



# Query Log Analysis: Detecting Anomalies in DNS Traffic at a TLD Resolver

Pieter Robberechts<sup>1</sup>(✉), Maarten Bosteels<sup>2</sup>, Jesse Davis<sup>1</sup>, and Wannes Meert<sup>1</sup>

<sup>1</sup> Department of Computer Science, KU Leuven, Leuven, Belgium

`pieter.robberchts@cs.kuleuven.be`

<sup>2</sup> DNS Belgium vzw, Leuven, Belgium

**Abstract.** We present QLAD, an anomaly detection system that is designed for the high query volume and the specific nature of DNS traffic at a TLD resolver. QLAD integrates three components that implement the complete anomaly detection process, ranging from the ingestion of raw traffic data to the visualisation of detected anomalies. With an initial analysis of query logs from the Belgian ccTLD registry, we showed that QLAD can archive data compactly, has a low computational cost and can detect a wide range of anomalies. We found several anomalies that are of interest to the registry operator, such as domain enumerations and DoS attacks. Other anomalies were caused by benign applications with unique traffic patterns. A user interface helps to distinguish these, but correctly identifying all anomalies remains a difficult and tedious task.

**Keywords:** Anomaly detection · DNS · Internet security

## 1 Introduction

The Domain Name System (DNS) is an Internet service that translates domain names into IP addresses [19]. While the Internet's infrastructure is based on IP addresses, domain names are alphanumeric to make them easier to remember. Every time you use a domain name, a DNS service must translate the name to the corresponding IP address. For example, the domain name [www.example.com](http://www.example.com) might translate to `198.105.232.4`.

At its core, DNS is a distributed and hierarchical database. Each level corresponds to a part in the domain name separated by dots and uses so-called authoritative name servers to provide pointers to the next lower level. At the top of the DNS hierarchy are 13 root name servers, which contain pointers to the name servers for all the generic top-level domains (gTLD) such as `.com` and `.org` as well as country-specific top-level domains (ccTLD) such as `.uk`, `.be` or `.fr`. Each TLD is managed by a registry operator which is responsible for the registration of new domain names and the resolution of queries for these domains. The other levels in this hierarchy follow the same principle.

On one hand, these TLD registries are themselves an interesting target for attackers. Given that most of the communication within the Internet starts with

several DNS lookups, a successful attack on servers high up in the DNS hierarchy can have far-reaching consequences. This was illustrated by the Dyn cyberattack that took place on October 21, 2016. It involved multiple distributed denial-of-service (DDoS) attacks targeting systems operated by DNS provider Dyn, which caused major Internet platforms and services to be unavailable to large swathes of users in Europe and North America [20]. On the other hand, TLD registries can leverage their unique position high up in the DNS hierarchy to detect threats against other stakeholders within their zone. Many threats such as phishing campaigns, spam attacks and Command and Control (C&C) communication used by botnets can be observed in their DNS traffic.

The TLD registries' business model relies on a high availability and resilience of their DNS service, as well as the high reputation and trustworthiness of their domain. Thus, it is in the registries' interest to quickly identify attacks targeting or abusing their infrastructure, and to identify misuse of their domain names. Nevertheless, threat detection is still mainly a post-hoc task. Most malicious behaviour is only observed after it has a noticeable impact on the service.

Using the Belgian TLD registry (DNS Belgium) as a use case, we will illustrate how proactive and real-time analysis of the continuous stream of DNS query logs can contribute to the resilience and security of a TLD registry's service. This paper will discuss the design and implementation of a query log analysis platform called *QLAD* (Query Log Anomaly Detection) that is able to detect attacks and other suspicious behaviour at a TLD resolver in near real time. *QLAD* performs a first set of attack and anomaly detection functions and it offers an interface for reporting and inspecting the detected anomalies. All code is open source and available at <https://github.com/DNSBelgium/qlad>.

