Effective DGA Family Classification using a Hybrid Shallow and Deep Packet Inspection Technique on P4 Programmable Switches

Ali AlSabeh*, Kurt Friday†, Jorge Crichigno*, Elias Bou-Harb†
*College of Engineering and Computing, University of South Carolina (USC), Columbia, South Carolina, USA
†The Cyber Center For Security and Analytics, Information Systems and Cyber Security Dept.
University of Texas at San Antonio (UTSA), San Antonio, Texas, USA
Email: *aalsabeh@email.sc.edu, †kurt.friday@utsa.edu, *jcrichigno@cec.sc.edu, †elias.bouharb@utsa.edu

***Abstract*—Domain Generation Algorithms (DGAs) are one of the most effective strategies for malware to obtain a connection with the adversary's Command and Control (C2) server. Moreover, the growing number of DGA families makes it increasingly challenging for defense strategies to promptly identify the DGA family behind a given compromise. State-of-the-art high-dimensional DGA detection models perform poorly in such multiclass classification scenarios because their domain name-based features fail to distinguish between DGA families.**

To this extent, this paper proposes a novel framework that harnesses the flexibility, per-packet granularity, and Terabits per second (Tbps) processing capabilities of P4 Programmable Data Plane (PDP) switches to swiftly and accurately classify DGA families. In particular, the P4 PDP switch is leveraged to extract a combination of unique network heuristics and domain name features through shallow and Deep Packet Inspection (DPI) with minimal throughput reduction. Such collected features cannot be tracked on commodity hardware without significantly degrading the throughput in high-speed networks, nor on traditional layer 2/3 switches due to their limited and fixed functionalities. We crawled hundreds of Gigabytes (GBs) of malware samples from different sources to obtain instances of 50 DGA families and show that the proposed approach can promptly classify each family with high accuracy. Such a reliable multiclass classification enables the immediate halting of malicious communications while allowing network operators to initiate appropriate mitigation, incident management, and provisioning strategies.

***Index Terms*—DGA family classification, botnets, P4 programmable switches, high-speed networks, DPI**

I. Introduction

A common evasive approach that is increasingly being relied upon by modern malware variants to communicate with their C2 servers is to employ a DGA [1, 2]. DGAs evade domain-based firewall controls by frequently changing the domain name selected from a large pool of candidates (reaching tens of thousands of domains per day). To determine the rendezvous point(s), the malware makes Domain Name System (DNS) queries in an attempt to resolve the IP addresses of these generated domains, where only a few IPs will typically be registered and associated with the C2. Non-Existent Domain (NXD) responses will coincide with the remainder of the DNS queries, denoting that the domain is not registered or the DNS server could not resolve it [3].

Currently, most DGA detection approaches perform binary classification in order to segregate DGAs from benign traffic. Generally, such approaches either rely on contextual network traffic collected retrospectively (context-aware) or analyze features of the domain name without depending on other metadata (context-less). Beyond the task of DGA detection is the desire to attribute DGA families, which is a much harder problem than binary classification of DGAs. This is the result of having to cope with multiple classes instead of just two classes (binary) within the context of the detection problem. Essentially, the multiclass classification of DGA families allows security professionals to assess the severity of the exploit and apply the appropriate remediation policies in the network [1]. While context-less approaches can obtain high accuracy, they require a general-purpose Central/Graphics Processing Unit (CPU/GPU) to process and analyze the domain names, which could create a bottleneck and significantly impact throughput due to the ubiquitous use of DNS on the Internet. On the other hand, context-aware approaches can be slow since they typically analyze batches of traffic offline (e.g., using NetFlow [4]).

