A Survey of DNS Traffic Analysis for Detecting DGA-based C2 Communication Channels

Snehal Patil[1], Sakshi Randive[2], Pranav Jagtap[3], Darshan More[4], Kunal Badhe[5]

[1]Asst.Prof, Department of Masters of Computer Applications and Management.

[2,3,4,5]Student, Department of Masters of Computer Application.

D.Y. Patil Institute of Master of Computer Applications and Management, Akurdi, Pune.

ABSTRACT

Malware families that rely on domain generation algorithms (DGAs) can produce hundreds of candidate domain names a day, and their bots try them one after another until one points to a live command-and-control (C2) server. Since most of those names are never registered, a static blocklist catches very little. Defenders have therefore turned to the Domain Name System itself, which every infected host must use before it can reach its operator. This survey reviews the research on spotting DGA-based C2 activity from DNS queries and responses. We sort the work into three groups: analysis of the domain strings, analysis of resolution behaviour and passive DNS records, and hybrid systems that feed both kinds of evidence into machine learning classifiers. Random Forest and other tree ensembles receive particular attention, because they are accurate enough, fast enough and readable enough to run in real networks. We also look at the datasets, features and metrics that the studies use, and we point out where differences in those choices make results hard to compare. The paper ends with open problems, including adversarial robustness, encrypted DNS and generalization across datasets.

Keywords: Domain Generation Algorithm, Command and Control, DNS Traffic Analysis, Passive DNS, Random Forest, Malware Detection, Network Security.

Introduction

Botnets are still among the most damaging things that happen on the internet. An infected machine waits for instructions from its operator and then gets used for spam, credential theft, denial-of-service floods or ransomware. For a long time the usual response was to find the C2 server and block it. That works until the operator moves, and operators move quickly. A server can be replaced in minutes, which turns blocking into a race that defenders rarely win for long.

Domain generation algorithms were designed to make that race harder to win. A DGA embedded in a malware sample takes a seed, often the current date, and produces a long list of candidate domains. The bot tries them in order. The operator needs to register only one or two of them in advance, so the vast majority of names the bot queries do not exist. Conficker, which spread widely from 2008 onwards, is probably the best-known early example. Measurement work has since catalogued dozens of families, and they do not all generate names the same way [6].

DNS is a useful place to look for these families, mainly because a bot cannot avoid it. Before a bot can talk to its server it has to resolve a name, and that lookup leaves a trace at the local resolver, at the network edge and in passive DNS databases. Many of the lookups fail with NXDOMAIN, and a host that produces a burst of failures is doing something an ordinary laptop rarely does. Most of the detection research follows from this observation, looking either at what the names look like or at how the querying clients behave [1], [2], [11].

The field has grown quickly. A decade and a half of papers, plus several earlier surveys of malicious domain detection in general [11], make it difficult for a newcomer to see how the pieces relate. This paper is narrower. It covers DGA-based C2 traffic observed through DNS, and it spends most of its time on supervised learning with Random Forest and similar tree ensembles. The reason is practical. Tree ensembles show up often in this literature, they train without much tuning, and their feature importance scores give analysts something concrete to check.

Problem Statement

DNS-based detection works, but the literature leaves several things unsettled. It is not clear which features carry over from one malware family to another, how a classifier holds up when traffic patterns change, or whether two reported accuracy figures can be compared at all. Many studies use private data with their own class balance, and very few ask what happens when the attacker knows how the detector works. A survey that ties together feature families, classifiers and evaluation habits can help two groups in particular: researchers who need a sensible baseline, and engineers who have to decide what to deploy.

Objectives

1. Review the main DNS-based approaches to detecting DGA-based C2 traffic and arrange them into a usable taxonomy.
2. Examine the features, datasets and metrics used in these studies, and identify where their results cannot be compared directly.
3. Judge how Random Forest and other tree-based classifiers stack up against lexical, behavioural and deep learning alternatives.
4. Describe the open problems and suggest where future work on robust DNS-based detection should go.

Literature Review

DGA Families and Generation Schemes

It helps to start with how the names are produced, because that determines what a detector can see. Arithmetic and seed-based generators compute names from a hash or a pseudo-random generator that is seeded with something like the date. Wordlist-based generators stitch dictionary words together, so their output reads more like language and is harder to separate from benign names by character statistics alone. Some families mix the two approaches or change their scheme over time. Plohmann et al. measured a large number of these families across an extended period, and the variety they documented is a good reason not to tune a detector to a single generator [6].

