# Identifying the True Origin of DNS Traffic Without Reference to Client Source Address

Joe Abley
Western University, London, Ontario, Canada
Afilias Canada, Toronto, Ontario, Canada
jabley@uwo.ca, jabley@afilias.info

***Abstract*—We demonstrate a model that is able to classify the originating system respobsible for stateless Domain Name System (DNS) traffic received at an authoritative DNS server without reference to source address. The ability to determine whether particular DNS query traffic received at an authoritative server is legitimately sourced from a particular client system is useful in identifying various classes of malicious traffic in production DNS systems.**

## I. Introduction

The Domain Name System (DNS) includes a wire protocol with which structured requests and responses are exchanged over a network. The DNS protocol was originally specified [1] [2] for use over both the Transmission Control Protocol (TCP) [3] and the User Datagram Protocol (UDP) [4] and the use of other transports have also been documented [5] [6] [7]. At present, however, UDP is the overwhelmingly dominant transport protocol in use; for example, according to statistics published by ICANN for queries received at the L root server, UDP accounts for 98% of all queries received[1].

Since UDP transport for DNS is stateless, consisting of single-datagram queries and responses with no setup or tear-down handshake, there are limited opportunities to verify the legitimacy of a source address. DNS servers are consequently frequently used as amplifiers in reflection attacks [8]. Although some such attacks are trivially identified, e.g. by Query Type (QTYPE), many are more difficult. By choosing query parameters that match legitimate, real-world use of the DNS, attackers can make it difficult for their traffic to be identified and blocked without causing collateral damage. This is especially true of amplification attacks against DNS resolvers.

The clients of authoritative DNS servers are most usually DNS resolvers. These client systems receive requests from end-user applications (or downstream resolvers). Different client resolver systems are observed to send different mixes of DNS traffic; for example, a resolver system that mainly serves end-users will send a different mixture of queries to authoritative servers than one which serves a specific application like Internet mail [9], which might reasonably be expected to have a much higher proportion of query traffic with `QTYPE=MX`.

[1]http://stats.dns.icann.org/plotcache/L-Root/transport_vs_qtype/2018-12-03T00:00-2018-12-03T23:59-all.html

Afilias Canada[2] operates authoritative DNS infrastructure for around 300 top-level domains, including several that attract high levels of query traffic such as INFO and ORG. This infrastructure is distributed globally using anycast service distribution [10], using commodity transit services, public peering and so-called Private Network Interconnects (PNIs). The real origin of queries received over a PNI can be known with high accuracy; the origin of queries received over the Internet, in contrast, cannot. We refer to the former as *trusted* paths, and the latter as *untrusted*. Trusted paths exist to Google Public DNS[3], a public DNS resolver system configured for use by a large number of end-users, and Facebook[4], whose resolver systems are mainly used by back-end systems that build previews for links shared between users of Facebook's social media platform. The traffic patterns of each are expected to be usefully different.

While real-time anomoly detection in DNS traffic remains an elusive problem, the ability to classify traffic apparently received by particular sources as being legitimate is useful in the forensic analysis of traffic spikes since it provides the opportunity to distinguish between illegitimate, unwanted traffic and traffic from clients that just happen to be busy, e.g. due to a burst in popularity in a particular web page, or changes in the Time To Live (TTL) parameters of high-use domain names. This paper describes a system that aims to provide such a classification.

A raw DNS dataset is collected in the form of individual (request, response) DNS messages received from and sent to a single apparent source over period of two weeks. We split the resulting query stream into five minute intervals and from each we extract a vector of variables that describes the traffic received from each client during that time. Each such vector, once normalised, represents a single observation related to a single client. Observations that correspond to traffic received from trusted sources can be used as a training dataset. Observations corresponding to DNS traffic that definitively did not arrive from a trusted source can also be incorporated as "other". The resulting model can be used to classify five-minute samples of query streams from purported single sources to classify the

[2]https://afilias.info/
[3]https://dns.google.com
[4]https://www.facebook.com

origin of the query traffic as “Facebook”, “Google” or “Other”. Since query sources for each category feature in an equal number of traffic samples, it is straightforward to produce a training dataset that is balanced across the three categories.

This paper is organised as follows. Section I introduces the problem and provides some high-level background on the DNS. Section II provides a short introduction to the algorithms and accuracy measures that are used to build the model. Some other work on applying machine learning techniques to problems in the DNS are described in section III. Data collection and preprocessing, feature engineering and choice of learning and validation algorithms are discussed in section IV. Section V describes the evaluation of the resulting model. Section VI provides a summary of the work described in this paper, and section VII identifies some areas for future study.