To address the aforementioned shortcomings, this paper proposes a novel framework for performing the multiclass classification of DGA families via the P4 PDP technology. The P4 language [5] is utilized to describe in software (i.e., the P4 program) the behavior dictating how packets are processed on the switch. Such programming allows network operators to customize their data plane to suit the requirements of their network. The P4 language overcomes the limitations of the traditional Software Defined Networking (SDN) protocol coupled with the OpenFlow protocol, which is restricted to a fixed set of header fields [6]. In turn, the proposed framework leverages the customization that P4 PDP switches offer to offload a unique hybrid feature extraction technique to the data plane. The selected features are a hybrid of context-aware and context-less attributes, extracted via shallow and deep inspection of the packet, respectively. Such a framework allows bypassing the classification latency associated with context-aware analysis and the potential bottlenecks and privacy issues coupled with context-less techniques. To this end, the contributions of this paper are summarized as follows:

- Presenting a novel framework that leverages P4 PDP switches to employ a hybrid context-aware and context-less feature extraction technique entirely in the data plane to overcome the associated obstacles of past DGA classification techniques.
- Implementing an in-network DPI mechanism on Intel's Tofino Application-Specific Integrated Circuit (ASIC) chipset [7] that processes the entirety of the domain name to extract its context-less features within $\approx$ 2-3 microseconds (μs). This preserves the privacy of users while reducing the overhead on the control plane associated with the feature extraction and preprocessing.
- Evaluating the proposed approach on 50 notable DGA families collected by crawling hundreds of GB of malware samples from multiple sources. The dataset includes new DGA families that are not reported in the literature. The proposed approach demonstrated that it can classify DGAs with very high accuracy from only a small number of NXD responses. The implementation code and the collected dataset are made publicly available to facilitate research developments in this area [8].

The remainder of the paper is organized as follows. In Section II, we begin by highlighting the related work and how it compares to our approach. Subsequently, Section III provides background on DGAs and P4 PDP switches. In Section IV, we describe the proposed approach, present the selected features, and detail the P4 implementation. We then present the experimental results of the proposed approach in Section V. Lastly, Section VI concludes the paper with a discussion and future work in this area.

II. Related Work

A. DGA Binary and Multiclass Classification

The vast majority of DGA detection techniques rely on using either context-aware or context-less features for binary classification. In terms of context-aware approaches, Grill et al. [4] utilized NetFlow whereas Iuchi et al. [9] relied on an SDN controller to perform the feature collection for DGA binary classification. However, such approaches can cause classification delays. For example, the average latency in [9] was approximately three seconds, which is larger than the timeout limit of the "nslookup" tool used for resolving domain names (two seconds).

Alternatively, context-less approaches reduce some of the latency of context-aware methods, at the risk of causing throughput degradation and bottlenecks in high-rate networks. In an attempt to alleviate such risk to some extent, EXPOSURE [10] reduced the volume of traffic being processed by sampling subsets of traffic. More recently, research efforts have found that assessing only the domains that resulted in NXD responses is more effective [3, 11–13], as this technique circumvents missed attacks due to sampling oversights. That said, the aforementioned techniques are at times not sufficient to counteract the ubiquitous use of DNS on the Internet.

Despite the challenges that multiclass classification has in comparison to its binary counterpart, the need for the granularity that DGA family classification offers has become increasingly apparent. EXPLAIN [1] aspired to satisfy this need by extracting 76 features from a domain name within tens to hundreds of μs on a dedicated ML GPU and achieving 81% accuracy. More recently, Tuan et al. [14] improved DGA family classification accuracy to 99% and 85.55% on two different datasets; however, the authors did not discuss key aspects pertaining to the deployment feasibility of their approach, such as the monitoring of DNS traffic, the time required to perform feature extract and classification, etc.

Table I compares past DGA approaches with our work, along with the Feature Extraction (F.E) time required for each. Essentially, our work is novel as it combines context-aware and context-less feature extraction techniques entirely in the P4 PDP, thus, accurately and swiftly classifying DGA families at line rate. In particular, the switch takes a few μs to extract the context-less features from NXD responses, which is orders of magnitude lower than CPU/GPU-based machines [1].

TABLE I
Comparison between DGA detection/classification approaches and our work.