Lexical and Statistical Analysis

The first detectors looked only at the names. Yadav et al. [10] noticed that generated names often have unusual character distributions. They compared the bigram distribution of a queried name against that of legitimate names using Kullback-Leibler divergence, and they added simple measures such as the share of consonants and digits. The appeal is obvious: each query can be judged on its own, with no history needed. The limitation is equally clear. Wordlist-based families can produce names that look statistically normal, and a short name simply does not contain enough characters to say much.

Passive DNS and Network Behaviour

Other researchers shifted attention from the names to the clients. Antonakakis et al. [1] observed that when a bot walks through a DGA list, the NXDOMAIN responses it generates form patterns that show up across many hosts. Grouping hosts by those patterns can reveal infected machines even when none of the individual names looks suspicious. Bilge et al. [2] approached the problem from passive DNS. Their EXPOSURE system looks at how many distinct addresses a domain resolves to, how widely those addresses are spread across networks, how the time-to-live values behave, and how long the domain has been around. A decision tree then makes the call. Schiavoni et al. [8] took a related route in Phoenix, pairing lexical features with IP-level information to follow DGA botnets and map their infrastructure.

Machine Learning Classifiers

Most recent work is supervised, and tree-based models have been especially popular. Support vector machines, logistic regression and naive Bayes all appear in the literature, but random forests keep coming back. Breiman [3] introduced them as a way to combine many decision trees, each grown on a bootstrap sample and each allowed to split only on a random subset of features, and then to let the trees vote. The randomness keeps the trees from memorizing the training set. In practice a forest copes well with the mixed inputs that DNS features produce, such

as counts, ratios and yes-or-no flags, and it needs little scaling or feature engineering beyond what the features already require. Scikit-learn [5] has made the models easy to deploy, which probably explains part of their popularity.

Deep Learning Approaches

Deep models skip most of the hand-built features. Woodbridge et al. [9] trained long short-term memory networks on legitimate domain names and flagged the names the network found hard to predict. Saxe and Berlin [7] used character-level convolutional networks on a wider range of malicious strings, including URLs. Their accuracy is often impressive, but they need more data and more compute, and an analyst gets little help from a network that says a name is suspicious without saying why. Whether that trade is worth it depends on the environment, so the decision is as much about operations as about accuracy.

Methodology

Figure 1 shows the pipeline that most of the reviewed systems share. Traffic is captured at a resolver or a border sensor, turned into features, and passed to a classifier. The classifier's output then triggers an alert or a block. The systems differ mostly in which features they compute and which model they choose.

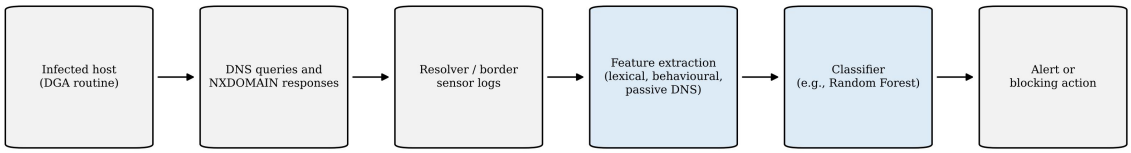


Fig. 1. Generic DNS-based pipeline for detecting DGA-based C2 communication.

How the Survey Was Done

We searched the main digital libraries and preprint servers for work on DGA detection, DNS-based malware detection and the identification of C2 traffic from DNS data. The search terms combined phrases such as domain generation algorithm, DGA, DNS traffic, passive DNS, command and control and machine learning. Titles and abstracts came first, and full texts followed for anything that still looked relevant. We kept papers that proposed or evaluated a DNS-based detection method and that described their features or evaluation clearly enough to be understood. We dropped papers about general web threats that did not analyse DNS at all.

Organizing the Literature

Each method was placed along three dimensions: where its evidence comes from (domain strings, DNS responses or client query behaviour), which feature family it uses, and whether it learns without labels, learns from labels, or combines the two. That structure is what Figure 2 and the discussion below rely on. We also recorded the dataset, the class balance and the metrics for each study wherever they were given. Those details often decide whether a reported number means anything.

Feature Families

Across the reviewed studies, features fall into four loose groups. Lexical features cover name length, character entropy, the ratio of digits to consonants and n-gram statistics. Structural features count the labels in a name and note the top-level domain. Behavioural features capture the NXDOMAIN ratio, how often a host queries, and how many distinct names it touches in a given window. Passive-DNS features describe how many IP addresses a name resolves to, what its TTL values look like and how old the domain appears to be.