## II. Background

Two multiclass classifiers are evaluated for this model in section II-A, below. The approach used to evaluate the accuracy of each is described in section **??**. These models are used to classify features of individual five-minute samples of DNS reponse data according to source system by treating each sample as a single observation. A brief discussion of other approaches that might usefully consider each sample as a point along a time series can be found in section VII.

### A. Classifier Models

We consider both Support Vector Machine and Random Forest models and select the most successful one based on 10-fold validation.

*1) Multiclass Support Vector Machine:* The classifier used in this paper was constructed as a series of Support Vector Machines (SVM), each used as a binary classifier. SVM represents $n$-dimensional support vectors in an $n$-axis hyperspace and identifies a hyperplane boundary between observations known to be in different categories to facilitate classification of unlabelled test sets. Those boundaries can then be used to classify unlabelled observations.

Multiclass classification is achieved using $k(k-1)/2$ *one-against-one* binary classifiers combined with a max-wins voting scheme, as discussed in [11].

The SVM implementation used to construct this model exposes several hyperparameters that can be tuned, as well as a native grid search to assist identification of optimal parameters for a supplied validation dataset.

*2) Random Forest:* Random Forests (RF) [12] combine many decision trees at training time into an ensemble learning model. RF uses bootstrap samples to introduce a random component into the tree-building process, whilst also reducing correlation amongst trees, adding noise to perturb the tree structure and using a random subset of available predictors each each split. Each of $m$ models in the resulting ensemble is used to generate a prediction for a new sample and those predictions are averaged to give the prediction from the entire forest.

The tuning parameters for the RF ensemble model are:

- the size of the subset of predictors randomly selected at each split, $m_try$. Breiman suggests setting $m_try$ to be one third of the number of predictors.
- the number of trees in the forest. Breiman has proved that random forests are immune from overfitting, but the accuracy benefit:computational cost is expected to decrease as the forest becomes larger.

### B. Accuracy Measures

We calculate a confusion matrix over a test dataset:

| | | Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual | Positive | TP | FN |
| | Negative | FP | TN |

Since we intend to ensure that we have a balanced dataset between the three classifications of traffic samples, we are able to use a straightforward measure of accuracy, A:

$$A = \frac{TP + TN}{TP + FP + TN + FN}$$

Without a particular application for the models, it is difficult to assess the relative importance of false positives or false negatives in our model. However, as a kindness to a reader with a business case in mind, we calculate the precision, P, the recall, R and the specificity, S:

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$S = \frac{TN}{TN + FP}$$

As is conventional, we also calculate the F1 score, F, as the harmonic mean of the precision and recall metrics:

$$F = \frac{2PR}{P + R}$$

## III. Related Work

Machine learning techniques were applied to the problem of classifying so-called core domains as part of a threat assessment in a production system at Nominum[5] [13] [14]. This problem has some similarities to the problem described in this paper, and illustrates the use of continuous learning to upadate an already-trained model on arrival of new data.

The .NZ registry maintains a set of business intelligence datasets which are constructed in part by analysis of queries received at authoritative DNS servers. In order to improve the accuracy of those datasets, machine learning techniques were used to build models that could classify query sources as DNS resolvers or other systems (e.g. systems performing active monitoring of the DNS). The work included extensive feature analysis and incorporated substantial domain knowledge derived from earlier analysis. [15] [16].

A study in the application of different machine learning techniques was presented in [17] as part of an attempt to train a model to identify Internet traffic tunnelled over the DNS protocol.

The approach described in this paper differs from other approaches described above in that it acknowledges the problems inherent in grouping DNS transactions together without the ability to be certain that the apparent sources of DNS queries are legitimate.

## IV. Methodology

### A. Overview

A complete set of DNS query data received with UDP transport over trusted and untrusted paths at a major anycast site in Ashburn, VA, USA was collected. This source data is based on raw packet captures in PCAP format[6], post-processed into *dnscount* objects to extract various parameters from the raw DNS messages: a timestamp; client and server adresses (IPv4 and IPv6); the query type; query name; transport protocol; response code and DNS message flags. Separate *dnscount* objects are stored for queries and responses; the query objects differ slightly in composition since they naturally do not include a response code.