To summarize, this paper makes the following contributions: (i) Highlight a number of data science challenges that we encountered while solving this problem; (ii) Discuss two statistical anomaly detection algorithms that jointly detect a wide range of threats in DNS query logs; (iii) Propose an architecture for storage, analysis and presentation of DNS query logs; (iv) Validate the approach on a real-world data set.

## 2 Related Work

SIDN Labs, the research unit of the Netherlands' TLD registry, were the first to present a complete framework that enables registries to increase the security and stability of their TLD [11]. However, compared to the approach presented in this work, they focus on the detection of domain names used in malicious activities instead of the detection of attacks against the DNS infrastructure. The operator of the .uk TLD developed Turing [21], a system that appears to be similar. However, Turing is a commercial solution and there is little publicly available information about its functionality and technical implementation.

Furthermore, there is scattered prior work on individual components for data storage, anomaly detection and threat mitigation. Traffic to ccTLD name servers produces gigabytes of traffic data each day. On the one hand, researchers have

resorted to Hadoop-based solutions [3, 4, 16, 32] as a way to address the high volume of data. Such cluster solutions are designed for long-term data storage and to support interactive queries on the stored data. On the other hand, tools such as the DNS Statistics Collector (DSC) [9] aggregate the data before archiving, reducing it to a size which can be stored in a traditional database.

Existing research about detecting the discussed attack vectors in the upper levels of the DNS hierarchy is limited. Most research focuses on anomaly detection in recursive resolvers and small authoritative resolvers [1, 23, 30]. Mostly, these methods do not scale to the level of a ccTLD. A couple of simple ideas are applicable at the TLD level, however. These methods are all based on the detection of changes in the frequency of packets with certain attributes [12, 13, 33]. For example, Karasaridis et al. [12] uses cross entropy to detect significant changes in the distribution of packet sizes. Considerably more research has been done on the broader scope of network anomaly detection [6]. These methods search for anomalies in IP or TCP traffic, but some can be applied on DNS traffic too [8].

### 3 Detecting Anomalies in DNS Server Logs

The core problem which we aim to solve is anomaly detection in a continuous stream of DNS query logs. We claim that due to the variety of types of anomalies this problem cannot be solved with a single approach. On the one hand, we need to analyse individual traffic flows to enable the detection of low-volume anomalies in the continuous stream of valid traffic. On the other hand, we need to look at global traffic patterns to detect anomalies that span multiple flows.

In this section, we firstly discuss the challenges posed by the task at hand and secondly introduce two complementary anomaly detection algorithms by which we address these challenges, referred to as respectively *QLAD-flow* and *QLAD-global*. Finally, we describe how to integrate these anomaly detection algorithms in a complete anomaly detection framework.

#### 3.1 Challenges

Analysing DNS server logs to identify anomalies poses a number of significant and non-trivial data science challenges.

**Volume of data.** Each server has to process about 1,000 queries per second.

This has two important consequences. First, efficiency and scalability are crucial: any algorithms used for analysis should have a low computational cost and should scale well to increasing traffic volumes. Secondly, due to the high volume of valid traffic, it is easy to mask malicious traffic.

**No labelled data.** A second challenge is the lack of labelled or clean training data. We have only access to the raw server logs. It is practically infeasible to manually label all anomalies in these logs. They are hidden between thousands of valid queries and one anomaly may correspond to thousands of unique queries, each valid on their own (*e.g.*, a DoS attack with spoofed

source IP addresses). Consequently, we can not use any supervised machine learning algorithms to learn a model for anomalous or valid traffic patterns. Moreover, any other algorithm should be robust enough to deal with a certain amount of unidentified anomalies in the training data. Also, evaluating anomaly detectors is difficult, since we can not compare our results to a ground truth.

**Adversarial setting.** There is a wide range of possible attack vectors, which are under constant evolution. Attackers have often successfully modified their attack vectors to circumvent new security patches [28] and extensions to the DNS protocol have introduced new vulnerabilities [2]. A straightforward and popular approach to solve the problem is the signature-based technique [17], which defines a separate model (signature) for each popular attack. This technique requires prior knowledge and new types of attacks can go undetected [17]. Moreover, generating a good signature for an anomaly is difficult [10]. The signatures should be general enough to capture small variations on the targeted attack vector, while being specific enough to allow valid traffic. Such models are often easy to mislead with a small change to the attack method [7].

