Down to earth! Guidelines for DGA-based Malware Detection

Bogdan Cebere
bogdan.cebere@cispa.de
CISPA Helmholtz Center for Information Security
Germany

Jonathan Flueren
jonathan.flueren@tu-dortmund.de
CISPA Helmholtz Center for Information Security
Germany

Silvia Sebastián
silvia.sebastian@cispa.de
CISPA Helmholtz Center for Information Security
Germany

Daniel Plohmann
daniel.plohmann@fkie.fraunhofer.de
Fraunhofer FKIE
Germany

Christian Rossow
rossow@cispa.de
CISPA Helmholtz Center for Information Security
Germany

ABSTRACT

Successful malware campaigns rely on Command-and-Control (C2) infrastructure, enabling attackers to extract sensitive data and give instructions to bots. As a resilient mechanism to obtain C2 endpoints, attackers can employ Domain Generation Algorithms (DGAs), which automatically generate C2 domains instead of relying on static ones. Thus, researchers have proposed network-level detection approaches that reveal DGA usage by differentiating between non-DGA and generated domains. Recent approaches train machine learning (ML) models to recognize DGA domains using pattern recognition at the domain's character level.

In this paper, we review network-level DGA detection from a meta-perspective. In particular, we survey 38 DGA detection papers in light of nine popular assumptions that are critical for the approaches to be practical. The assumptions range from foundational ones to assumptions about experiments and deployment of the detection systems. We then revisit if these assumptions hold, showing that most DGA detection approaches operate on a fragile basis. To prevent these issues in the future, we describe the technical security concepts underlying each assumption and indicate best practices for obtaining more reliable results.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Security and privacy** → **Intrusion detection systems**; • **General and reference** → **Surveys and overviews**; • **Networks** → **Firewalls**.

KEYWORDS

Machine Learning, Intrusion detection systems, Domain Generation Algorithms (DGAs), Meta-study

RAID 2024, September 30–October 02, 2024, Padua, Italy

ACM ISBN 979-8-4007-0959-3/24/09
https://doi.org/10.1145/3678890.3678913

ACM Reference Format:
Bogdan Cebere, Jonathan Flueren, Silvia Sebastián, Daniel Plohmann, and Christian Rossow. 2024. Down to earth! Guidelines for DGA-based Malware Detection. In *The 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2024), September 30–October 02, 2024, Padua, Italy*. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3678890.3678913

1 INTRODUCTION

Malware must inevitably reach out to its command & control (C2) infrastructure to receive instructions or leak sensitive data. To this end, malware–except a few peer-to-peer botnets [81]–usually relies on centralized C2 servers. In the simplest case, malware can hard-code the C2's address (e.g., a domain or IP address). To avoid this single point of failure, various malware families leverage Domain Generation Algorithms (DGAs) [9, 74]. Instead of using static C2 addresses, DGAs algorithmically generate the addresses, optionally based on an agreed-upon seed (e.g., the current day). DGA-empowered malware can then use the Domain Name System (DNS) to dynamically resolve a C2 server's IP address at runtime.

Consequently, for over a decade, we have observed a constant stream of academic proposals—54 peer-reviewed papers by now, to the best of our knowledge—to detect the presence of such automatically generated domains (AGDs) in DNS traffic. Indeed, C2 communication is a potent angle from which to detect malware activity at the network level. The early works focused on a context-aware detection strategy, with 16 papers tackling the problem by buffering various indicators per user and modeling them using clustering techniques. However, since 2016, the focus has shifted to *contextless detection*, with 38 out of 54 papers aiming to detect DGAs by inspecting character patterns of the queried domains. The continuous academic interest in such contextless DGA detection is likely spurred by the availability of labeled datasets, which ease the application of modern machine learning (ML) models. Not surprisingly, contextless DGA detection papers report staggering accuracy and thus allegedly promising network-based botnet detectors.

But are these contextless DGA detection approaches indeed helpful for detecting botnets? We find that research mainly reduces the problem of DGA-based botnet detection to a machine-learning task in a static setting. By doing so, research often overlooks both practical aspects that hinder wider success in practice, *and* the fact that

we enter an arms race with an unclear outcome. As promising as it sounds, the problem of botnet detection goes much beyond just being able to distinguish between human- and algorithmically-created domain names. By neglecting the technical inner workings of botnets or caveats of deployed ML models, such as (simple) evasion strategies, we widen the already-existing gap between the academic literature and what is useful in practice even more.

In this paper, we, therefore, review network-level DGA detection from a meta-perspective. In particular, we survey all 38 context-less DGA detection papers in light of nine popular assumptions that are required for the approaches to be practical. The assumptions range from *foundational* ones (e.g., AGDs are distinctive from human-generated domains, AGDs stem from bots, and any malware using a DGA would indeed activate the DGA) to assumptions about *experiments* (e.g., clean training data sets) and *deployment* of the detection systems (e.g., inference speed, ability to generalize). We then revisit if these assumptions hold, showing that most DGA detection approaches operate on a shockingly fragile basis.

The highlights of our analyses underline that botnet detection requires *both* ML expertise *and* domain expert knowledge. Unfortunately, we find that most papers do not sufficiently emphasize one of these two dimensions. For example, 22 papers implicitly assume that generically distinguishing AGDs from human-registered (non-DGA) domains is sufficient to detect malicious DGAs, completely ignoring several use cases of *benign-origin* AGDs that would trigger false positives. More fundamentally, all 38 papers operate assuming they can identify AGDs, but 36 papers did not test their approach against evasive, more difficult DGAs. Likewise, to our knowledge, none of the papers discusses DGAs being used as *backup* C2 mechanism only, which tacitly assumes that the sheer presence of a DGA in malware is sufficient for detection—ignoring that 1/3 of the botnets would activate their DGA only if the botnet's C2 infrastructure is unreachable.

These problematic trends show an apparent disconnect between academic proposals and practical needs for botnet detection. For this reason, we provide concrete guidelines that researchers can leverage to design and present future detection approaches. This is an attempt to improve the quality of academic AGD detection research by highlighting subtle pitfalls that can harm the field's progress. This meta-analysis and the resulting constructive suggestions for future work were inspired by prior meta-studies such as [11, 82]—this time, providing insights for DGA detection.

We summarize our contributions as follows:

- We formalize nine popular assumptions about DGA detection, grouped by their point of presence: (a) how the research is motivated (*foundational assumptions*), (b) how the research is presented (*experimental assumptions*), and (c) how the research is intended to be used in practice (*deployment assumptions*).
- We discuss the implications of the assumptions and emphasize some pitfalls that can inhibit the impact of the DGA detectors.
- We empirically investigate the claims' downsides and highlight the consequences of ignoring them.
- For each assumption and its limitations, we compile a set of recommendations and best practices, hopefully improving the quality of future research.

2 BACKGROUND

2.1 Domain Generation Algorithms

Domain Generation Algorithms (DGAs) are pseudo-random algorithms that aim to create unseen domain names (AGDs) and are frequently used for malicious purposes.

A fundamental part of most botnets is identifying and contacting their command-and-control (C2) server. The C2 server can further instruct the botnet on initiating various attacks, such as large-scale Distributed Denial of Service (DDoS) attacks [49], spamming attacks [57], or different kinds of online frauds [69, 101]. The most straightforward approach to contact the C2 server is to hardcode the attacker-controlled domains or IP addresses into the malware binary. However, this approach can be easily susceptible to takedowns due to a single point of failure. To counter this, many botnets thus leverage DGAs that create new domain names for C2 servers based on a random seed (derived from timestamps, Twitter feeds, Google Trends, etc.). The botnet owner, aware of the DGA and the random seed, registers one (or multiple) domains and points them to the actual C2 server. This additional level of indirection by leveraging standard DNS name resolution eliminates the single points of failure, making the actual C2 communication harder to identify and block.

Over time, researchers and malicious actors investigated different strategies for generating AGDs, which vary significantly in terms of ease of detection. Plohmann et al. [74] introduced the DGArchive, a collection of AGDs, with 92 botnet families available at the time of writing. They also pointed out that the pseudo-random domain generators can be grouped as follows:

- *Arithmetic DGA (*74/92 *families).* The domain names are generated using simple arithmetic operations derived from a random seed (usually a timestamp).
- *Hash DGA (*11/92 *families).* The generative process can include any seeded random generator, finalized by applying a hashing function on top of it, such as MD5, SHA-1, etc.
- *Wordlist/Dictionary/Stealthy DGA (*4/92 *families).* A trickier generative method combines common words, phrases, or dictionary entries. The difficulty comes from the hardness of finding a decision boundary between the non-DGA and wordlist DGA samples.
- *Permutation DGA (*1/92 *families).* Another difficult-to-detect DGA category is based on the permutation of a few seed domains.
- *Adversarial/Evasive DGA (*2/92 *families).* This generative process assumes knowledge of the non-DGA samples, decision models, or past families used in the detector and tries to exploit the decision boundary with hard-to-detect samples. On top of the DGArchive families, 9 papers on domain generation provide other sources of adversarial families.

Table 1 contains sample 2LDs from each DGA category, i.e. the domain without the subdomain and the TLD.

2.2 Academic DGA research

In this section, we review the DGA research from several points of view: detection (Section 2.2.1), generation (Section 2.2.2), and other surveys (Section 2.2.3) that we deem slightly related to ours.

Category	Example 2LDs
Arithmetic DGA	hngajjkuknzwdliqfj ciobanarianaqh
Hash DGA	2e468aef1f98372b8009d0196ef1d951 7817b2bcf25367beb24b3270232e67e5
Wordlist DGA	allowallowclient windcalendarcommittee
Permutation DGA	gertadobeflashplaye ntexplotreredo
Adversarial DGA	wegannas pongsgite

Table 1: *Examples of 2LDs from each DGA category.*

2.2.1 AGD Detection. A significant effort has been invested in DGA detection techniques, resulting (to our knowledge) in 54 peer-reviewed papers. Based on the type of input information, we group the solutions into (1) contextless and (2) context-aware detection.

Contextless Detection. The main focus of our study is contextless detection research. In this approach, the detection system operates only on the domain name, with no user-correlated information or past activity. Some systems might include additional information, such as the DNS response or WHOIS information—still remaining contextless as the detection does not spawn multiple DNS transactions. Contextless detection is considered more scalable and faster than context-aware techniques, more privacy-friendly, and thus can be better suited for firewalls running at the ISP level. At the same time, the character-level analysis opened the door for many novel machine-learning techniques. Since 2016, due to these incentives, this is the most active research area in DGA detection, with 38 publications.

Table 2 summarizes these works grouped by their underlying technique to model domain names, i.e., Random forests, Ensembles, Multilayer perceptron variants, Recurrent neural networks, Convolutional networks, pattern matching or Residual neural networks. Depending on the modeling solution, the input features might require processing. Common approaches for creating the feature space are extracting statistical features of the domains [4, 16, 59, 66, 70, 79, 86, 93, 95, 108] and creating text embeddings from the domains [27, 30–35, 48, 48, 65, 72, 92–94, 97, 102, 105, 107, 109, 110, 115].

Context-aware Detection. Early works on DGA detection focused on context-aware methods, which buffer multiple indicators of compromise per device/user over time and model them using clustering algorithms. As no prior knowledge of particular DGAs is required, these methods allow for the detection of known DGAs or the discovery of unknown DGAs. Previous work [3, 9, 17, 18, 40, 42, 56, 83–85, 89, 103, 106, 112, 116] provided context-aware detection solutions, assuming it was possible to create a buffer of domains per client and to run clustering or anomaly detection algorithms on that buffer. This approach can be better suited for firewalls in local networks, as it is hard to scale to larger networks and might be privacy-invasive. This category has not received much research attention since 2016, so it is outside the scope of our analysis.

2.2.2 Domain Generation. Generative methods attempt to create stealthy AGDs that evade classifiers. They build on the wealth of prior knowledge about botnets, collected in DGArchive [74], the Open-Source Intelligence (OSINT) DGA feed from Bambenek Consulting, and AmritaDGA [100]. The generative models use recurrent neural networks [50], Generative Adversarial Networks [7, 111], or character manipulations from a baseline dataset [39, 63, 71, 91]. We replicated these methods and included them as baselines in our benchmarks.

2.2.3 DGA Surveys. Plohmann et al. [75] provided one of the earliest in-depth analyses of the DGA ecosystem, facilitating several works with the DGArchive dataset. Since then, other works have reflected on the state of the field [5, 14, 58, 77, 113]. However, these works focus on the broader landscape of DNS attacks. To our knowledge, we are the first to specifically focus on DGA detection practices and to provide guidelines for potential future directions.

3 POPULAR ASSUMPTIONS IN CHARACTER-LEVEL DGA DETECTION

Seeing the ever-increasing literature on DGA detection on the one hand but little practical deployment on the other, we take a step back. In this section, we revisit and challenge nine popular assumptions made by state-of-the-art approaches. To this end, we review 38 papers, representing all peer-reviewed articles we know on this subject. We then reflect on these papers' assumptions about DGA detection's limitations and implications, the underlying experiments, and deployment requirements.

This analysis was inspired by Arp et al. [11], who surveyed typical pitfalls when applying machine learning techniques to security problems. Similar pitfalls exist when researchers study DGA detection approaches, but they are not limited to ML idiosyncrasies. Instead, our analysis is more specific to DGA detection and, hence, orthogonal to existing guidelines such as by Arp et al. [11] or research on how to design malware experiments by [82].

In particular, we observe nine popular assumptions specific to DGA contextless detection papers. We group these assumptions into three categories:

- *Foundational Assumptions (Section 3.2).* First, we discuss perennial motivations in the problem statements, which act as pillars in designing the detection solution. To that end, we study the usefulness of non-existing domains in DGA detection, the preference of DGAs over hard-coded domains in botnets, and the difficulty of separating non-DGA samples from generated domains.
- *Experimental Assumptions (Section 3.3).* The design of the experiment directly impacts the results and conclusions, i.e., a biased approach can lead to inaccurate results. Therefore, we discuss potential pitfalls related to the training set quality, the sample sizes, and the improvements in the benchmark reports.
- *Deployment Assumptions (Section 3.4).* DGA detection papers aim for practical use cases. We thus also discuss some subtleties that might prevent these solutions from being deployed in real-world networks. This entails their runtime performance and their promise regarding generalizability.

