Automatic Detection of DGA-Enabled Malware Using SDN and Traffic Behavioral Modeling

Jawad Ahmed, Hassan Habibi Gharakheili, Craig Russell, and Vijay Sivaraman

***Abstract*—Enterprise networks are under enormous threats from sophisticated cyber-attacks. Various kinds of malware are installed by attackers on compromised hosts, acting as bots that typically use Domain Generation Algorithms (DGAs) to communicate with their Command and Control (C&C) servers. It is computationally expensive to inspect all network packets of every host connected to a large enterprise network in "real-time" at scale with hundreds of gigabits per second data rates. This paper combines Software Defined Networking (SDN) and machine learning to develop an accurate, cost-effective, and scalable system for detecting infected hosts communicating with external C&C servers, subsequent to the resolution of DGA query names. Our solution dynamically selects network flows for diagnosis by trained models in real-time, and relies more on the behavioral traffic profile, rather than packet content.**

Our first contribution highlights the prevalence and activity pattern of DGA-enabled malware across internal hosts. We draw insights into the behavioral profile of DGA-enabled malware flows when communicating with C&C servers. For our second contribution, we identify malware traffic attributes and train three specialized one-class classifier models using behavioral attributes of malware HTTP, HTTPS and UDP flows. We develop an SDN-based monitoring system to automatically mirror TCP/UDP flows pertinent to DGA queries for diagnosis by the trained models. Finally, we evaluate the efficacy of our approach by testing suspicious traffic flows (selectively recorded by SDN reactive rules), identifying infected hosts, and verifying our detection with an off-the-shelf Intrusion Detection System (IDS) software tool.

***Index Terms*—Malware, DGA, SDN, Behavioral Modeling**

I. Introduction

Cyber threats and data breaches continue to increase in both frequency and complexity, placing businesses and individuals at constant risk. According to Cybersecurity Ventures [1], cybercrime damages will cost the world $6 trillion annually by 2021. Enterprises, small and large, remain among the top lucrative targets of automated attacks [2], [3]. Enterprise networks are often complex, with applications that rely on a mix of local and cloud-based services, and hence difficult to manage securely [4]. Enterprise hosts often include powerful servers, personal computing devices, mobile phones, and unmanaged Internet of Things (IoTs). These devices may use a mixture of statically or dynamically assigned addresses from several public and private Internet Protocol (IP) address ranges. Poorly administrated assets, like personal computers or unpatched servers [5], are not only potential victims of cyber-attacks but are also sources of risk for other entities on the Internet. Hosts sitting behind the enterprise border firewall can be infected by malware coming from phishing emails, security holes in browser plugins, or other infected local devices.

Malware-infected machines, forming a botnet, are typically managed remotely by an adversary (aka botmaster) via a C&C channel. The botnet is primarily used by cyber-criminals for malicious activities such as stealing sensitive information, disseminating spam, or launching denial-of-service attacks. Therefore, law enforcement agencies routinely perform take-down operations on the blacklisted C&C servers [6], disrupting their botnet activities. In response to these efforts, botmasters have developed innovative approaches to protect their infrastructure. The use of DGAs is one of the most effective techniques that has gained increasing popularity [7].

DGAs make use of a "seed" (a random number that is accessible to both the botmaster and the malware agent on infected hosts) to generate a large number of custom domain names. Generating numerous time-dependent domain names and registering only the relevant one(s) "just shortly" before an attack allows a botnet to shift their C&C domains on the fly and remain invisible for longer [8]. The botmaster waits for the malware to successfully resolve a Domain Name System (DNS) query for the registered domain, enabling the C&C communications to take place. Note that even if a C&C server is taken offline or blacklisted, this process can simply be restarted and a new server can come online. To date, more than 80 collections of DGA domains (each corresponding to a malware family) have been recorded by DGArchive [9] and are publicly available.

Problem Statement: There exist a number of research works [10]–[16] that analyze DNS traces to identify malicious activities, detecting C&C servers, infected hosts, or malicious domains. Their proposed methods largely require the extraction of information from DNS packets, correlating queries and responses, and maintaining many states over a reasonably long duration. These processing steps collectively demand heavy compute resources and hence make it difficult to scale cost-effectively. On the other hand, existing firewalls and intrusion detection systems rely primarily on inspecting every packet traversing the network, which makes them expensive. Further, correlating malicious DNS queries with subsequent C&C communication flows would significantly impact their inferencing accuracy and cost.

In this paper, we employ the SDN paradigm to judiciously combine selective packet inspection (only DNS proactively) and flow behavioral analysis (reactively), in order to intel-

J. Ahmed is with the School of Computer Science and Engineering, University of New South Wales, Sydney, 2052 Australia. H. Habibi Gharakheili, C. Russell, and V. Sivaraman are with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, 2052 Australia. (e-mails: j.ahmed@unsw.edu.au, h.habibi@unsw.edu.au, craig.russell@unsw.edu.au, vijay@unsw.edu.au). J. Ahmed is with Data61, CSIRO, Sydney, Australia (e-mail: jawad.ahmed@data61.csiro.au)

ligently detect malware-infected hosts on the network. The novelty of this paper arises from the dynamic inter-relation of SDN and machine learning technologies for a sophisticated yet cost-effective cyber-security solution. Commodity SDN switches today can forward data very cost-effectively at Terabits-per-second. When combined with intelligent machine learning algorithms in software, it provides the flexibility and agility to deal with existing and emerging threats given the separation of forwarding and control in SDN as well as the advantage of centralized control in software. This is ideal for dynamically selecting flows (subsequent to malicious DNS queries), and inferring their behavioral health using trained models. We use public data of malware families to develop our machine learning models. Our key contributions are summarized as follows.

1) We highlight the prevalence and activity pattern of more than twenty DGA-enabled malware families running on internal hosts of a university campus network by analyzing full DNS traffic (consisting of 2.4B records) collected over 75 days from the network border (outside of the firewall). We also analyze a PCAP trace of full campus Internet traffic (collected during the peak hour with a total load of about 10Gbps) to draw insights into the behavioral pattern of DGA-enabled malware flows between suspicious internal hosts and malware servers on the Internet.
2) We identify key traffic attributes of malware and train one-class classifier specialized models by attributes of malware HTTP, HTTPS, and UDP flows obtained from a public dataset. We then develop a monitoring system that uses SDN reactive rules to automatically and selectively mirror TCP/UDP flows (between enterprise hosts and malware servers pertinent to DGA queries) for making inferences by the trained models.
3) We evaluate the efficacy of our proposed method and system during a 50-day trial. We record suspicious traffic flows mirrored by SDN reactive rules, examine them by applying our trained models, identify malware-infected hosts, and verify our detection with an off-the-shelf IDS software tool.

The rest of this paper is organized as follows: §II describes relevant prior work. We present our data analysis and highlight the prevalence of DGAs in our campus network in §III. In §IV, we characterize key attributes of malware network behavior, followed by the performance evaluation of our trained machine learning models in §V. The paper is concluded in §VI.

II. Related Work

Malware behavioral analysis has been widely studied by many researchers [7], [16]–[21] using different tools and techniques [22]. The most relevant works to ours can be divided into three categories: (a) detection of malicious traffic based on unusual DNS queries (predicting the presence of DGA domains) [7], [16], [21], [23]–[30], (b) network behavioral analysis of known malware and botnet by inspecting their traffic data and/or metadata [18]–[20], [31]–[33], (c) use of SDN and/or programmable networking in detecting cyber-attacks [34]–[37].

Malicious DNS Queries: DNS traffic has been analyzed to identify malicious network activities [15], [16], [38], [39]. Over the past decade, there has been an increasing number of works [10]–[13] on detecting malicious network activities mostly related to DNS exfiltration, DNS tunneling, and C&C communications [40], [41]. To detect DNS exfiltration and tunneling, attributes like length, entropy, and the number of labels for domain names of benign queries are considered to train a model [14], [42], [43].

In the past, blacklists were used to detect C&C communications between servers and infected hosts. However, blacklisting has been defeated by attackers since they migrated from a static domain mapping to the use of algorithmically generated domain names. In response to this change, researchers have attempted to automatically detect DGA domains using statistical modeling of DNS traffic [7], [16], [21], [23]–[25], or machine/deep learning techniques [26]–[30], [44]. Antonakakis et al. [7] develop a clustering-based method for detecting new (unknown) algorithmically generated domains (AGD) as well as classifying known AGDs using supervised learning. The authors evaluated their proposed solution in a large ISP network and found several new families (unseen before) of DGAs, operating on the network. They used statistical attributes including entropy and n-grams measures as well as structural attributes such as length and label count in the domain name, extracted from NXD (nonexistent domains) responses on a per-host basis. Schuppen et al. [25] employ manually-engineered features to train a binary classifier using Random Forest algorithm to determine whether domains in NXDomain-failed DNS queries (*i.e.,* queries to nonexistent domain names) are benign or malicious. It remains unclear how this model performs in classifying unseen malicious domains. We note that generating a rich training dataset of benign NXDs is nontrivial, and a labeled benign dataset may get polluted by some new (but unknown) malicious NXDs.