**Nature of DNS traffic.** DNS traffic is subject to both periodic and slow varying trends in terms of the volume, type, origin and content of queries. The number of user generated DNS queries decreases at night, over the weekend and during holidays. Administrative queries like SOA type queries are less affected by these factors. Moreover, apart from trends in the global amount of traffic, individual domains often have unique traffic patterns. Furthermore, the unique characteristics of the DNS data flow, such as small in packet size and little in message amount, make it more difficult to distinguish anomalous behaviours from normal ones [33].

**Packet spoofing.** An attacker can easily replace some fields in the DNS queries by random values. This will make it more difficult to identify anomalies, since all queries will seem unrelated.

### 3.2 QLAD-flow

The first algorithm's goal is to identify low volume anomalies. Therefore, it splits the traffic into different flows, for example based on the source IP of the DNS query. The partitioning makes it possible to detect anomalies which are otherwise hidden in the global flow of traffic. We employ the statistical anomaly detection approach proposed by Dewaele et al. [8]. We first briefly describe this method for completeness, and next highlight precisely what is different.

The algorithm analyses a continuous stream of packets within a sliding time-window of duration  $T$ . Each packet is identified by its time of arrival and a set of packet attributes (*i.e.*, source IP, destination IP, source port and destination port). First, all packets within a window are hashed to  $N$  buckets, using the packet attributes as hashing keys. Second, for each bucket, the algorithm counts the number of packets that arrive during a short aggregation period. This is done for multiple aggregation levels, such that each bucket is transformed to multiple

series of packet counts. Each of these series is modelled by Gamma distributions  $\Gamma_{\alpha,\beta}$ . Then, we can estimate the average value and variance for the  $\alpha$  and  $\beta$  parameters for each aggregation level and identify the buckets for which the  $\alpha$  and  $\beta$  values deviate more than a given threshold. Finally, the algorithm repeats this procedure with different universal hash functions. Each hash function will result in a different set of anomalous buckets. The intersection of all these buckets should correspond to the set of all anomalous packets.

The method was originally designed to detect low-intensity anomalies in the TCP/IP layer [8]. CZ.NIC, the Czech domain registry, implemented and slightly modified the algorithm for the purpose of DNS traffic monitoring [18]. They implemented a set of modules, called policies, in order to test various packet identifiers. Such a policy defines which packet features are used as the hashing keys for dividing packets into sketches. CZ.NIC designed and implemented<sup>1</sup> the first two policies, we added the third.

- 1. IP Address Policy** [18] uses only the source IP address as the hash key. It is based on the original algorithm, which uses the whole TCP/IP connection identifier (source IP, destination IP, source port, destination port) as the hash key. However, destination address and destination port show little to no variability in TLD DNS traffic.
- 2. Query Name Policy** [18] is based on application layer data. It extracts the first domain name from the DNS query and uses it as the hash key.
- 3. ASN Policy** is a generalization of the *IP Address Policy*. For each source IP, it first looks up its autonomous system number (ASN). Each network on the Internet is uniquely defined by such an ASN. Therefore, this policy should be able to detect anomalies which are linked to a network, rather than an individual server.

### 3.3 QLAD-global

*QLAD-flow* will fail to detect attacks that use random spoofed IP addresses, since each packet will belong to a different flow. Especially DoS attacks often use this technique and will therefore remain undetected by *QLAD-flow*. *QLAD-global* tries to address this issue by looking at the global traffic flow.

The method is based on the observation that all common traffic anomalies cause changes in the distribution of one or more traffic features. In some cases, feature distributions become more concentrated on a small set of values; for example the distribution of source IP addresses during a reflection attack on a couple of servers in the same network. In other cases, feature distributions become more dispersed; for example when source addresses are spoofed in a DoS attack, or during a zone enumeration attack when lots of domains are queried only once. An analysis based on these traffic feature distributions can capture fine-grained patterns in traffic distributions that simple volume based metrics cannot identify [15].