We study the prevalence of these assumptions in scientific works to better understand their validity. To this end, we study how many

Detection Arch.	Papers	Classification Type	Input Type
Random Forests	[4, 12, 61, 66, 70, 86, 93, 95]	binary	FQDN
Ensembles	[108]	binary	
MLP	[16, 59, 61, 79]	binary	
LSTM/RNN/GRU	[12, 27, 48, 65, 80, 88, 90, 92–94, 97, 102, 104, 105, 107, 109, 110]	binary & multiclass	
CNN	[15, 48, 104, 109, 110, 114, 115]	binary & multiclass	
Pattern Matching	[6, 36, 37]	multiclass	
ResNet	[30, 32–35]	multiclass	FQDN, e2LD

Table 2: High-level overview of the contextless character-level DGA detection papers.

papers (i) state the assumption explicitly/implicitly, (ii) do not discuss it, or (iii) do not base their methodology on the assumption. For each assumption, we count the papers in these three categories using the following coloring scheme for illustration:

Assumed | Omitted Discussion | Not assumed

After showing the prevalence of each assumption, we will also study the conditions under which each assumption holds. As part of this, we rely on mechanisms and datasets typically used by the DGA papers under study. Therefore, the following subsection introduces an experimental setup that we will leverage in subsequent evaluations when revisiting the assumptions empirically. We will then return to the concrete assumptions in the subsequent subsections.

Appendix A includes the detailed survey methodology and the labeling criteria for each assumption.

3.1 Empirical Setup

Datasets. First, we introduce a labeled dataset that is representative of the surveyed DGA detection papers. For the non-DGA domains, we use Tranco [76], a research-oriented and curated list of popular domains. We survey alternative datasets in Appendix E, showing that Tranco is statistically similar to other non-DGA sample lists (e.g., Alexa's top domain list). For the malicious domains, we included all botnet families from DGArchive. In addition, we added a set of academic and synthetic AGD generators, including adversarial generators ([7, 71, 91, 111]), dictionary/stealthy DGAs ([39]), and simple mutation/permutation DGAs. By including DGAs from all known categories, we can objectively evaluate the detection models and understand the corresponding challenges.

The datasets we evaluate comprise domain names represented as strings. Following recent works [33], we extract the effective second-level domain (e2LD) token of the domains, ignoring the TLD and the subdomain (unless it is a subdomain of a dynamic DNS service). For most DGA families, extracting the 2LD is enough—e.g., 'morningignore.com' from Matsnu. However, families like Bobax, Corebot, Necro, and Symmi rely on Dynamic DNS services. For example, Bobax generates domains like "sjxhzwvtj.dyndns.org". This behavior motivates the concept of e2LD instead of 2LD.

Classification Models. We then replicate 13 popular classification models from DGA detection papers. This includes models that work on tabular statistical features: logistic regression [26], SVM [25], k-nearest neighbors [73], decision trees [20], random forests [19], XGBoost [22]. In addition, to capture deep learning approaches, we replicate classifiers based on plain MLP [46], RNN/GRU/LSTM [41], CNN [62], Transformer [99] and ResNet [47]. We released these implementations as open-source to foster follow-up, detailed in Appendix B. Appendix D explains how we represent the datasets for these models.

Benchmarks. We evaluate each model using 5-fold cross-validation. For each benchmark, we focus on binary classification (DGA-vs-non-DGA, one DGA-vs.-another, etc.). We denote true positive as TP, false positive as FP, true negative as TN, and false negative as FN. Similar to other studies (cf. Appendix C), we use *Precision* to measure the model's capability to make correct positive predictions (TP/(TP+FP)), *Recall* to calculate the ratio of positive instances (TP/(TP+FN)), and the F1-score, which combines the two as follows $F1 = \frac{2\times Precision\times Recall}{Precision+Recall}$. An F1 score of 1 indicates perfect detection, while a score of 0.5 indicates that the model's performance is close to random choice. We also include the Matthews correlation coefficient (MCC), a more robust evaluation metric for imbalanced datasets [23]. The MCC score is a statistical rate that produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (TP, FN, TN, FP), proportionally both to the size of positive elements and the size of negative elements in the dataset. The general formula is $MCC = \frac{TP\times TN - FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$. The MCC value ranges from -1 to 1. -1 means inverse, 0 random prediction, and 1 perfect prediction, respectively.

3.2 Foundational assumptions

This section reviews some recurrent arguments used as building blocks in DGA research. These arguments are not necessarily used in developing the method or the experiments but as a pillar for conducting the investigation. These assumptions are commonly found in the introductory sections of the papers and include the difficulty of separating non-DGA samples from AGDs (**A1**), the risks of hiding multiple DGAs under a single detection label (**A2**), preconceptions about botnet behaviors (**A3**) and the NXDOMAINs' utility (**A4**).

A1: AGDs' structural patterns differ from non-DGA samples, regardless of the generator family.

100% papers

Description. Settled on the contextless detection perspective, the question is whether a decision model could discriminate AGDs from non-DGA samples. While a fundamental question, the answer is not always clarified in related work. Every family has a different complexity (unless they share the same domain generator), and visual and statistical aids can help the readers to understand the task's difficulty. Some DGA families, like Hash or Arithmetic DGAs,

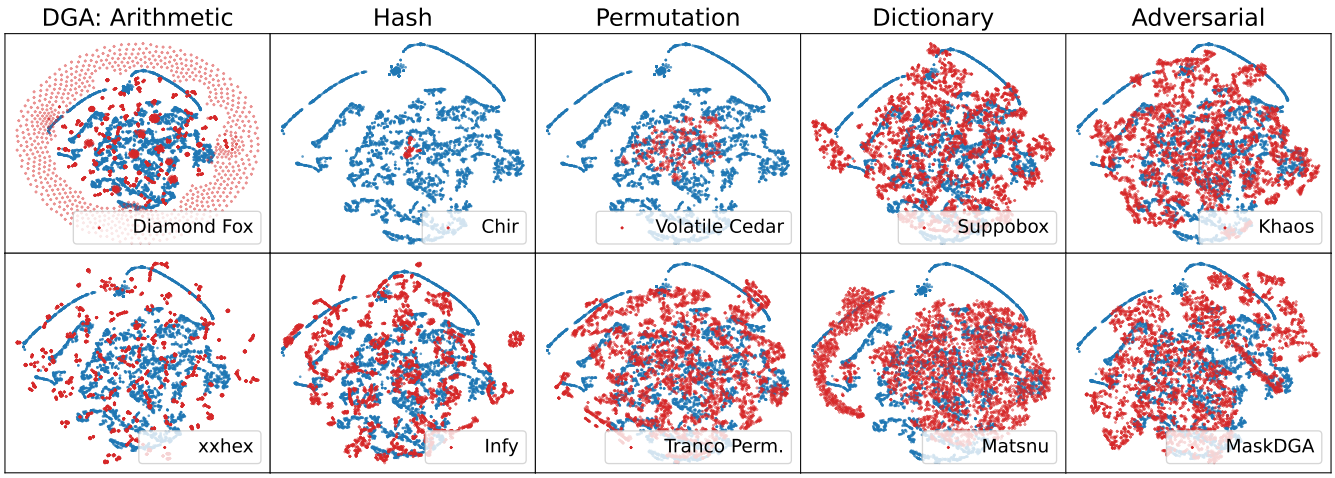

Figure 1: *t-SNE plots comparing the distributions of Tranco (non-DGA, blue) vs some DGA families (red), grouped by their category.* Each point represents a domain name projected in the 2D space. The plots highlight how various DGA distributions are represented compared to non-DGA datasets. On the left (Arithmetic, Hash DGA categories), we can observe visibly distinct distributions with different patterns and locations, making discriminating against non-DGA samples easier. Other examples (from Dictionary and Adversarial DGAs) are clustered close to non-DGA samples, making them more challenging to detect.

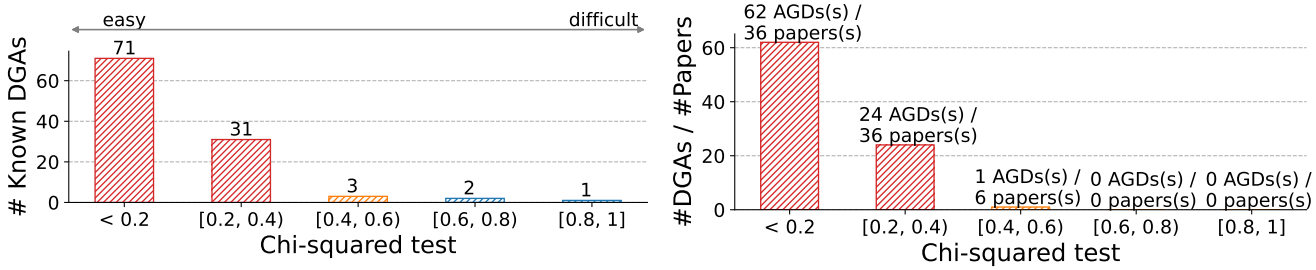

Figure 2: *The Chi-squared test between the non-DGA dataset (Tranco) and the DGA families.* Left: The Chi-squared statistic of DGA families reported against the Tranco Dataset. Right: The presence of the DGA in the papers' experimental section, grouped by the Chi-squared metric.

might employ simple patterns that are detectable with a simple model, while difficult classes, such as Dictionary or Adversarial DGAs, might need a different, more advanced detection model.

Understanding the difficulty of DGAs using t-SNE plots. Before evaluating the detection models, we take a step back and try to understand the non-DGA and DGA datasets visually and statistically. Figure 1 illustrates the t-SNE plots of a few DGA families (in red), grouped by category when compared to the non-DGA dataset (Tranco, in blue). t-SNE plots are a common technique for representing high-dimensional data in the 2D space [98]. In such a plot, each point represents a domain from each dataset, projected in the 2D space. A clear red pattern, distant from the blue points - as we can see in the examples from the Arithmetic and Hash families - can be trivially picked up by models. However, this does not apply to all AGD families. We can visualize some difficult examples on the right - in the Dictionary and Adversarial examples. The red and blue dots are extremely close (but not overlapping, as the points are distinct), meaning separating those AGDs from non-DGA examples is challenging. This shows that different DGA families pose challenges and should be analyzed in depth.

Understanding the difficulty of DGAs using statistical tests. For the statistical insights, we evaluate the Chi-squared test, which measures how similar two distributions are. The test can be performed on datasets using the following steps: (1) group the values of the distributions into bins; (2) count the observations included in each bin; (3) compute the expected frequencies using one of the distributions; (4) compute the chi-squared statistic using $\chi^2 = \sum \frac{(O-E)^2}{E}$, where O are the observed frequencies, and E are the expected frequencies. Intuitively, the metric is 0 for unrelated distributions and 1 for identical distributions.

Figure 2 depicts the Chi-squared test between the DGA and the non-DGA datasets. The metric highlights that permutation and dictionary-based DGA are statistically more indistinguishable from non-DGA data than other variants. We also represent the densities of these families in the research literature in Figure 2 (right). We can observe that many papers report statistically easier DGA families, while the more challenging ones are rarely or never analyzed.

Appendix F includes the measurements using two other statistical distances, the Wasserstein Distance and the Generalized Jaccard

index. From all perspectives, statistically difficult DGA families receive very little attention in scientific papers.

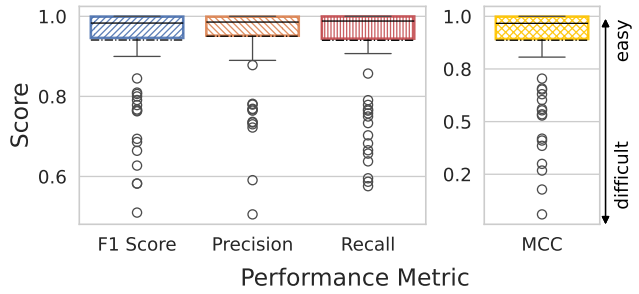

Figure 3: *Models' predictive performance when comparing the non-DGA dataset against each DGA family.* Each test evaluates each DGA family individually against the non-DGA dataset in a binary classification task. The gray spots denote the outliers, which are hard-to-detect families. The solid horizontal line is the median performance, and the dashed horizontal line is the average performance.

DGA	F1 Score	MCC
Dictionary DGA	0.582 ± 0.01	0.178 ± 0.01
CharBot	0.627 ± 0.01	0.299 ± 0.02
Permutation DGA	0.686 ± 0.01	0.419 ± 0.01
Khaos	0.694 ± 0.01	0.410 ± 0.01
Deception	0.764 ± 0.01	0.529 ± 0.01

Table 3: *Detailed information about hard-to-detect/outlier families from Figure 3.* The model evaluation confirms the difficulty of some adversarial and dictionary families, on par with the visual and statistical insights.

Understanding the difficulty of DGAs using ML benchmarks. Finally, to round up the discussion around this assumption, we benchmark the predictive performance of machine learning models when comparing non-DGA datasets against each DGA family. Figure 3 depicts the detection performance of each DGA family against a non-DGA dataset. For each boxplot, the middle line is the mean score across all families. As observed in the statistical tests, most families have distinctive statistical patterns. This is confirmed by the average scores line, located above 0.9 for all the metrics, meaning that most families are visibly distinct from non-DGA samples for classification models. Nevertheless, the presence of the outliers confirms what the statistical tests have shown: a handful of techniques - often overlooked in research papers - are exceptionally challenging to spot. We zoom in on some of the outliers in Table 3, showing five adversarial/dictionary AGDs that are hard to distinguish from non-DGA samples.