Approach	**DGA multiclass.**	**Context-less**	**Context-aware**	**F.E. latency**
[4]			✓	*minutes* ●
[9]			✓	*seconds* ●
EXPOSURE [10]		✓	✓	*minutes* ●
FANCI [3]		✓		*ms* ●
ANCS [11]		✓		*ms* ●
[12]		✓		*ms* ●
[13]		✓		*ms* ●
EXPLAIN [1]	✓	✓		100's μs ●
[14]	✓	✓		*ms* ●
Our approach	✓	✓	✓	**2-3** μs ⋆

⋆ : ASIC processing ● : CPU/GPU processing

B. DNS and DPI in P4

Meta4 [15] is a network monitoring framework that parses a limited number of labels and characters of the domain name in DNS replies. P4DDPI [16] is a previous work of ours that increases the number of parsed labels in domain names for the purpose of applying security functionalities. Kaplan et al. [17] handle the vast majority of DNS packets in the data plane without tampering with the DNS packets. Despite the novelty and effectiveness of the aforementioned DNS inspection approaches, none of them parse the whole domain name regardless of its size.

Our approach herein is the first approach that leverages P4 programmable switches to analyze network traffic on the fly in order to classify DGA-based malware infections. In particular, the P4 programmable switch can (1) perform real-time analysis and DPI on domain names, (2) flexibly collect relevant features that are not available in traditional routers, (3) process Tbps of network traffic, which is infeasible in general-purpose servers.

III. Background

A. DGAs

The growing diversity of DGA families is primarily attributed to their ability to dramatically improve the resistance of many malicious activities to takedowns [18, 19]. For instance, the `Locky` ransomware generates domains with 5-18 characters while using one of 14 Top-Level Domains (TLDs). Other families, such as `Gozi` and `Matsnu` combine random dictionary words to make it more challenging for humans and traditional defenses to detect. Existing DGA multiclass classification approaches have solely relied on the features of the domain name [1, 14, 20, 21]. The network characteristics of DGAs (e.g., Inter-Arrival Time (IAT) between packets, protocol counts, etc.) are not considered for several reasons, such as avoiding the exhaustion of the proposed system under high traffic loads and preserving the privacy of the users. However, the aggregation of network heuristics was proven to be highly effective in malware and botnet detection [22]. To this end, such heuristics are incorporated into the proposed approach without significantly affecting throughput.

B. Programmable Switch Primer

The Protocol Independent Switch Architecture (PISA) is a data plane programming model that includes the following elements: programmable parser, programmable match-action pipeline consisting of multiple stages, and programmable deparser. The programmable parser is represented as a state machine that can define the headers that need to be parsed (e.g., Ethernet, IP, DNS, or even custom headers). The programmable match-action pipeline consists of multiple match-action units to match against packet header fields and apply actions with supplied action data. Each unit can include one or more Match-Action Tables (MATs) that are coupled with Static Random Access Memory (SRAM) or Ternary Content Addressable Memory (TCAM) for storing lookup keys and action data. Additional action logic can be implemented using stateful objects, such as registers that are stored in SRAM. Lastly, the programmable deparser defines how packet headers are reassembled when they exit the switch [23]. The high-level language for programming PISA is P4. Unlike general-purpose programming languages, P4 is domain-specific and optimized to handle Tbps of network traffic.

Despite the increasing number of emerging applications in P4 (e.g., network telemetry, security, etc. [6, 24]), preserving Tbps throughput requires limiting the complexity of the pipeline stages and permitting only elementary actions (e.g., removing looping operations and allowing only simple arithmetic). Such limitations require efforts and knowledge of the language and architecture in order to devise workarounds when implementing complex functionalities in the switch.