---

<sup>1</sup> Project repository can be found at [git://git.nic.cz/dns-anomaly/](https://git.nic.cz/dns-anomaly/).

In this paper, we focus on eight traffic features: the number of requests for each TLD and SLD (second level domain name), the query types, the response codes, the number of requests by each client, ASN and country, and finally the response sizes. Obviously, these are not the only fields that could be used to detect anomalies, but they are general enough to encompass most other fields and we found that they suffice to detect the most common anomalies.

The distribution of these traffic features is high dimensional, and so is hard to work with directly. Therefore, we perform anomaly detection on the entropies of these feature distributions instead. The entropy is defined as:

$$H(X) = - \sum_{i=1}^n p(x_i) \log(p(x_i)).$$

Here,  $X$  is a feature that can take values  $\{x_1, \dots, x_n\}$  and  $p(x_i)$  is the probability mass function of outcome  $x_i$ . Entropy is a measure of the uncertainty associated with a random variable [24]. Although, in this context, it can be interpreted as a measure for the degree of concentration or dispersal of a distribution [15].

Besides concentration and dispersal of the underlying distribution, the entropy also depends on the number of distinct values  $n$ . In practice, this means that entropy tends to increase when the traffic volume increases [15]. This is an advantage if normal traffic volumes are constant, since it allows the detection of volume anomalies with entropy measures. However, in our application, traffic volume is highly periodic, so we mitigate the effect of this phenomenon by normalizing entropy values. Therefore, we divide by  $\log(n)$  (the maximum entropy), as proposed by Nychis et al. [22].

Figure 1 illustrates that entropy can be effective for anomaly detection with a simple example. The left plot shows the distribution between queries for valid (NOERROR) and non-existent domain names (NXDOMAIN) during 100 successive one-minute windows. The right plot shows the entropy values for these distributions. During two short periods (around minute 90 and around minute 125), the percentage of NXDOMAIN requests increases significantly, i.e., the distribution of response code values becomes more dispersed. These anomalies cannot be observed in the global traffic volume, but they stand out clearly on the right plot. The entropy increases, corresponding to a distributional dispersion around the NOERROR and NXDOMAIN response codes.



**Fig. 1.** Two anomalies in the ratio between DNS queries for valid and non-existent domain names over time (left), viewed in terms of entropy (right). Transforming the distribution to an entropy time series allows automated detection with standard anomaly detection techniques.

For each of the eight traffic features, we compute the entropy of the distribution in one-minute windows. This reduces the problem of anomaly detection in DNS traffic to a problem of anomaly detection in (correlated) time series. The research community has defined a wide range of algorithms to solve this type of problem [29]. One popular approach compares windows or individual points with a pre-trained profile of legitimate traffic [5]. Due to the lack of clean training data, we can not apply this approach to our problem. Alternatively, the subspace method [15] uses principal component analysis to detect unusual variations in the correlation between multiple time series. This method gave poor results, due to the high level of periodicity and unequal effects of this periodicity on each entropy time series. Therefore, we apply anomaly detection separately on each time series. There exists algorithms that should be able to learn the periodicity in these time series, but since the time series are influenced by both the hour of the day and the day of the week and since the data is contaminated with anomalies, these algorithms would require data of several weeks to learn a good model [31].

Despite its simplicity, we found a simple Exponential Moving Average (EMA) filter to work best. To quantify the range around the EMA that could be considered normal, we compute the Exponential Moving Standard Deviation (EMS):

$$\text{EMS}_t = \sqrt{w * \text{EMS}_{t-1}^2 + (1 - w) * (y_t - \text{EMA}_t)^2} \quad (1)$$

An anomaly is reported when  $|y_t - \text{EMA}_t| > n * \text{EMS}_t$ , where  $n$  controls the sensitivity of the alarm to the entropy measurement  $y_t$ .