From the survey, only two papers employ t-SNE plots for comparing dictionary DGAs to the non-DGA dataset, but no paper visually compared adversarial DGAs (e.g., Khaos, Deception, etc.). Only one paper analyzed the evaluated DGAs statistically (i.e. character-level statistics), but without focusing on dictionary or adversarial DGAs.

Recommendations. The challenge of distinguishing DGAs from non-DGA samples varies depending on the family. Consequently, the difficulty of DGAs should be transparent before benchmarking a detection model. As a best practice, we suggest the researchers observe the similarity between DGA and non-DGA patterns by plotting the distributions with t-SNE and reporting statistical metrics such as Wasserstein distance, Chi-squared test, or Jaccard index.

A2: DGAs can be used only for malicious purposes.

Description. Assuming clear discrimination exists between AGDs and non-DGA domains, the next question is how to label the identified classes. Another widespread foundational assumption is that all AGDs can be grouped under a single detection tag (*benign* vs *malicious*). This assumption simplifies the detection procedure and implies that *any* AGD stems from malware. The premise is particularly fundamental in detection systems that use binary classification, i.e., distinguishing between AGDs and human-registered domains. Stacking all the DGA families under a single label, a binary classifier which identifies DGA vs non-DGA classes, can achieve an F1 score over 0.874. However, this approach leads to some subtle issues.

One notable downside of the general-purpose binary DGA detectors is the AGDs of benign origin. In their seminal work on DGA detection and discovery, Antonakakis et al. [9] highlight benign situations that trigger DGA detection, such as DNS network tests in browsers, domain typos, or some domains within the *.it, .gov, .edu* TLDs. Since then, 360 Netlab has reported benign DGA use cases for the TeleRU Android application (e.g., 93b375dd6cd9f2704d613d1016dbe0f2.tk) and the TcpRoute2 tool (e.g., 1492590065dshsdjhsdsgsevstyhndrdrntrtvsvstbruiuok095g.com). DGArchive includes another benign DGA family, DNSBenchmark - a DNS testing utility (e.g., y3ofpq5upb3uxaijydqk1uhr5h.com).

On one hand, the benign families do not attempt to create evasive samples. However, as the examples from each family suggest, benign DGAs create domains at the same DNS hierarchy level as malicious families. Consequently, if binary classification models were tested in open-world settings (e.g., in real networks), they could spot such benign AGDs, leading to false positives. This fundamental problem remains hidden when approaches are evaluated in closed-world settings with binary datasets. Worse, attackers could create evasive samples that resemble AGDs of benign sources. Nevertheless, identifying the exact family of the generator could help with further forensics.

One solution is to train a binary classifier per DGA family. The models can jointly be used in a prediction ensemble, where the output class is selected from the model with the highest confidence. This strategy offers granularity by explicitly detecting the exact DGA family, benign or malicious. The ensemble can also be easily extended with new families, as the models are trained independently. However, such a solution might hit scalability issues as the model count grows with the number of known DGA families.

Another viable solution is to train a multiclass ML model that jointly learns patterns per DGA family and outputs the most probable class. This version can simultaneously learn the representations from multiple DGA families, avoiding duplicate knowledge and the

overhead of multiple independent models. On the downside, any new DGA family requires a dedicated spot in the model's output layer, implying an architecture change (and retraining).

We can highlight the individual families and spot edge cases, such as benign DGAs, by evaluating the detection models in a multiclass setting. While this strategy is a step forward, it is not always bulletproof. The main reason is that botnet families can share DGAs (discussed further in **A9**), and benign DGAs might reuse generators from malicious variants. Nevertheless, those colliding families can still be flagged by a multiclass classifier and treated separately.

Recommendations. Future work should recognize benign use cases of DGAs and design their experiments accordingly. In particular, multiclass DGA detection - the ability to detect the exact DGA family - gives room for DGA detection of any origin, benign or malicious. DGA detection papers should include multiclass benchmarks. We further investigate multiclass classification best practices in **A7**.

A3: Botnets that contain DGAs will also use them.

100% papers

Description. Once the domains can be accurately classified, it is tempting to imply that the DGA classifiers can be used for botnet detection. Indeed, the implicit assumption of all surveyed papers is that botnets with DGA functionality will also use AGDs to find a C2 server. That is, the papers do *not* discuss that the botnet may use DGAs only as a backup mechanism that becomes active if bots can no longer reach their preconfigured (hardcoded) C2 server.

If the botnet's DGA functionality remains dormant, even a perfect classifier cannot detect the bot due to the lack of DGA activity—unless the primary C2 channel becomes inactive, e.g., in abandoned botnets.

We reviewed the available 92 families in DGArchive and their behavior. The main source of information are the reports on the botnets from their prime days, including [8, 13, 21, 64, 68, 87, 96]. Of these, 29 families first try to reach a hardcoded IP or domain before returning to the DGA. That is, ∼ 31% of known DGA botnet families only use the DGA as a backup channel. DGA-based detection of those families only becomes possible once the primary C2 mechanism is offline. Even more, this statistic is skewed and represents a lower bound, given that we analyze a dataset of botnets known to include DGAs. The bigger landscape of botnets, including those with hidden DGAs that have not become active, will likely increase this percentage even further.

We also conducted a cursory evaluation to estimate the prevalence of known malware families with DGAs. To this end, we collected information about all submissions to Malware Bazaar, an open malware-sharing platform popular among practitioners, between Oct. 2022 and Feb. 2024. Of 176,879 submissions, 152,984 had a signature tag identifying their malware family. Comparing the top 100 signatures (96.81%) of submissions against families listed in DGArchive, we noticed the presence of only five families: Emotet (1.14%), Qakbot (0.97%), Tofsee (0.93%), Gozi (0.65%), BumbleBee (0.10%). Further investigating the top-100 families by reviewing their sandbox runs, we noticed that only samples for BumbleBee actively use a DGA (as primary rallying mechanism), while the samples for all other families in the given versions rely on hardcoded IP addresses and domains to contact their C2s.

This analysis is not a representative image of the entire botnet landscape but tries to highlight some cases when the DGAs are just a fallback mechanism for contacting the C2. To our knowledge, a more comprehensive study about the C2 communication choices from all botnets is not yet available, and it would be beneficial to put the importance of DGA detection into perspective.

Recommendations. Researchers that propose DGA classifiers for malware detection should be fully transparent about the fact that DGAs oftentimes are just *backup* C2 channels. We recommend that future papers discuss the implications of this insight. That is, DGA classifiers cannot be described as a general botnet detection technique but rather should be put into perspective with respect to their true capabilities (e.g., finding infections of dormant botnets or when hardcoded C2 endpoints are temporarily unavailable).

A4: Bots using DGAs trigger NXDOMAIN responses, which are visible long before the C2 communication.

47% papers 45% 8%

Description. Given that DGA bots may attempt to resolve many AGDs, a common assumption is that DGA bots also resolve nonexistent domains and thus cause NXDOMAIN responses. Consequently, 27 papers suggest focusing (only) on NXDOMAINs and neglecting successful DNS lookups for efficient botnet detection. In this section, we review the main assumptions surrounding the value of NXDOMAINs in DGA detection and highlight why they are not entirely valid.

However, this assumption overlooks the many situations when botnets with DGAs cannot be detected based on NXDOMAIN responses. First and foremost, any DGA botnet for which a valid C2 server *can* be reached does not need to resolve potentially unregistered AGDs. This not only entails those ∼ 31% of botnets with dormant DGAs (see **A3**). Active DGAs that immediately reach registered C2 domains will not cause NXDOMAINs. Even more, if the domain generation rate is low, the attackers can also register all future domains as long as they are predictable. From the DGArchive families, two generate unpredictable DGAs (no predictable seed), 37 are time-independent DGAs, 8 generate domains every week (or less often), 39 generate new domains daily, and 6 create domains every few hours (less than a day). This implies that for 47 families, the attackers could likely register a domain daily, resulting in no failed DNS resolutions.

This strategy of pre-registering AGDs and "aging" them has been observed in malicious campaigns, with the technique being named registered domain generation algorithms (RDGA) ([28]). Some currently known campaigns which employ RDGAs include: (a) The Sparkle malware, which aged the AGDs for three months before using them ([28]); (b) The VexTrio malicious actor, which maintains over 70k registered AGDs ([54]); (c) A phishing campaign against the United States Postal Service which registers 40-160 new AGDs daily ([55]); or (d) The Prolific Puma, a malicious actor which provides link-shortener service based on AGDs for other malicious actors, maintains over 35k pre-registered AGDs ([53]).

Several existing DGA detection approaches successfully discover DGA usage only if the botnet causes NXDOMAINs. The stream of NXDOMAINs can indicate unknown DGA variants not part of any labeled training dataset [9]. However, this strategy effectively discovers dormant infections in abandoned botnets but is not necessarily suited for detecting active botnets. The domains generated by abandoned botnets can be helpful in reversing the generation algorithms and proactively detecting future reuses of the same generator. Nevertheless, this is not a complete solution.

In addition to these fundamental problems, ignoring successful DNS lookups and focusing on NXDOMAINs does not live up to the promises papers make. We observed two often-claimed benefits of NXDOMAINs. First, papers argue that the NXDOMAIN volume is significantly smaller and faster to process by detection systems. However, the volume of non-existing domains is still significant and varies between authoritative and recursive DNS servers. From the authoritative DNS point of view, [67] presents the DNS statistics for a 90-day window and reports that 9.9% of the traffic is from NXDOMAINS, meaning 744.9 billion requests; APNIC [10] reports that 50% turn into NXDOMAINs; ICANN [52] reports that 53% of the traffic results in non-existent domains. At the recursive DNS level, Cloudflare [24] reports ~ 5% of the traffic results in NXDOMAINs. All these reports suggest that the volume of NXDOMAINs is far from insignificant and cannot be used to support longer inference processing. Second, papers prefer NXDOMAINs from a privacy perspective, as they can allegedly prevent user identification. In the contextless detection setup, this precaution is not entirely justified. Instead, in the contextless setup, the models receive only a domain as input—without any personally identifiable information. Privacy leakages can occur in context-aware models, where the per-user activity is profiled for anomaly detection. To that end, the privacy benefit of the NXDOMAINs does not make any difference in the contextless character-level detection.

All the arguments hardly relate to the problem of botnet detection, and they can do more harm than good. In the contextless setup, the detection is non-trivial anyway (as highlighted in **A1** and **A2**), and any additional assumption makes it even more prone to errors and unlikely to be adopted.

Recommendations. There should not be a blind spot regarding registered domains. A registered domain might be parked in the future, creating an issue in an NXDOMAIN-only character-level DGA detector. At the same time, malicious actors can actively register and maintain AGDs (RDGAs). These scenarios can be problematic, and we suggest that papers openly discuss these limitations when using NXDOMAIN traffic only.

3.3 Experimental assumptions

After setting the threat model, most recent works focus on designing an ML-based solution. This section reviews three experimental assumptions commonly used in evaluating these solutions.

In particular, we discuss conceptual assumptions on clean training data (**A5**), dataset imbalances (**A6**), and representation of benchmarks (**A7**).

A5: The non-DGA class in the training data can be consistently cleaned of DGA examples.

74% papers 26%

Description. One primary concern in security-critical systems is excluding "corrupted non-DGA samples" from the training dataset. By corrupted non-DGA samples, we refer to samples that are controlled by an attacker, which can lead to evasion from detection for the DGA samples. Only a few papers discuss the risks, while most papers assume that training data cleanup can be achieved by applying simple preprocessing steps. Prior works [90, 92, 109] highlighted that Alexa could lead to a weakly labeled non-DGA dataset. Yu et al. [109] suggested that a heuristically engineered non-DGA dataset should be preferred. This section reviews three edge cases to clean up the training data.

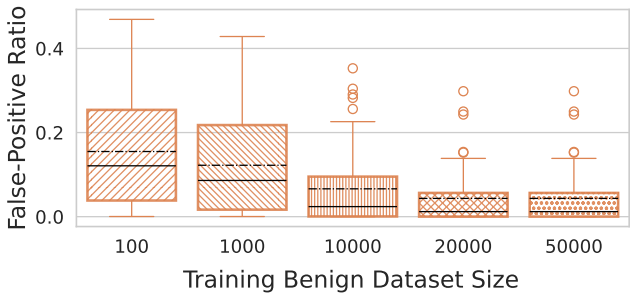

Figure 4: *False-Positive Ratio on unseen Tranco samples, by the number of selected top-N samples.* Each boxplot is constructed by training each DGA family individually against the selected top-N non-DGA samples and testing on 150k unseen Tranco samples. The dashed line is the average ratio, while the solid line is the median ratio across all families.

Keep only top-N non-DGA samples. The most prevalent approach is to keep only top-N examples from the non-DGA dataset (38 papers). This strategy can be biased by commercial trends and not robust enough for security applications [76, 94], and some papers apply further preprocessing steps. Even so, restricting only to top-N domains can omit valid non-DGA patterns. Figure 4 illustrates this issue. The plot is constructed by training all available models on Tranco top-N samples, where $N \in \{100, 1000, 10000, 20000, 50000\}$, and evaluating them on 150k unseen samples from the same dataset. We keep only the model with the smallest FP ratio for each (non-DGA sample size, family) combination and add it to the boxplot. As expected, the FP ratio is reduced with a larger non-DGA sample but never to zero. Using a 50k non-DGA samples sample size, the Deception, Matsnu and Suppobox families still inflict large false-positive ratios; 25%+ of unseen Tranco samples. While most of the false positives have unsuspecting forms, such as "myhomeweekly" or "nwfdailynews", some of the non-DGA samples look like "2d4c9479175de54fbecdb875fb6fbe4ddc6cf725" or "iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea". These samples are included in the Tranco list with the ".com" TLD, which again, raises the question: "What is a non-DGA sample at character level after all?"