Malware and Botnet Behavioral Analysis: Flow-based analysis has been used to detect malware and botnet traffic. However, it is difficult and computationally expensive to monitor high-rate network traffic in large and complex enterprise environments. Works in [20], [31]–[33], [45] develop flow-based features like packet count and distribution of packet length, by analyzing every packet of the network to model the patterns in encrypted traffic. Similarly, Anderson et al. [18], [19] employed supervised learning algorithms to classify malware and benign traffic. Their model was trained by a variety of host-level attributes including packet size, flows inter-arrival, DNS query (*e.g.,* TLD, TTL, domain rank in Alexa), HTTP data (*e.g.,* server code, content type, Accept-language, and location), and TLS data (*e.g.,* TLS cipher suites and TLS extension) that are measured over a period of time.

Our behavioral modeling approach differs from prior works in three ways: (a) we only monitor the behavior of selected flows pertinent to certain servers (resolved by DGA responses) instead of monitoring all traffic of every host on the network, (b) we choose to extract statistical attributes of flows (encrypted and unencrypted) to train our models, without the need for inspecting payloads like HTTP content type or TLS handshake cipher keys, and (c) our models are

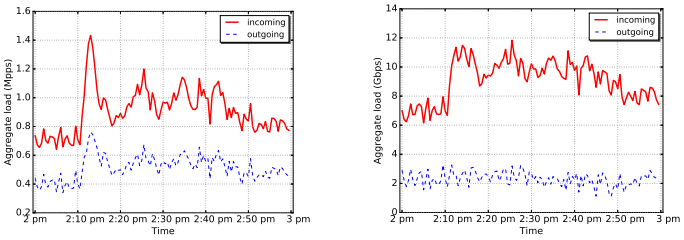
built by one-class classification algorithms and hence become sensitive to changes in any attribute while multi-class models become sensitive to changes in only discriminative attributes. Also, it is important to note that we only use a database of known DGA domains, and check domain queries against this database in real-time instead of classifying a domain as benign or malicious. Our objective is to detect infected hosts of enterprise networks by monitoring their selected flows.

Programmable Networking for Network Security: Programmable networking has recently gained popularity among researchers, specifically for network security use-cases [34]–[37], [46]. Ceron et al. [46] developed an automated system for offline malware analysis, recording the network behavior of a given malware in a controlled sandbox environment orchestrated by an SDN controller. A known host on their sandbox is infected by malware (from a set), and the SDN controller inspects every packet from/to this infected host for taking required actions like rate-limiting, blocking, or re-configuring the topology upon finding certain patterns in the packet payload (*i.e.,* regex signature) or headers (*e.g.,* contacting specific IP address and/or TCP/UDP port numbers). Our work, instead, develops an automatic SDN-based system for detecting malware-infected hosts in real-time by relying more on the behavioral activity profile of "selected flows" rather than the content of packets. Further, our SDN switch does not send any network packets to the controller (protecting it from overload from the data-plane and allowing the solution to scale to high rates). Instead, packets that need to be inspected in the software are sent as copies on a separate interface of the switch, to which a software inspection engine is attached.

Gupta et al. [34] develop scalable telemetry (*i.e.,* partitioning different types of traffic such as TCP and ICMP) that can be used to collect and analyze the network traffic in real-time using a programmable data-plane. The authors show that their approach reduces the workload of the overall system to be able to operate at line rates. Similarly, Zhang et al. [37] proposed a method that uses P4 programmable switches to better defend against DDoS attacks. The authors considered the case of volumetric DDoS attacks. They provided the defense strategies in a modular fashion that can be adopted for each network and can be used for new defense strategies other than just the DDoS. These prior works primarily leverage the programmability features in the data plane. We instead employ the programmable control-plane available by Openflow-based SDN to dynamically select suspicious flows for diagnosis by trained machine learning models.

III. Analyzing Network Traffic Data: Prevalence of DGA-Enabled Query Names and Network Behavior of Malware

In this section, we begin by analyzing the DNS traffic of a campus network to demonstrate the prevalence of DGA-enabled domain names (obtained from a public dataset) which are found in DNS queries of internal hosts. We, then, analyze a one-hour PCAP trace of the entire campus traffic (in/out) to understand the network behaviors of internal hosts when they communicate with Internet-based servers following their DGA-related DNS query.



(a) Aggregate packet rate. (b) Aggregate bit rate.

Fig. 1. Aggregate load: (a) packet rate, and (b) bit rate, during peak hour (2pm-3pm) of a weekday (31-May-2019).

A. Our Datasets

In this work, we use four different datasets including (a) 75 daily DNS PCAPs collected from the border of a university campus network, (b) a one-hour PCAP trace of the entire traffic of the university campus network to/from the Internet, (c) 82 archived files containing more than 65 million domain names used by DGA-related malware families, and (d) public network traces (PCAPs and NetFlow records) of known malware and benign traffic.

PCAP Traces of DNS Traffic: We collected daily DNS PCAP traces from the border of the University campus network. Each PCAP has a size of about 15 GB on average. The IT department of the campus network provisioned a full mirror (both inbound and outbound) of its Internet traffic (on a 10 Gbps interface each) to our data collection system from its border router (outside of the firewall), and we obtained appropriate ethics clearance (Human Research Ethics Advisory Panel approval number HC17499) for this study. We extracted DNS packets from each of the enterprise Internet traffic streams in real-time by configuring rules to match incoming/outgoing IPv4 and IPv6 UDP packets on port 53 in an OpenFlow switch. This work analyzes data collected over 75 days from 16-Sept-2019 to 1-Dec-2019. Our detailed analysis is described later in this section (§III-B, §III-C, and §III-D).

One-Hour PCAP Trace of Full Campus Traffic: In addition to DNS packet traces, we recorded all incoming/outgoing packets (only the first 96 bytes of each packet) of the campus network using `tcpdump` tool during the peak hour (2pm-3pm) of a weekday on 31st May 2019. This PCAP trace, with a size of over 250 GB, consisting of 3.2B packets, represents traffic of large-scale enterprise networks. Fig. 1 shows the aggregate load (packet rate and bit rate, moving averaged over 30-sec intervals) on the Internet link of the campus network. It can be seen that on average more than a million packets per second are exchanged between internal hosts and the Internet, resulting in an aggregate load of 10 Gbps. We will use this relatively large dataset in §III-E to highlight the behavioral profile of malware flows (needles in the haystack of enterprise network traffic) pertinent to DGA queries.

DGArchive: A group of researchers conducted an extensive study [8] on several families of DGA-based malware. Authors made their dataset ("DGArchive") available to the public [9],

TABLE I
DGA-RELATED DOMAIN FAMILIES FOUND IN THE CAMPUS NETWORK (FROM 16-SEP-2019 TO 1-DEC-2019).

Malware family	ModPack	Tinba	Gameover	Nymaim	Bamital	Suppobox	Ranbyus	Matsnu	Ramnit	Cryptolocker	Gozi	Tsifiri	Simda	Urlzone	Vawtrak	Banjori	Downloader	Necurs	Dyre	Locky	Bobax	Ccleaner	Pandabanker	Mirai	Murofet	Dircrypt
DNS queries count	552,877	22,284	9,419	1,338	842	768	399	324	260	123	94	83	72	69	34	30	28	12	11	8	7	6	4	2	2	2
	93.85%	3.78%	1.60%	0.77%																						
Unique domain names	6	327	305	38	396	175	13	15	77	7	13	2	6	1	8	5	1	3	4	2	4	3	2	1	1	1

and they have been consistently expanding their database over time. For this study, we have used the version of the public database uploaded on 7th Jan 2019, which contains records of domain names from 86 DGA families. We excluded four families for our study because they overlap with legitimate domains like `github.com`, `oracle.com`, or `doodle.com`. Such overlaps are seen because their algorithms are restricted to generate only short- to medium-sized strings (with 6 to 15 letters) that can lead to the generation of existing legitimate domains. Note that datasets of these four families contain more than 20 million records of domain names, and hence it was infeasible to manually check all domains, filtering out the legitimate ones. Therefore, we used the files corresponding to the remaining 82 malware families and developed our database consisting of about 83 Million records of domain names in total (65 Million unique records). Note that some families share a number of records with other families – the details of the overlap across various families can be found in [8].

