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

Ali AlSabeh
University of South Carolina, SC

Agenda

- Introduction
- Motivation
- Contribution
- Related Work
- Programmable switches
- Proposed system
- Implementation and Evaluation
- Conclusion and Discussion

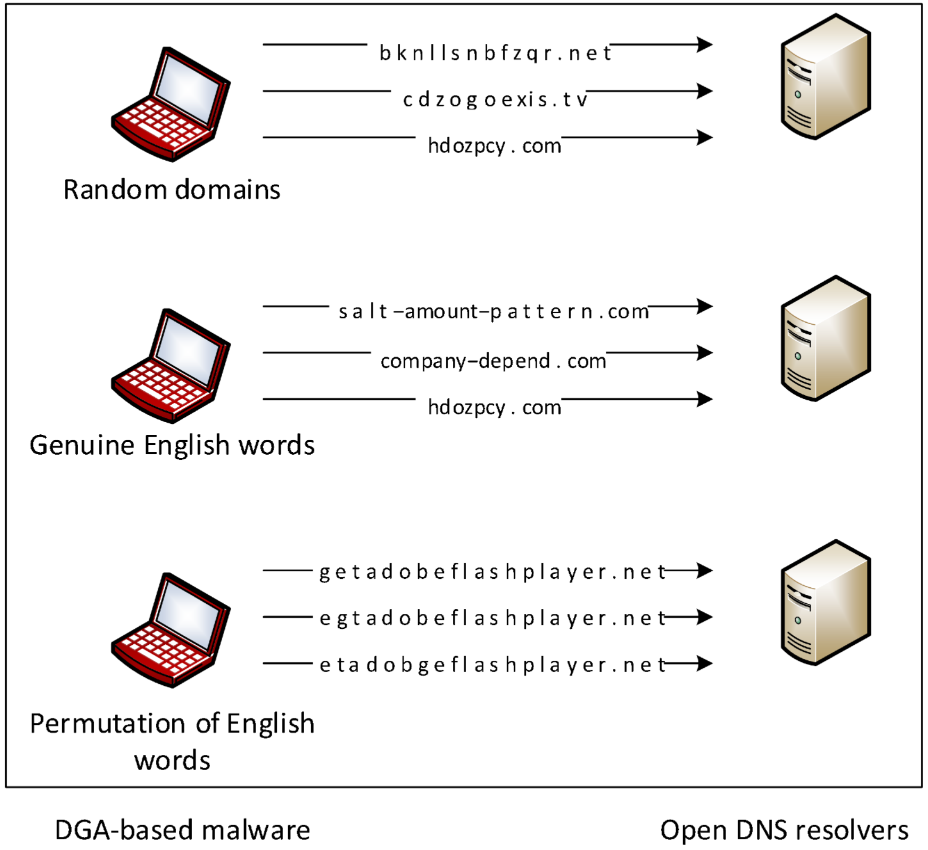
Introduction

- Attackers often use a Command and Control (C2) server to establish communication and send commands to infected machines for malicious acts
- Communication with the C2 server can either be static or dynamic
  - ➢ Static communication: the C2 server has a fixed IP address and domain name
  - ➢ Dynamic communication: the C2 server’s IP and/or domain name change frequently
- Domain Generation Algorithms (DGAs) are the de facto dynamic C2 communication method used by a broad array of modern malware, including botnets, ransomware, and many others[1]

[1] “Dynamic Resolution: Domain Generation Algorithms.” [Online]. Available: https://tinyurl.com/44hz9hpm.

DGA Attacks

- DGAs evade domain-based firewall controls by frequently changing the domain name selected from a large pool of candidates
- The malware makes Domain Name System (DNS) queries in an attempt to resolve the IP addresses of these generated domains
- 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



Existing Mitigation Strategies

- Most research efforts focus on DGA detection, i.e., they perform binary classification in order to segregate DGAs from benign traffic
- Approaches rely on contextual network traffic analysis (context-aware) or domain name analysis, without considering network traffic (context-less)
- In addition to DGA detection, it is helpful to classify DGA malware based on the family (Trojan, Backdoor, etc.)
  - ➢ 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]

[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.