IV. Proposed System

A. Overview

The proposed approach aims to profile (i.e., attribute) the DGA family in the network rather than detect it amongst

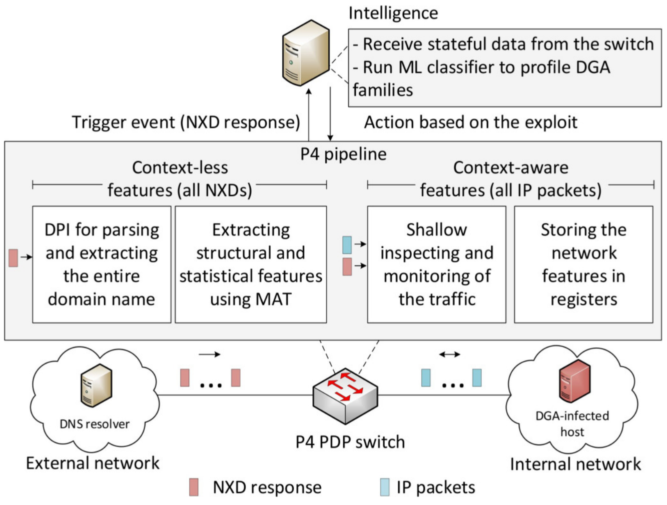
Intelligence
- Receive stateful data from the switch
- Run ML classifier to profile DGA families
Trigger event (NXD response)
Action based on the exploit
P4 pipeline
Context-less features (all NXDs)
Context-aware features (all IP packets)
DPI for parsing and extracting the entire domain name
Extracting structural and statistical features using MAT
Shallow inspecting and monitoring of the traffic
Storing the network features in registers
DNS resolver
External network
P4 PDP switch
DGA-infected host
Internal network
NXD response
IP packets

Fig. 1. System overview.

normal traffic. Since the detection of DGAs is a much easier problem than the multiclass classification [1], our approach can be easily extended to detect DGAs by incorporating the traffic of normal users. The high-level architecture of the proposed approach is shown in Fig. 1 and can be summarized in the following steps. (1) The P4 PDP switch monitors the communication of the internal network's hosts with the Internet to collect the context-aware features (i.e., network features) and store them in registers. (2) When an NXD response is received, the switch parses the whole domain name and extracts its context-less features (i.e., statistical and structural domain name features). (3) Next, the switch sends the aggregated context-less and context-aware features (associated with the host receiving the NXD response) to the control plane via message digests. Such features are then passed through a trained ML classifier to classify the DGA on the network. (4) Once the DGA family is profiled, the security operator managing the control plane can immediately apply the appropriate security policy based on the DGA family (i.e., the family of malware behind the attack.)

B. Feature Selection

The features extracted in the P4 PDP switch are divided into context-aware and context-less features. The former is applied for all IP packets and requires shallow inspection; thus, the throughput is not affected. The latter performs DPI only on NXD responses. The features are selected to adhere to the hardware limitations of P4 PDP switches.

1) Context-aware Features: The context-aware features aim to characterize the network behavior of DGAs while they attempt to contact the C2 server. Our approach collects features in the data plane without involving the control plane until an NXD response is received. For each host in the internal network, the features aggregated in the data plane include the number of IPs contacted, the IAT between such IP packets, and the number of DNS requests made. Additionally, the time it takes for the first NXD response to arrive is also included as a feature, as well as the IAT between subsequent NXD responses. Such features are relevant to the multiclass classification of DGA families. As an example, the IAT between malicious Algorithmically Generated Domains (mAGDs) could range from hundreds of μs to a few seconds depending on the family. This requires the nanosecond granularity offered by the P4 PDP switch to track the shorter IATs.

2) Context-less Features: Our work aims to identify the features that have been proven to be effective in the literature [25, 26] and can be implemented in P4. In particular, we utilize the randomness of a certain domain by implementing the n-gram character frequency as a statistical feature, in addition to a number of structural features of the domain name.

The 2-grams (bigrams) of a domain name d is a multiset of all character sequences b, such that $b \in d$ and $|b| = 2$. For instance, the bigrams of the word "google" are: "$g", "go", "oo", "og", "gl", "le", and "e$". We utilize the bigram frequency because the distribution of frequencies can vary between families based on their algorithm [20], in addition to their practicality in being stored in the limited SRAM in the data plane. To arrive at the bigram frequency feature on the switch, the domain name d is first divided into multiple subdomains, where each subdomain s can be viewed as a separate word. Subsequently, the bigram frequency distribution is applied to each subdomain separately. Next, we calculate the randomness of a domain name d according to the following formula [25]:

$$score\ (d) = \sum_{\forall\ subdomain\ s\ \in\ d} \left(\sum_{\forall\ bigram\ b\ \in\ s} f_s^b \right)$$

where f_s^b is the frequency of the bigram b in the subdomain s, read from the pre-computed match-action tables. The frequency of the bigrams utilized in this equation was obtained by processing the English dictionary and counting the number of occurrences of each bigram within every word.

