Multi-dimensional Aggregation for DNS Monitoring

Lautaro Dolberg, Jérôme François, Thomas Engel

Multi-dimensional Aggregation for DNS Monitoring

Lautaro Dolberg, Jérôme François, Thomas Engel
SnT (Interdisciplinary Centre for Security, Reliability and Trust), University of Luxembourg
4, rue Alphonse Weicker, L-2721 Luxembourg, Luxembourg
firstname.lastname@uni.lu

***Abstract*—DNS is an essential service in the Internet as it allows to translate human language based domain names into IP addresses. DNS traffic reflects the user activities and behaviors. It is thus a helpful source of information in the context of large scale network monitoring. In particular, passive DNS monitoring garnered much interest for the security perspectives by highlighting the services the machines want to access. In this paper, we propose a new method for assessing the dynamics of the match between DNS names and IP subnetworks using an efficient aggregating scheme combined with relevant steadiness metrics. The evaluation relies on real data collected over several months and is able to detect anomalies related to malicious domains.**

***Index Terms*—Security, Monitoring, DNS, Malicious domains, Aggregation**

I. Introduction

The DNS (Domain Name System) [1] plays a major role in the networks and generally in Internet. It maps human-readable names such as *www.uni.lu* to IP addresses. Moreover, reallocating or distributing resources dynamically may be done transparently with little effort (dynamic DNS, round robin DNS for load balancing).

However, these properties make it a good target and tool for attackers. For example by disrupting a DNS server with a denial of service attack, DNS clients can be prevented from locating services. DNS spoofing attacks are even worse, because their objective is to falsify the IP address mapping of a domain name and so transparently redirect users to malicious machines [2]. DNS Security Extensions (DNSSec) [3] have been developed, but it is not largely deployed yet and can help against only some specific threats like spoofing attacks. However, other threats such as botnets [4] and in particular fast-flux services [5] occur and become of the primary interest. Fast-flux is used to support various kinds of malicious activities (DDoS attacks, spam, espionage, etc.) through the coordination of an army of zombies in a botnet. This is done by hosting offensive content or synchronization information on multiple machines with a kind of round robin mechanism for IP address allocation to a DNS name.

DNS traffic monitoring can help to identify these two major threats, fast-flux and phishing, and can also provide insights into other malicious activities [6], [7], [8]. It is a complementary to the standard approaches like detecting misbehaviors on a computer from a system point of view [9], using a firewall or a Intrusion Detection System [10] or fully monitoring the traffic for tracking anomalies [11] or suspects devices [12]. Firstly, domains used in malicious activity are intuitively requested over short periods of time (during an attack campaign, for example, and/or before being disrupted by defensive measures). Secondly, the IP addresses associated with a malicious domain can be highly variable, fast flux being the best example. These features have been used in [13], [14]. In this paper, we also consider them dynamically, as they change over time, by aggregating corresponding data in a multi-dimensional structure. As both DNS and IP space follow hierarchical organisation, we propose using MAM (Multidimensional Aggregated Monitoring) [15] to handle the IP and DNS space simultaneously.

To summarize our contributions, we propose to use a hierarchical and multi-dimensional aggregation scheme for evaluating the mapping steadiness between the domain names and the IP addresses. Dedicated metrics are defined and assessed through long term experiments for detecting important changes (macroscopic view) and discovering domains and IP addresses which are responsible for these changes (local view). Finally, the scalability of our method is evaluated.

Hence, a network operator can leverage our method to monitor if there are infected hosts in the operated network.

The rest of this paper is organized as follows: section II describes the general scheme of our approach. Section III introduces our multi-dimensional aggregation and formally defines its steadiness. Evaluation is presented in section IV. Section V focuses on related work. Finally, concluding remarks are presented.

II. Background and Architecture

A. DNS name mapping and anomaly detection

DNS names are organized in a hierarchical tree and a Fully Qualified Domain Name (FQDN) represents the exact position of a domain in this tree. Considering the FQDN of our university webserver, *www.uni.lu.*, the root of the DNS tree is symbolized by the last dot while the other levels, separated by dot characters, are named. The hierarchy is read from right to the left: the first level, also called the TLD (top level domain), is *lu*, the second, *uni* and the third, *www*. Each represents a single node in the tree which can have multiple children. For instance all **.uni.lu* names are children of the node corresponding to *uni.lu*, which is the child node *uni* of the *lu* TLD. Thus, the number of levels considered qualifies a domain as a n-level domain (n-LD). For example, *snt.uni.lu* is a 3-LD and is a subdomain of *uni.lu*. The DNS hierarchy is illustrated in Figure 1. For full functionalities and operational details, the reader may refer to [1].