## 4 QLAD Architecture

Our goal is to develop an anomaly detection system that is tuned to the unique nature of DNS traffic. Ideally, the system has the following characteristics:

**Accurate.** It should be able to detect suspicious and unexpected behaviour.

This requires being sensitive to a wide range of possible attacks, including low volume anomalies.

**Efficient.** It should be able to handle high volumes of traffic and perform its analysis in near real time.

**Unsupervised.** It does not need any initial knowledge about the analysed traffic in the form of labelled data or a description of the possible attack vectors.

**Provide insight.** It should allow the operator to pinpoint the cause of an anomaly and determine the correct countermeasures.

**Extendable.** The architecture should enable future growth and improvements to the system.

The *QLAD* system was designed with these requirements in mind. Figure 2 displays its modular architecture, which can be divided into three layers: a data transformation, anomaly detection, and presentation layer. The transformation layer is a crucial component. Storing the raw pcap files is undesirable because

they have a large storage cost and are inefficient to query. Therefore, we considered two alternative solutions for preprocessing and storing these pcap files.

The first solution, based on the DNS Statistics Collector (DSC) [9], transforms the pcap files into time aggregated traffic features such as the number of queries by query type, the most popular domains queried, and the length of DNS reply messages. The aggregated data is small enough that it can be directly stored in a traditional database system. This means that this component can be deployed easily and at a very low cost in terms of infrastructure. The aggregated data suffices to detect anomalies, but does not allow a detailed manual analysis.

The alternative, ENTRADA [32], is a big data platform designed to ingest and quickly analyse large amounts of network data. It was built by SIDN Labs, the research department of the Dutch domain registry, to enable fast data analysis on the network traffic of their authoritative name servers. ENTRADA is built entirely on open-source tools, and was open sourced itself in January 2016 [26]. The ENTRADA platform continuously processes pcap files in order to convert them to Apache Parquet files (a columnar storage format), which are ultimately stored in a Hadoop cluster (HDFS). Once archived, these files are available for analysis via Impala<sup>2</sup> [25].

Each approach has its own advantages and disadvantages. Effectively, the choice between both methods is a trade-off between deployment costs and more extensive data analysis possibilities.

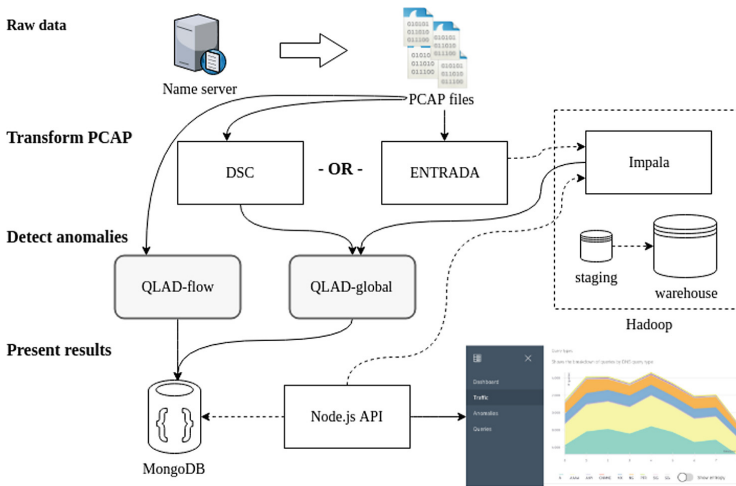


Fig. 2. A global overview of the QLAD system.

<sup>2</sup> Impala provides an SQL-like interface to query data stored in a HDFS. See <https://impala.incubator.apache.org/overview.html> for more information.