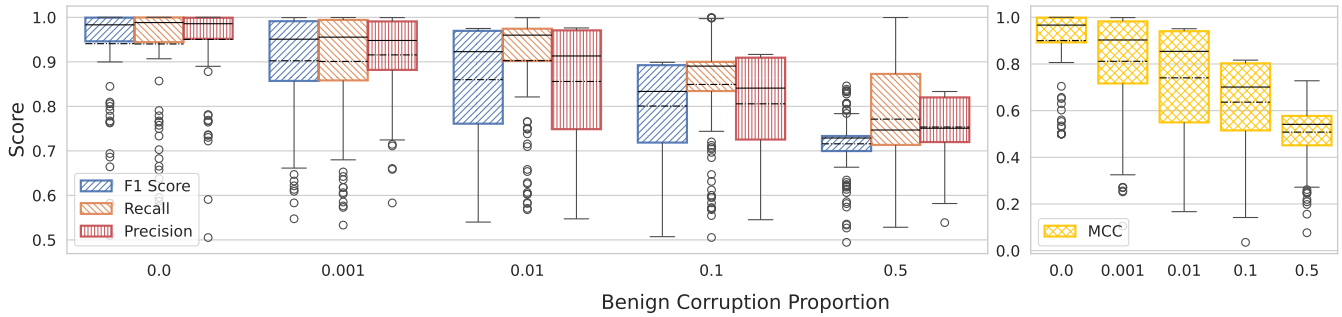

Figure 5: *The impact of corrupting the non-DGA dataset with DGA samples by the corruption ratio.* For each DGA family, we measure the impact of leaking some samples in the non-DGA dataset. The solid horizontal line is the median performance, and the dashed horizontal line is the average performance.

Filter out known blocklists. Another popular cleanup strategy is filtering out DGArchive samples from the "non-DGA" dataset (10 papers). While this method might clean up some known malicious samples, it misses out on the actual patterns of the generating algorithms. Some other domains in the "non-DGA" dataset might follow a known AGD pattern, yet they have not been observed in DGArchive. This leads to a corrupted non-DGA dataset, and the detection models might wrongfully learn to allowlist the patterns.

We test if a known DGA family can evade detection if it follows a pattern leaked in the non-DGA samples, and the results are presented in Figure 5. For each known family, we leak DGA samples as a proportion of the non-DGA data, include some other samples in the malicious training set, and re-evaluate the detection of new, unseen samples from the same family. We can see that only 0.1% of corrupted non-DGA samples affect the detection ability of some families, which otherwise were easy to detect. The examples include the Shifu and Vawtrak families. These families use an Arithmetic DGA, with samples like "ehimmun" or "fopwhwr". Yet, polluting the non-DGA dataset with a few of these samples forces the detector to learn this pattern as non-DGA, allowlisting the entire families. Going to a larger percentage of non-DGA corruption reduces the average performance even further, missing several other DGA families from detection.

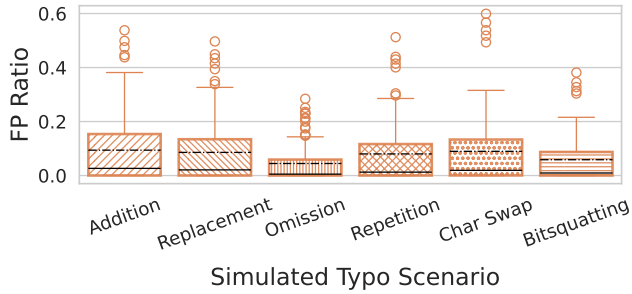

Figure 6: *False Positive Ratio on known Tranco samples with typos.* Each boxplot is constructed by individually training each DGA family against the top-20k Tranco samples and testing against six synthetic typo datasets with typo domains derived from the non-DGA training domains.

Filter out NXDOMAINs from the non-DGA samples. Another common strategy is to filter out NXDOMAINs from the non-DGA dataset (5+ papers). This approach can lead to concept drifts, as domains are registered and deregistered daily. Also, common human errors like typos can lead to false positives. We test the impact of innocent mistakes by creating six synthetic datasets out of the non-DGA training dataset, using common sources of human and computer errors - summarized in Figure 6: (1) Addition - adding a character to the domain by mistake (facebook → fabcebook); (2) Replacement - replacing one character in the domain with another (google → googxe); (3) Omission - one character is missing from the original domain (amazonaws → aazonaws); (4) Repetition - one character from the domain is repeated (youtube → youtubee); (5): Character swapping - two characters from the domain are interchanged (google → gooleg); (6) Bitsquatting - the generalization of the Replacement test case, where one bit from the binary representation of the domain is flipped (fbcdn → fbcdN). The Addition and Replacement scenarios have the highest average false positive ratio; ∼ 0.09 of the samples are marked as malicious. Another interesting aspect are in the outlier families, as different DGA families draw different decision boundaries in models. In the Addition typo test, the DGAs underneath the Pykspa, Nymaim, Proslikefan, Pushdo, Permutation DGA, and Deception2 families inflict a 0.3+ FP ratio. In the Omission test, Deception2, Virut, Permutation DGA, and dictionary DGAs like Suppobox and Gozi force a 0.2+ FP ratio on the non-DGA side. In the Repetition scenario, the families that inflict a high FP ratio (0.3+) are the ones that include repeated consecutive characters in their AGD: Symmi (28% samples with consecutive repetitions); Deception2 (22% samples with repetitions), and Matsnu (41% samples with repetitions). The Character-swapping test case has the highest FP rate among outliers, with the Deception and Permutation DGA families leading to over 50% false positive. The Replacement and Bitsquatting have similar families that affect the non-DGA detection performance, with Virut, Pykspa, Proslikefan, Conficker, and Deception causing a 0.3+ FP ratio in both scenarios.

Summarizing our discussion on all three preprocessing strategies, training dataset cleanup is not trivial and should not be treated only as excluding a static blocklist. ML models are famous for learning representations and patterns, and any corrupted non-DGA sample can have severe consequences.

Recommendations. A machine learning model is only as good as the data it is fed. Thus, a robust strategy for preparing and cleaning

the training dataset is paramount. We highlighted three augmentation strategies for better quality non-DGA data: (1) Select a larger sample size from the Tranco dataset; (2) Train a detector for each known DGA family and use them to manually inspect the non-DGA training data. Compared to simply excluding the known AGDs from the training set, this approach can leverage the learning capabilities of machine learning models to discover more samples following a similar pattern. (3) Augment the training dataset with synthetic non-DGA typos, for better resilience against adversarial samples. Future work should investigate and bring extensive guidelines for data preparation for DGA detection.

A6: The DGA population size per family does not affect the detector's performance.

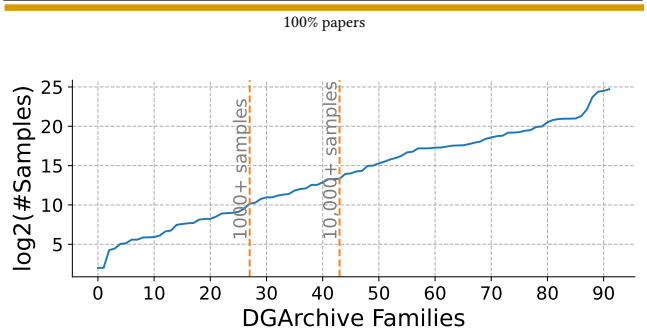

Figure 7: *Cumulative unique sample counts from the DGArchive families, in log2 scale.* 27 families have < 1000 samples available (first red dotted vertical line). 49 families have $\leq 10k$ samples (after the second red dotted vertical line)

Description. When evaluating detection models, one common approach is to use a fixed number of DGA samples, usually around 20,000 domains per family. Including such many domains is possible if the DGA was reverse-engineered and additional samples from the same algorithm can be generated. However, not all DGAs are well-documented, and models thus have to rely on limited datasets for some families. At the same, as pointed out in **A1**, different AGD families have different detection difficulties, hence a different threshold of sample count at which their pattern can be robustly detected. In other words, we should not use a global value for the population size, regardless of the DGA family, as this value can be impacted by the DGA difficulty.

Relying only on live collected traffic might not result in equally large training datasets. Various families have a low rate of domain generation, including Corebot, which generates 40 AGDs/month, AlienBot, which produces 3 AGDs/month, and CCleaner, which generates 1 AGD/month. This problem is more widespread than assumed. Figure 7 depicts the log2 sample sizes available in DGArchive, including $\sim$ 27 families with less than 1000 unique known examples, which can be challenging to use for generalization.

Consequently, we study if DGA detection remains possible under a DGA scarcity constraint. For several families, like hash-based or most arithmetic-based DGAs, the patterns can be visible from a few samples due to outstanding length and character statistics. However, Figure 8 highlights the behavior of 25 of the evaluated families, which are more challenging to spot with a low sample count. The affected families are either dictionary-based (e.g., Suppobox, Matsnu, Gozi), adversarial (e.g., Khaos, Deception, CharBot), permutation-based, and even arithmetic-based DGAs (e.g., Bobax, Conficker). While these families are trivial to detect using a large sample count (20k samples), their patterns are not clear enough for models with 100 samples, leading to a median F1 score of 0.6 and a median MCC score of 0.2. This underlines the importance of evaluating models with the DGA sample count sensitivity, as some family samples might be scarce.

Recommendations. A model's generalization capabilities depend on the number of samples per DGA family and the difficulty of detecting the AGD samples. As models are diverse, it is not feasible to suggest a minimum number of global training samples for all families. Instead, to adapt to the details of each model and scenario, we recommend performing a sensitivity analysis to transparently highlight when the results converge for each DGA family.

A7: The average score of multiclass classification sufficiently reflects the overall performance.

Description. 16/38 papers report the performance results in a multiclass classification setting. The approach is beneficial for identifying the exact DGA family - whether malicious or benign. The common approach is to report the F1 score for each label and report the unweighted mean - also known as the "macro" average. Unfortunately, reporting the macro averaging alone across labels can be misleading. Next, we will highlight a few shortcomings.

Consider the following scenario: Given N labels - the non-DGA class and $N-1$ DGA families - the multiclass F1 score works by setting each label as the positive label, while the rest of $N-1$ families are marked as the negative class. This can negatively affect the identification of the non-DGA class, as, at all times, it is either mixed with other DGAs in the negative class or compared against all the N - 1 DGA families as the positive class. The non-DGA class should not have the same weight or be reported separately - as we already pointed out in **A1**. To that end, we reiterate that it is critical to identify the non-DGA samples first before jumping into multiclass labeling.

However, this is not the only issue. Assuming that non-DGA samples have been identified and we only want to do multiclass classification between N DGA families, reporting only the F1 macro average might be misleading. Indeed, the macro average might conceal a bad performance of some DGAs (benign or malicious). For example, assume we have 5 DGA families, and the multiclass F1 scores are $[0.55, 0.5, 0.99, 0.99, 0.99]$. Reporting only the average score—0.804—would encourage us to believe the classification quality is reasonable.

Instead, the preferred output should include both the F1 macro-average and the *standard deviation* ($\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\bar{x})^2}{N}}$, where x_i is each performance score, $\bar{x}$ is the mean performance, and N is the number of DGA families). The standard deviation quantifies how much the scores deviate from the average value. A low σ indicates that the values are close to the mean, while a high σ indicates that the values are spread out over a broader range. This can be useful

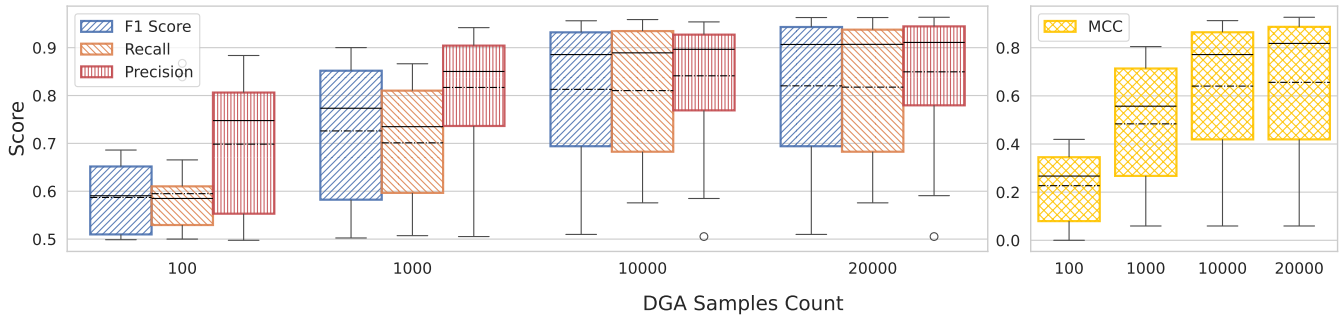

Figure 8: *Detection sensitivity to the available DGA sample count for 25 difficult families.* The horizontal solid and dashed lines report the median and average performance, respectively. The detection performance significantly depends on the training data size. For optimal performance, a model thus needs enough samples, e.g., as derived from a fully reverse-engineered DGA.

in spotting outlier scores. Reporting the standard deviation puts our example in a different light. An F1 score of 0.804 ± 0.22 clearly outlines that the model works only by chance for some families.

Note that the standard deviation is not the only possibility for spotting outliers. Other practical alternatives are the *variation*, the *95% confidence interval*, or the top-k lowest F1/MCC scores.

Metric	Avg. Score	Min	Median	Max
F1 Score	0.941 ± 0.10	0.510	0.983	1.0
Recall	0.940 ± 0.10	0.576	0.988	1.0
Precision	0.950 ± 0.09	0.505	0.986	1.0
MCC	0.887 ± 0.19	0.059	0.967	1.0

Table 4: *Improved multiclass benchmark reports, including the average, stddev, min, max, and median performance.* The average and deviations of the scores when benchmarking the 108 DGA families against non-DGA data are the same as in Figure 3.