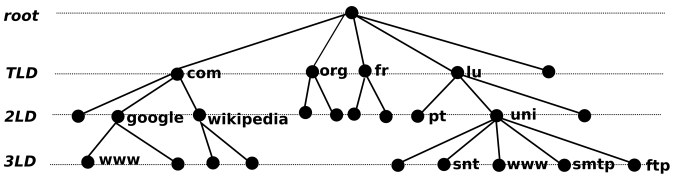

Fig. 1: DNS tree and level domains

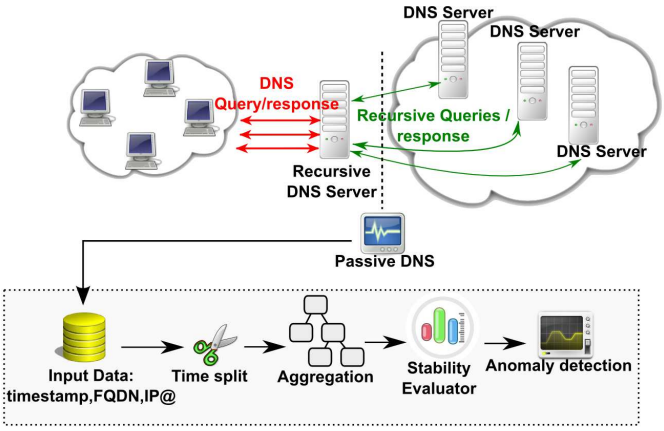

Fig. 2: System overview

In brief, the hosts under the same authority are usually configured to request the IP address of a DNS name through the DNS server of their operators which is then responsible to iteratively contacts the proper DNS name server for each level. Assuming *www.uni.lu*, it contacts known root servers (pre-configured list) to retrieve the name server responsible for *.lu*, which will then be requested to locate the name server for *uni.lu*, which is the authoritative server for indicating the IP address corresponding to *www.uni.lu*.

As highlighted in the introduction, the association lifetime between a domain name and an IP address may reveal malicious activities [16]. In this paper, we propose to avoid single DNS name evaluation by leveraging aggregation. For example, globally monitoring *uni.lu* may sometimes be important but a more specific domain like *gforge.uni.lu* can also be helpful. The same is valid for IP addresses, since they are monitored by blocks (/24, /28 subnets for example) for scalability reasons. Moreover, the granularity of our aggregation varies over the feature space, especially with respect to the number of DNS requests, *e.g.* some IP addresses will be aggregated by /24 subnets and others by /16 subnets. Similarly, the number of hierarchical levels separated by dots in DNS names is variable (there is no regular split, unlike most of the current approaches [13], [17]). Hence, this allows efforts to be concentrated on activities having a visible impact on the network, *i.e.* reaching a certain threshold.

Using aggregation is an advantage as single malicious domain detection may be easily evaded by attackers through mutltiple subdomain registrations. For example, an attacker can register the domain *safenetworking.com* and then derives an infinite number of subdomains freely. Hence, if *aaa.safenetworking.com* is detected as malicious and so blacklisted, the attacker can easily replace it by *bbb.safenetworking.com*. Using aggregation, our approach aims to learn that the domain to blacklist is *safenetworking.com*. A naive approach would consider the entire domain except the top level domain (.com) but this is impracticable as malware may be also hosted in a subdomain of a legitmated domain which, evidently, must not be blacklisted. The best example in this case is the dynamic DNS services.

The same reason is valid for IP addresses. For example, a small company hosting a service (like a website) would probably use a single IP address while a large company would use load balancing and so associate a single DNS name to several IP address. Unlike [13], [17], the granularity is automatically adjusted by our method.

B. System overview

As highlighted in Figure 2, a passive DNS probe [18] is deployed between the recursive server(s) of the monitored network. This probe can collect all traffic for iterative resolutions. In this way, the original IP addresses of clients are kept private. We are interested in this paper in the mapping between IPv4 addresses and requested names (A records). Thus, all other requests are excluded by filtering on resource record types. As the objective is to assess the steadiness of the mapping between IP addresses and names, a timestamp is used. Hence, the input data of our system is a list of tuples $< timestamp, FQDN, IP\ address >$. To evaluate the steadiness over time, data is divided into time windows as shown in Figure 2. In each window, the aggregation process takes place and creates a summarized tree-based view of DNS data. These two processes are executed by *MAM* (Multidimensional Aggregation Monitoring), which is described in the next section. Based on such sequences of trees, our approach computes a steadiness metric to evaluate the extent to which the IP addresses of a domain belong to a same subnet and whether the subdomains of a domain also belong to the same subnet.

Based on the steadiness, an anomaly detection engine is built to detect abnormal changes in steadiness due to unusual behaviour and to identify the responsible domains and/or IP subnets.

III. Assessing steadiness in DNS

A. Multidimensional Aggregation using MAM

MAM stands for *Multidimensional Aggregation Monitoring* [15]. *MAM* aggregates large collections of network-related data using multiple features such as IP Addresses, FQDNs, GPS coordinates, services, etc.

Each feature is associated to a dimensional space where data can be aggregated. For example, IP addresses can be aggregated by subnetwork and DNS names by subdomains. *MAM* is leveraged in this paper due to two major advantages. Firstly, the data space is not divided a priori and the aggregation granularity varies over the space according to the

events being monitored, *i.e.* reaching a user-defined volume-based threshold (number of bytes, alerts, etc). To illustrate, IP addresses are aggregated into subnets but the subnet size is not fixed manually and *MAM* can decide for example to aggregate part of data using a /16 prefix, while using /24 for another, or even keeping single host information (/32). The second advantage is that there is no order relationships between dimensions. When, for example, both domain names and IP addresses are monitored with a less flexible tool, the user may decide to aggregate data on addresses first and then on subdomains. With *MAM*, there is no need to make such a choice.

We are interested in the mapping between domain names and IP addresses which, hence, correspond to the two monitored dimensions. To represent such data, *MAM* uses a tree structure composed of a root node and dependent children. Figure 3 shows a short example. Each node contains the following information:

1) dimension names and values (e.g.: {dns:biz.ROOT}, {ip:64.0.0.4/4}, etc)
2) percentage of aggregated activity (activity volume defined as vol) for the current node
3) cumulative percentage of activity of the node and its subtree defined as acc_vol

As noted, the IP feature is defined as prefix and a prefix size. As described in section II-A, DNS follows a hierarchical structure where the level domains basically represent intermediate nodes. We consider the root level (ROOT) and a final level denoted by $. This allows to differentiate a node corresponding to all `*.uni.lu` names like `www.uni.lu` or `snt.uni.lu` and a direct request to `uni.lu` which is represented by `$.uni.lu`. Each unique mapping between a domain name and an IP address is considered to have a volume $vol = 1$. Hence, it only represents the existence of such a mapping and is independent from the number of times the domain has been requested. From our perspective, counting the number of requests/answers by tuple of domain name and IP address would have been helpful. However, due to caching at the recursive servers, such a source of information is not reliably available.