The *dnscount* objects are centralised using Redis[7] message brokers for integration in other Afilias traffic measurement systems. Since the response objects contain a superset of the information contained within the query objects (they include a response code), and since they represent the results of queries that are known to be well-formed to the extent that a nameserver can produce a response, only the response objects for a sample period were extracted for use in training this model. The production Redis message brokers were not used since doing so would involve a release engineering process which would introduce unnecessary cost and delay to the collection process.

### B. Data Reduction

Query summaries at just this site represent around 500GB of data when compressed using bzip2[8], and hence an in-place reduction and summarisation process was undertaken:

1. Only data from the first two weeks in November 2018 were considered. This seems intuitively like a long enough period to accommodate different workday and weekend behaviour without represending an unmaageable data set, although it seems intuitively true that a longer sample period would result in a better model (see also section VII);
2. Query *dnscount* objects were discarded, since the corresponding response objects contain a superset (see section IV-A);
3. Response objects with TCP transport were discarded, since transactions over an established TCP session have authenticated endpoint addresses through the TCP setup handshake;
4. Collections of the remaining objects within five-minute sample buckets were used to produce a set of summary observations for each (site, client, bucket), as described in section IV-C.

The resulting summaries (compressed again using bzip2) occupied around 5GB, which is a more manageable data volume for transport over a network to a central location. This data also contains no query names, allowing greater confidence that it contains no personally-identifiable information and hence presents no significant threat to personal privacy.

### C. Data Extraction

Individual *dnscount* records were summarised in five-minute intervals in order to characterise the nature of DNS traffic for the corresponding (*timestamp*, *sitecode*, *client*). The resulting observations for each contained the following variables. These were selected based on general domain knowledge about the DNS and about the nature of the end-systems that trigger DNS queries to be sent from Google and Facebook resolvers.

- (*timestamp*, *sitecode*, *client*)
- number of responses counted in each five-minute sample interval
- length of the largest observed label in all query name
- the mean length of all observed labels in all query names
- the number of unique top-level labels observed in all query names
- the number of unique second-level domains observed in all query names

[5]Nominum was acquired by Akamai in November 2017

[6]PCAP, named after the C library `libpcap`, is the file format used by the `tcpdump` utility

[7]https://redis.io/

[8]http://www.bzip.org/

- the proportion of query names that consisted of 1, 2, 3 or 4 labels (exposed as four separate variables)
- the proportion of responses with response code[9] 0, 1, 2, 3, 4 5, 6, 7, 8, 9 or 10 (eleven separate variables)
- the proportion of responses with query type[10] 1, 2, 5, 6, 15, 16, 28, 48 and 255 (eight separate variables)

Closer examination of this summary set revealed data for around two million clients; of those two million, 80% of responses observed during the sample window were sent to just ten thousand. This is a decidedly asymmetric distribution with a long tail.

Training and validation datasets were extracted from these summary sets by collecting all observations for (*t*, *sitecode*, *client*) $\forall t$, *sitecode* = IAD1 and each of:

1. *client* is known to be reachable via the Google PNI (candidate Google dataset);
2. *client* is known to be reachable via the Facebook PNI (candidate Facebook dataset);
3. *client* is not reachable via any PNI (candidate "other" dataset).

*Describe the resulting datasets*

### D. Feature Engineering

- derive day-of-week and hour-of-day from date field; discard month and year since they would be the same for all observations
- discard site code, since it's the same for all observations

### E. Validation Process

We validate each model described in section II-A using k-fold cross-validation over the training dataset with $k = 10$. This provides an assessment of each model through ten folds of the source data, giving a better measure of validation than a single training/validation split.

The selected model is then trained over the entire training set, and applied to untrained data to measure its accuracy.

## V. Evaluation

Results of the process I applied

Can include a paragraph describing what languages, packages and libraries were used.

Possibilities: accuracy measures, graphs showing tuning process, tables and graphs comparing different approaches, tuned parameter ranges and selected values.

No code.

[9]https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-6

[10]https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4

## VI. Conclusion

Short summary of the paper/report

Should include problem description, how I solved it and the main results.