We will use the DGArchive dataset in §III-B, §III-C, and §III-D in our detection system. We acknowledge that our detection method primarily relies upon the knowledge-base of DGArchive and hence may miss some "novel" malicious query names that are not captured by this database. Note that this limitation is inherent to any signature-based detection method (ours included). This public database is actively updated at a frequency of weeks to months. Therefore, in practice, one can check this public repository daily or weekly to obtain the latest signatures (domain names used by latest malware families).

Network Traces of Known Malware and Benign Traffic: Authors of [24] released a public dataset called "CTU-13" that contains packet traces of malicious traffic as well as labeled NetFlow records of benign traffic. Malicious traffic traces consist of 13 PCAPs (76.8M packets) from network activities of seven real botnets including Menti [47], Murlo [48], Neris [49], NSIS.ay [50], Rbot [51], Sogou [52], and Virut [53] on Windows operating systems – executable binary files of these malware were installed on lab computers [54] at the CTU University, Czech Republic, in 2011. Benign traces contain 3.6M NetFlow records of normal traffic (matching certain conditions [24]) from a controlled and known set of computers on a testbed. We will use this dataset in §IV to train our classifier models, validate, and test their performance in distinguishing malicious flows from benign ones. Our analysis is motivated by evidence [55] that Web-based "reusable" tools for remote command of malware are available for sale on the Internet. Also, malware writers may generate a large number of polymorphic variants of the same malware using executable code obfuscation techniques, however, these variants will ultimately display similar activity patterns when executed [56].

TABLE II
TOP TEN MOST FREQUENTLY USED DGA DOMAIN NAMES FOUND IN THE CAMPUS NETWORK.

Domain name	DGA family	# occurrences
`gvaq70s7he[.]ru`	ModPack	530,647
`76236osm1[.]ru`	ModPack	22,151
`vqponckshykx[.]in`	Tinba	301
`uecrbipuperq[.]online`	Tinba	284
`qipnhdggsteb[.]org`	Tinba	284
`xllqwgtppipp[.]info`	Tinba	269
`edyrsdetxwnu[.]info`	Tinba	189
`rkcrurklbstr[.]in`	Tinba	185
`jdlrshfmxkqdeprhypbejn[.]org`	Gameover	179
`vwxwvcmicwnu[.]org`	Tinba	177

B. DGA-Fueled Malware Families

We begin by analyzing DGA-based DNS queries found in the campus network traffic. Out of 2.4B DNS queries made (during 75 days) by internal hosts of the campus network, about 589K were found in DGArchive, and hence considered as DGA-based queries belonging to a total of 26 known families. Table I shows the breakdown of queries count across these families. It is seen that "ModPack" heavily dominates (93.85%), followed by "Tinba" and "Gameover" respectively contributing to 3.78% and 1.60% of total DGA-based queries – other 23 families are not very frequent (their collective contribution is less than one percent). These 26 families in total generated 1416 unique domain names. Table II lists the top ten most frequently used domain names found in DGA queries and their corresponding family. Surprisingly, only two domain names (*i.e.,* `gvaq70s7he[.]ru` and `76236osm1[.]ru`) dominate almost all the queries for the ModPack family. Other families like Tinba, however, use a variety of domain names in their DNS queries.

Various malware types participate in various malicious actions [8] such as unauthorized access to victim machines, stealing personal information, or actively taking part in denial-of-service (DOS) attacks. "ModPack" [9] was found by the Canadian Centre for Cyber Security (CCIRC) [57] which potentially relates to Andromeda [58]. Andromeda is malware that infected millions of computers across the world [59] to perform its botnet activities (*i.e.,* to steal, to destroy websites, or to spread malicious code). "Tinba" (Tiny banker was first discovered in 2012), is a malware program, targeting banking websites to steal online banking data [60]. Similarly, "Gameover" Zeus looks for personal and sensitive data *e.g.,* banking information, customer data, and secret corporate information [61].

We note that the occurrence of DGA-based queries is at least three orders of magnitude less than typical DNS queries made by internal hosts, as shown by a weekly trace in Fig. 2.

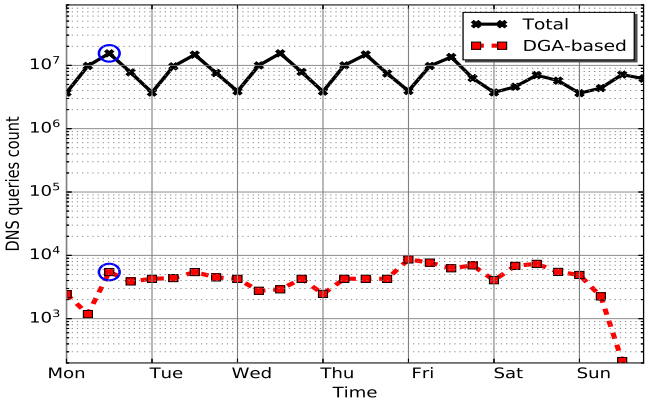


Fig. 2. A weekly trace of DNS queries count: total versus DGA-based queries (from 25-Nov-2019 to 1-Dec-2019) – blue circles highlight representative points to illustrate that count of DGA-enabled queries is at least three orders of magnitude less than count of total DNS queries.

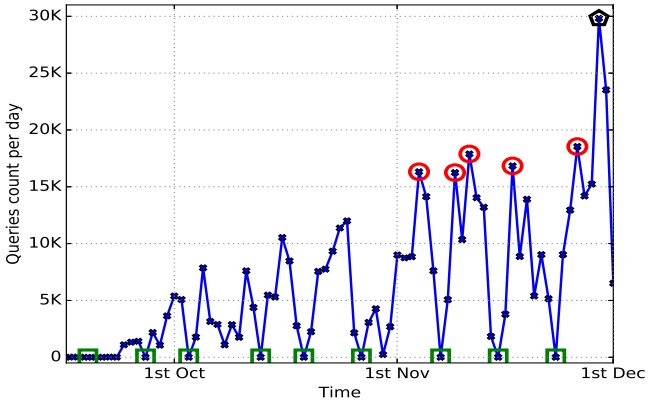
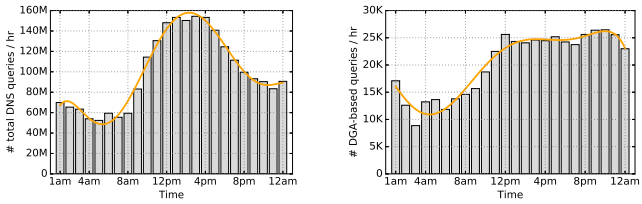


Fig. 3. Time-trace of DGA-based DNS queries across 75 days (between 16-Sep-2019 and 1-Dec-2019) – green squares represent no DGA queries, red circles represent more than 15K DGA queries, and the black pentagon highlights DGA queries daily count peaks at 30K.



(a) Total DNS queries. (b) DGA-based DNS queries.

Fig. 4. Hourly histogram of: (a) total DNS queries, and (b) DGA-based DNS queries, across 75 days (between 16-Sep-2019 and 1-Dec-2019).

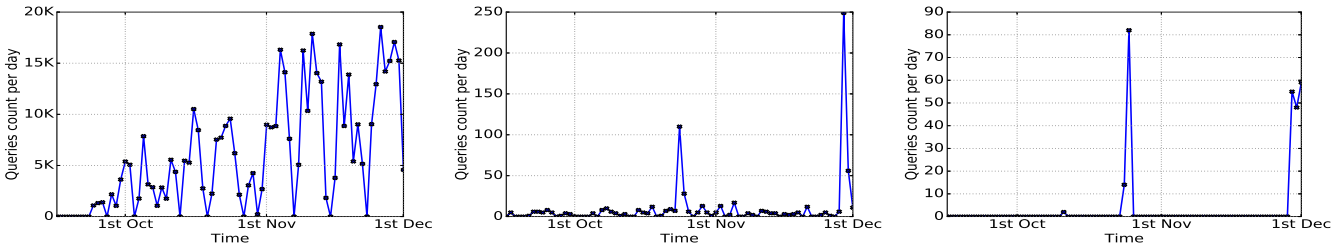
Each data-point in this plot represents the number of DNS queries over a 6-hour window. For example, it can be seen that on Monday at 12pm (as highlighted by blue circles), out of 15.4M DNS queries sent out of the campus network, only 5.4K are DGA-enabled. Such low-profile activity allows various malware families to go undetected in large enterprise networks [62].