## 5 Empirical Evaluation

We evaluated *QLAD* on two days of real-world traffic to one of DNS Belgium’s servers. The absence of a ground truth makes evaluating anomaly detectors notoriously difficult [27]. Since manually identifying all true anomalies in the traffic is infeasible, it is impossible to determine which anomalies are missed by the detectors. Hence, we cannot compute metrics like recall. The research community takes two approaches to address this problem [14]. A first approach is to manually inspect each detected anomaly in order to pinpoint its root cause. This approach fits best with a real-world application of the detector. A second approach injects the data with synthetically generated attacks. This approach has the advantage that the parameters of an attack (duration, traffic volume, etc.) can be carefully controlled, which enables sensitivity analysis of the detection algorithm.

With *QLAD* we focussed on the detection of a wide range of anomalies. Our primary interest is to find out which type of real-world anomalies *QLAD* is able to detect. Therefore, we use the first approach. An evaluation with the second approach is left as future work. Specifically, we address the following questions:

- Q1:** What are the space/time trade-offs between DSC and ENTRADA storage?
- Q2:** What is the runtime performance of the anomaly detectors?
- Q3:** Which types of real-world anomalies can *QLAD* detect?

### 5.1 Q1: Data Preprocessing and Storage

The traffic used for evaluation originates from one of DNS Belgium’s name servers, collected between Sunday 12 February and Monday 13 February 2017. Traffic was captured for 44h and amounts to 58,345,819 queries or 42 GB of pcap packet dumps. We processed these logs with both ENTRADA and DSC. ENTRADA reduces the original logs to 3.8 GB of Parquet files, which is about 9% of the original volume. Note, however, that—given the default replication factor of three in Hadoop—you still need 11.4 GB of actual storage space. With DSC, the volume is reduced to 39 MB, which is about 0.09% of the original size.

### 5.2 Q2: Anomaly Detection Performance

Both *QLAD-flow* and *QLAD-global* are computationally very efficient and capable of analysing real-time traffic. *QLAD-flow* analyses the complete dataset in 3 m 36 s (source IP policy), 5 m 13 s (query name policy) and 21 m 45 s (ASN policy).<sup>3</sup> We used a window size of 10 min, 8 aggregation levels, 25 hash functions, a hash table of size 32 and 1.2 as a detection threshold. The same parameters are used in further experiments. For a more extensive performance analysis with varying parameters, we refer to Mikle et al. [18].

Using the DSC setup, *QLAD-global* can analyse the full dataset in 1 m 18 s. Its performance is mainly determined by the time needed to fetch the data from

<sup>3</sup> 2.3 GHz dual-core Intel Core i5 processor with 8 GB RAM.

the MongoDB database. We do not count the time needed to extract the traffic feature distributions from the query logs.

### 5.3 Q3: Classification of Detected Anomalies

Table 1 shows the total number of anomalies detected by each anomaly detection algorithm and our expert’s classification. We tried to group anomalies together as much as possible. For example, when the same attack is launched from multiple IP addresses, we count it as only one anomaly. Furthermore, there was a large overlap between the anomalies detected by each detector. We write the number of anomalies that are not detected by any other detection method in parentheses. *QLAD-flow* with the ASN policy detected a subset of the anomalies detected with the source IP policy, but no new anomalies.

**Table 1.** A manual classification of the detected anomalies.

Anomaly type	<i>QLAD-flow</i> (source IP)	<i>QLAD-flow</i> (query name)	<i>QLAD-global</i>	Total
Benign				
Caching resolver	12 (10)		2	12
Email marketing	8 (7)		2 (1)	9
Other	1 (1)	2 (2)		3
Malicious				
Spam sender	3 (3)			3
Domain enumeration	5 (3)		2	5
Reflection attack	1		1	2
Phishing	1 (1)			1
DoS attack	3 (1)	2 (1)	1	4
Unknown	1		1	1
Total	35 (26)	4 (3)	9 (1)	39

In total, we found 39 unique anomalies, but only a minority of them is interesting for a TLD operator. *QLAD-flow* pinpoints flows which are statistically different from the other flows in some time window. Not all of these flows are malicious; there are some benign applications with unique traffic patterns too. Although, they are anomalous in some sense, they are not interesting for a TLD operator. Examples of such applications we found include caching resolvers, email marketing services, a cloud provider, security appliances and a cryptocurrency mining pool.