For aggregation purposes, *MAM* creates leaf nodes representing unique data instances (/32 IP address and a domain like $.*.ROOT) and insert them iteratively by creating intermediate nodes if necessary. Only nodes representing a minimum volume, $vol > \alpha$, are retained, otherwise they are aggregated into their parents by summing vol meanwhile.

Data is also aggregated over time by specifying η, a time window size in seconds. Hence all events within a single window will be aggregated using the space aggregation scheme described above. This is an essential feature of *MAM* since our goal is to study the dynamic of the DNS mapping over time.

To keep memory consumption low, the tree size is limited to 3000 nodes using a Least Recently Used (LRU) strategy.

For a detailed and formal description of *MAM* and associated algorithms, the interested reader can refer to [15]. In this paper, we analyze time series of the trees constructed by *MAM* as illustrated in Figure 3 which are constructed using tuples $< timestamp, FQDN, IP\ address >$ as shown in Figure 2. Such tuples serve as input of *MAM*, with the IP address being the address matching a domain name in a DNS answer.

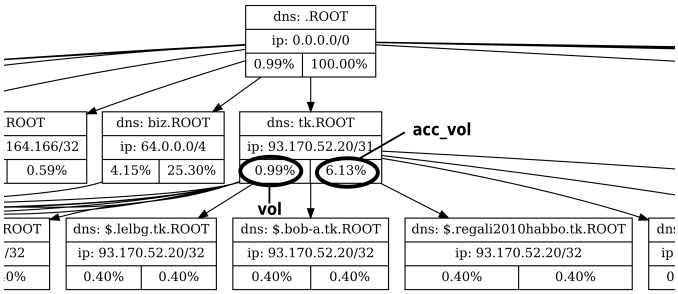

Fig. 3: Sample tree for IP Address and FQDN (For clarity, only a subpart is presented as shown by unterminated arrow)

B. Steadiness metrics

DNS names are usually associated with a group of IPv4 Addresses, and this association tends to be stable over time for benign domains. However, inconstant mappings between domains and IP subnets are suspect [19], [8]. This concept is extended to the trees obtained from *MAM*. The aggregation helps to increase the visibility of distributed events like fast flux, and so avoid bias due to local phenomena.

We propose a metric to characterize the steadiness of domain names associated with subnetworks over a time sequence. While this can be taken as a measure of the local steadiness of a node in a tree, it is also possible to evaluate the global persistence of all nodes over a sequence of trees, and thus assess the general steadiness of the mapping between IP and DNS spaces.

Assuming a sequence of N trees $S = \{T_1, T_2, \ldots, T_N\}$, each node $n \in Node$ in a tree is defined by the following elements:

- n_{acc_vol} is the cumulated proportion of activity volume as defined in *MAM*,
- n_{prefix} and n_{prefix_len} are respectively the IP prefix and its size,
- n_{dns} is the set of level names of the FQDN (for example $\{www, uni, lu\}$ for `www.uni.lu`)

When evaluating the steadiness of a node $n1$ in a tree T_i, $stead(n1)$, the objective is to check for the presence of $n1$ in the previous tree T_{i-1}. However, while aggregation improves scalability, exact information is lost and precise correspondence between nodes is rare. Therefore it is necessary to find the most similar node to $n1$ in the tree T_{i-1}. The notion of similarity is based on the hierarchical structure of IP addressing (subnets) and DNS naming space (subdomains). Assuming two nodes, $n1$ and $n2$, the similarity is positive only if one of the following condition is verified:

- $n1$ and $n2$ represent the same domain and IP subnet

- $n1$ is a child of $n2$ for at least one dimension (DNS or IP) and is the same as $n2$ for the remaining ones
- $n2$ is a child of $n1$ for at least one dimension (DNS or IP) and is the same as $n1$ for the remaining ones

In brief, nodes with locations that are disjoint in at least one space will have a similarity of 0, i.e. none of them can be a child of the the other in a *MAM* tree. For example, 192.168.10.0/24 and 192.168.20.0/24 represent disjoint portions of the IP space; and `www.uni.lu` and `snt.uni.lu` are disjoint in the DNS space whereas `www.uni.lu` is a child (subdomain) of `uni.lu`.

Based on this definition, the built tree allows easy location of the most similar node to $n1$, denoted as $most\_sim(n1)$, since parent-child relationships are preserved. Assuming a current node $n0$ having a strictly positive similarity with $n1$, *i.e.* $sim(n1, n0) > 0$, the algorithm considers each child node of $n0$ in the tree to find the node $n0'$ such that $sim(n_1, n_0') > sim(n_1, n_0)$ and $sim(n1, n0') = max_{n2 \in child(n0)} sim(n1, n2)$. The goal is to find the child node with the maximum similarity. If no child guarantees $sim(n_1, n_0') > sim(n_1, n_0)$, the algorithm stops because this means that all child nodes represent divergent spaces relative to $n1$. The most similar node is thus $n0$, the current node. The algorithm is initialized by starting from the root.