For completeness, we revisit the results in Figure 3 from a numerical point of view. Again, we benchmark 108 DGA families and the non-DGA dataset. Table 4 reports the multiclass classification more transparently by highlighting both the average performance and the outliers. Instead of relying on the mean F1 score of 0.941, the standard deviation of 0.1 and the minimum score of 0.51 tell us a different story, with seven families having ≤ 0.7 F1 score. This transparency is critical for translating the research into practical applications.

Recommendations. Multiclass benchmarks have several benefits. However, to be completely transparent, we must augment the average performance with information about possible outlier classes (e.g., by providing stddev and discussing outlier scores).

3.4 Deployment Assumptions for Firewalls

The primary goal and promise of the DGA detection methods is practicality. To this end, these methods should seamlessly bridge the DNS firewalls and machine learning models. However, this link is unclear about how it should translate to practical situations, and we highlight a few edge cases.

A8: Detection models have negligible inference duration.

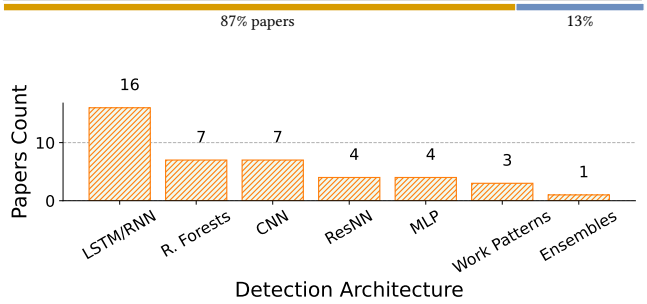

Figure 9: *Distribution of architectures used for detection in research papers.* Most papers rely on neural net-based architectures: MLP, RNN, LSTM, CNN, or ResNets. However, the impact on resource usage and usability in real-life firewalls is not discussed.

Description. The papers focus on well-known text classifiers. Figure 9 illustrates the choices of architectures in various detection papers. However, an often overlooked detail is how long the detection takes.

For DNS firewalls, the inference time is critical, and live DNS traffic analysis requires either significant resources or an efficient classifier. Some papers report their inference durations for a single domain: Koh and Rhodes [59] reported a 3ms inference duration on CPUs; Berman [16] reported 4.08ms for CNN, 21.58ms for an LSTM, 60ms for a bidirectional LSTM, and 2.86ms for a CapsNet on GPUs; Nie et al. [66] provide a compromise between performance and processing speed, with 1.46ms needed for 0.99 AUC and 0.51ms required for 0.69 AUC on CPUs.

We also benchmark our reproduced models separately, with the results summarized in Table 5. The benchmarks show the average and outliers for 1000 domains tested over ten rounds. We notice that the preprocessing steps alone could take significant time. The statistical feature extraction (required for logistic regression, SVM, random forest, or XGBoost) requires an average of 12.3ms. However, the implementation is not optimized and could be tailored per model. On the other hand, the tokenization step uses an optimized Rust backend (applicable with any model) and still requires 0.17ms processing time.

Model	Inference Time (ms)		
	min	mean	max
Feature Extraction	12.2 ± 0.5	12.3 ± 0.5	15.3 ± 2.1
Logistic Regression	0.12 ± 0.1	0.15 ± 0.1	1.91 ± 0.4
SVM	0.12 ± 0.1	0.13 ± 0.1	1.24 ± 0.3
Random Forest	2.96 ± 0.1	3.09 ± 0.1	6.09 ± 0.6
XGBoost	0.39 ± 0.1	0.41 ± 0.1	2.10 ± 0.5
Tokenization	0.16 ± 0.1	0.17 ± 0.1	3.20 ± 1.9
MLP	0.36 ± 0.1	0.38 ± 0.1	1.59 ± 0.4
LSTM	2.02 ± 0.1	2.29 ± 0.4	4.02 ± 1.8
CNN	1.29 ± 0.1	1.41 ± 0.1	3.10 ± 1.5
Transformer	1.80 ± 0.1	1.99 ± 0.1	24.79 ± 40.1
ResCNN	2.90 ± 0.1	3.17 ± 0.1	5.44 ± 1.3

Table 5: *Benchmarks on inference time for evaluating a single domain on a CPU.* The tests are averaged over 1000 domains.

Nevertheless, classification takes significant time even when these preprocessing steps are ignored. In fact, it never reaches more than 8000 domains per second in any model, with several models even being significantly slower.

When we relate these reported inference times to DNS server query volumes, we can infer whether the models can cope with real-world DNS data. The contextless detection methods should be beneficial mainly to large DNS servers (such as ISPs), as the detection needs to be as fast as possible on a big volume of data. From the authoritative DNS point of view, NS1 [67] reports that it processes an average of one million queries per second every second of every day. From a recursive resolver point of view, Cloudflare [24] reports that it processes 1.3T requests daily, translating it to ~ 2.1 million queries per second. Even in large corporate networks, the task is non-trivial, as the devices (workstations, servers, cloud replicas, or security tools) might generate millions of DNS requests per day. However, on a smaller scale, from a local network perspective (i.e., small and medium enterprises and consumer networks), the number of DNS queries and the resources available for filtering are significantly lower. An average household DNS workload can range between 1k-500k queries/day ([1, 2]), depending on the number of devices on the network and their type, which will make the models more practical.

Consequently, practical deployments may require a caching strategy for the DGA detectors, which none of the papers evaluates on real-world traffic.

Recommendations. A model is impractical if it is too expensive or impossible to operate for the system on which the DGA detection will be deployed. The detection solution should benchmark the inference times and possibly provide caching strategies for performance improvements.

A9: Models can generalize to unseen DGA families.

Description. The set of known DGA families increases over time. This represents an instantiation of the common problem of distributional shifts for ML models [51, 60, 78]. Trained detection models do not receive real-time feedback from malware intelligence sources. Hence, they risk missing new DGA families that are not part of the training. To cope with this problem, 4 papers argue that their model is good in practice if periodically retrained, which partially remedies this problem apart from several practical burdens. However, 10 papers argue that the detection model generalizes well to unseen DGA families, while the rest do not address the issue.

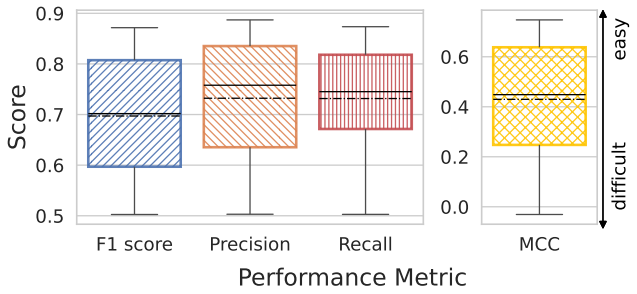

Figure 10: *Evaluation using one unseen DGA family in the test set.* While some DGA families have similarities with others and can still be detected, the general performance is degraded for unseen families. Any generalization claim does not hold for families with F1 score ≤ 0.7.

However, unseen DGA families can be recognized only if their generator creates similar (if not identical) patterns to a family in the training set. To support this argument, Figure 10 highlights the difficulty of spotting DGA families without prior knowledge. The evaluation is conducted on each DGA family, treated as an unseen family, with no examples in the training. The data suggests that many families do not share any obvious traits unless they are explicitly added to the training set. On the easy-to-spot side, families like Bamital, Banjori, Bedep, Blackhole, Chinad, Corebot, Cryptolocker, Dyre, Emotet, Gameover, MoneroMiner, Murofet, PandaBanker, Qakbot, Ramnit, Rovnix, or Tinba, can be detected with an F1 score ≥ 0.8, even without being previously observed. These families are either from the DGA Arithmetic or Hash families, and their patterns are similar. However, the papers claiming to generalize to unseen families should not only use such families as support for their claim. In stark contrast, families like Deception, Khaos, CharBot, Pitou, Nymaim2, or with basic permutations on the non-DGA dataset, have an F1 score ≤ 0.6, meaning they are prone to misclassification. Any paper claiming to generalize to unseen families should also discuss these families.

Figure 11 supports these temporal distribution shifts. It summarises the DGArchive families' temporal drifts by the years they were first observed, using the MCC metric. The prediction is good at first but steadily degrades when facing newer families.

Finally, another important but overlooked detail is that DGArchive includes botnet families, not necessarily DGA families. However, families might share the DGA generator or use a similar implementation. One known example is the Ares botnet family (2022), which uses the same DGA as QakBot (2013) with a different seed. Another set of families that share the DGA generator implementation is Tufik (2007), Murofet (2010), and GameoverP2P (2014). Ignoring these details might make us believe that a model generalizes to unseen families, while instead we just leak the test set from families that share the DGA core implementation.

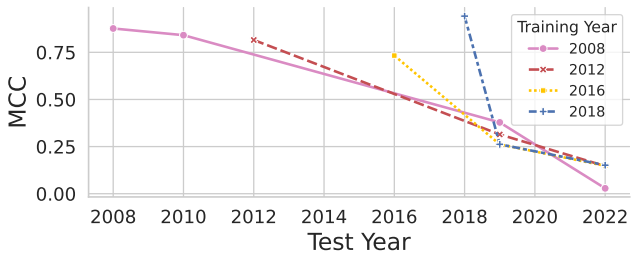

Figure 11: *Evaluation by years, using DGArchive.* The effects of training a model with data up to a year and testing it with future data. The training set includes samples up to the "training year", and the test set includes samples from the "test year".

Recommendations. As time passes, the models will see their accuracy affected by the emergence of new DGAs. Papers aiming for family-agnostic detection models should carefully evaluate whether they discover new DGA families. One subtle pitfall can be when unseen DGA families share the same domain generator underneath with a training DGA family, falsely supporting the generalization claim. Alternatively, the models could incorporate online learning support with a feedback loop for new malicious AGDs, or allow for incremental training with new data.

4 DISCUSSION

In this section, we reflect on the findings and discuss potential improvements for future works. We group the discussion around the three assumption categories: foundational (Section 4.1), experimental (Section 4.2), and deployment (Section 4.3).

4.1 The importance of factual information in problem statements

A proper foundation in the problem statements is critical and must be factually accurate. Otherwise, we risk getting a cascade effect of follow-up works or citations without proper sanity checks. Inheriting unchecked assumptions such as "NXDOMAINs are enough for DGA detection", or "DGA detection implies C2 detection" can confuse future attempts to improve the existing solutions. We find it crucial to present any DGA detection idea transparently, including all foundational limitations. DGA detection is not a panacea for botnet detection. While it has its value, for holistic botnet detection, DGA detection is only a small add-on to other orthogonal botnet detection approaches [43–45].

4.2 The need for reproducible benchmarks

Machine learning benchmarks, applied to other sciences (applied ML), can often be inconclusive. This lack of trust comes from multiple sources. Non-ML researchers can be skeptical about the clairvoyant abilities of black box models. Some reports naively use biased metrics affected by imbalance (AUCROC) or FP-rate (accuracy). Other reports fail to report a confidence interval for their results but only provide an average score. Another issue is the lack of standardized benchmarks for the problems of interest. Almost every paper we reviewed used a different subset of DGA families, a different non-DGA dataset snapshot, and different metrics. It is hard to compare and draw any conclusions in such an ecosystem. Unfortunately, only six papers we reviewed published their code, and none offered the details about the exact evaluation datasets (e.g., random seeds, sampling strategies). This practice blocks any reproducibility attempt and decreases trust in the results.

Additionally, the detection models are as good as their training data. As highlighted in **A5**, preparing the training data for cybersecurity tasks should be a meticulous and robust process. We highlighted some potential strategies for DGA detection tasks, but an in-depth analysis is required in future works.

4.3 Why is DGA academic research not translating well to real-life use cases?

Contextless character-level detection is a compromise between performance, memory usage, and promising results. However, the limitation is that it is hard to robustly link linguistic features to malicious intentions. At the same time, while it is an exciting opportunity to test the latest machine learning architectures, it is hard to define the exact target audience for these solutions.

If the targeted audience is the machine learning community, the lack of standardized benchmarks can make this task uninteresting. At the same time, the DGA research field has settled itself in the text classification problem. More interesting future research questions could be training data cleanup strategies, shrunk detection models, detection from a small number of samples, or finding new prediction indicators.

For the security research community, the solution's main limitation is its skewed usefulness. In reality, a DGA detector does not represent a general framework for detecting botnets (**A3**), but rather a small (sometimes optional) step. Another limitation is the performance overhead (**A8**); practical solutions require caching strategies to cope with common DNS traffic volumes. Ultimately, it is dangerous to derive malicious behavior only from character-level statistics: Unseen DGA families can be complicated to detect (**A1**, **A9**); and it is non-trivial to clean up the training set of corrupted samples (**A5**). To cope with this risk, any practical solution should use multiple indicators of compromise, including DNS data, ownership information, and temporal patterns.

5 CONCLUSION

Recent DGA detection research focused on character-level domain analysis, with an emphasis on ML techniques, yet overlooking some fundamental limitations. We revisited 38 papers on contextless DGA detection, and we analyzed nine popular assumptions belonging to three categories: (1) foundational and recurrent ideas, which fuel the research; (2) experimental assumptions, which should spotlight the added value; and (3) deployment assumptions, which should facilitate the adoption into practice. For each assumption, we outlined some overlooked perils and discussed some potential remedies.

Our findings provide a strong argument that botnet detection requires both ML and domain expert knowledge. On a foundational level, we outlined beliefs that domain experts should review as well, not just ML experts. These assumptions can skew the role of DGAs in botnet detection or recommend harmful optimizations (such as NXDOMAIN-only filtering), thus limiting future research.

On the experimental level, we highlighted dangerous yet overlooked details when reporting DGA detection results. Preparing a robust training dataset is a daunting task. Although we provide specific recommendations for improving the quality of the training dataset, we encourage future work to develop novel preprocessing techniques. At the same time, empirical studies should be transparent, e.g., highlighting the sensitivity to the sample counts or outstanding hard-to-detect families.