Motivation

- Context-aware approaches analyze the network traffic behavior to fingerprint DGAs
  - ➢ Slow since they typically analyze batches of traffic offline
- Context-less approaches obtain high accuracy with advanced ML models
  - ➢ Require a general-purpose CPU/GPU to process and analyze the domain names, which could create a bottleneck due to the ubiquitous use of DNS on the Internet
- There is a need for a system that uses context-aware and context-less features to classify DGAs without degrading high-throughput networks

Contribution

- Proposing a novel P4 scheme that uses a hybrid context-aware and context-less feature extraction technique entirely in the data plane
- Implementing an in-network Deep Packet Inspection (DPI) on Intel's Tofino ASIC that extracts and analyzes the entirety of the domain name within 3 microseconds
- Evaluating the proposed approach on 50 DGA families collected by crawling GBs of malware samples
- Highlighting the effectiveness of the proposed work in terms of accuracy, performance

Related Work

- DGA binary and multiclass classification
  - [1, 2] use NetFlow and an SDN controller to collect context-aware features
  - [3] uses ML models on context-aware and context-less features on batches of DNS traffic
  - [4-7] use machine learning trained on features of the domain name (statistical, structural, linguistic, etc.)
- DGA multiclass classification
  - EXPLAIN [8] and [9] extract numerous features from a domain name to classify DGAs

Approach	**DGA multiclass.**	**Context-less**	**Context-aware**	**F.E. latency**
[1]			✓	*minutes* •
[2]			✓	*seconds* •
EXPOSURE [3]		✓	✓	*minutes* •
FANCI [4]		✓		*ms* •
ANCS [5]		✓		*ms* •
[6]		✓		*ms* •
[7]		✓		*ms* •
EXPLAIN [8]	✓	✓		100's μs •
[9]	✓	✓		*ms* •
Our approach	✓	✓	✓	**2-3** μs ⋆

⋆ : ASIC processing • : CPU/GPU processing

Overview P4 Switches

- P4 switches permit programmer to program the data plane
- Customized packet processing
- High granularity in measurements
- Per-packet traffic analysis and inspection
- Stateful memory processing
- If the P4 program compiles, it runs on the chip at line rate

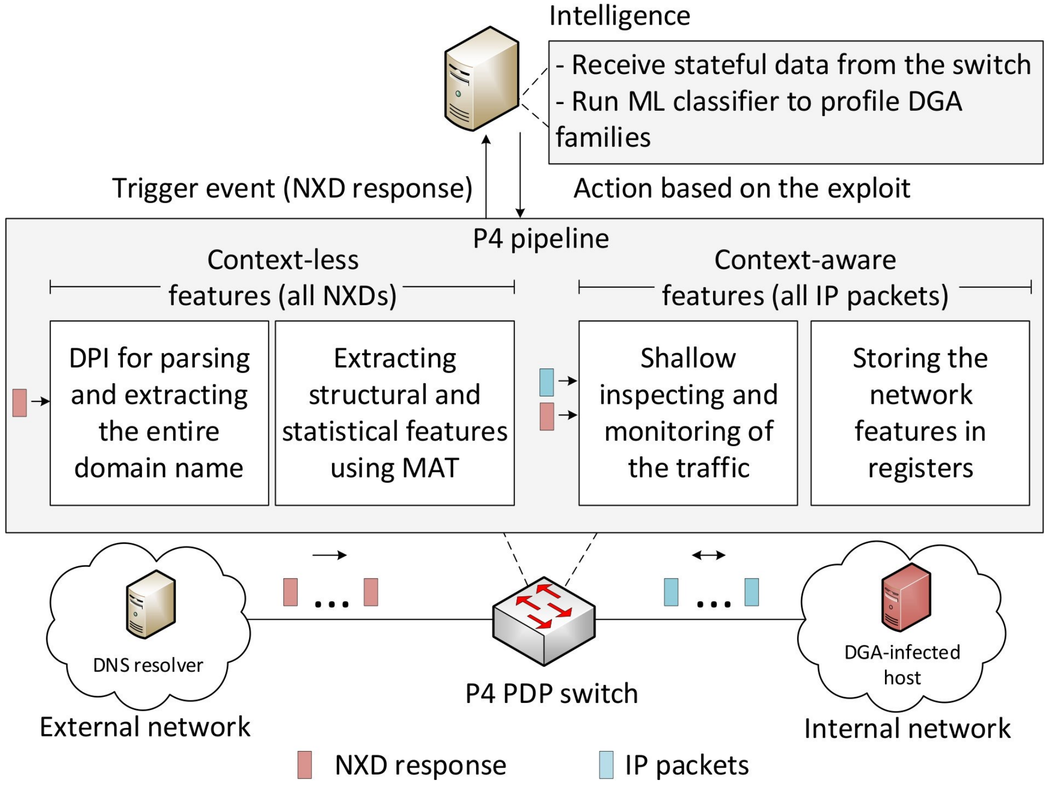
```
/*************************************************************************
*********************** P A R S E R  ***********************************
*************************************************************************/

    state parse_ethernet {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            TYPE_IPV4: parse_ipv4;
            default: accept;
        }
    }

    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        verify(hdr.ipv4.ihl >= 5, error.IPHeaderTooShort);
        transition select(hdr.ipv4.ihl) {
            5             : accept;
            default       : parse_ipv4_option;
        }
    }
```