The similarity between two nodes, $n1$ and $n2$, is calculated over the different features and the activity volumes as follows:

$$\begin{aligned} sim : Node \times Node \rightarrow [0;1] \\ sim(n1, n2) = \quad & \alpha \times IP\_sim(n1, n2) \\ & + \beta \times DNS\_sim(n1, n2) \\ & + \gamma \times vol\_sim(n1, n2) \end{aligned} \tag{1}$$

subject to:

$$IP\_sim : Node \times Node \rightarrow [0;1]$$
$$IP\_sim(n1, n2) = 1 - \frac{|n1_{prefix\_len} - n2_{prefix\_len}|}{32} \tag{2}$$

$$DNS\_sim : Node \times Node \rightarrow [0;1]$$
$$DNS\_sim(n1, n2) = \frac{|n1_{dns} \cap n2.dns|}{|n1_{dns} \cup n2_{dns}|} \tag{3}$$

$$vol\_sim : Node \times Node \rightarrow [0;1]$$
$$vol\_sim(n1, n2) = 1 - |n1_{acc\_vol} - n2_{acc\_vol}| \tag{4}$$

α, β and γ can be fine tuned to give preference to a certain feature based on the user knowledge ($\alpha + \beta + \gamma = 1$).

Being inspired by [20], [15], equation (2) considers the size difference between two IP subnets with respect to the number of IP addresses and normalized with respect to the total number, 2^{32}, of IPv4 addresses. This yields a value bounded between 0 and 1, which is subtracted from one to obtain a similarity value. Obviously, this is only meaningful only if one subnet is a subnet of the others, which, in our case, is guaranteed by searching for the most similar node. Therefore, the more similar the prefix size, higher the similarity.

The second part of the metric focuses on the variation of DNS names in equation (3) (*e.g.*, `www.google.com` vs. `google.com`) by splitting the DNS name into a set of labels using the dot as a separator (`www`, `google` and `com` for `www.google.com`). Based on these sets, the Jaccard index is computed (the ratio between the number of elements in the intersection and the union) to evaluate the similarity between pairs of sets. Again, searching for the most similar node guarantees that compared domains are the same or, that one is a subdomain of the other. Thus, the similarity is proportional to the number of shared level names.

Therefore, the similarity is a value bounded between 0 and 1. The steadiness of the node $n1$ in T_i is evaluated regarding the similarity of $n1$ to the most similar node in the previous tree, T_{i-1}, and the steadiness of the latter. This smoothing allows the steadiness of the node $n_1$to be dependent not only on the previous time window but also the earlier ones.

$$\begin{aligned} & n1 \in T_i, n2 \in T_{i-1}, n2 = most\_sim(n1) \\ & stead(n1) = sim(n1, n2) + \mu \times stead(n2) \end{aligned} \tag{5}$$

To avoid giving preference to any parameter, $\alpha = \beta = \gamma = \mu = 1/4$ in equations (1) and (5)

Although the previous equation is helpful in calculating the individual steadiness of a node (local steadiness), the global persistence (macroscopic steadiness) of the tree T_i is the average steadiness of all nodes in T_i

$$pers(T_i) = \frac{\sum_{n \in T_i} stead(n)}{|\{n \in T_i\}|} \tag{6}$$

	Domains	IP Address
Normal	661968	164559
Malicious	173066	174619
Total	835034	339178

(a) General information

Malware	45%
Phishing	30%
Harmful	13%
Malvertising	4%
Trojan	3%
Others	5%

(b) Malicious domains

TABLE I: Datasets

IV. Evaluation

A. Datasets and methodology

We built a dataset using DNS Records from a Passive DNS database provided by two major Internet providers in Luxembourg in collaboration with the University of Luxembourg. The records give the associations between domain names and IPv4 addresses. DNS records were collected from 2011-04-23 to 2012-06-30. Due to a non-disclosure agreement, the exact probe location cannot be revealed. To establish the ground truth, two subsets were extracted from A records (IPv4 addresses):

- a malicious set by extracting DNS records related to domains obtained in the following malware databases: MalwareDomains [1], Exposure [2] and FIRE [3].
- a normal set excluding domains from the previous set

[1] http://www.malwaredomains.com/, accessed on 06/19/2012
[2] http://exposure.iseclab.org/, accessed on 06/19/2012
[3] www.maliciousnetworks.org/, accessed on 06/19/2012

As highlighted in table I(a), we considered that 662k normal domains would be representative. In comparison, 173k malicious domains were represented. However, the number of distinct IP addresses was similar, justifying steadiness of the mapping between domains and IP addresses as a relevant metric.

The blacklists constitute the ground truth in our experiments and correspond to malicious domains (details in table I(b)). As the blacklists may contain errors, a bias may be introduced. Another solution, largely leveraged in related work like [8], [13], is whitelisting using Alexa [4] which ranks the top 1,000,000 websites by access count. However, this inevitably results in a strong bias as it represents popular sites, which are highly constant by nature. To keep our evaluation valid, such an approach was rejected.

The objectives of the experiments were to:

- check that steadiness is a feature that discriminates between malicious and normal domains
- to verify that it is possible to detect attacks, *i.e.* that the time series are highly steady for normal domains compared to malicious ones
- show how to extract suspect data that has has an abnormal effect on the steadiness
- assess the dependency between detection accuracy and aggregation granularity
- evaluate the impact of aggregation on performance

B. Steadiness Distribution

In this experiment, the validity of the proposed metric for evaluating the steadiness in section III-B was verified.

For the complete dataset, the local steadiness of the nodes was computed using equation (5) over the two disjoint datasets: malicious and normal domains gathered from the passive DNS database. For each dataset, the aggregation relies on a time window (η) of one week and an aggregation threshold of 2% (α). This tuning is the same for following sections, unless otherwise mentioned, and was arrived at by running preliminary experiments over a small sample of data.

By considering every tree (and so every week), Figure 4 shows the cumulative distribution of nodes regarding their steadiness (buckets of $[i; i + 0.1]$ have been constructed). In both distributions, there is a significant increase on the highest values of steadiness. This phenomenon occurs because some generic nodes qualified at the higher levels exhibit a high steadiness independently of the dataset. For example, the `.com` domain and `0.0.0.0/0` are always present.