C. Daily Activity Pattern of DGA-Based Domains

Let us now focus on the temporal activity pattern of various DGA-enabled malware families. Fig. 3 illustrates the daily count of DGA queries during 75 days *i.e.,* from 16-Sep-2019 to 1-Dec-2019. It can be seen that there is no specific pattern of daily activity at an aggregate level. For certain days, *i.e.,* mostly Thursdays, Fridays, and Saturdays, they become completely inactive (zero queries as highlighted by green squares), and some other days they are heavily active (more than 15K queries per day as highlighted by red circles). We observe a growing trend in the queries count daily over this period peaking at 30K towards the end of Nov 2019 (as highlighted by the black pentagon at the top right of the plot). Focusing on individual families, we found that almost all families (except Tsifiri, Downloader, Pandabanker, and Mirai) became active on the same day, resulting in significant peaks between 28th Nov 2019 and 1st Dec 2019.

To better understand the time-of-day activity of DGA-enabled malware, we plot in Fig. 4 the hourly histogram of DNS queries (overall versus malicious) count during the 75-day study. Starting from total load in Fig. 4(a), it can be seen that the distribution of overall DNS queries reflects the daily activity pattern of users – it starts rising in the morning (8-9am) when students and staff come to the university campus, peaks at around noon (12-1pm) when most of the users go online during their lunch break and starts falling in the afternoon (4-5pm) when the users leave the campus network at the end of working hours. Moving to the distribution of DGA queries in Fig. 4(b), we observe that the probability of finding DGA queries on the campus network is relatively higher from noon to midnight, and it is lower between post-midnight and pre-noon (1am-11am). We note that the temporal activity pattern of DGA queries does not correlate with that of the overall network traffic (*i.e.,* mostly benign), especially during afternoon and evening hours. This increasing trend in malware activity starting from mid-day could be due to waking times hard-coded in their software, possibly configured in a time zone different from ours. One may rightly ask *how many* internal hosts, of *which* type, and from *where* in the network make these DGA-based DNS queries? We will provide insights into possible infected hosts later in Table III (briefly) and Section IV (in detail).

By analyzing the DNS activity pattern of various DGA families, we categorize them into three groups, namely (a) Frequent and Heavy, (b) Frequent and Light, and (c) Bursty. For brevity reasons, we illustrate the activity pattern of only representative families from these three groups in Fig. 5. Fig. 5(a) corresponds to the most frequent and the heaviest family, the "ModPack". We observe that ModPack is highly active (thousands of queries) during most of the days (starting from 1st October), except for nine days on which it becomes completely inactive. Moving to the "Suppobox" family in Fig. 5(b) representing a frequent and light family, it is seen that the daily queries count is fairly low (less than 20), but it rarely goes inactive on a day. Lastly, as a bursty pattern family shown in Fig. 5(c), we see that 'Ramnit" displays 2 bursts (end of October and end of November) during the entire period of our analysis, and it remains completely inactive otherwise. It is important to note that these activities could be due to either a large number of infected hosts or certain infected hosts making a large number of DGA queries – we will discuss in §III-D the challenge of identifying infected hosts purely based on DNS traffic. We also found some forms of coordination across various DGA families in terms of their activity. For example,



(a) ModPack (frequent and heavy). (b) Suppobox (frequent and light). (c) Ramnit (bursty).

Fig. 5. Time-trace of daily DNS queries count for various DGA-enabled malware families: (a) ModPack, (b) Suppobox, and (c) Ramnit.

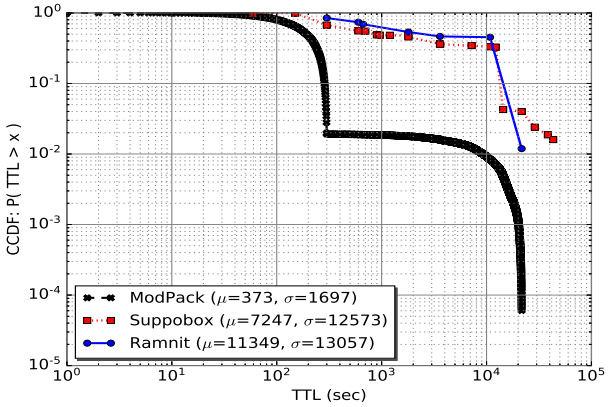


Fig. 6. CCDF of TTL value in DGA-related DNS responses across representative malware families.

TABLE III
ENTERPRISE HOSTS MAKING DGA-ENABLED QUERIES.

Querying hosts (# hosts)	**# DGA queries [%]**
Recursive resolvers (3)	552,669 [93.82%]
Enterprise servers (8)	14,217 [2.41%]
End hosts (13)	283 [0.05%]
WiFi NAT gateways (19)	21,929 [3.72%]

Suppobox and Ramnit simultaneously became heavily active on 24th Oct as well as towards the end of November (*i.e.,* from 28-Nov to 01-Dec) as shown in Figures 5(b) and 5(c). We will highlight some examples of such coordination in §III-E.

To further investigate the periodicity of these three representative families, we extracted the TTL value from their DNS responses. Fig. 6 shows the CCDF of TTL values for each family. It is seen that the ModPack family (dashed black lines with cross markers) tends to use fairly short TTLs with an average of 373 seconds – 90% of ModPack DNS responses will live less than three minutes. The TTL values in the Suppobox family are relatively longer, averaging at 7247 seconds while 50% are more than 15 minutes. Lastly, the Ramnit family has the longest TTL values with an average of 11349 seconds. Similar to the Suppobox family, Ramnit also has 50% of its responses with a TTL greater than 15 minutes. Overall, the distribution of TTL values in various families explains (to a great extent) their frequency of occurrence – the shorter the TTL, the more frequent they become. We will use (in §IV-B) TTL values for timeout setting of reactive flow entries installed by the SDN controller, to enable scalable management of switch TCAM table.

TABLE IV
DGA-ENABLED MALWARE AND INFECTED CAMPUS HOSTS FOUND BY ANALYSIS OF ONE-HOUR PCAP OF FULL CAMPUS TRAFFIC.

	Count
DGA-enabled families found in DNS queries of campus hosts	8
C&C servers identified in responses to DGA queries	14
C&C servers exchanged traffic with internal hosts	5
DGA-enabled families involved in C&C traffic exchange	2
Num. of malware-infected hosts exchanged traffic with C&C servers	17
End hosts	8
WiFi NAT gateways	7
Enterprise servers	2

D. Infected Enterprise Hosts

We now look at enterprise hosts that make DGA-enabled queries. Table III shows that a very high majority (93.8%) of DGA queries are sourced from DNS recursive resolvers, and hence the original querying end-hosts are invisible, except a limited number of hosts (8 enterprise servers and 13 regular hosts) which are probably configured to directly use public DNS resolvers (*e.g.,* Google `8.8.8.8`). We also note that 3.7% of DGA queries are generated by 19 WiFi NAT gateways – the identity of infected hosts (WiFi clients) that generate these queries remains unknown, as they reside behind NAT gateways inside the network (we collect data from the border of the network).

Obviously, enterprise recursive resolvers and NAT gateways hide the identity of infected hosts, and hence host analysis purely based on DNS traffic will not suffice for identifying infected hosts. To better understand the network activity of (possibly) infected hosts following their suspicious DNS queries (found in DGArchive), we analyze (in what follows) a 1-hour PCAP trace of full traffic dump from the campus network. We focus on subsequent TCP/UDP traffic exchanged between Internet-based C&C servers (following the response of DGA-based DNS queries) and their respective enterprise hosts – we will show in §V how our SDN-based method in conjunction with trained models (§IV) will be able to identify infected enterprise hosts.

One may choose to inspect packets of enterprise DNS servers to gain richer insights into the health of internal hosts purely based on DNS traffic. However, obtaining all (incoming/outgoing) packets of various DNS servers requires significant changes to the distributed infrastructure of large enterprise networks [15]. Our solution (§IV), instead, is designed to be a "bump-in-the-wire" on the Internet link of the

enterprise network and provides different visibility (a judicious combination of DNS and selected subsequent TCP/UDP flows) into activities of connected hosts from a different perspective. Our detection system is transparent to the network, requires minimal change to existing infrastructure, easy to deploy, and does not modify packets in any way.