The structural features of a domain name include the length of the domain name, the number of subdomains, the TLD, whether the domain has a valid TLD, and if the domain has a single-character subdomain. These features have been shown to be useful for distinguishing DGA families [19].

C. P4 implementation

Parsing the entire domain name and applying complex arithmetic operations are not trivial in P4 due to hardware limitations and resource constraints. An example of such restrictions is the lack of support for variable length headers (which would require looping operations) that extend deeper into the packet [15, 16]. Moreover, even if the domain name is fully extracted in the parser, the limited number of stages in the ingress and egress pipeline would not allow the manipulation of all these additional headers (i.e., via applying MATs) for feature extraction purposes. To address this issue, in each pipeline pass, we extract seven characters from a subdomain at maximum, and if more characters exist in the subdomain, then recirculation is applied until the entirety of the subdomain is parsed. This procedure is repeated until all such subdomains are parsed.

The characters extracted from all the subdomains in the programmable parser are used to compute the bigram frequency f_s^b in the ingress pipeline. Each bigram b of these extracted characters will be applied to a MAT, which returns the given frequency. The match keys of the MAT are the bigrams, and the MATs are pre-populated by the control plane with the frequency of each bigram based on the English dictionary. A MAT is required to calculate the frequency of every bigram. The summation of all the frequencies f_s^b over the domain d is assigned to a custom header that is recirculated with the packet. In turn, once the domain d is fully extracted, its overall bigram frequency will already be computed.

The egress pipeline is used to count and aggregate the context-aware features using registers. The P4 program successfully compiles and runs on a Tofino hardware switch and the source code is made publicly available [8].

V. Evaluation

A. Dataset

Due to the increased interest in DGA detection, a plethora of mAGD datasets exist [19, 27]. These datasets only comprise domain names and therefore are not sufficient for techniques such as ours that leverage network-related features. Moreover, there is no online repository comprising a wide variety of DGA traffic behavior.

To tackle this shortcoming, we crawled and retrieved hundreds of GB of malware samples from notable cyber security services and websites including VirusTotal (2017 until 2021) [28], VirusShare [29], Malpedia [30], and Triage [31]. The hashes of the obtained samples were submitted to VirusTotal to retrieve their metadata information and determine if they were previously reported to have DGA behavior. Subsequently, each sample was instrumented in an isolated environment to capture its network traffic behavior, i.e., the Packet Capture (PCAP), file for a variation of 10 to 30 minutes. To further validate that the collected samples demonstrate DGA behavior, only PCAP files that have NXD responses registered in DGArchive [19] are considered. DGArchive contains a database of well-known DGAs and their generated samples. Furthermore, there were a number of malware samples that showed DGA behavior (having numerous NXD responses), but their generated domain names were not registered in DGArchive. We attribute such samples to three new DGA families that were not previously reported in the literature and incorporate them into our experiments. The resulting dataset includes 1,311 samples containing 50 DGA families. We kindly refer interested readers to our dataset which we make publicly available for additional details pertaining to the collected DGA families [8].

B. Experimental Setup

The collected dataset of DGA samples was used to train ML models offline on a general-purpose CPU. In particular, 80% of the dataset was used for training and 20% for testing. During the training phase, 5-fold Cross Validation (CV) was used to

TABLE II
ACCURACY, F1 SCORE, AND PRECISION OF DIFFERENT ML CLASSIFIERS DURING THE FIRST 8 NXD RESPONSES RECEIVED.