For normal domains, the curve continuously remains increasing beyond 0.75 unlike malicious domains. This means that there is a significant proportion of nodes having a steadiness higher than 0.75 for normal domains.

Nevertheless, the ere is one intermediate high increase, at 0.6 for the malicious dataset and at 0.75 for the normal dataset. Therefore, the steadiness metric is able to reveal the differing impact of malicious and normal domains in constructed trees.

[4]http://www.alexa.com/topsites

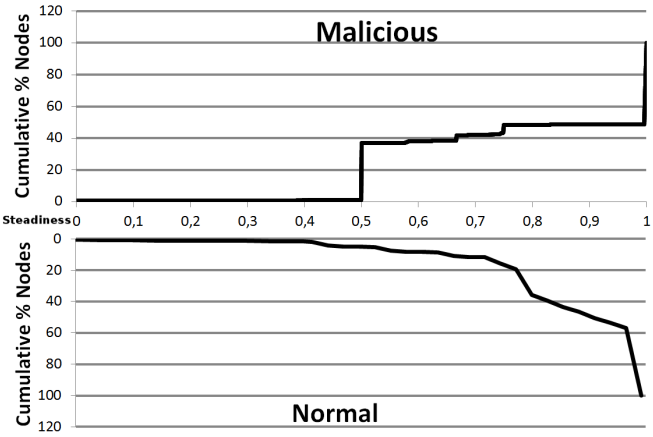

Fig. 4: Cumulative distribution functions of nodes according to steadiness values

C. Steadiness Evolution

While the previous experiment assessed the steadiness of nodes independently of time, the next experiment considered macroscopic steadiness (equation (6)) of the trees over time. To conduct this experiment, 10 consecutive weeks, from 2011-07-09 to 2011-09-24, were chosen, since they represent the greatest number of records. The total size of the dataset for 10 weeks is 81806 DNS Records (both malicious and normal domains).

For this research, we were interested in observing how steadiness varies when different percentages of malicious domains are present. Thus data sets were generated by injecting records of malicious domains into the normal dataset. In this way, it was possible to create datasets with various proportions (0.1%, 1%, 10%) of malicious domains, so strengthening the validation. It is important to note that this was not a random injection since we mixed datasets by respecting the time scale, *i.e.* only data of from the same week i from both malicious and normal datasets was mixed. Consequently, it was not always possible to reach the desired proportion of malicious domains due availability of normal data. However, this different proportion of malicious does not variate significantly to cause bias. This is equivalent to evaluating our approach in worst case conditions. In order to compare results to a baseline and contrast the variation in steadiness, the steadiness evolutions for purely malicious and for normal domains were calculated.

In Figure 5, the average steadiness is smoothed using a sliding window: the value at week i is the average from week $i - 2$ to week $i + 1$. The chart represents the average steadiness of the ten consecutive trees except for the first, since its steadiness cannot be calculated with respect to a previous week (according to equation (5)). The results for pure datasets (normal and malicious) are easy to distinguish. Although the normal dataset shows a fairly steady value around 0.85, the malicious one reveals variations between 0.65 and 0.83. Generally, as expected, the steadiness of trees constructed from

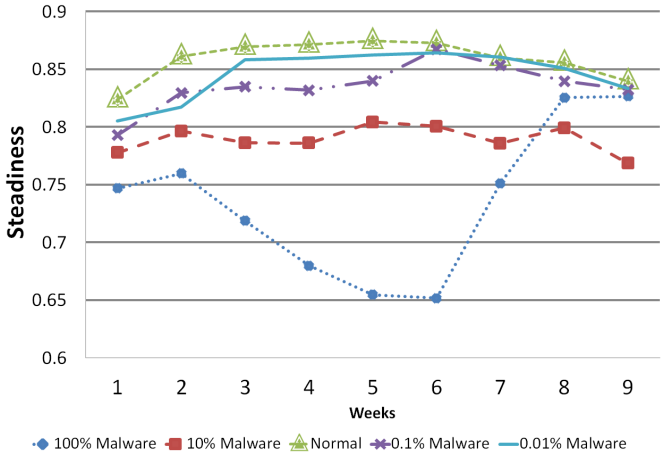

Fig. 5: Comparision of average steadiness for 10 weeks period

malicious domains is lower and varies considerably, meaning that the steadiness of malicious domains differs from one to another more that would be expected for normal ones.

When both malicious and normal domains are mixed together, steadiness is impacted and the values are logically lower than when only the normal dataset is used. Since data from malicious and normal domains can be aggregated into shared nodes, there is however no clear ordering between the curves, i.e. the curves may cross. But even with only 0.1% of malicious domains injected into the normal database, steadiness is lower than for the normal domains. Therefore, this experiment confirms the viability of steadiness, as we have defined it, as a discriminative feature for tracking malicious domains.

D. Macroscopic Steadiness during attacks

In the previous experiment, malicious domains clearly impacted steadiness, but the differences could be very low (see weeks 8,9 in Figure 5). However, a monitoring system should detect an anomaly when it occurs, whereas the previous experiment focused on the continuous presence of malicious data. Assuming a clean database to initialise the system (as we did with blacklists), this experiment shows that steadiness drops in the presence of malicious data.

The dataset used in this experiment was made by first aggregating 40 consecutive weeks, then every 5 weeks, malicious domains related DNS records were injected for the next 5 consecutive weeks. The periods where malicious domains are included are annotated as *Attack* in Figure 6.