## VII. Future Study

- different algorithms
- incorporate new classes (new trusted paths to resolvers)
- string-similarity metrics for QNAMEs
- quantitative measurements of query entropy
- larger sample period
- continuous learning model
- recurrant neural networks or other models that exhibit temporal dynamic behaviour for a time sequence

It is reasonable to expect that the grouping and ordering of DNS queries might be relevant in the classification of a query stream as originating from a particular DNS resolver. For example, the DNS Security Extensions (DNSSEC) specification [18] accommodates flexibility in the order in which DNSSEC resource record sets are retrieved when a resolver with an empty cache performs validation on an answer from a signed zone; certain applications[11] are also known to exhibit specific behaviour when using the DNS, and resolvers that serve a community of such applications might exhibit corresponding identifying behaviour. Particular web services use signature combinations of content distribution network or embedded advertiser beacons that might well provide a useful signature through a resolver, even with the significant caching potential of answers obtained from top-level domain authoritative nameservers.

The models described in this paper treat each query stream as an unordered set of observations. The applicability of other models whose training can be influenced by the ordering of data, e.g. those based on recurant multilayer perceptron networks, seem worthy of future investigation.

## VIII. Colophon

This document has been written in R Markdown[12]; the code used to produce the output included in this document is consequently included with the document source[13]. The production of this document in IEEEtran style from R Markdown was informed by a pseudonomymously-attributed community project[14].

[11]For example, qmail sends queries with QTYPE=ANY in an attempt to accelerate the process of retrieving answers that otherwise would require separate queries with QTYPE=A, AAAA and MX. The rarity of this approach was exposed when support for ANY queries on the server side started to become constrained. See https://fanf.livejournal.com/122220.html for related commentary.

[12]https://rmarkdown.rstudio.com

[13]https://github.com/ableyjoe/uwo-mesc/tree/master/ECE-9603A-001-GF18/project

[14]https://github.com/mathematicalcoffee/IEEEtran-rmarkdown

## REFERENCES


[1] "Domain names - concepts and facilities," RFC 1034, Nov. 1987. [Online]. Available: https://rfc-editor.org/rfc/rfc1034.txt

[2] "Domain names - implementation and specification," RFC 1035, Nov. 1987. [Online]. Available: https://rfc-editor.org/rfc/rfc1035.txt

[3] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://rfc-editor.org/rfc/rfc793.txt

[4] "User Datagram Protocol," RFC 768, Aug. 1980. [Online]. Available: https://rfc-editor.org/rfc/rfc768.txt

[5] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," RFC 7858, May 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7858.txt

[6] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8484.txt

[7] C. Huitema, M. Shore, A. Mankin, S. Dickinson, and J. Iyengar, "Specification of DNS over Dedicated QUIC Connections," Internet Engineering Task Force, Internet-Draft draft-huitema-quic-dnsoquic-05, Jun. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-huitema-quic-dnsoquic-05

[8] F. Neves and J. Damas, "Preventing Use of Recursive Nameservers in Reflector Attacks," RFC 5358, Oct. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5358.txt

[9] D. J. C. Klensin, "Simple Mail Transfer Protocol," RFC 5321, Oct. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5321.txt

[10] K. E. Lindqvist and J. Abley, "Operation of Anycast Services," RFC 4786, Dec. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4786.txt

[11] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass svm method? an empirical study," in *Multiple Classifier Systems*, N. C. Oza, R. Polikar, J. Kittler, and F. Roli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 278–285.

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[13] Y. Yuzifovich, H. Liu, A. Sarychev, and A. Asiaee, "Augmented Intelligence to Scale Humans Fighting Botnets," in *Proceedings of botconf 2017*, Dec. 2017. [Online]. Available: https://www.botconf.eu/2017/augmented-intelligence-to-scale-humans-fighting-botnets/

[14] Y. Yuzifovich, "What's lurking in core domains," in *Proceedings of OARC 27*, Sep. 2017. [Online]. Available: https://indico.dns-oarc.net/event/27/contributions/456/

[15] J. Qiao. (2018, Nov.) Detecting resolvers at .nz. [Online]. Available: https://blog.nzrs.net.nz/detecting-resolvers-at-nz/

[16] J. Qiao and S. Castro, "Resolver detection using machine learning," in *Proceedings of OARC 29*, Oct. 2018. [Online]. Available: https://indico.dns-oarc.net/event/29/contributions/655/

[17] M. Sammour, B. Hussin, and M. F. I. Othman, "Comparative analysis for detecting dns tunneling using machine learning techniques," *International Journal of Applied Engineering Research*, 2017. [Online]. Available: https://www.ripublication.com/ijaer17/ijaerv12n22_137.pdf

[18] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends, "DNS Security Introduction and Requirements," RFC 4033, Mar. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc4033.txt