NXD count	**RF**			**SVM**			**MLP**			**LR**			**GNB**		
	Acc	**F1**	**Prec**	**Acc**	**F1**	**Prec**	**Acc**	**F1**	**Prec**	**Acc**	**F1**	**Prec**	**Acc**	**F1**	**Prec**
NXD 1	0.923	0.907	0.902	0.872	0.856	0.847	0.87	0.843	0.829	0.716	0.679	0.667	0.726	0.688	0.688
NXD 2	0.951	0.943	0.943	0.899	0.893	0.893	0.904	0.897	0.9	0.76	0.741	0.747	0.727	0.701	0.707
NXD 3	0.964	0.958	0.964	0.918	0.913	0.914	0.924	0.914	0.912	0.767	0.74	0.743	0.649	0.668	0.732
NXD 4	0.966	0.961	0.963	0.906	0.905	0.912	0.916	0.909	0.915	0.79	0.765	0.758	0.633	0.635	0.692
NXD 5	0.97	0.966	0.967	0.915	0.91	0.911	0.919	0.91	0.907	0.77	0.735	0.746	0.604	0.615	0.689
NXD 6	0.975	0.972	0.973	0.914	0.911	0.912	0.922	0.915	0.918	0.794	0.767	0.783	0.617	0.627	0.716
NXD 7	0.977	0.976	0.979	0.92	0.915	0.915	0.929	0.924	0.93	0.799	0.771	0.78	0.61	0.613	0.714
NXD 8	0.98	0.979	0.981	0.917	0.912	0.914	0.93	0.923	0.921	0.764	0.73	0.735	0.631	0.618	0.65

avoid overfitting the model. Since some class imbalance was observed in our dataset, i.e., a different number of samples for each DGA family, we assigned weights for every class so that classes with few samples are not underrepresented. For hyperparameter tuning, the grid search method was applied to select the best parameters of each model. The experiments were performed 10 times for each model and their average is reported.

We tested five ML models on the features extracted by the P4 PDP switch, namely, Random Forest (RF), Support Vector Machine (SVM), Multi-layer Perceptron (MLP) classifier, Logistic Regression (LR), and Gaussian Naive Bayes (GNB). Table II shows the accuracy (Acc), F1 score (F1), and Precision (Prec) of the models amid varying the number of NXD responses. The RF model performed best among the tested models, where the accuracy starts at 92% from the first NXD response received and reaches 98% by the 8^{th} NXD response. As more NXD responses are received from a certain host, the accuracy is expected to increase as more information is collected. Note that while MLP neural network architectures in general are known for their performance, they necessitate large amounts of training data to maximize their classification potential. As such, it can be observed that the RF outperformed the MLP by a fair amount.

To demonstrate the effectiveness of the proposed approach on the various families of DGA samples collected, we grouped

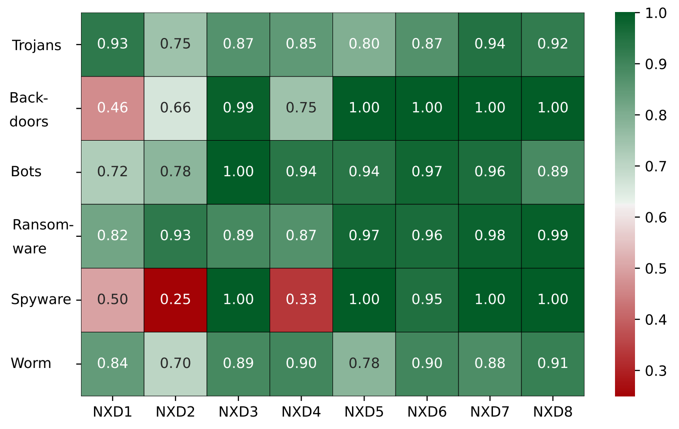

Fig. 2. Performance of the proposed approach amid varying NXD responses on a subset of samples grouped by their attack category.

such families based upon the overarching attacks they fall under, namely trojan, backdoor, bots, ransomware, spyware, and worm. Fig. 2 shows the grouping of such attacks and the RF's ability to accurately classify them. The accuracy of critical attacks, such as ransomware, is high from the first NXD capture, and the majority of attacks are classified with high confidence by the 5^{th} NXD response.