It can be seen that there is a drop in average steadiness correlated when malicious domains are injected. At the right of Figure 6 the tail of the chart appears to remain stable, which seems to be due to a steadiness that has remained low even without malicious data (weeks 30 - 34). This might be interpreted as a false positive in the context of anomaly detection. It could be due to limited knowledge provided by blacklists, which cannot guarantee 100% coverage of all malicious domains in our passive DNS database.

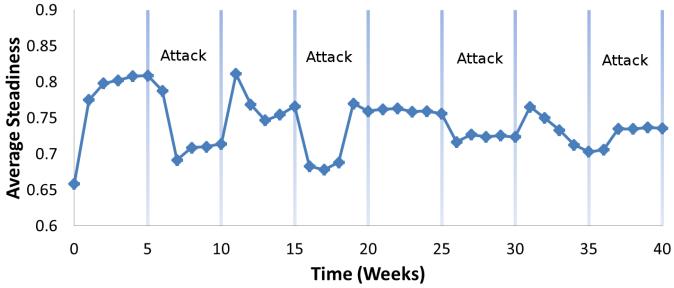

Fig. 6: Macroscopic evaluation of average steadiness for 40 weeks alternating DNS Records containing malicious domains every 5 weeks

E. Detection

Although the previous experiment allowed the detection of windows containing anomalies (malicious domains), detecting the exact anomaly is more interesting. The goal of the next experiment was to determinate an experimental threshold such that nodes having a steadiness lower than this threshold can be identified as malicious, since Figure 4 shows that most of nodes related to malicious domains are less steady than normal ones.

Due to the aggregation scheme, both abnormal domains and IP addresses can be detected. However, obtaining a representative blacklist of IP addresses is quite unfeasible as most of services offer limited online checking [5] or simply block entire countries or regions. Therefore, our evaluation relies on the blacklists we described in Section IV-A, which contain only DNS names.

As the blacklists only include the FQDNs of final hosts, there is no ground truth for entire domains. Hence, any assessment of our detection scheme can only be based on the *final domain* nodes, *i.e.* the nodes in the tree representing final names like `$.*.ROOT`. Such nodes mostly correspond to leaf nodes, which are usually collapsed into their ancestors during aggregation. Hence, α is set to zero to preserve entire trees and so track suspect domains at a specific time. Nevertheless, the steadiness of corresponding *final domain* nodes still takes the tree of the previous time window into account, possibly leading to computing the similarity in equation (1) with a non *final domain* node. Thus, when evaluating steadiness at time i, the previous trees are still aggregated ($\alpha > 0$). Therefore, the steadiness metrics retain the advantage of aggregation and provide a richer comparison than with either DNS names or IPv4 Addresses (as for example with pair-wise comparisons).

Figure 7 shows the cumulative distributions of final domain nodes according to the local steadiness. Supposing a steadiness threshold, the curves representing malicious and normal domains can be considered respectively as the True Positive Rate and False Positive Rate.

As shown in Figure 4, a steadiness of 0.5 seems to characterize malicious domains. Setting the threshold to 0.5 is equivalent in Figure 7 to detecting 73% of malicious

[5] As an example, the Spamhaus Project, www.spamhaus.org/

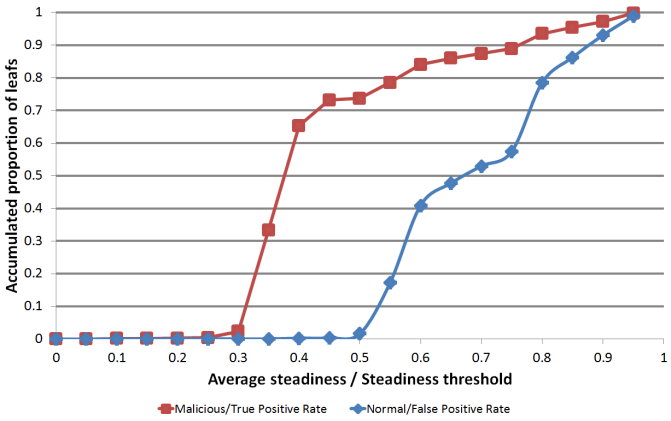

Fig. 7: Cumulative distribution of local steadiness of final domain nodes equivalent to the true positive and false positive rates assuming the x-axis as the threshold for considering a domain as normal

domains malicious domains with fewer than 1.6% of false positives. While 1.6% may represent too high a number of false positives, our approach can be considered as a means of selection suspicious domains for further checking with a more in-depth investigation like to MalwareDomains [6]. Furthermore, 0.45 is a better value according to Figure 7 as the same true positive rate (73%) can be achieved with only 0.3% of false positives. This represents a highly acceptable result.

F. Performance

Scalability was one of the design objectives for *MAM*. The reader can refer to [15] for details about the computational complexity of *MAM* itself (tree construction). To compute the steadiness of a node in a tree, the worst case may need to iterate over all nodes of the previous tree following the process described in section III-B (linear complexity), leading to a quadratic complexity if the steadiness of every node is evaluated (equivalent to pair-wise comparisons). Hence, the number of nodes is the main parameter which impacts scalability and this depends on the aggregation threshold α. Thus, a final experiment tests distinct aggregation thresholds.

We can observe in table II that the tree size decreases as α grows. Since the steadiness evaluation is performed after aggregation and depends on the size of the tree (number of nodes), this helps in improving scalability. In cases where scalability and memory consumption are critical factors, the tool is still usable, since the number of nodes after aggregation at the end of a time window is drastically reduced. The average tree size is around 2200 nodes with the default parameter used in previous experiments ($\alpha = 2\%$) which thus greatly limits the overhead when computing the steadiness, even though the complexity is quadratic in the worst case. For reference, the average number of distinct IP addresses and domains per week are respectively around 13000 and 5300.