Finally, we outlined potential impediments that prevent these solutions from becoming practical, ranging from performance issues to temporal distribution shifts. Addressing these limitations requires collaboration between domain and ML experts, e.g., designing detection caching systems or online feedback loops, which could make the research practical.

We reiterate that DGA detection is part of a bigger problem—botnet detection. Any potential solution should be factual, adjusted to the expected impact, and transparent about the limitations of DGA detection in general. To that end, the research should not be limited to character-level classifiers but should explore novel sources of information and compatibility with non-DGA C2 detection technologies.

ACKNOWLEDGMENTS

We would like to thank Simon Ofner and the Fraunhofer FKIE Cyber Analysis & Defense department for granting us access to DGArchive. We want to thank our shepherd and reviewers for their insightful comments.

REFERENCES

[1] [n. d.]. GLOBAL DNS INFRASTRUCTURE. https://www.opendns.com/about/global-dns-infrastructure/.
[2] [n. d.]. Pi-hole reddit. https://www.reddit.com/r/pihole/comments/11ocub5/490k_dns_queries_per_day_whats_normalaverage/.
[3] Jasper Abbink and Christian Doerr. 2017. Popularity-based Detection of Domain Generation Algorithms. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM, Reggio Calabria Italy, 1–8. https://doi.org/10.1145/3098954.3107008
[4] Jawad Ahmed, Hassan Habibi Gharakheili, Craig Russell, and Vijay Sivaraman. 2022. Automatic Detection of DGA-Enabled Malware Using SDN and Traffic Behavioral Modeling. *IEEE Transactions on Network Science and Engineering* 9, 4 (July 2022), 2922–2939. https://doi.org/10.1109/TNSE.2022.3173591
[5] Sameer Ajmera and TR Pattanshetti. 2020. A survey report on identifying different machine learning algorithms in detecting domain generation algorithms within enterprise network. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 1–5.
[6] Mohammadhadi Alaeiyan, Saeed Parsa, Vinod P., and Mauro Conti. 2020. Detection of algorithmically-generated domains: An adversarial machine learning approach. *Computer Communications* 160 (July 2020), 661–673. https://doi.org/10.1016/j.comcom.2020.04.033
[7] Hyrum S. Anderson, Jonathan Woodbridge, and Bobby Filar. 2016. DeepDGA: Adversarially-Tuned Domain Generation and Detection. http://arxiv.org/abs/1610.01969 arXiv:1610.01969 [cs].
[8] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*. 1093–1110.
[9] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away traffic to bots: Detecting the rise of DGA-Based malware. In *21st USENIX Security Symposium (USENIX Security 12)*. 491–506.
[10] APNIC. [n. d.]. Measuring NXDOMAIN responses. https://blog.apnic.net/2023/07/12/measuring-nxdomain-responses/.
[11] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*. 3971–3988.
[12] Md. Ahsan Ayub, Steven Smith, Ambareen Siraj, and Paul Tinker. 2021. Domain Generating Algorithm based Malicious Domains Detection. In *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, Washington, DC, USA, 77–82. https://doi.org/10.1109/CSCloud-EdgeCom52276.2021.00024
[13] Johannes Bader. [n. d.]. bumblebee report. https://bin.re/blog/the-dga-of-bumblebee/.
[14] Moran Baruch and Gil David. 2018. Domain generation algorithm detection using machine learning methods. *Cyber security: power and technology* (2018), 133–161.
[15] Franziska Becker, Arthur Drichel, Christoph Muller, and Thomas Ertl. 2020. Interpretable Visualizations of Deep Neural Networks for Domain Generation Algorithm Detection. In *2020 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE, Salt Lake City, UT, USA, 25–29. https://doi.org/10.1109/VizSec51108.2020.00010
[16] Daniel S. Berman. 2019. DGA CapsNet: 1D Application of Capsule Networks to DGA Detection. *Information* 10, 5 (April 2019), 157. https://doi.org/10.3390/info10050157
[17] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. 2014. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Transactions on Information and System Security (TISSEC)* 16, 4 (2014), 1–28.
[18] Federica Bisio, Salvatore Saeli, Pierangelo Lombardo, Davide Bernardi, Alan Perotti, and Danilo Massa. 2017. Real-time behavioral DGA detection through machine learning. In *2017 International Carnahan Conference on Security Technology (ICCST)*. IEEE, Madrid, 1–6. https://doi.org/10.1109/CCST.2017.8167790
[19] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.
[20] Leo Breiman. 2017. *Classification and regression trees*. Routledge.
[21] Broadcom. [n. d.]. bamital report. https://docs.broadcom.com/doc/trojan-bamital-13-en.
[22] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
[23] Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics* 21 (2020), 1–13.
[24] Cloudflare. [n. d.]. Delegated and NXDOMAIN requests through Cloudflare DNS. https://stats.labs.apnic.net/cfnxdata/.
[25] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20 (1995), 273–297.
[26] David R Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 20, 2 (1958), 215–232.
[27] Ryan R. Curtin, Andrew B. Gardner, Slawomir Grzonkowski, Alexey Kleymenov, and Alejandro Mosquera. 2019. Detecting DGA domains with recurrent neural networks and side information. http://arxiv.org/abs/1810.02023 arXiv:1810.02023 [cs].
[28] Darby Wise. [n. d.]. [RDGAs: The New Face of DGAs. https://blogs.infoblox.com/cyber-threat-intelligence/rdgas-the-new-face-of-dgas/.
[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423
[30] Arthur Drichel, Nils Faerber, and Ulrike Meyer. 2021. First Step Towards EXPLAINable DGA Multiclass Classification. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*. ACM, Vienna Austria, 1–13. https://doi.org/10.1145/3465481.3465749
[31] Arthur Drichel, Mehdi Akbari Gurabi, Tim Amelung, and Ulrike Meyer. 2021. Towards privacy-preserving classification-as-a-service for DGA detection. In *2021 18th International Conference on Privacy, Security and Trust (PST)*. IEEE, 1–10.
[32] Arthur Drichel, Marc Meyer, and Ulrike Meyer. 2024. Towards Robust Domain Generation Algorithm Classification. *arXiv preprint arXiv:2404.06236* (2024).
[33] Arthur Drichel and Ulrike Meyer. 2023. False Sense of Security: Leveraging XAI to Analyze the Reasoning and True Performance of Context-less DGA Classifiers. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. ACM, Hong Kong China, 330–345. https://doi.org/10.1145/3607199.3607231
[34] Arthur Drichel, Ulrike Meyer, Samuel Schüppen, and Dominik Teubert. 2020. Analyzing the real-world applicability of DGA classifiers. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ACM, Virtual Event Ireland, 1–11. https://doi.org/10.1145/3407023.3407030
[35] Arthur Drichel, Ulrike Meyer, Samuel Schüppen, and Dominik Teubert. 2020. Making use of NXt to nothing: the effect of class imbalances on DGA detection classifiers. In *Proceedings of the 15th International Conference on Availability,*

Reliability and Security. ACM, Virtual Event Ireland, 1–9. https://doi.org/10.1145/3407023.3409190

[36] Arthur Drichel, Justus von Brandt, and Ulrike Meyer. 2022. Detecting unknown DGAs without context information. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*. 1–12.

[37] Xin Fang, Xiaoqing Sun, Jiahai Yang, and Xinran Liu. 2020. Domain-Embeddings Based DGA Detection with Incremental Training Method. http://arxiv.org/abs/2009.09959 arXiv:2009.09959 [cs].

[38] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. 2019. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 2681–2690.

[39] Yu Fu, Lu Yu, Oluwakemi Hambolu, Ilker Ozcelik, Benafsh Husain, Jingxuan Sun, Karan Sapra, Dan Du, Christopher Tate Beasley, and Richard R. Brooks. 2017. Stealthy Domain Generation Algorithms. *IEEE Transactions on Information Forensics and Security* 12, 6 (June 2017), 1430–1443. https://doi.org/10.1109/TIFS.2017.2668361

[40] Dragos Teodor Gavrilut, George Popoiu, and Razvan Benchea. 2016. Identifying DGA-Based Botnets Using Network Anomaly Detection. In *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, Timisoara, Romania, 292–299. https://doi.org/10.1109/SYNASC.2016.053

[41] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[42] Martin Grill, Ivan Nikolaev, Veronica Valeros, and Martin Rehak. 2015. Detecting DGA malware using NetFlow. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 1304–1309.

[43] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. 2008. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. (2008).

[44] Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. 2007. Bothunter: Detecting malware infection through ids-driven dialog correlation.. In *USENIX Security Symposium*, Vol. 7. 1–16.

[45] Guofei Gu, Junjie Zhang, and Wenke Lee. 2008. BotSniffer: Detecting botnet command and control channels in network traffic. (2008).

[46] Simon Haykin. 1994. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[48] Kate Highnam, Domenic Puzio, Song Luo, and Nicholas R. Jennings. 2021. Real-Time Detection of Dictionary DGA Network Traffic Using Deep Learning. *SN Computer Science* 2, 2 (April 2021), 110. https://doi.org/10.1007/s42979-021-00507-w

[49] Nazrul Hoque, Dhruba K. Bhattacharyya, and Jugal K. Kalita. 2015. Botnet in DDoS Attacks: Trends and Challenges. *IEEE Communications Surveys & Tutorials* 17, 4 (Fourthquarter 2015), 2242–2270. https://doi.org/10.1109/COMST.2015.2457491

[50] Xiaoyan Hu, Hao Chen, Miao Li, Guang Cheng, Ruidong Li, Hua Wu, and Yali Yuan. 2023. ReplaceDGA: BiLSTM-Based Adversarial DGA With High Anti-Detection Ability. *IEEE Transactions on Information Forensics and Security* 18 (2023), 4406–4421. https://doi.org/10.1109/TIFS.2023.3293956

[51] Chip Huyen. 2022. *Designing machine learning systems*. " O'Reilly Media, Inc.".

[52] ICANN. [n. d.]. M3: DNS Root Traffic Analysis. https://ithi.research.icann.org/graph-m3.html.

[53] Infoblox. [n. d.]. Infoblox. Prolific Puma: Shadowy Link Shortening Service Enables Cybercrime. https://blogs.infoblox.com/threat-intelligence/prolific-puma-shadowy-link-shortening-service-enables-cybercrime/.

[54] Infoblox. [n. d.]. Infoblox. VexTrio Operates Massive Criminal Affiliate Program. https://blogs.infoblox.com/threat-intelligence/cybercrime-central-vextrio-operates-massive-criminal-affiliate-program/.

[55] Infoblox. [n. d.]. Phishers Weather the Storm: The DNS Landscape of U.S. Postal Smishing Attacks. https://blogs.infoblox.com/threat-intelligence/phishers-weather-the-storm-the-dns-landscape-of-us-postal-smishing-attacks/.

[56] Haoran Jiao, Qing Wang, Zhaoshan Fan, Junrong Liu, Dan Du, Ning Li, and Yuling Liu. 2022. DGGCN: Dictionary based DGA detection method based on DomainGraph and GCN. In *2022 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, Honolulu, HI, USA, 1–10. https://doi.org/10.1109/ICCCN54977.2022.9868932

[57] Wazir Zada Khan, Muhammad Khurram Khan, Fahad T. Bin Muhaya, Mohammed Y. Aalsalem, and Han-Chieh Chao. 2015. A Comprehensive Study of Email Spam Botnet Detection. *IEEE Communications Surveys & Tutorials* 17, 4 (Fourthquarter 2015), 2271–2295. https://doi.org/10.1109/COMST.2015.2459015

[58] Aminollah Khormali, Jeman Park, Hisham Alasmary, Afsah Anwar, Muhammad Saad, and David Mohaisen. 2021. Domain name system security and privacy: A contemporary survey. *Computer Networks* 185 (2021), 107699.

[59] Joewie J. Koh and Barton Rhodes. 2018. Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, Seattle, WA, USA, 2966–2971. https://doi.org/10.1109/BigData.2018.8622066

[60] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*. PMLR, 5637–5664.

[61] Nikos Kostopoulos, Dimitris Kalogeras, Dimitris Pantazatos, Maria Grammatikou, and Vasilis Maglaris. 2023. SHAP Interpretations of Tree and Neural Network DNS Classifiers for Analyzing DGA Family Characteristics. *IEEE Access* 11 (2023), 61144–61160. https://doi.org/10.1109/ACCESS.2023.3286313

[62] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE, 253–256.

[63] Wanping Liu, Zhoulan Zhang, Cheng Huang, and Yong Fang. 2021. CLETer: A Character-level Evasion Technique Against Deep Learning DGA Classifiers. *ICST Transactions on Security and Safety* (Feb. 2021), 168723. https://doi.org/10.4108/eai.18-2-2021.168723

[64] media.blackhat.com. [n. d.]. suppobox report. https://media.blackhat.com/us-13/US-13-Geffner-End-To-End-Analysis-of-a-Domain-Generating-Algorithm-Malware-Family-WP.pdf.

[65] Juhong Namgung, Siwoon Son, and Yang-Sae Moon. 2021. Efficient Deep Learning Models for DGA Domain Detection. *Security and Communication Networks* 2021 (Jan. 2021), 1–15. https://doi.org/10.1155/2021/8887881

[66] Lihai Nie, Laiping Zhao, Keqiu Li, Xiaoyang Shan, and Tie Qiu. 2023. A Game-based Adversarial DGA Detection Scheme using Multi-level Incremental Random Forest. *IEEE Transactions on Network Science and Engineering* (2023), 1–13. https://doi.org/10.1109/TNSE.2023.3308126

[67] NS1. [n. d.]. Global DNS Traffic Report Insights into the Health of Networks in 2023. https://ns1.com/writable/resources/ns1_Global_DNS_Traffic_Report_2023.pdf.