P4 code

Programmable chip

Proposed System

- The P4 PDP switch collects and stores the context-aware features of the hosts
- When an NXD response is received, the switch performs DPI on the domain name to extract its context-less features
- The switch sends the collected features to the control plane
- The control plane runs the intelligence to classify the DGA family and initiate the appropriate incidence response



Proposed System

- Context-aware features
  - It characterizes the network behavior of DGAs while they attempt to contact the C2 server
  - For each host in the network, the following features are stored in the data plane:
    - Number of IPs contacted
    - Number of DNS requests made
    - Time it takes to for the first NXD response to arrive
    - Inter-arrival Time (IAT) between subsequent NXD responses
  - Collected in the data plane without involving the control plane (until an NXD response is received)

Proposed System

- Context-less features
  - It computes the bigram of the domain name; a bigram model may suffice to predict whether a domain name is a legitimate human readable domain
  - Other domain name attributes include length of the domain name and number of subdomains
  - For each NXD response received, the data plane extracts the following features from the domain name
    - Randomness of a domain name *d* according to its bigram frequency

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

Where f_s^b is the frequency of the bigram b in the subdomain *s*

- **Example:** bigrams of “google” are: “$g”, “go”, “oo”, “og”, “gl”, “le”, “e$”

P4 Implementation

- The parser parses DNS packets in the data plane
  - ➢ Packet recirculation maybe required for certain domain names
  - ➢ To compute the randomness of a domain, each bigram *b* will be applied to a Match-Action Table (MAT)
  - ➢ The frequencies of the bigrams are computed offline using the English dictionary; thus, the lower the score the more it is considered random
  - ➢ The MATs are pre-populated by the control plane with the frequency of each bigram

Algorithm 1: Pseudocode of the P4 code

```
Parser():
    Parse_headers(ETH, IP, UDP, DNS)
    if pkt == IPv4 && DNS.type == NXD then
        part1 ← pkt.extract(p.domain_label1.part1) // Extract 2^0 bytes
        part2 ← pkt.extract(p.domain_label1.part2) // Extract 2^1 bytes
        part4 ← pkt.extract(p.domain_label1.part4) // Extract 2^2 bytes
SwitchIngress():
    table bigram_tabel1
        key : part1;
        actions : add_bigram_val;
    ...

    for i = 0, 1, 2 do
        if part2^i.isValid() then
            Apply (2^i − 1) bigrams of part2^i
        if part2^(i−1).isValid() && part_2^i.isValid() then
            Calculate the bigram between part2^(i−1) and part2^i
        if domain.is_fully_parsed == False then
            recirculate();
        else
            check_validity_TLD();
        calc_domain_length();
        set_invalid(part2^i);
SwitchEgress():
    register unique_ips_contacted;
    register nb_DNS_requests;
    register unique_NXDs;

    unique_ips_contacted.update();
    nb_DNS_requests.update();
    unique_NXDs.update();
```

Evaluation

- Dataset
  - Hundreds of GB of malware samples from cyber security websites were crawled
  - Each sample was instrumented in an isolated environment to capture its network traffic behavior
  - To collect DGA-based malware, only samples that receive NXD responses containing domain names generated by DGAs (based on DGArchive[1]) are considered
  - The resulting dataset includes 1,311 samples containing 50 DGA families
- Experimental setup
  - The collected dataset was used to train ML models offline on a general-purpose CPU
  - 80% of data was used for training and 20% for testing
  - 5-fold Cross Validation (CV) was used to avoid overfitting the model
  - Weights were assigned for every class (DGA family) to deal with class imbalance

[1] D. P LOHMANN, “DGArchive.” [Online]. Available: https://tinyurl. com/yc6whwrc.