Other anomalies were clearly malicious. We found something that looked like a reflection attack on Twitter, several domain enumeration attacks, three servers sending spam and four DoS attacks (although not large enough to have

a real impact). Furthermore, one server generated bizarre queries for domains such as [www-axa-fr.be](http://www-axa-fr.be) and [www-keytradebank.be](http://www-keytradebank.be). Since these refer to Belgian and French banks, these queries were probably generated by a script looking for unused domain names to be used in phishing.

To conclude, we verified whether any of the IP addresses corresponding to the anomalies we discovered was reported previously for engaging in malicious activity.<sup>4</sup> Two were reported for engaging in DDoS attacks and two others for spreading spam. For all four, this corresponds to the anomalies we observed.

## 6 Conclusions

We presented the design, implementation and evaluation of the *QLAD* framework—a proof of concept anomaly detection system for query logs of a TLD resolver. We showed how existing solutions can be integrated with custom development, to create a system that is adapted to the particular nature of DNS traffic at a TLD resolver and that can detect a wide range of anomalies in near real-time.

**Acknowledgements.** The authors acknowledge the partial support of KU Leuven Research Fund C14/17/070 and C22/15/015 (PR and JD), FWO-Vlaanderen SBO-150033 (JD and WM) and Interreg V A project NANO4Sports (PR and JD).

## References

1. Alonso, R., Monroy, R., Trejo, L.: Mining IP to domain name interactions to detect DNS flood attacks on recursive DNS servers. *Sensors* **16**(8), 1311 (2016)
2. Ariyapperuma, S., Mitchell, C.J.: Security vulnerabilities in DNS and DNSSEC. In: Proceedings of the 2nd International Conference on Availability, Reliability and Security, pp. 335–342. IEEE Computer Society, Washington, DC (2007)
3. Baker, M.: Packetpig - open source big data security analysis. <http://blog.packetloop.com/2012/03/packetpig-open-source-big-data-security.html>
4. Bär, A., Finamore, A., Casas, P., Golab, L., Mellia, M.: Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis. In: 2014 IEEE International Conference on Big Data, pp. 165–170, October 2014
5. Bereziński, P., Jasiul, B., Szpyrka, M.: An entropy-based network anomaly detection method. *Entropy* **17**(4), 2367–2408 (2015)
6. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: methods, systems and tools. *IEEE Commun. Surv. Tutor.* **16**(1), 303–336 (2014)
7. Cheng, T.H., Lin, Y.D., Lai, Y.C., Lin, P.C.: Evasion techniques: sneaking through your intrusion detection/prevention systems. *IEEE Commun. Surv. Tutor.* **14**(4), 1011–1020 (2012)
8. Dewaele, G., Fukuda, K., Borgnat, P., Abry, P., Cho, K.: Extracting hidden anomalies using sketch and non Gaussian multiresolution statistical detection procedures. In: Proceedings of the 2007 Workshop on Large Scale Attack Defense, pp. 145–152. ACM, New York (2007)

---

<sup>4</sup> We used the AbuseIP database (<https://www.abuseipdb.com>).