Evaluation Metrics

Precision, recall, F1-score and the false positive rate are the numbers most papers report. For DNS detectors the false positive rate matters most. A rate that looks tiny on paper can still bury an analyst when a busy resolver handles millions of benign queries an hour. ROC curves and their area (ROC-AUC) show performance across thresholds, and precision-recall curves tell a clearer story when attacks are rare. Cross-validation also needs care.

If samples from the same family or time window end up on both sides of the split, the scores are inflated, and a number like that will not survive deployment.

Results and Discussion

Figure 2 arranges the approaches into four families and lists the features that go with each. Table 1 compares what each family offers and what it costs. The comparison is qualitative on purpose. The studies used different datasets, class balances and metrics, so a ranking by reported accuracy would be misleading, and we do not rank them that way.

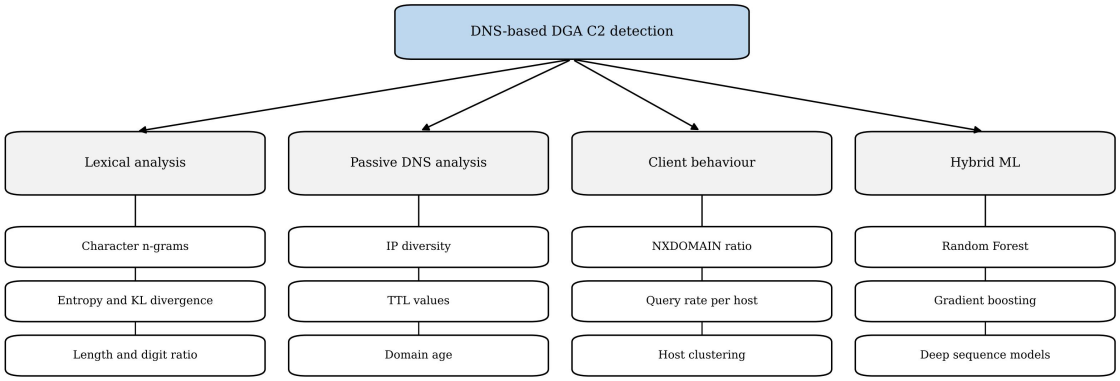


Fig. 2. Taxonomy of DNS-based approaches to DGA-based C2 detection.

Table 1 summarizes the trade-offs across the main families.

Approach	Main inputs	Strengths and limitations
Lexical analysis	Domain name strings	Cheap and needs no history, but weak against wordlist-based DGAs, and short names carry little signal
Passive DNS analysis	Historical resolution records	Captures infrastructure reuse and domain age, but needs access to passive DNS data and gives little history for new domains
Client behaviour analysis	Resolver or border logs and NXDOMAIN patterns	Finds infected hosts even when names look ordinary, but needs enough traffic per host and is affected by NAT and caching
Hybrid machine learning (e.g., Random Forest)	Combined lexical, behavioural and passive features	Good accuracy, readable feature importances and fast inference, but depends on labelled data and can drift over time
Deep sequence models	Raw character sequences	Learns features automatically, but is data hungry, expensive to run and hard to explain to analysts

Table 1. Comparison of DNS-based approaches to DGA-based C2 detection

Tree ensembles deserve a closer look. Across the studies we read, random forests and gradient-boosted trees often came close to deep models once the features were engineered carefully. That is roughly what one would expect. Tabular features tend to suit tree ensembles, and a trained forest can run on modest hardware inside a resolver or a SIEM without needing an accelerator. For many operators that practical point settles the question.

Data quality came up more than anything else. Benign traffic is often gathered in a different month, or on a different network, from the malicious samples. A classifier can then learn the difference between two collection setups instead of the difference between good and bad behaviour. Class imbalance makes this worse. A real network sees far more benign queries than DGA queries, yet some reported results use balanced test sets that look nothing like production. A detector that looks excellent on such a set can behave quite differently once the base rate is realistic.

Attackers adapt, and the literature has not caught up with that fully. A family that character entropy catches can be rewritten to draw on dictionary words. A behavioural detector can be sidestepped by slowing the query rate or

by mixing in ordinary lookups. Very few of the reviewed papers tested against an adversary who knew the detector, so we know little about how long these systems stay useful once they are in service.

Encryption changes the picture further. DNS over HTTPS and DNS over TLS keep query contents away from anyone on the path, which removes much of what lexical and passive methods depend on. Detection then has to happen closer to the resolver, on the endpoint, or in flow metadata. Work on that shift is still sparse, and it seems likely to matter a great deal in the coming years.