C. Comparison with State-of-the-art

Since existing DGA multiclass classification approaches use context-less features, they do not discuss the context-aware features of DGAs and do not publish any network traces [1, 14]. Consequently, we cannot apply their datasets to our experiments. That being said, as EXPLAIN's code has been open-sourced, we alternatively performed some additional analysis of EXPLAIN's feature extraction time and compare our findings to that of the proposed approach.

While EXPLAIN reports that their feature extraction time on a dedicated GPU is in the hundreds of μs, we were rather interested in measuring the feature extraction time on a general-purpose CPU. To this extent, we apply EXPLAIN's available source code to a general-purpose machine with 64 GB RAM and a 2.9 GHz processor with 8 cores. Comparatively, we measure the time it takes for the P4 PDP switch to perform DPI and extract the context-less features from a domain name according to our proposed approach. The context-aware feature extraction time is performed at line rate without degrading the throughput, thus, it is not reported herein. Fig. 3 shows the Cumulative Distribution Function (CDF) of the feature extraction time of both approaches. Our proposed approach allows the P4 PDP switch to extract all the relevant features within 2-3 μs. Conversely, EXPLAIN requires up to a few ms using a CPU with one thread.

VI. CONCLUSION AND DISCUSSION

In this work, we propose a hybrid feature extraction technique relying on context-aware and context-less features to classify DGA families. In particular, the context-aware feature extraction characterizes the network traffic behavior of the DGAs and requires shallow packet inspection, therefore it does not degrade the throughput. The context-less feature extraction aims to study the structural and statistical characteristics of domain names relating to NXDs using DPI. For every NXD

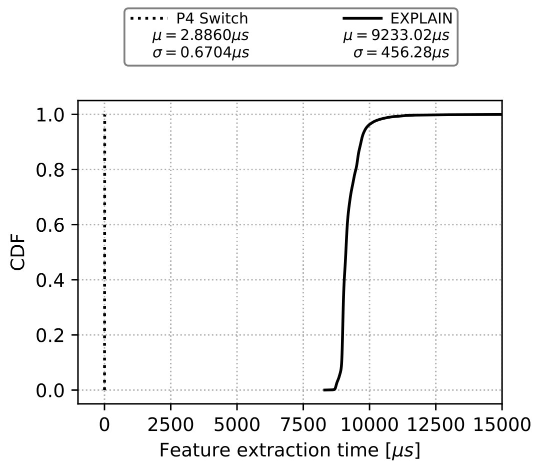

Fig. 3. Feature extraction time of our work and EXPLAIN.

reply received, the P4 PDP switch sends the extracted features to the control plane to run the intelligence, thus, minimizing the bulk of processing on the server and preserving the privacy of the users in the network. With 50 DGA families collected by crawling hundreds of GB from notable security websites, the proposed approach achieves 92% accuracy with a RF classifier from the first NXD collected and reaches up to 98% by the 8^{th} NXD.

Our work is crucial in deployments where high throughput is required while preserving the security and privacy of the network. In the future, we aim to explore other techniques that are robust against encrypted DNS traffic. More DGA-based malware samples will also be collected to characterize additional DGA families and extend our publicly available dataset in order to promote related research efforts.

Acknowledgement

This work was supported by the U.S. National Science Foundation, awards 2118311 and 2104273.

References

[1] A. Drichel, N. Faerber, and U. Meyer, "First step towards explainable dga multiclass classification," in *The 16th International Conference on Availability, Reliability and Security*, pp. 1–13, 2021.

[2] Y. Li, K. Xiong, T. Chin, and C. Hu, "A machine learning framework for domain generation algorithm-based malware detection," *IEEE Access*, vol. 7, pp. 32765–32782, 2019.

[3] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "{FANCI}: Feature-based automated {NXDomain} classification and intelligence," in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1165–1181, 2018.

