al. state that 9 of these features have not been used ever before to find botnet activity in DNS traffic.

Now, the system is ready to classify new domains. For the domain of each DNS request, the feature set is determined and the kNN algorithm is applied to classify the domain. This works by considering the feature values as a vector, finding the k nearest neighbours is then done by calculating the Euclidean distance between two vectors. The new domain is then added to the group most of the k nearest neighbours belong to. In order to always be updated on the current status of benign and malicious domains, the authors propose to retrain the system every once in a while, so that the detection rate stays as high as possible. Predentifier does not employ any kind of clustering afterwards, it can only distinguish between benign and malicious.

A large part of the paper consists of the evaluation of the features as well as the whole framework. This evaluation shows that the application of all the new features significantly increases the detection rate (from 58.86% using only the five already "known" features to 94.19%). So, with the best choice of k being two, the overall detection rate lies at roughly 94.2% with a false positive rate of 8.7%.

The training required by Predentifier must be repeated at times to keep the detection quality as high as possible which can be resource-intensive and lower the practicability of the approach. However, the authors make no statement about the performance of their system. Moreover, the initial input is crucial for the whole detection accuracy. Even though the system is flexible and can learn new types of malicious domains with the help of many different features, if the initial training set is badly chosen and lacks important domain groups, the detection rate can suffer.

The authors state that "only DNS answers are stored – information that is publicly available within the domain name system" and therefore, privacy is preserved. However, privacy is still an issue because the training sets still store DNS answers – even completely legitimate ones – with the IPs still visible. Storing the pages queried by an IP address can be secretly used for a wide variety of purposes – from personalized advertisement to identifying users accessing illegal pages. This means that the privacy is definitely affected by the framework. Such issues could be avoided, e.g., by employing an anonymization of the IPs with the help of hashing.

4.4 Privacy-Preserving Detection with Bloom Filters

Guerid et al. designed a framework using the NXDomains returned to bots, just like Pleiades [7]. The big difference of their system compared to all other presented is that they put emphasis on privacy issues. Before any processing for the botnet detection takes place, the input data is anonymized. The identification and clustering of bots and their C&C server is achieved by using Bloom Filters.

The first step of the framework is the anonymization of the IP addresses of the scanned traffic. This is done by hashing the addresses together with a periodically changing salt. This hash is subsequently called the "identifier" of the response. Then, the NXDomains are cleared of any responses that are easily identifiable as uninteresting, like queries with invalid characters or Top Level Domain. Afterwards, the remaining queries are sent to the *Community Construction Layer*.

The Bots Detection Module, first part of the *Community Construction Layer*, creates a Bloom Filter for each host that appears in the received NXDomains. A Bloom Filter is a data structure consisting of an array of bits. An input is hashed by a number of different hash functions and the bit in position in the array that corresponds to the hash result is set to 1. By default, it is 0. That way, all NXDomains queried by a certain host are stored in its filter (this framework here uses only one hash function per filter). If the filling ratio of the Bloom Filter is below 0.5% or above 10% (with the total size of the Bloom Filter being 1000 Bits), its host is dropped because it is considered uninteresting noise and it is no longer considered for any further processing.

The Bots Grouping Module, second part of the *Community Construction Layer*, is called by the Bots Detection Module and receives all newly collected identifiers and their Bloom Filters. It calculates how similar each host's queries are to each other to group bots by comparing their Bloom Filters. After grouping, all groups that contain more hosts than a specific paramater are passed on to the next module. This also means that any noise is filtered out, for example a host only with a misspelled, benign domain query.

In a last step, the *Malicious Domain Name Detection Layer* is responsible for the detection of the C&C servers. First, the C&C detection module is called and gets all the groups created by the Bots Grouping Module. It then creates new Bloom Filters for each host, this time with all the successfully resolved domain names. The filters within each group are then compared again to find overlapping domains. These domains are subsequently considered candidate C&C domains. Finally, in the Domain Name Validation Module, the system finds out if an identified domain could be benign and has no relations with botnet communication by computing a traffic dispersion indicator. If it is low, it means that a certain domain does not appear in many different bot groups which means that it is very likely a DGA domain (as different DGAs do not produce the same domain names).

Unfortunately, the evaluation by the authors lacks important information. No tests were performed to determine a reliable detection or false positive rate. Moreover, even though the paper was published in 2013, the DNS captures used were from 2009 and 2010. We assume that within that period, the landscape of botnets could have significantly changed so that the expressiveness of the evaluation is also questionable. An issue of the whole framework is that it requires the different bots of the same botnet to query the same domain names at roughly the same time. With the current implementation, the different bots could request the same domain with a larger time span in-between, for example more than one hour – as the authors state that they need a DNS history "of less than one hour [...] to detect malicious domain names" [7].

Nevertheless, performance-wise, the tool is able to work in real time as it is capable of processing half a million DNS queries per second (according to the authors, this is more than a typical DNS server has to handle). Moreover, the approach to preserve the users' privacy is commendable as there is no necessity to save the IP addresses of the DNS replies if only the C&C servers are supposed to be discovered. But if we want to identify the bots in the botnets, we are currently unable to do that with this implementation (although it would certainly be possible to monitor the IPs trying to access a previously identified C&C domain).