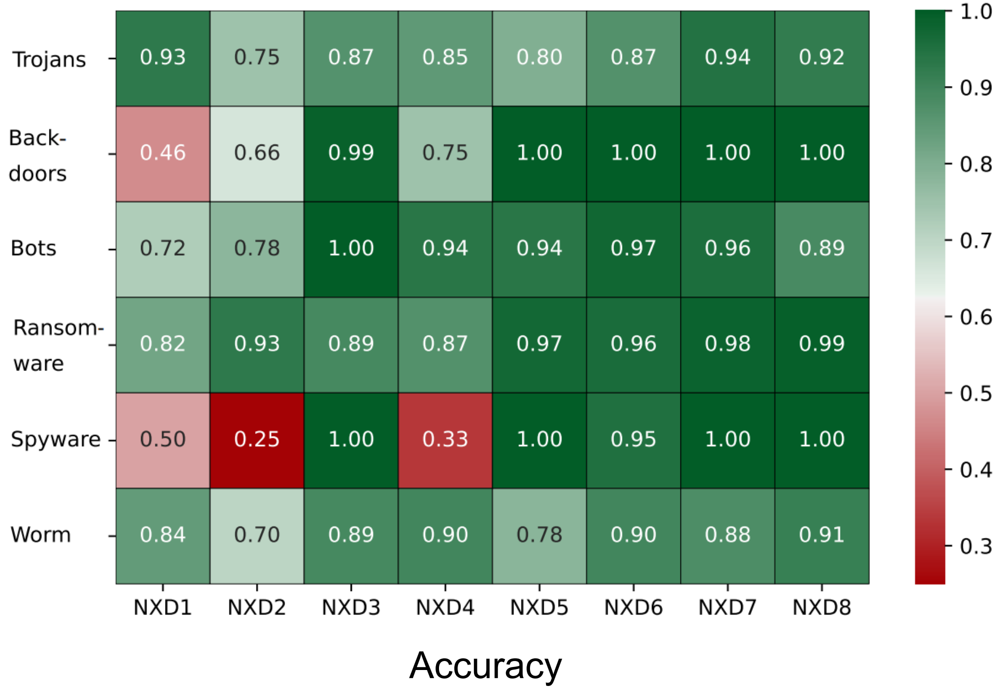
Evaluation

- Accuracy (Acc), F1 score, and Precision (Prec) of different ML classifiers during the first 8 NXD responses received were reported
- The Random Forest (RF) model performed best
  - The Accuracy (Acc) starts at 92% from the first NXD response received and reaches 95% by the 8th NXD response