[6]http://www.malwaredomains.com/

TABLE II: Average tree size

α	0%	2%	5%	10%
Average #nodes	84651	2238	1755	1525

Assuming now the previous case for detecting malicious domains (section IV-E), the number of comparisons is given by the number of leaf nodes (*final domain* names) multiplied by the number of nodes in the previous aggregated tree, which corresponds to an average of around 14400×2200 comparisons considerably fewer than a pair-wise comparison of leaf nodes (14000^2).

V. Related Work

Passive collection of DNS data was proposed in [18] and opened a new direction for several research projects. This appears to be relevant for tracking malicious domains. The approaches differ in the features and the algorithms used. This section provides a qualitative comparison. A quantitative analysis is not feasible due to the confidential nature of DNS data (user profiling) which thus cannot be shared publicly.

The authors in [5] reveal interesting features for tracking fast flux (number of IP addresses, name servers or Autonomous System Numbers) and especially for discarding CDNs (Content Delivery Networks). This is complementary to [21] which highlights facts and statistics regarding manually investigated malicious domains. Further features like the TTL (Time-To-Live) or the registrar might be provided to a Bayesian classifier [16]. In [22], a study is made of additional features, like the countries where hosts are located, the types of connection (cable, DSL, etc.) or the average uptime obtained by an active probing.

While these works focus exclusively on fast-flux, observing the registration details [23] and tracking the initial behaviour of a domain [7] also provide valuable information. Observing cache changes in multiple recursive servers is helpful in detecting cache poisoning attacks [24]. However, all the above approaches, except [21] which is mainly manual, require additional sources of information, which is gathered actively using common tools like *dig* or *whois*. Traceroute based probing is leveraged in [25] for fast-flux detection.

Like our approach, EXPOSURE [8] relies only on a pure passive DNS analysis to detect any kind of malicious activity. These can be reflected into domain features as this paper shows, and into further features like usage of numerical characters, or time-based features for detecting short life domains or daily patterns in requests. Additionally, looking for the longest meaningful words in domain is considered. In [26], the system requires accessing the DNS data of DNS servers near the top in the hierarchy (root and TLDs) to determinate IP addresses of recursive servers relaying requests of their clients. Some methods also take in account shared behaviours between hosts and domains [27], [14] and so cannot be used to analyse a single domain. Failure graphs are helpful for locating suspicious activities because they link domains and hosts from where requests originate. However, standard passive DNS

does not provide access to the request originator due to legal constraints. Studying user trends and in particular country-based behaviours is explored in [28].

N-grams have been analyzed to track misuse of DNS for tunneling [29] or fast-flux [30]. In [31], [32], an approach generating domain names based on lexical features and word semantics is proposed and used in [6], [33] to identify phishing domains. In fact, phishing detection usually relies partially on domain name features but may also consider many others offered by the full URL [34], [35] (length, IP address obfuscated in the URL, etc.).

All these works employ similar features related to the domain name itself and the associated IP addresses. Therefore, our paper focuses on these two areas. The aggregation which summarizes the information, may be used in related work (IP dispersion within a domain, the number of subdomains) without any predefined parameters like the level of domains being considered (X level domains) or the IP prefix to consider (/X networks). This remains the common approach, even in very recent work like [13]. In practice, such operations are necessary for scalability reasons, which is why this paper leverages an aggregation technique. Our aggregation is optimized, as the granularity is not defined a priori and can vary over the space in which both dimensions are considered together. Since, as shown in this paper, malicious domains often exhibit a higher variability in terms of IP addresses, our paper evaluates the dynamics of the aggregated data for detecting suspicious domains and/or IP addresses. From this perspective, [19] is a closely related work use manually fixed aggregation levels and does not aim to track suspect domains.

TreeTop [36] employs aggregation on domain names and IP addresses as well but differs from this paper because individual trees for each dimension are built before being linked together in order to analyse requests sent to blacklist services.

VI. Conclusion

In this paper, we leverage *MAM* to study the mapping between IP addresses and domain names. Thanks to the aggregation scheme, the volume of data to analyze is reduced and distributed behaviours can be identified more easily, which is helpful in the security context. It is useful in tracking entire harmful IP subnets or domain names while keeping the computational overhead low. We have defined a specific metric to study the steadiness of time series of trees produced by *MAM*. The evaluation on real data has proven its efficiency for detecting global abnormal changes (macroscopic steadiness) and for tracking specific domains and IP subnets which are responsible for them (local steadiness). Hence, our approach is helpful for detecting malicious domains. Scalability has also been assessed, since our tool leverage the analysis of aggregated data reducing the post processing time requirements. In future work, we plan to integrate other features into our scheme, such as countries and TTL values. Additionally, we plan to benchmark the performance of systems performing similar techniques such as [24], [21].

VII. Acknowledgment

Acknowledgement: this work was partially funded by IoT6, a European FP7 funded project under the grant agreement 288445 and MOVE, a CORE project funded by FNR in Luxembourg.

References

[1] P. Mockapetris, "RFC 1035: Domain Names - Implementation and Specification," 1987.

[2] N. Alexiou, S. Basagiannis, P. Katsaros, T. Dashpande, and S. A. Smolka, "Formal analysis of the kaminsky DNS cache-poisoning attack using probabilistic model checking," in *International Symposium on High-Assurance Systems Engineering (HASE)*. IEEE, 2010.

[3] D. Atkins and R. Austein, "RFC 3833: Threat Analysis of the DNS," 2004.

[4] J. François, S. Wang, R. State, and T. Engel, "Bottrack: tracking botnets using netflow and pagerank," in *NETWORKING 2011*. Springer Berlin Heidelberg, 2011, pp. 1–14.