Challenges and Future Directions

The most obvious gap is the lack of shared benchmarks. A public dataset that covered many DGA families, realistic benign traffic and a clean time-based split would make results much easier to compare. Authors should report class balance, the false positive rate at thresholds an operator would actually use, and how far apart in time the training and test data were collected. Adversarial testing is the second gap. Evaluations should include attacks built against the detector, such as wordlist generators or deliberately paced query schedules, so that robustness gets measured rather than assumed.

Generalization is the third concern. A model trained on one family or one network often loses accuracy when moved to another, and transfer learning or domain adaptation might help. Explainability is the fourth. An analyst who receives a flag needs some reason to trust it, and feature attribution methods applied to forests or neural networks are a reasonable way to provide one. The last direction is combining signals. DNS data alone may not be enough, and pairing it with flow records, endpoint events or threat intelligence could improve reliability, though it raises privacy questions that cannot be brushed aside.

Conclusion

DNS traffic gives defenders a practical view of DGA-based C2 activity. The field has moved from inspecting domain strings toward systems that combine lexical, behavioural and passive evidence with machine learning. Random Forest and related ensembles remain attractive because they strike a reasonable balance between accuracy, speed and interpretability. Their results still depend heavily on the data and features behind them.

This survey has offered a taxonomy of the field, reviewed the features and classifiers that appear most often, and pointed out the habits that make results hard to compare. The main gaps are the absence of shared benchmarks, thin adversarial evaluation and the growing use of encrypted DNS. Until those gaps are closed, DNS-based detectors should be treated as one layer of defence rather than a complete answer.

References

[1] L. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists: Learning to detect malicious web sites from suspicious URLs,” in Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 1245–1254.

[2] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, “A comprehensive measurement study of domain generating malware,” in Proc. 25th USENIX Security Symposium, 2016, pp. 263–278.

[3] P. Mockapetris, “Domain names: Implementation and specification,” IETF, RFC 1035, Nov. 1987.

[4] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, “From throw-away traffic to bots: Detecting the rise of DGA-based malware,” in Proc. 21st USENIX Security Symposium, 2012, pp. 491–506.

[5] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, “EXPOSURE: Finding malicious domains using passive DNS analysis,” in Proc. Network and Distributed System Security Symposium (NDSS), 2011.

[6] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, “A survey on malicious domains detection through DNS data analysis,” ACM Computing Surveys, vol. 51, no. 4, Art. no. 67, 2018.

[7] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, “Detecting algorithmically generated domain-flux attacks with DNS traffic analysis,” IEEE/ACM Transactions on Networking, vol. 20, no. 5, pp. 1663–1677, 2012.

[8] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, “Detecting algorithmically generated malicious domain names,” in Proc. 10th ACM SIGCOMM Conference on Internet Measurement (IMC), 2010, pp. 48–61.

[9] M. Mowbray and J. Hagen, “Finding domain-generation algorithms by looking at length distributions,” in Proc. IEEE 25th International Symposium on Software Reliability Engineering Workshops, 2014.

[10] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, “Building a dynamic reputation system for DNS,” in Proc. 19th USENIX Security Symposium, 2010.

[11] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, “EXPOSURE: A passive DNS analysis service to detect and report malicious domains,” ACM Transactions on Information and System Security, vol. 16, no. 4, Art. no. 14, 2014.

[12] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, “Phoenix: DGA-based botnet tracking and intelligence,” in Proc. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2014.

[13] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[14] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[15] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[16] A. Liaw and M. Wiener, “Classification and regression by randomForest,” R News, vol. 2, no. 3, pp. 18–22, 2002.

[17] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, “Predicting domain generation algorithms with long short-term memory networks,” arXiv preprint arXiv:1611.00791, 2016.

[18] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] J. Saxe and K. Berlin, “eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys,” arXiv preprint arXiv:1702.08568, 2017.

[20] P. Lison and V. Mavroeidis, “Automatic detection of malware-generated domains with recurrent neural models,” arXiv preprint arXiv:1709.07102, 2017.

[21] K. Highnam, D. Puzio, S. Luo, and N. R. Jennings, “Real-time detection of dictionary DGA network traffic using deep learning,” SN Computer Science, 2021.

[22] R. Vinayakumar, K. P. Soman, P. Poornachandran, S. Akarsh, and S. Sachin Kumar, “Detecting malicious domain names using deep learning approaches at scale,” Journal of Intelligent & Fuzzy Systems, vol. 34, no. 3, 2018.

[23] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” The Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.

[24] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[25] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in Proc. IEEE Symposium on Security and Privacy, 2010, pp. 305–316.