4.5 Anchor Domains

Gao et al. present a system that uses the correlation with predefined anchor domains to find new malicious domains [6]. Their system is not specifically designed to find botnets, but malicious domains in general. However, as their approach is different to the other ones and as they actually found new domains that are used presumably for botnet communication, it is mentioned here. The idea behind their framework is that it is very often the case that once a machine contacts a known malicious domain, it is very likely that it wants to contact other malicious domains at the same time or within a short period of time. These additional domains are supposed to be found.

As input, the system requires beside the DNS traffic a set of known malicious domains A, the *anchor domains*. Additionally, a time window T_w must be specified. All domains that are requested half that time before or half that time after a malicious domain is queried are analyzed. All the domains d collected around one anchor domain are called a segment s. The set S contains all segments. In the next step, by calculating two values m_{tf} and m_{idf}, the TF-IDF (term frequency – inverse document frequency) metric is applied. It measures how many times a domain name occurs while taking into account that there are popular domains that are uninteresting for us. The term frequency m_{tf} of a domain d and a segment s holds the number of appearances of the domain in that segment. The inverse document frequency $m_{idf} = |S|/|\{s \in S : d \in s\}|$ mitigates the impact of legitimate, well-known domains that are likely queried often by all hosts. If both values are above certain thresholds T_{tf} and T_{idf}, the corresponding domain is kept. All other domains are disregarded.

In the next step, the clustering is done with the help of the X-Means clustering [15] algorithm and the *pattern of co-occurrence* with the anchor domain (this means that domains appearing in the same segments may be more likely grouped than domains not having that in common). After splitting the segments in more subsegments and applying two more filters for further improvement of the results, the clustering is finished and a set of clusters is presented that should be analyzed in more detail.

In an evaluation of their approach, Gao et al. detected 6890 previously unknown malicious domains, among them also domains supposedly generated by a DGA. Each anchor domain could be expanded to an average of 53 newly detected malicious domains. Their false positive rate lies at 3.6%. As each anchor domain can be treated independently, their approach is easily parallelizable, leading to a good performance.

The largest issue of that framework is that it is only as good as the list of anchor domains it is provided with. Any malicious domain that stands in no correlation with any of the anchor domains will never be detected. This is especially a problem with DGAs, as one of their core feature is to regularly generate new domains and to never use the old ones again. So, either the blacklist must be updated very quickly or such DGA-based malware will never be found. That is why this approach is not perfectly suited to find botnets with these DGA-based DNS queries. No specific measures are taken to protect the privacy and anonymize the collected input, as well.

5. CONCLUSION

In this paper, we gave an overview on recently published systems to detect botnets and their C&C servers by monitoring the DNS traffic. This survey shows that there are many different approaches. Most of them show a high detection rate for botnets using DGAs or fast-flux techniques of up to 99.7%. Additionally, the performance of most frameworks was also proven in a live scenario at large DNS nodes which makes them actually deployable. Nevertheless, except for one system, privacy issues are largely disregarded. The frameworks handle a large amount of data, therefore, they must take care of the protection of a user's privacy.

However, the development of botnet malware will never stop and malware developers will also have noticed the effort taken to detect botnets – be it by DNS or otherwise. This means that the development of new malware trying to evade and trick the different mechanisms presented here will continue. Therefore, the progress of botnet development must be closely watched to be able to quickly react on new communication techniques and to develop new frameworks to stop them.

6. REFERENCES

[1] 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 *Proceedings of the 21st USENIX security symposium*, 2012.

[2] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton. Real-time detection of fast flux service networks. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 285–292. IEEE, 2009.

[3] V. Cardellini, M. Colajanni, and S. Y. Philip. Dynamic load balancing on web-server systems. *IEEE Internet computing*, 3(3):28–39, 1999.

[4] D. Dittrich and S. Dietrich. P2p as botnet command and control: a deeper insight. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 41–48. IEEE, 2008.

[5] T. Frosch, M. Kührer, and T. Holz. Predentifier: Detecting botnet c&c domains from passive dns data.

[6] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An empirical reexamination of global dns behavior. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 267–278. ACM, 2013.

[7] H. Guerid, K. Mittig, and A. Serhrouchni. Privacy-preserving domain-flux botnet detection in a large scale network. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–9. IEEE, 2013.

[8] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS*, 2008.

[9] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu. Social network-based botnet command-and-control: emerging threats and countermeasures. In *Applied Cryptography and Network Security*, pages 511–528. Springer, 2010.

[10] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182. ACM, 2001.

[11] F. Leder and T. Werner. Know your enemy: Containing conficker. *The Honeynet Project, University of Bonn, Germany, Tech. Rep*, 2009.

[12] P. Lin. Anatomy of the mega-d takedown. *Network Security*, 2009(12):4–7, 2009.

[13] P. Mockapetris. RFC 1034 - Domain Names - Concepts and Facilities, 1987.

[14] P. Mockapetris. RFC 1035 - Domain Names -

Implementation and Specification, 1987.

[15] D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[16] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.