[5] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Network and Distributed System Security Symposium (NDSS)*, 2008.

[6] S. Marchal, J. François, R. State, and T. Engel, "Proactive discovery of phishing related domain names," in *Recent Advances in Intrusion Detection*, ser. LNCS. Springer, 2012. [Online]. Available: http://lorre.uni.lu/~jerome/files/raid12.pdf

[7] S. Hao, N. Feamster, and R. Pandrangi, "Monitoring the initial DNS behavior of malicious domains," in *ACM SIGCOMM Internet Measurement Conference (IMC)*. New York, NY, USA: ACM, 2011.

[8] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis," in *Network and Distributed System Security Symposium - NDSS*, 2011.

[9] C. Wagner, G. Wagener, R. State, and T. Engel, "Malware analysis with graph kernels and support vector machines," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*. IEEE, 2009, pp. 63–68.

[10] J. Koziol, *Intrusion Detection with Snort*. Sams, 2003.

[11] F. Simmross-Wattenberg, J. Asensio-Perez, P. Casaseca-de-la Higuera, M. Martin-Fernandez, I. Dimitriadis, and C. Alberola-Lopez, "Anomaly detection in network traffic based on statistical inference and alpha-stable modeling," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, no. 4, 2011.

[12] H. J. Abdelnur, R. State, and O. Festor, "Advanced network fingerprinting," in *Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg, 2008, pp. 372–389.

[13] R. Perdisci, I. Corona, and G. Giacinto, "Early detection of malicious flux networks via large-scale passive DNS traffic analysis," *Transactions on Dependable and Secure Computing*, pp. 714–726, 2012.

[14] S. Marchal, J. François, C. Wagner, R. State, A. Dulaunoy, T. Engel, and O. Festor, "DNSSM: A large-scale Passive DNS Security Monitoring Framework," in *IEEE/IFIP Network Operations and Management Symposium*, 2012.

[15] L. Dolberg, J. Francois, and T. Engel, "Efficient multidimensional aggregation for large scale monitoring," in *Large Installation System Administration Conference (USENIX LISA)*. [Online]. Available: http://lorre.uni.lu/~jerome/files/lisapaperMAM.pdf

[16] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "Fluxor: Detecting and monitoring fast-flux service networks," in *International conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. Springer, 2008.

[17] G. C. Moreira Moura, R. Sadre, A. Sperotto, and A. Pras, "Internet bad neighborhoods aggregation," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, April 2012.

[18] F. Weimer, "Passive DNS replication," 2005.

[19] A. Berger and E. Natale, "Assessing the real-world dynamics of DNS," in *International conference on Traffic Monitoring and Analysis (TMA)*. Springer, 2012.

[20] C. Wagner, J. François, T. Engel *et al.*, "Danak: Finding the odd!" in *International Conference on Network and System Security (NSS)*. IEEE, 2011.

[21] B. Zdrnja, N. Brownlee, and D. Wessels, "Passive monitoring of DNS anomalies," in *International conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. Springer, 2007.

[22] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive DNS traces," in *Annual Computer Security Applications Conference (ACSAC)*, 2009.

[23] M. Felegyhazi, C. Kreibich, and V. Paxson, "On the potential of proactive domain blacklisting," in *Conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*. USENIX, 2010.

[24] M. Antonakakis, D. Dagon, X. Luo, R. Perdisci, W. Lee, and J. Bellmor, "A centralized monitoring infrastructure for improving DNS security," in *Recent Advances in Intrusion Detection*, ser. LNCS. Springer, 2010.

[25] H.-T. Lin, Y.-Y. Lin, and J.-W. Chiang, "Genetic-based real-time fast-flux service networks detection," *Computer Networks*, vol. 57, no. 2, pp. 501 – 513, 2013.

[26] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, II, and D. Dagon, "Detecting malware domains at the upper DNS hierarchy," in *USENIX Security*, 2011.

[27] H. Choi and H. Lee, "Identifying botnets by capturing group activities in DNS traffic," *Comput. Netw.*, vol. 56, no. 1, pp. 20–33, Jan. 2012.

[28] L. Deri, L. L. Trombacchi, M. Martinelli, and D. Vannozzi, "Towards a passive DNS monitoring system," in *Symposium on Applied Computing (SAC)*. ACM, 2012.

[29] K. Born and D. Gustafson, "Detecting dns tunnels using character frequency analysis," *Arxiv preprint arXiv:1004.4358*, 2010.

[30] S. Yadav, Reddy, A.K.K., Reddy, AL, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *ACM SIGCOMM Internet Measurement Conference (IMC)*. ACM, 2010.

[31] C. Wagner, J. François, R. State, T. Engel, A. Dulaunoy, and G. Wagener, "SDBF: Smart DNS Brute-Forcer," in *IEEE/IFIP Network Operations and Management Symposium - NOMS*, 2012.

[32] S. Marchal, J. François, C. Wagner, and T. Engel, "Semantic exploration of DNS," in *IFIP/TC6 Networking 2012*, Prague - Czech Republic, 2012.

[33] S. Marchal, J. François, R. State, and T. Engel, "Semantic based DNS Forensics," in *Workshop on Information Forensics and Security - WIFS*, IEEE, Ed., Tenerife, Spain, 2012. [Online]. Available: http://lorre.uni.lu/~jerome/files/wifs12_semantic1.pdf

[34] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: Url names say it all," in *INFOCOM*. IEEE, 2011.

[35] P. Prakash, M. Kumar, R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *INFOCOM*. IEEE, 2010.

[36] D. Plonka and P. Barford, "Context-aware clustering of DNS query traffic," in *Internet Measurement Conference (IMC)*. ACM SIGCOMM, 2008.