9. DNS-OARC: The DNS Statistics Collector, February 2016. <https://www.dns-oarc.net/tools/dsc>
10. Gascon, H., Orfila, A., Blasco, J.: Analysis of update delays in signature-based network intrusion detection systems. *Comput. Secur.* **30**(8), 613–624 (2011)
11. Hesselman, C., Moura, G.C.M., Schmidt, R.d.O., Toet, C.: Increasing DNS security and stability through a control plane for top-level domain operators. *IEEE Commun. Mag.* **55**(1), 197–203 (2017)
12. Karasaridis, A., Meier-Hellstern, K., Hoefflin, D.: NIS04-2: detection of DNS anomalies using flow data analysis. In: *IEEE Globecom 2006*, pp. 1–6 (2006)
13. Kreibich, C., Warfield, A., Crowcroft, J., Hand, S., Pratt, I.: Using packet symmetry to curtail malicious traffic. In: *Proceedings of the 4th Workshop on Hot Topics in Networks* (2005)
14. Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. *SIGCOMM Comput. Commun. Rev.* **34**(4), 219–230 (2004)
15. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. *SIGCOMM Comput. Commun. Rev.* **35**(4), 217–228 (2005)
16. Lee, Y., Lee, Y.: Toward scalable internet traffic measurement and analysis with hadoop. *SIGCOMM Comput. Commun. Rev.* **43**(1), 5–13 (2012)
17. Li, Z., Das, A., Zhou, J.: USAID: unifying signature-based and anomaly-based intrusion detection. In: Ho, T.B., Cheung, D., Liu, H. (eds.) *PAKDD 2005*. LNCS (LNAI), vol. 3518, pp. 702–712. Springer, Heidelberg (2005). [https://doi.org/10.1007/11430919\\_81](https://doi.org/10.1007/11430919_81)
18. Mikle, O., Slany, K., Vesely, J., Janousek, T., Survy, O.: Detecting hidden anomalies in DNS communication. Technical report, CZ.NIC (2011)
19. Mockapetris, P.: Domain names - concepts and facilities. STD 13, RFC Editor, November 1987. <http://www.rfc-editor.org/rfc/rfc1034.txt>
20. Newman, L.H.: What we know about Friday’s massive internet outage, October 2016. <http://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/>
21. NOMINET: NOMINET adds machine learning to Turing network analytics and monitoring tool, February 2017. <https://www.nominet.uk/nominet-adds-machine-learning-to-turing-network-analytics-and-monitoring-tool/>
22. Nychis, G., Sekar, V., Andersen, D.G., Kim, H., Zhang, H.: An empirical evaluation of entropy-based traffic anomaly detection. In: *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, pp. 151–156 (2008)
23. Satam, P., Alipour, H., Al-Nashif, Y., Hariri, S.: Anomaly behavior analysis of DNS protocol. *JISIS* **5**(4), 85–97 (2015)
24. Shannon, C.E.: A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.* **5**(1), 3–55 (2001)
25. SIDN Labs: ENTRADA Documentation. <http://entrada.sidnlabs.nl/docs/introduction/overview/>
26. SIDN Labs: SIDN Labs Open-Sources ENTRADA. <https://www.sidnlabs.nl/a/weblog/sidn-labs-open-sources-entrada>
27. Silveira, F., Diot, C., Taft, N., Govindan, R.: ASTUTE: detecting a different class of traffic anomalies. In: *Proceedings of the ACM SIGCOMM Conference*, pp. 267–278 (2010)
28. Son, S., Shmatikov, V.: The Hitchhiker’s guide to DNS cache poisoning. In: Jajodia, S., Zhou, J. (eds.) *SecureComm 2010*. LNICSSITE, vol. 50, pp. 466–483. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-16161-2\\_27](https://doi.org/10.1007/978-3-642-16161-2_27)
29. Teng, M.: Anomaly detection on time series. In: *2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 1, pp. 603–608 (2010)

30. Trostle, J., Van Besien, B., Pujari, A.: Protecting against DNS cache poisoning attacks. In: 6th IEEE Workshop on Secure Network Protocols, pp. 25–30 (2010)
31. Vallis, O., Hochenbaum, J., Kejariwal, A.: A novel technique for long-term anomaly detection in the cloud. In: Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing, pp. 15. USENIX Association, Berkeley (2014)
32. Wullink, M., Moura, G.C.M., Muller, M., Hesselman, C.: ENTRADA: a high-performance network traffic data streaming warehouse. In: 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 913–918. IEEE, April 2016
33. Yuchi, X., Wang, X., Lee, X., Yan, B.: A new statistical approach to DNS traffic anomaly detection. In: Cao, L., Zhong, J., Feng, Y. (eds.) ADMA 2010. LNCS (LNAI), vol. 6441, pp. 302–313. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17313-4\\_30](https://doi.org/10.1007/978-3-642-17313-4_30)