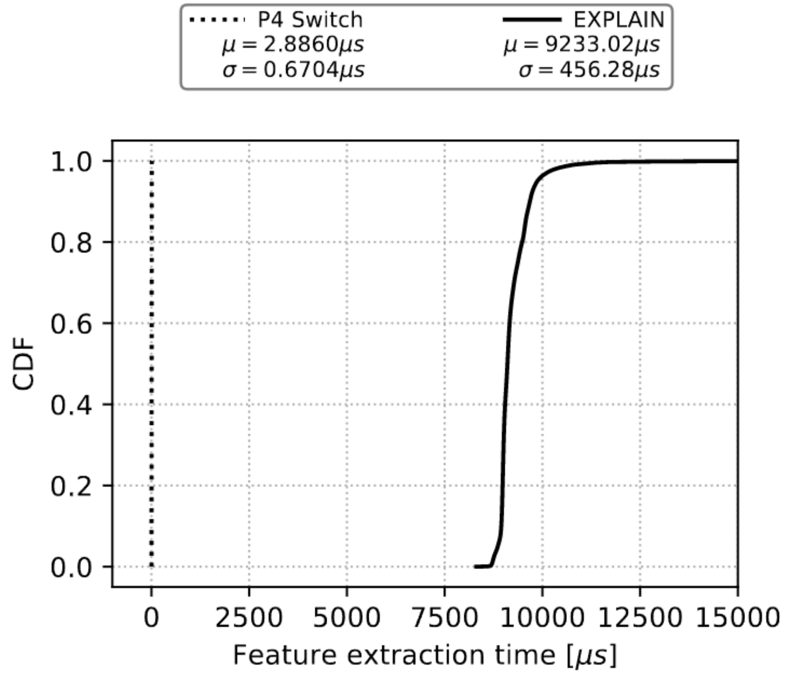
NXD count	RF			SVM			MLP			LR			GNB		
	Acc	F1	Prec	Acc	F1	Prec	Acc	F1	Prec	Acc	F1	Prec	Acc	F1	Prec
NXD 1	0.923	0.907	0.902	0.872	0.856	0.847	0.87	0.843	0.829	0.716	0.679	0.667	0.726	0.688	0.688
NXD 2	0.951	0.943	0.943	0.899	0.893	0.893	0.904	0.897	0.9	0.76	0.741	0.747	0.727	0.701	0.707
NXD 3	0.964	0.958	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

Evaluation

- Performance of the proposed approach amid varying NXD responses on a subset of samples grouped by their attack category
- The accuracy of critical attacks, such as ransomware, is high from the first NXD response
- The majority of attacks are classified with high confidence by the 5th NXD response

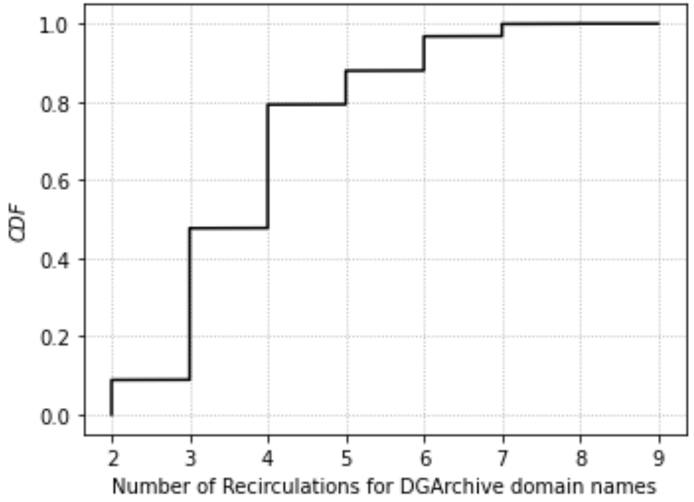


- Feature extraction time of our work and EXPLAIN
- EXPLAIN's available source code was tested on a general-purposed CPU with 64 GB RAM, 2.9 GHz processor with 8 cores



Evaluation

- Our approach only recirculates NXD responses
  - NXDs account for 0.01% of the traffic in campus traffic[1]
  - The rest of the traffic undergoes shallow packet inspection (few hundreds of nanoseconds)
- Number of recirculations for domain names in DGArchive
  - 80% of the domains require a maximum of four recirculations



[1] Garcia, Sebastian, et al. "An empirical comparison of botnet detection methods." *computers & security* 45 (2014): 100-123.

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
- Context-aware features characterize the network traffic behavior of the DGAs and require shallow packet inspection (no degradation to the throughput)
- Context-less features study the statistical and structural characteristics of the domain names relating to NXDs using DPI
- With 50 DGA families analyzed, the proposed approach achieves 92% accuracy with RF classifier from the first NXD response and reaches up to 98% by the 8th NXD response
- In the future, we aim to explore other techniques that are robust against encrypted DNS traffic, in addition to collecting more DGA families

Acknowledgement

- Thanks to the National Science Foundation (NSF)
- Activities in the CI Lab at the UofSC are supported by NSF, Office of Advanced Cyberinfrastructure (OAC), awards 2118311 and 2104273

References

[1] 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.

[2] 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.

[3] 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.

[4] S. Schuppen, 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

Thank You!

Ali AlSabeh
aalsabeh@email.sc.edu