[68] nsfocusglobal.com. [n. d.]. banjori report. https://nsfocusglobal.com/banking-trojan-banjori-analysis-report/.

[69] Kyle O'Meara, Deana Shick, Jonathan Spring, and Edward Stoner. 2016. Malware capability development patterns respond to defenses: Two case studies. *White Paper, Software Engineering Institute, Carnegie Mellon University* (2016), 1–11.

[70] Constantinos Patsakis and Fran Casino. 2021. Exploiting Statistical and Structural Features for the Detection of Domain Generation Algorithms. *Journal of Information Security and Applications* 58 (May 2021), 102725. https://doi.org/10.1016/j.jisa.2020.102725 arXiv:1912.05849 [cs].

[71] Jonathan Peck, Claire Nie, Raaghavi Sivaguru, Charles Grumer, Femi Olumofin, Bin Yu, Anderson Nascimento, and Martine De Cock. 2019. CharBot: A Simple and Effective Method for Evading DGA Classifiers. http://arxiv.org/abs/1905.01078 arXiv:1905.01078 [cs, stat].

[72] Mayana Pereira, Shaun Coleman, Bin Yu, Martine DeCock, and Anderson Nascimento. 2018. Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*. Springer, 295–314.

[73] Leif E Peterson. 2009. K-nearest neighbor. *Scholarpedia* 4, 2 (2009), 1883.

[74] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. 2016. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*. 263–278.

[75] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. 2016. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*. 263–278.

[76] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2018. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156* (2018).

[77] Morteza Safaei Pour, Christelle Nader, Kurt Friday, and Elias Bou-Harb. 2023. A comprehensive survey of recent internet measurement techniques for cyber security. *Computers & Security* 128 (2023), 103123.

[78] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2008. *Dataset shift in machine learning*. Mit Press.

[79] Vinayakumar Ravi, Mamoun Alazab, Sriram Srinivasan, Ajay Arunachalam, and K. P. Soman. 2023. Adversarial Defense: DGA-Based Botnets and DNS Homographs Detection Through Integrated Deep Learning. *IEEE Transactions on Engineering Management* 70, 1 (Jan. 2023), 249–266. https://doi.org/10.1109/TEM.2021.3059664

[80] Fangli Ren, Zhengwei Jiang, Xuren Wang, and Jian Liu. 2020. A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity* 3, 1 (Dec. 2020), 4. https://doi.org/10.1186/s42400-020-00046-6

[81] Christian Rossow, Dennis Andriesse, Tillmann Werner, Brett Stone-Gross, Daniel Plohmann, Christian J. Dietrich, and Herbert Bos. 2013. P2PWNED: Modeling and Evaluating the Resilience of Peer-to-Peer Botnets . In *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)* . San Francisco, CA.

[82] Christian Rossow, Christian J. Dietrich, Christian Kreibich, Chris Grier, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten van Steen. 2012. Prudent Practices for Designing Malware Experiments: Status Quo and Outlook . In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P)* . San Francisco, CA.

[83] Akihiro Satoh, Yutaka Fukuda, Toyohiro Hayashi, and Gen Kitagata. 2020. A Superficial Analysis Approach for Identifying Malicious Domain Names Generated by DGA Malware. *IEEE Open Journal of the Communications Society* 1 (2020), 1837–1849. https://doi.org/10.1109/OJCOMS.2020.3038704

[84] Akihiro Satoh, Yutaka Fukuda, Gen Kitagata, and Yutaka Nakamura. 2021. A Word-Level Analytical Approach for Identifying Malicious Domain Names Caused by Dictionary-Based DGA Malware. *Electronics* 10, 9 (April 2021), 1039. https://doi.org/10.3390/electronics10091039

[85] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-based botnet tracking and intelligence. In *International Conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 192–211.

[86] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. 2018. FANCI: Feature-based automated NXDomain classification and intelligence. In *27th USENIX Security Symposium (USENIX Security 18)*. 1165–1181.

[87] securityaffairs.com. [n. d.]. gozi report. https://securityaffairs.com/51744/malware/goznym-botnet-profiling.html.

[88] Jose Selvi, Ricardo J. Rodriguez, and Emilio Soria-Olivas. 2021. Toward Optimal LSTM Neural Networks for Detecting Algorithmically Generated Domain Names. *IEEE Access* 9 (2021), 126446–126456. https://doi.org/10.1109/ACCESS.2021.3111307

[89] Yong Shi, Gong Chen, and Juntao Li. 2018. Malicious domain name detection based on extreme machine learning. *Neural Processing Letters* 48, 3 (2018), 1347–1357.

[90] Lior Sidi, Yisroel Mirsky, Asaf Nadler, Yuval Elovici, and Asaf Shabtai. 2020. Helix: DGA Domain Embeddings for Tracking and Exploring Botnets. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, Virtual Event Ireland, 2741–2748. https://doi.org/10.1145/3340531.3416022

[91] Lior Sidi, Asaf Nadler, and Asaf Shabtai. 2020. MaskDGA: An Evasion Attack Against DGA Classifiers and Adversarial Defenses. *IEEE Access* 8 (2020), 161580–161592. https://doi.org/10.1109/ACCESS.2020.3020964

[92] Raaghavi Sivaguru, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, and Martine De Cock. 2018. An Evaluation of DGA Classifiers. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, Seattle, WA, USA, 5058–5067. https://doi.org/10.1109/BigData.2018.8621875

[93] Raaghavi Sivaguru, Jonathan Peck, Femi Olumofin, Anderson Nascimento, and Martine De Cock. 2020. Inline Detection of DGA Domains Using Side Information. *IEEE Access* 8 (2020), 141910–141922. https://doi.org/10.1109/ACCESS.2020.3013494

[94] Jan Spooren, Davy Preuveneers, Lieven Desmet, Peter Janssen, and Wouter Joosen. 2019. Detection of algorithmically generated domain names used by botnets: a dual arms race. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. ACM, Limassol Cyprus, 1916–1923. https://doi.org/10.1145/3297280.3297467

[95] Hatma Suryotrisongko, Yasuo Musashi, Akio Tsuneda, and Kenichi Sugitani. 2022. Robust Botnet DGA Detection: Blending XAI and OSINT for Cyber Threat Intelligence Sharing. *IEEE Access* 10 (2022), 34613–34624. https://doi.org/10.1109/ACCESS.2022.3162588

[96] threatpost.com. [n. d.]. matsnu report. https://threatpost.com/matsnu-botnet-dga-discovers-power-of-words/109426/.

[97] Duc Tran, Hieu Mac, Van Tong, Hai Anh Tran, and Linh Giang Nguyen. 2018. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* 275 (Jan. 2018), 2401–2413. https://doi.org/10.1016/j.neucom.2017.11.018

[98] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[99] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[100] R Vinayakumar, KP Soman, Prabaharan Poornachandran, Mamoun Alazab, and Sabu M Thampi. 2019. Amritadga: a comprehensive data set for domain generation algorithms (dgas) based domain name detection systems and application of deep learning. In *Big Data Recommender Systems: Application Paradigms*. Institution of Engineering and Technology, 455–485.

[101] Gernot Vormayr, Tanja Zseby, and Joachim Fabini. 2017. Botnet Communication Patterns. *IEEE Communications Surveys & Tutorials* 19, 4 (Fourthquarter 2017), 2768–2796. https://doi.org/10.1109/COMST.2017.2749442

[102] Harald Vranken and Hassan Alizadeh. 2022. Detection of DGA-Generated Domain Names with TF-IDF. *Electronics* 11, 3 (Jan. 2022), 414. https://doi.org/10.3390/electronics11030414

[103] Tzy-Shiah Wang, Hui-Tang Lin, Wei-Tsung Cheng, and Chang-Yu Chen. 2017. DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. *Computers & Security* 64 (Jan. 2017), 1–15. https://doi.org/10.1016/j.cose.2016.10.001

[104] Yu Wang, Rui Pan, Zuchao Wang, and Lingqi Li. 2023. A Classification Method Based on CNN-BiLSTM for Difficult Detecting DGA Domain Name. In *2023 IEEE 13th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. IEEE, Beijing, China, 17–21. https://doi.org/10.1109/ICEIEC58029.2023.10200702

[105] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. 2016. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. http://arxiv.org/abs/1611.00791 arXiv:1611.00791 [cs].

[106] Sandeep Yadav and AL Narasimha Reddy. 2011. Winning with DNS failures: Strategies for faster botnet detection. In *International Conference on Security and Privacy in Communication Systems*. Springer, 446–459.

[107] Luhui Yang, Guangjie Liu, Yuewei Dai, Jinwei Wang, and Jiangtao Zhai. 2020. Detecting Stealthy Domain Generation Algorithms Using Heterogeneous Deep Neural Network Framework. *IEEE Access* 8 (2020), 82876–82889. https://doi.org/10.1109/ACCESS.2020.2988877

[108] Luhui Yang, Jiangtao Zhai, Weiwei Liu, Xiaopeng Ji, Huiwen Bai, Guangjie Liu, and Yuewei Dai. 2019. Detecting Word-Based Algorithmically Generated Domains Using Semantic Analysis. *Symmetry* 11, 2 (Feb. 2019), 176. https://doi.org/10.3390/sym11020176

[109] Bin Yu, Daniel L. Gray, Jie Pan, Martine De Cock, and Anderson C. A. Nascimento. 2017. Inline DGA Detection with Deep Networks. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, New Orleans, LA, 683–692. https://doi.org/10.1109/ICDMW.2017.96

[110] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. 2018. Character Level based Detection of DGA Domain Names. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Rio de Janeiro, 1–8. https://doi.org/10.1109/IJCNN.2018.8489147

[111] Xiaochun Yun, Ji Huang, Yipeng Wang, Tianning Zang, Yuan Zhou, and Yongzheng Zhang. 2020. Khaos: An Adversarial Neural Network DGA With High Anti-Detection Ability. *IEEE Transactions on Information Forensics and Security* 15 (2020), 2225–2240. https://doi.org/10.1109/TIFS.2019.2960647

[112] Xiaodong Zang, Jian Gong, and Ping Zong. 2021. Identifying DGA Malware via Behavior Analysis. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Nanjing, China, 1–6. https://doi.org/10.1109/WCNC49053.2021.9417570

[113] Zhibo Zhang, Hussam Al Hamadi, Ernesto Damiani, Chan Yeob Yeun, and Fatma Taher. 2022. Explainable artificial intelligence applications in cyber security: State-of-the-art in research. *IEEE Access* 10 (2022), 93104–93139.

[114] Kejun Zhao, Wei Guo, Fenglin Qin, and Xinjun Wang. 2022. D3-SACNN: DGA Domain Detection With Self-Attention Convolutional Network. *IEEE Access* 10 (2022), 69250–69263. https://doi.org/10.1109/ACCESS.2021.3127913

[115] Shaofang Zhou, Lanfen Lin, Junkun Yuan, Feng Wang, Zhaoting Ling, and Jia Cui. 2019. CNN-based DGA Detection with High Coverage. In *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, Shenzhen, China, 62–67. https://doi.org/10.1109/ISI.2019.8823200

[116] Yonglin Zhou, Qing-shan Li, Qidi Miao, and Kangbin Yim. 2013. DGA-Based Botnet Detection Using DNS Traffic. *J. Internet Serv. Inf. Secur.* 3, 3/4 (2013), 116–123.

A SURVEY METHODOLOGY

Review process. This section discusses the survey process and the criteria for analyzing and labeling each reviewed paper.

Two authors independently analyzed each paper using the questionnaire included in this section. In case of disagreement, a third author analyzed the uncertain assumption status to break the tie. The following paragraphs outline our internal survey questionnaire and label criteria per assumption. Note that we do not include details per surveyed paper in our work. On the one hand, doing so would provide full transparency and foster reproducibility. On the other, we would like to avoid negative implications for individual surveyed papers. In fact, we do not want to point out flaws in individual papers but survey the state of the overall literature body. This type of data representation is also in line with prior related

studies such as [11] or [82]. Authors of surveyed papers can contact us at any time to learn how we categorized their papers.

Questionnaire for A1: AGDs' structural patterns differ from non-DGA samples, regardless of the generator family.
Survey Questions:
- *Q1A:* Are the distribution of difficult DGAs (i.e. dictionary DGAs and adversarial DGAs) visually compared to the non-DGA data?
- *Q1B:* Are the distributions of difficult DGAs (i.e. dictionary DGAs and adversarial DGAs) statistically compared to the non-DGA data?
- *Q1C:* Is the paper explicitly claiming that the structural patterns of any DGA can be distinguished from the non-DGA data?

Labeling Criteria. Each detection paper introduces a novel detection method, thus implicitly assuming partial discrimination is possible between DGA and non-DGA distributions. However, if the difficult DGA families are omitted from the discussion, we consider it a 'shortcut' to get easy decision boundaries, thus an explicit **A1** assumption. If there is an explicit claim to distinguish any DGA families in the text body, we consider it an explicit assumption. In short, the assumption will be considered:
- *Q1A* = FALSE and *Q1B* = FALSE $\implies$ **assumed**.
- *Q1C* = TRUE $\implies$ **assumed**.
- otherwise, $\implies$ **not assumed**.

Questionnaire for A2: DGAs can be used only for malicious purposes.
- *Q2A*: Is the paper discussing the use of DGA only in malicious conditions?
- *Q2B*: Is the paper experimenting only with binary classification?

Labeling Criteria. If the experimental section of the paper handles only binary classification, this implies the generator family cannot be individually detected, including DGA families or benign origin. Otherwise, multiclass DGA classification leaves room for handling DGA families of any origin. At the same time, ignoring DGAs of benign sources can have a negative impact on the detection model performance, as it can lead to a significant number of false positives. In short, the assumption will be considered:
- *Q2A* = TRUE and *Q2B* = TRUE $\implies$ **assumed**.
- otherwise, **not assumed**.