E. Network Behavior of DGA-Fueled Malware

As explained in the previous section, purely monitoring DNS traffic does not lead to finding the hosts that are indeed suspected of malware infection. Furthermore, to draw insights into the behavior of suspected hosts (in terms of services used and/or any possible coordination across infected hosts while communicating with their corresponding C&C servers), we analyze a very short but fairly active period of the full campus traffic dump. This full dump of the entire Internet traffic contains 3.2B packets of which only 2M packets are DNS – less than 0.1%.

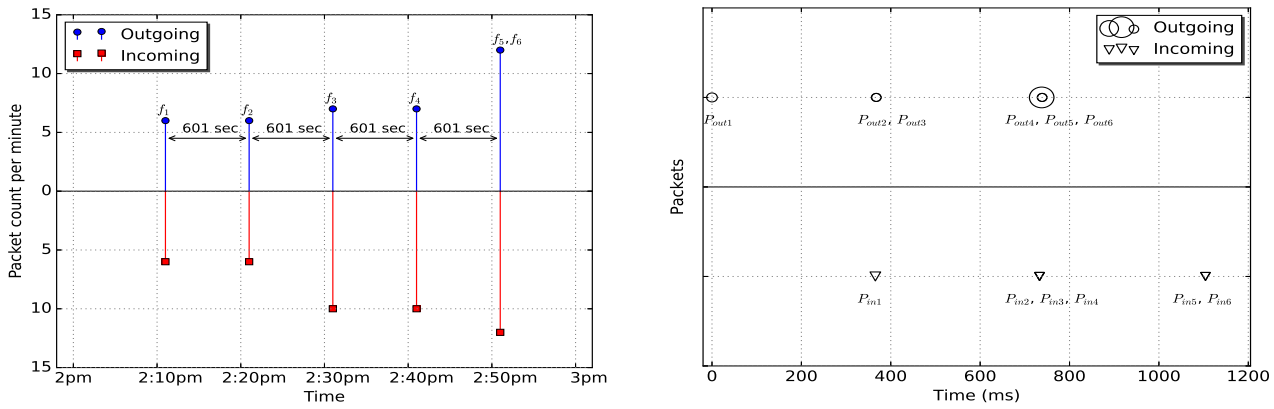
Table IV summarizes our findings from this analysis. During this one hour, eight DGA-enabled malware families were found in the DNS packets of this 1-hour PCAP trace. Also, a total of 14 unique C&C servers were identified from DNS responses, and only five of these C&C servers (corresponding to two DGA families) exchanged TCP traffic with enterprise hosts following their DGA-based DNS resolution. Analyzing these follow-up TCP flows, we found 17 hosts (we call them "suspicious hosts") communicated with the five C&C servers. By reverse DNS lookup, we verified that 8 of the suspicious hosts were regular end-hosts, 2 were enterprise servers, and 7 were WiFi NAT gateways. These 17 suspicious hosts accessed HTTP and HTTPS services offered by their corresponding C&C servers, generating 33 suspicious HTTP flows (75%) and 11 suspicious HTTPS flows (25%) that collectively exchange a total of only 365 packets which contribute to a tiny fraction (10^{-7}) of total packets (3.2 B) recorded in this one-hour PCAP trace from campus Internet traffic.

Additionally, we analyzed the behavior of these suspicious flows to highlight their activity patterns, or possibly find coordination across suspicious hosts [63]. Fig. 7(a) shows the pattern of communications (on a flow basis) between a suspicious host (a WiFi NAT gateway in this case) and a C&C server of Ramnit family with IP address "`89.185.44.100`" resolved by a query for domain name "`lvxlicygng.com`" (with the response TTL value of 300 sec) – we verified this address is blacklisted [64]. We observe that this WiFi gateway (possibly on behalf of a number NATed hosts) initiates six HTTPS flows (each with a duration of less than a few seconds) to the server in the time period between 2:10pm and 2:50pm. Note that two flows f_5 and f_6 are established concurrently. The height of each flow in the plot (Fig. 7(a)) indicates the number of packets sent and received (outgoing direction shown by blue circles and incoming direction shown by red squares). Interestingly, there is a clear periodicity in the arrival of these flows – they are well spaced by 601 sec ($\approx$10 minutes). Some of these flows (*e.g.,* f_1 and f_2) are symmetric in terms of incoming and outgoing packet count, and some are asymmetric (*e.g.,* f_3 and f_4).

To further understand the network activity of these suspicious flows, we zoom in on the arrival and size of individual packets within each flow per direction. As an example, we show in Fig. 7(b) the time trace of packets in the flow f_2. The x-axis is time (in ms) and the y-axis indicates the direction (top row corresponds to outgoing packets and the bottom row corresponds to incoming packets). Also, each marker represents a packet (circles for outgoing and triangles for incoming), and the size of the markers is indicative of the relative length of the corresponding packet. The duration of this flow is about 1100 ms over which six packets are sent and six packets are received. In this flow, all incoming packets from the server have the same size of 60 bytes. Within less than 400ms from the commencement of the flow, the three-way handshake is completed, *i.e.,* SYN (P_{out1}) $\rightarrow$ SYN-ACK (P_{in1}) $\rightarrow$ ACK (P_{out2}). Right after establishing the TCP connection, the host sends 60 bytes data over SSL (P_{out3}). In response, the server sends SSL data of 60 bytes (P_{in2}) followed by FIN-ACK (P_{in3}) and TCP-RST (P_{in4}) packets. Next, the host sends SSL data of 139 bytes (P_{out4}), followed by ACK (P_{out5}) and FIN-ACK (P_{out6}) packets. Lastly, the server sends two more TCP RST packets (P_{in5} and P_{in6}) back-to-back. Observing such activity patterns will help us (in §IV-A) identify flow-level attributes needed for modeling malware traffic behavior. As stated earlier in §II, our main objective in this paper is to develop a "cost-effective", yet "accurate" solution for detecting malicious flows and infected hosts in "large-scale enterprise networks". We will use flow-level attributes (as opposed to computationally expensive packet-level attributes) to diagnose whether selected suspicious traffic is malicious, or not.

As another example, we show in Fig. 8 the flow activity of an end-host establishing a suspicious HTTP flow with its C&C server. It can be seen that in this case, the server sends some data (not just acknowledgments) to the internal host. In each direction, 5 packets are exchanged between the end-host and the server over this flow with a duration of about 4600 ms. The three-way handshake (SYN:P_{out1} $\rightarrow$ SYN-ACK:P_{in1} $\rightarrow$ ACK:P_{out2}) completes within the first 160 ms. Following the establishment of this connection, the end-host sends an `HTTP GET` request (P_{out3} with the size 420 bytes) to the server at the time about 4100 ms. The server responds with an ACK (P_{in2}) followed by 1001 bytes `HTTP OK` (P_{in3}) – we were unable to inspect the packet payload since in our PCAP trace packets are truncated to their first 96 bytes. Right after that, the server initiates termination of this TCP flow by sending a FIN-ACK (P_{in4}) – this packet is followed by ACK (P_{out4}) and FIN-ACK (P_{out5}) packets from the host, and the final ACK (P_{in5}) from the server.

We also found two incidents of possible coordination [33] across suspicious hosts while contacting their C&C servers at around 2:10pm and 2:20pm. In the first instance (at 2:10pm), two suspicious end-hosts simultaneously initiated HTTP flows: one host initiated four flows and the other one initiated eight flows both with a C&C server from Suppobox family with IP address "`184.168.131.241` (corresponding to domain name "`strengthstorm.net`" with the response TTL value of 600 sec) – we verified that this IP address is blacklisted [65]. In the second instance (at 2:20pm), a WiFi NAT gateway and an



(a) Sequence of flows.

(b) Sequence of packets in flow "f_2", for example.

Fig. 7. Communication pattern of a suspicious host with its C&C server of Ramnit family: (a) sequence of HTTPS flows, and (b) sequence of packets in a selected HTTPS flow.

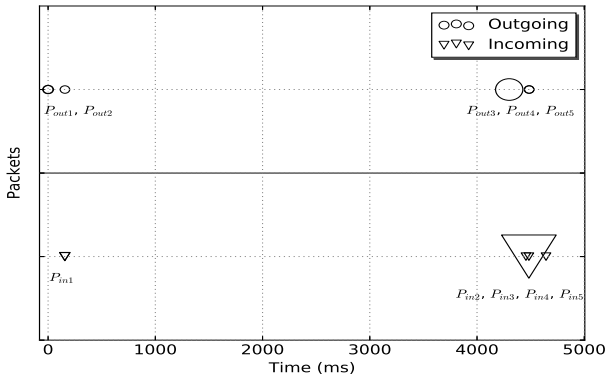


Fig. 8. Time trace of packets (outgoing/incoming) in a selected HTTP flow.

end-host respectively initiated one and two HTTP flows with a C&C server from the Suppobox family with the same IP address "`184.168.131.241`".