[4] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak, "Detecting dga malware using netflow," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1304–1309, IEEE, 2015.

[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[6] A. AlSabeh, J. Khoury, E. Kfoury, J. Crichigno, and E. Bou-Harb, "A survey on security applications of p4 programmable switches and a stride-based vulnerability assessment," *Computer Networks*, vol. 207, p. 108800, 2022.

[7] Intel, "Intel Tofino P4-programmable Ethernet switch ASIC." [Online]. Available: https://tinyurl.com/2p97j3pe.

[8] Aalsabeh Github, "P4-DGA-Multiclass." [Online]. Available: https://github.com/aalsabeh/P4-DGA-Multiclass.

[9] Y. Iuchi, Y. Jin, H. Ichise, K. Iida, and Y. Takai, "Detection and blocking of dga-based bot infected computers by monitoring nxdomain responses," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pp. 82–87, IEEE, 2020.

[10] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, pp. 1–28, 2014.

[11] L. Fang, X. Yun, C. Yin, W. Ding, L. Zhou, Z. Liu, and C. Su, "Ancs: Automatic nxdomain classification system based on incremental fuzzy rough sets machine learning," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 4, pp. 742–756, 2020.

[12] K. Highnam, D. Puzio, S. Luo, and N. R. Jennings, "Real-time detection of dictionary dga network traffic using deep learning," *SN Computer Science*, vol. 2, no. 2, pp. 1–17, 2021.

[13] B. Yu, D. L. Gray, J. Pan, M. De Cock, and A. C. Nascimento, "Inline dga detection with deep networks," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 683–692, IEEE, 2017.

[14] T. A. Tuan, H. V. Long, and D. Taniar, "On detecting and classifying dga botnets and their families," *Computers & Security*, vol. 113, p. 102549, 2022.

[15] J. Kim, H. Kim, and J. Rexford, "Analyzing traffic by domain name in the data plane," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pp. 1–12, 2021.

[16] A. AlSabeh, E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4ddpi: Securing p4-programmable data plane networks via dns deep packet inspection," in *Proceedings of the 2022 Network and Distributed System Security (NDSS) Symposium*, pp. 1–7, 2022.

[17] A. Kaplan and S. L. Feibish, "Dns water torture detection in the data plane," in *Proceedings of the SIGCOMM'21 Poster and Demo Sessions*, pp. 24–26, 2021.

[18] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 263–278, 2016.

[19] D. P LOHMANN, "DGArchive." [Online]. Available: https://tinyurl.com/yc6whwrc.

[20] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, "Dga botnet detection using supervised learning methods," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*, pp. 211–218, 2017.

[21] M. Zago, M. G. Pérez, and G. M. Pérez, "Umudga: A dataset for profiling algorithmically generated domain names in botnet detection," *Data in Brief*, vol. 30, p. 105400, 2020.

[22] K. Friday, E. Kfoury, E. Bou-Harb, and J. Crichigno, "Inc: In-network classification of botnet propagation at line rate," in *European Symposium on Research in Computer Security*, pp. 551–569, Springer, 2022.

[23] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with p4: Fundamentals, advances, and applied research," *arXiv preprint arXiv:2101.10632*, 2021.

[24] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, 2021.

[25] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A bigram based real time dns tunnel detection approach," *Procedia Computer Science*, vol. 17, pp. 852–860, 2013.

[26] V. Tong and G. Nguyen, "A method for detecting dga botnet based on semantic and cluster analysis," in *Proceedings of the seventh symposium on information and communication technology*, pp. 272–277, 2016.

[27] Netlab360, "DGA Families." [Online]. Available: https://tinyurl.com/348u3pyt.

[28] "VirusTotal." [Online]. Available: https://tinyurl.com/2p9cscna.

[29] "VirusShare." [Online]. Available: https://virusshare.com/.

[30] "Malpedia." [Online]. Available: https://tinyurl.com/28k2snbk.

[31] "Triage." [Online]. Available: https://tria.ge/.