Questionnaire for A3: Botnets that contain DGAs will also use them.
Survey Questions:
- *Q3A*: Does the paper mention the possibility that botnets with DGA support may actually not use the AGDs to establish contact with the C2?

Labeling Criteria. A3 investigation focuses on the introductory part of the papers, searching for any discussion around the use of DGAs as backup channels, and how to circumvent this edge case.
- *Q3A* = FALSE $\implies$ **assumed**.
- *Q3A* = TRUE $\implies$ **not assumed**.

Questionnaire for A4: Bots using DGAs trigger NXDOMAIN responses, which are visible long before the C2 communication.
Survey Questions:
- *Q4A*: Is the paper explicitly claiming that every DGA will generate a list of non-existent domains before contacting the C2?

Labeling Criteria. A4 analysis focuses on the papers' introductions and experimental section, looking for any quote that NXDOMAIN pre-filtering is enough for DGA or C2 detection. In short, the assumption will be considered:
- *Q4A* = TRUE $\implies$ **assumed**.
- *Q4A* = FALSE $\implies$ **not assumed**.
- no discussion $\implies$ **omitted**.

Questionnaire for A5: The non-DGA class in the training data can be consistently cleaned of DGA examples.
Survey Questions:
- *Q5A*: Is the paper discussing the possibility of attacker-controlled samples in the non-DGA datasets (Tranco, Alexa, etc)?
- *Q5B*: Is the paper taking precautions against corrupted non-DGA data?

Labeling Criteria. This section focuses on the non-DGA dataset creation and curation for the experimental sections in papers. The analysis looks for any preprocessing steps and precautions against corrupt non-DGA samples (e.g., filter out the DGArchive samples from the non-DGA dataset, train a filtering model to curate non-DGA data, etc.). In short, the assumption will be considered:
- *Q5A* = TRUE or *Q5B* = TRUE $\implies$ **not assumed**.
- otherwise, $\implies$ **omitted**.

Questionnaire for A6: The DGA population size per family does not affect the detector's performance.
Survey Questions:
- *Q6A*: Is the paper benchmarking each DGA family at various sample counts?

Labeling Criteria. This section looks for any discussion or sensitivity analysis around the sample counts per DGA family. In short, the assumption will be considered:
- *Q6A* = TRUE $\implies$ **not assumed**.
- otherwise, $\implies$ **omitted**.

Questionnaire for A7: The average score of multiclass classification sufficiently reflects the overall performance.
Survey Questions:
- *Q7A*: Is the paper benchmarking binary classification only?
- *Q7B*: Does the paper report the multiclass classification's confidence intervals?
- *Q7C*: Is the paper reporting the outlier DGA families, i.e. which are the hardest and easiest to predict for the proposed detection model?

Labeling Criteria. The analysis extends the discussion from A2 to a general experimental setup and looks for transparency in the hard-to-detect domain generators. If the papers work only with binary classification, we will consider the assumption omitted. If the papers work with multiclass classification but do not report confidence intervals or outliers, we consider A7 assumed. Otherwise, if there is transparency about the prediction edge cases, we consider this 'not assumed'. In short, the assumption will be considered:
- *Q7A* = TRUE $\implies$ **omitted**.
- *Q7A* = FALSE and (*Q7B* = FALSE and *Q7C* = FALSE) $\implies$ **assumed**.

- *Q7A* = FALSE and (*Q7B* = TRUE or *Q7C* = TRUE) ⟹ **not assumed**.

Questionnaire for A8: Detection models have negligible inference duration.
Survey Questions:
- *Q8A*: Is the paper benchmarking the inference time overhead of the proposed detection model?

Labeling Criteria. All the papers introduce novel ML architectures, and the section looks for transparency regarding the expected delay in the model prediction logic. In short, the assumption will be considered:
- *Q8A* = TRUE ⟹ **not assumed**.
- otherwise, ⟹ **omitted**.

Questionnaire for A9: Models can generalize to unseen DGA families.
Survey Questions:
- *Q9A*: Is the paper explicitly claiming the detection can generalize to unseen DGA families without retraining?

Labeling Criteria. The section investigates the generalization claims to unseen domain generators and the benchmarking methodology. In short, the assumption will be considered:
- *Q9A* = TRUE ⟹ **assumed**.
- *Q9A* = FALSE ⟹ **not assumed**.
- No discussion about generalization ⟹ **omitted**.

B DATA AVAILABILITY

To ease reproducibility, we have released the code that we used to (1) create the evaluation datasets, (2) create the synthetic test cases, and (3) conduct the benchmark analyses.

https://github.com/bcebere/Guidelines-for-DGA-based-Malware-Detection.

C METRICS USED BY DGA DETECTION PAPERS

While comparing the benchmarks with prior work, we reviewed the choice of metrics for presenting the results. Figure 12 summarizes the reported metrics from other empirical studies.

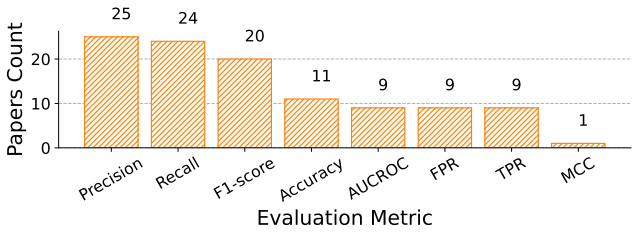

Figure 12: *Distribution of evaluation metrics used in related work*

D DATA REPRESENTATION

For models like Random Forest and Logistic Regression, the expected input is tabular, with numerical or categorical feature values. We follow the approach in [30] and extract statistical features from the domains. These features include string entropy information, n-gram statistics, character statistics, and string length. This preprocessing outputs 136 features and is used for training and inference for some models.

For neural-network-based models, we rely on text embedding tokenizers. A tokenizer prepares the input strings by splitting them into tokens and encoding the tokens in a numerical representation digestible by the neural networks. For the tokenization, we use a pre-trained BERT tokenizer [29], trained using the English dictionary.

E DATASETS DESCRIPTION

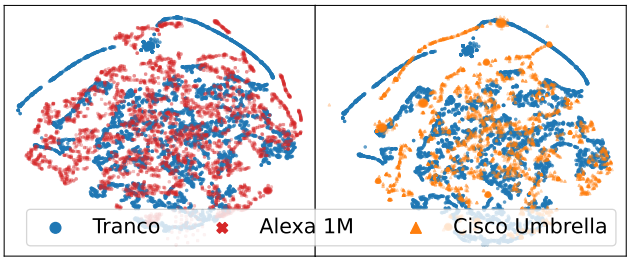

Figure 13: *t-SNE plots comparing the distributions of non-DGA datasets.* Reference plot comparing the distributions of similar but not identical datasets.

non-DGA Datasets. The related work relies on three non-DGA datasets: (1) Alexa 1M domain dataset, the most popular non-DGA option, which is formed of the most popular domain queries over HTTP but which was discontinued in 2022; (2) Cisco Umbrella, a list of the most popular domains queried over any protocol in the Umbrella global network; and (3) Tranco [76], a research-oriented and curated domain list. While the Alexa and Cisco Umbrella datasets are used in 32 related papers, an extensive study from the Tranco whitepaper showed how these lists can be contaminated with adversarial malicious samples. On top of that, their content changes frequently, and their bias towards commercial trends makes them inadequate for conclusive security reports. To that end, we employ the Tranco dataset as the non-DGA reference. We observe the difference in sources from the e2LD intersections from the top 5,000 domains in each list: Tranco ∩ Alexa = 1421 e2LD, Tranco ∩ Cisco = 837 e2LD, and Alexa ∩ Cisco = 170 e2LD. Despite this small overlap, we find that the domain lists are statistically similar. To illustrate this, we compare the statistical features of the three datasets in a t-SNE plot in Figure 13. We can observe a relatively small distance in the t-SNE representations but not a significant overlap.

Malicious Datasets. DGArchive [75] is the most popular source of AGD samples, with 19 papers from related work relying on it. Another nine papers rely on the Open-Source Intelligence (OSINT) DGA feed from Bambenek Consulting, two papers on the DGA feed from Network Security Research Lab at 360, while the rest work on synthetic DGA data. DGArchive is the most complete collection of DGAs and is still maintained, so we use it in our study.

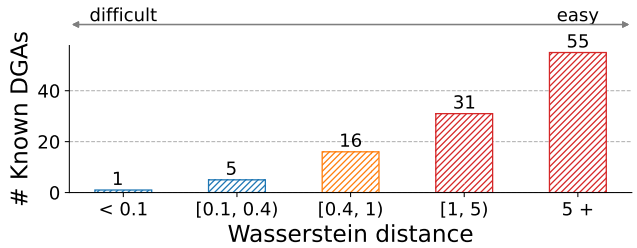

Figure 14: *The Wasserstein distance between DGA families and the non-DGA dataset (Tranco), grouped by difficulty.* Wasserstein Distance is a measure of the distance between two probability distributions - the effort to convert one distribution into another. Blue bins denote families that are difficult to detect, while red bins include statistically different DGA families compared to non-DGA samples.

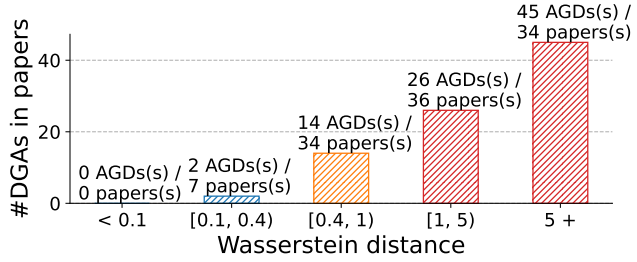

Figure 15: *The Difficulty of the DGA per papers, using the Wasserstein distance between the non-DGA dataset (Tranco) and the DGA families.* The count of unique DGAs in the experimental sections of related work, grouped by the Wasserstein distance.

F NON-DGA VS. DGAS: ADDITIONAL STATISTICS

Understanding the difficulty of DGAs using the Wasserstein Distance. Another statistical metric is the Wasserstein distance between two distributions, which intuitively shows the minimum effort to convert one distribution into another. Concretely, the general formula is $W_p(\mu,\nu) = \left(\inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^d\times\mathbb{R}^d} \|x-y\|^p d\pi(x,y)\right)^{\frac{1}{p}}$, where μ,ν are the distributions we want to compare, $\Pi(\mu,\nu)$ are all the transport strategies, p is the order of the Wasserstein distance, and $\|x-y\|^p$ is the distance between two points of the distributions in the Euclidean space. We evaluate the Wasserstein distance using Sinkhorn iterations [38], an iterative algorithm for solving the optimal transport problem.

We employ this metric to understand the difficulty of the DGA families by measuring the distance from each DGA family compared to the Tranco non-DGA dataset. Concretely, we measure the cost of transforming the Tranco non-DGA dataset into each one of the available DGA families. Figure 14 summarizes these results. A smaller cost means that the DGA is very similar to the non-DGA dataset and might be difficult to spot, while a large value means that the DGA stands out in contrast to the non-DGA data. We can observe that a small set of DGAs has a significant difficulty: variants permutation DGAs constructed on top of Tranco and the English dictionary, dictionary-based DGAs using multiple sources (English dictionary, non-DGA domains), and the Deception DGA. Figure 15 compares this information with the unique DGA families' presence in the experimental section of the DGA research papers. Each bucket represents unique DGA families, but a paper might be contained in multiple buckets. Figure 15 illustrates the tendency to use "easy" DGAs in the experimental section and rarely/never evaluate difficult classes.

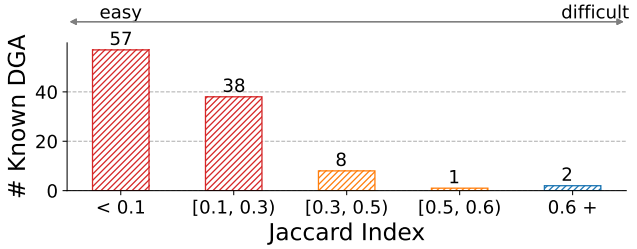

Figure 16: *Jaccard Index test between the non-DGA dataset and the DGA families.* The Jaccard similarity measures the similarity between two sets. The sets are constructed from 2-grams and 3-grams of the domains in the datasets.

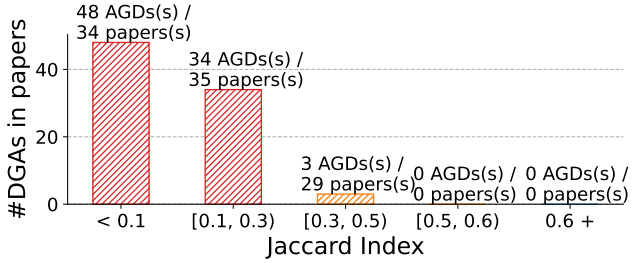

Figure 17: *The Difficulty of the DGA per papers, using the Jaccard Index the non-DGA dataset (Tranco) and the DGA families. The count of unique DGAs in the experimental sections of related work, grouped by the Jaccard Index.*

Understanding the difficulty of DGAs using the Generalized Jaccard Index. Lastly, we include the Generalized Jaccard Index score, which measures the similarity and diversity of two multisets, where each element can occur multiple times. The formula is $GJI(A,B) = \frac{\sum_{x\in A\cap B}\min(\text{count}(x,A),\text{count}(x,B))}{\sum_{x\in A\cup B}\max(\text{count}(x,A),\text{count}(x,B))}$, where A,B are the two evaluated datasets. We prepare each dataset for this metric by collecting n-grams from the domains of each dataset. Concretely, the metric is 0 for unrelated and 1 for identical datasets, and it has been a popular choice for highlighting potential DGAs in the past.

Figure 16 and Figure 17 illustrate the Generalized Jaccard Index results, a similarity metric evaluated (here) on sets of 2-grams and 3-grams of the non-DGA dataset and the DGA families. A value closer to zero means unrelated data sets, and a value of 1 means identical sets of n-grams.