Compute Cost: Our insights into network activities of malware were only obtained by manually analyzing a short but representative PCAP trace of the campus network. It is important to note that it becomes very expensive (practically infeasible) to capture and analyze packets of all network flows at high rates (10Gbps or more). We also note that malicious traffic (specifically for malware and botnet) often contributes to a very small fraction of the total network traffic – the majority of packets are benign. Therefore, it is needed to employ a systematic and scalable method to capture only suspicious traffic (corresponding to servers that are resolved as a result of DGA queries) and check whether it is malicious or not. In the next section, we will leverage the ability of SDN to dynamically mirror suspicious traffic flows, and develop learning-based models (based on insights obtained in this section) to automatically detect malicious flows their associated infected hosts inside the enterprise network.

IV. Modeling and Mirroring Traffic of Suspicious Malware Servers

In this section, we begin by developing our protocol-specialist models (one corresponding to each of HTTP, HTTPS, and UDP protocols) using CTU-13 network traces (discussed in §III-A). We, next, develop a system to automatically select, mirror and diagnose traffic flows corresponding to suspicious malware servers. We employ SDN reactive rules to select and mirror suspicious traffic to our packet processing engine. The engine feeds our trained models by a set of flow attributes for diagnosis, determining whether these selected TCP/UDP flows are malicious, or not.

A. Modeling Traffic Behavior of Malware

We use malware PCAP traces of the CTU-13 dataset [24]. Authors of [24] primarily aimed to detect malicious "hosts" by developing clustering-based models from their dataset. They employed host-level attributes including the count of remote IP addresses, count of remote/local transport port numbers, average packet size, and the average count of packets transmitted over windows of every two-minute. We note that computing these attributes in near real-time for every internal host (tracking metadata of all packets) will be computationally prohibitive, especially at scale. Also, traffic modeling at the host-level becomes slightly coarse-grained (aggregate of benign and malicious flows) which can result in reduced visibility into the activity of individual malware flows. Our approach, instead, aims to characterize the behavior of malware activity on a per-flow basis, and diagnoses a fraction of network flows, only those suspicious TCP/UDP flows pertinent to a DGA-related DNS query. We develop a set of machine learning models (a model per protocol) to determine if a flow is malicious or not. Many cybersecurity researchers [24], [66], [67] employed multi-class decision trees to distinguish malicious and benign traffic, however, balancing the training dataset to avoid overfitting remains a nontrivial challenge [68]. It has been shown [42], [69] that one-class classifiers or anomaly detection models are able to learn the distribution of training data (malicious flows in the context of this paper), and detect any deviations (benign flows) during the testing phase. Our protocol-specialist models will generate "negative" output for malicious instances, and "positive" output otherwise. This use of one-class models means that each model can be re-trained/updated (in case of extending the malware dataset), independent of the other models.

1) Attributes and Classifiers for Malware Flows: Inspired by [24], we identify eight attributes on a per-flow basis for

TABLE V
DISTRIBUTION (μ AND σ) OF ATTRIBUTES VALUE FOR MALICIOUS FLOWS IN CTU-13 DATASET.

	Outgoing				**Incoming**			
	flow volume (B)	flow duration (s)	# pkts	Avg. pkts size (B)	flow volume (B)	flow duration (s)	# pkts	Avg. pkts size (B)
HTTP	(980, 666)	(7.3, 29.5)	(5, 1)	(172.5, 94.1)	(1470, 1589)	(1.5, 7.6)	(4, 1)	(270.5, 235.4)
HTTPS	(1597, 4392)	(761.3, 2832.3)	(12, 41)	(81.1, 59.4)	(5805, 47043)	(760.6, 2832.1)	(7, 20)	(187.4, 390.9)
UDP	(5968, 100239)	(708.8, 2326.2)	(14, 165)	(201.6, 164.6)	(5968, 100239)	(712.3, 12451.2)	(14, 165)	(201.6, 164.6)

malware traffic – 4 per each direction (in/out). Our attributes of a malware flow are as follow.

- *flow volume* in bytes (in/out).
- *flow duration*, *i.e.,* the gap between the arrival time of the first packet and the last packet (in/out).
- *number of packets* (in/out).
- *average packet size* (in/out).

We computed the above attributes for all (labeled) malicious flows in the CTU-13 dataset. We show in Table V the distribution (μ and σ) of raw values of attributes across the entire dataset. Note that the malware flows in the CTU-13 dataset are from three protocols, namely HTTP, HTTPS, and UDP. By analyzing these values, we observe that flow volume, flow duration, and packet count for both incoming and outgoing directions are key attributes in characterizing the three categories of malware flows. As an example, considering the outgoing direction, the mean (μ) volume of flow in HTTP, HTTPS, and UDP malware is about 1000, 1600, and 6000 bytes, respectively (first column of Table V).

Let us make some high-level observations on the range of attributes across the three types of malicious flows. Note that the variation of flow volume is much larger in HTTPS (with $\sigma \approx 4400$) and UDP (with $\sigma \approx 100,000$) flows than in HTTP flows (with $\sigma \approx 700$). A slightly similar pattern is observed in the flow duration and packet count attributes. UDP and HTTPS flows, compared to HTTP flows, are generally longer in duration (mean 700s versus mean 7s), and carry a larger number of packets (mean $12 - 14$ packets versus mean 5 packets). Such a clear distinction between the three categories (HTTP, HTTPS, and UDP) can also be seen in the values of attributes for the incoming direction. Therefore, we train three separate models (each specific to a protocol), increasing the accuracy of detecting malware flows. We split the malicious data of each protocol-specific model into 60% (for training and validation) and 40% (for testing only).

2) Model Training: We used `scikit-learn` and its APIs, an open-source machine-learning package written in Python, to train and test our models. Our prediction models are one-class classifiers trained by four popular algorithms, namely Isolation Forest (iForest) [70], Extended iForest (EiF) [71], K-means [72], and one-class support vector machines (OC-SVM) [73] using attributes of malicious flows obtained from the CTU-13 dataset. The models classify a flow: if the subject flow is tested negative by its corresponding model (*i.e.,* HTTP, HTTP, or UDP), then it is classified as "malicious", otherwise it is "benign". Later in this subsection, we will compare the performance of one-class models against that of multi-class classifiers.

The iForest algorithm works based on the concept of isolation without employing any distance or density measure. The algorithm divides instances into sub-samples to construct a binary tree structure – by randomly selecting the attribute, and then randomly selecting the split values from a range (within min and max obtained from training) for that particular attribute – splitting values is always done by an "axis-parallel" hyperplane (*e.g.,* rectangular shape in 2D space of attributes). If the value of a given instance is less than the split value, the point is directed to the left branch of the tree structure otherwise it goes to the right side branch. This branching is performed recursively until either a predefined height limit of the tree is approached or a single point is isolated in the dataset. The algorithm then marks the instances that travel less into the tree structure as an anomaly while the ones that travel deeper into the tree structure are classified as benign. To avoid issues due to randomness, the process is repeated several times, and the average path length is calculated and normalized.

For the training phase of the iForest models, we consider three tuning parameters, namely the number of trees (*n_estimators*), height limit of trees (*max_samples*), and contamination rate. For each of the three models, we tune the value of each parameter while fixing the other two parameters and validate the accuracy of our specialized models for the malicious flows in the CTU-13 dataset. The default value for the number of trees is 100, the height limit of trees is set to "auto", and the contamination rate is 10%. After tuning the individual three models, we found the optimal value of these tuning parameters as follows: the number of trees equal to 10, the height limit of trees equal to 8, and contamination rate of 1% for all of the three models.

It has been recently shown [71] that while iForest is a computationally efficient algorithm, it suffers from a bias (affecting the anomaly score) arisen by its use of axis-parallel hyperplanes. Authors of [71], therefore, enhance the standard iForest algorithm, by developing Extended Isolation Forest (EiF) which performs data splitting with "random-slope" hyperplanes. Our EiF models are tuned in the same way as iForest.

The K-means algorithm finds groups of instances (aka clusters) for a given class that are similar to one another. Every cluster is identified by its centroid, and an instance is associated with a cluster if the instance is closer to the centroid of that cluster than any other cluster centroids. For better performance of K-means models, it is important to pre-process our data and tune certain parameters. We begin by recording the Z-score (*i.e.,* computing mean μ and standard deviation σ) of each attribute. We then normalize our dataset instances by calculating the deviation from the mean divided by the standard deviation with respect to each attribute. To tune a K-means model, we need to compute the optimal number

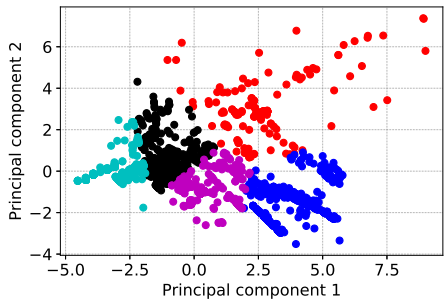


Fig. 9. Clusters of K-means model for HTTP flows.

TABLE VI
DISTRIBUTION (μ AND σ) OF ATTRIBUTES FOR K-MEANS HTTP CLUSTERS.

	Outgoing				Incoming			
	# pkts	**flow vol.**	**Pkt size**	**flow dur.**	**# Pkts**	**flow vol.**	**Pkt size**	**flow dur.**
	(5, 0.3)	(1103, 1645)	(214, 26)	(2, 7)	(4, 0.5)	(995, 325)	(218, 77)	(0.42, 1.4)
	(7, 0.8)	(838, 216)	(124, 30)	(7, 19)	(7, 1.1)	(4993, 1222)	(767, 222)	(4.0, 15.1)
	(4, 1.1)	(358, 138)	(83, 19)	(7, 32)	(3, 0.7)	(200, 95)	(71, 2)	(0.6, 2.6)
	(7, 0.6)	(2457, 420)	(366, 55)	(1, 6)	(6, 0.9)	(2770, 992)	(449, 104)	(0.3, 0.8)
	(5, 0.9)	(663, 118)	(125, 27)	(15, 44)	(5, 0.8)	(764, 295)	(172, 80)	(2.3, 9.7)

of clusters that is obtained by the elbow method [74]. By applying this method, we found the optimal number of clusters for all three models to be equal to 5. We show in Fig. 9 the resulting (color-coded) clusters of training instances for HTTP flows. Note that our instances are multi-dimensional (*i.e.,* each instance contains 8 attributes), and thus cannot be easily visualized. Therefore, for illustration purposes, we employ Principal Component Analysis (PCA) to project the data instances onto two-dimensions.

Additionally, Table VI provides further insights into attributes of these five clusters. We make a few observations: these clusters cannot be distinctly identified by their packet count attribute – almost similar across all clusters; cyan and black seem to represent top heavy clusters (cyan by total flow volume in both directions and black by the average size of incoming packets); comparing red (top row) and purple (bottom row) clusters we find that the average size of packets in the red cluster is about 20% (incoming) to 70% (outgoing) larger than that of the purple cluster – also, duration of flows in the red cluster is one order of magnitude shorter than that in the purple cluster.

OC-SVM is an algorithm that identifies anomalous instances by constructing a hyperplane boundary around expected training instances. It comes with three main tuning parameters, namely *Kernel*, *gamma*, and *nu* with default values, respectively equal to "radial basis function (rbf)", "scale" (inverse of product of attributes count and attributes variance) and "0.5". We tune OC-SVM similar to iForest and EiF where a parameter is fixed, and others are varied to find the best prediction. The optimal tuning parameters are found to be as follows: "rbf" kernel, gamma equals 0.125, and nu equals 0.05.

3) Model Validation: We validate the performance of our trained models against training instances (only 60% of the malicious dataset since benign instances are not used for training our one-class models). Validation results are shown by top row in Tables VII (HTTP model), VIII (HTTPS model), and IX (UDP model). It is observed that three algorithms namely iForest, EiF, and K-means perform fairly well (giving consistently high accuracy of more than 97% across the three protocol-specialist models) during the validation phase. However, the OC-SVM algorithm performs relatively poorly even for validation, with the best malicious detection rate of less than 73% given by the UDP model.

To better understand the inferencing capability of top-performing algorithms (*i.e.,* iForest, EiF, and K-means) we

TABLE VII
ACCURACY OF HTTP MODELS
IN SUCCESSFULLY DETECTING MALICIOUS AND BENIGN FLOWS.

		iForest	**EiF**	**K-means**	**OC-SVM**	**RandomForest**
Validation	**malicious**	98.71%	99.05%	98.27%	63.18%	94.13%
	benign	–	–	–	–	95.68%
Testing	**malicious**	97.80%	98.85%	98.11%	61.09%	81.32%
	benign	93.55%	93.91%	91.64%	59.43%	79.18%

TABLE VIII
ACCURACY OF HTTPS MODELS
IN SUCCESSFULLY DETECTING MALICIOUS AND BENIGN FLOWS.

		iForest	**EiF**	**K-means**	**OC-SVM**	**RandomForest**
Validation	**malicious**	98.96%	99.13%	98.42%	65.59%	93.89%
	benign	–	–	–	–	96.73%
Testing	**malicious**	97.16%	98.92%	98.45%	64.90%	80.27%
	benign	94.51%	94.59%	91.92%	62.94%	82.52%

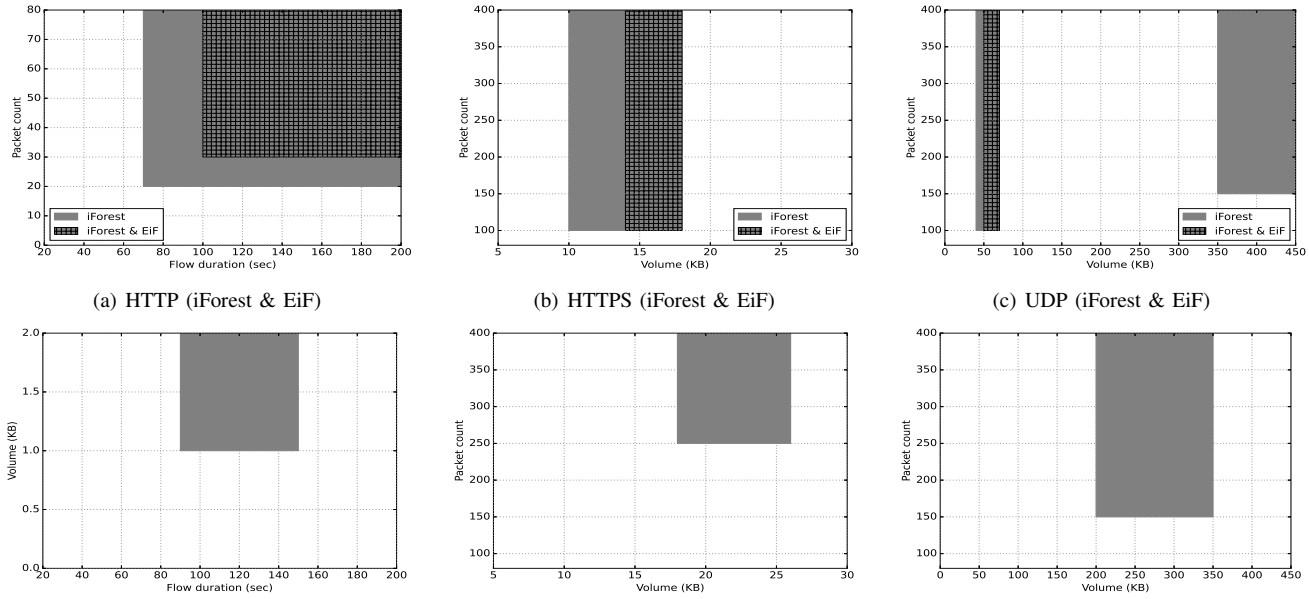
TABLE IX
ACCURACY OF UDP MODELS
IN SUCCESSFULLY DETECTING MALICIOUS AND BENIGN FLOWS.

		iForest	**EiF**	**K-means**	**OC-SVM**	**RandomForest**
Validation	**malicious**	96.96%	97.60%	98.76%	72.91%	95.32%
	benign	–	–	–	–	96.21%
Testing	**malicious**	96.28%	97.03%	97.58%	73.35%	82.92%
	benign	92.14%	92.99%	92.03%	68.52%	80.18%

now focus on misclassified flows across these models. Fig. 10 visualizes an approximation of two-dimensional regions for key attributes of misclassified flows. We found that a diagnosed flow is misclassified (with a probability of more than 90%) by respective models if its attributes fall in the highlighted regions of Fig. 10. Let us start with the iForest and EiF models, on the top row. It is seen that they misclassify those HTTP flows which are large in packet count and long in duration, as shown in Fig. 10(a), while for HTTPS and UDP flows, having a large packet count (more than 100 packets) will probably lead to misclassification, as shown in Figures 10(b) and 10(c). K-means models, on the other hand, tend to misclassify HTTP flows with smaller volume but medium-length (Fig. 10(d)), and HTTPS and UDP flows with larger packet count (Fig. 10(e) and 10(f)).

Note that we employ four widely used one-class algorithms to compare their performance. Taking the above observations into account, none of these models seem distinct except by their overall accuracy. Hence, we choose the best-performing model (EiF) for our trial evaluation in §V.

4) Models Testing: Following validation, we quantify the performance of our trained one-class models against testing malicious instances (the remaining 40% of malicious flows) as well as the entire set of benign instances. Testing results



(a) HTTP (iForest & EiF) (b) HTTPS (iForest & EiF) (c) UDP (iForest & EiF)

(d) HTTP (K-means) (e) HTTPS (K-means) (f) UDP (K-means)

Fig. 10. Attributes of misclassified flows: (a,d) HTTP, (b, e) HTTPS, and (c,f) UDP, across top performing classifiers.

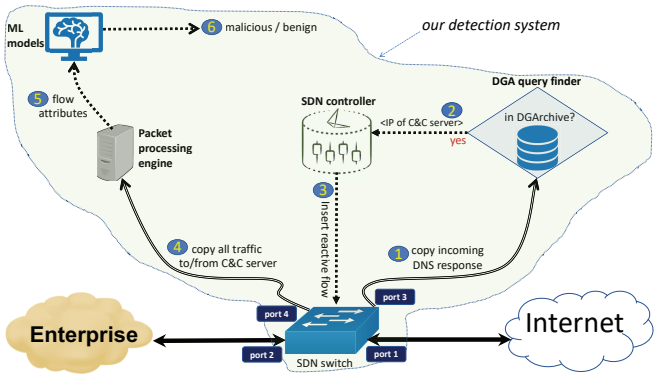


Fig. 11. System architecture of our detection system.

are shown by bottom row in Tables VII (HTTP model), VIII (HTTPS model), and IX (UDP model). We observe that EiF consistently gives the best accuracy for both malicious and benign testing flows, across the three models – true negatives of more than 97% and true positives of at least 93%. Unsurprisingly, OC-SVM is found to perform very poorly (compared with iForest, EiF, and K-means) during the testing phase with (malicious and benign) detection rates of mostly less than 70%.

One-class versus Multi-class Models: Lastly, in order to highlight the shortcomings of multi-class models in diagnosing the health of network traffic, we consider a two-class (malicious and benign) Random Forest classifier. It is trained with 60% of the entire CTU-13 dataset and its performance is tested with the remaining 40% of instances. Validation and testing results are shown by the last column of Tables VII, VIII, and IX. It can be seen that Random Forest's detecting rates (true positives and true negatives) are around 80% which is lower than those of iForest, EiF, and K-means particularly during the testing phase, though it gives acceptable detection rates ($\approx 95\%$) during the validation phase.

B. Dynamic Traffic Selection using SDN

Fig. 11 shows the functional blocks in our system architecture applied to a typical enterprise network. Enterprise users are on the left and can be on an access network (wired and/or wireless). The Internet is on the right. Our solution is designed to be a "bump-in-the-wire" on the link at which traffic monitoring/management is desired (active management) – an alternative approach is to feed our system a mirror of all network traffic (passive monitoring). Our system is therefore transparent to the network and does not modify packets in any way. Further, no packet is sent to the SDN controller (for dynamic management of flow rules); instead, selected traffic (DNS packets, C&C flows) that need inspection or diagnosis are sent as copies on separate interfaces of the switch, to which specialized traffic analytics engines (software inspection) are attached. This protects the controller from overload from the data-plane, allowing it to scale to high rates and to serve other SDN applications. Our solution comprises a DGA query finder (top right of Fig. 11) which is fed by real-time incoming DNS responses, an SDN switch whose flow-table rules will be managed dynamically by API calls from the DGA finder to the SDN controller (center of Fig. 11), a packet processing engine that extracts flow attributes (of suspicious network traffic only) feeding machine learning-based models (top left of Fig. 11).

Our DGA query finder engine (handling DNS traffic of our university Internet link with a peak load of about 10 Gbps) runs on a virtual machine with 6 CPU cores, 8GB of memory, and storage of 500GB. We believe that cost and complexity of processing DNS packets in software can be reasonably managed at scale. In terms of processing costs, empirical results of our previous research studies [14], [15] on traffic analysis of our university campus network show that DNS constitutes a tiny fraction (less than 0.1%) of total network traffic by volume. This corroborates with the analysis of the one-hour full campus traffic trace collected during the peak hour (§III-E), revealing that out of the total 3.2B packets,

TABLE X
DISTRIBUTION OF MALWARE FAMILIES AMONG SUSPICIOUS FLOWS, SELECTED AND MIRRORED BY OUR SDN SYSTEM.

DGA-enabled malware (# C&C servers)	# flows [%]
ModPack (7)	35941 [63.10%]
Matsnu (2)	20806 [36.53%]
Ramnit (2)	169 [0.30%]
Suppobox (8)	37 [0.06%]
Bamital (1)	4 [0.01%]

only 2M are DNS – about 0.06%. In terms of complexity of measurement, it is relatively easy to capture DNS traffic with only a few flow entries (*i.e.,* mirroring IPv4 and IPv6 UDP packets to/from port 53) in an SDN switch.

We believe that this paper's key contribution is our detection method that dynamically (using SDN) and confidently (using specialized classifiers) identifies malicious flows established after DGA queries. Our system infers suspicious hosts by employing a broad signature from the public database DGArchive. It verifies its initial inference by applying specialized one-class classifiers. The key advantage of our method is its scalability and low rate of false alarms.

For the SDN switch, we use a fully Openflow 1.3 compliant NoviSwitch 2122 [75] which is controlled by the Ryu SDN controller [76]. The switch provides 240 Gbps of throughput, up to one million TCAM flow-entries, and millions of exact-match flow-entries in DRAM, and we found it to amply cater to the requirements of this project. We use a combination of proactive and reactive entries in the switch flow table. A proactive entry is statically pushed by the controller so that all DNS response packets (*i.e.,* UDP source port 53) received from the Internet are forwarded (on port 2) and mirrored (on port 3) to the DGA finder, as shown by step ① in Fig. 11. The finder looks up the queried domain name against DGArchive, and if found, it extracts the IP address of the server resolved by DNS responses and subsequently calls the SDN controller (step ②) that results in the insertion of a reactive flow-entry, shown by step ③. Note that our malware detection method is limited by the knowledge-base provided by the DGArchive repository. This means that in order to detect new families of DGA-enabled malware flows, whose DNS query name is outside of this database, one needs to update the DGArchive – it has been consistently maintained over the past five years.

We note that DGArchive contains all domains queried by various malware families, but there are two challenges: (a) queried domains are not necessarily malicious, (b) DGA queries do not necessarily lead to C&C communications. As highlighted earlier in §III-A, some of the more recent malware families, for some reasons (evading security appliances or bugs in their code), tend to make queries for very legitimate domains like "`github.com`" or "`oracle.com`" or some relatively legitimate domains found in the Majestic Million database [77] (a public database of trusted websites). In addition, merely making DNS queries (even for malicious domains) itself is not harmful unless a subsequent communication occurs with a C&C server. Results from our 50-day trial (§V) show that only 12% of DGA queries lead to a subsequent communication with a C&C server. Therefore, observing a domain name found in DGArchive does not warrant the host is infected (a high possibility of false-positive) unless their flow-level behavior is further verified (by specialized models) to be similar to that of known malware (§IV-A). One may also want to examine the registration date of domain names prior to mirroring the traffic, which is beyond the scope of this paper. We believe that examining the registration date of domains in "real-time" can be challenging since many TCP/UDP flows (between internal host and external C&C server) often commence shortly (less than 100ms) after their DNS resolution (will be discussed later in §V, Fig. 15), and hence the delay of registration look-up may cause missing the C&C communication flow. In addition, mirroring "selected flows" which carry a fairly small number of packets and are relatively short (will be discussed later in §V, Fig. 16) would not incur significant cost of software processing or switch TCAM entries (will be discussed later in §V, Fig. 17).

TABLE XI
SYSTEM SPECIFICATIONS.

Requirement	System component	Specifications
Hardware	**DGA query finder**	6 CPU cores, 8GB RAM, and 500GB storage
	Packet processing engine	4 CPU cores, 6GB RAM and 200GB storage
	SDN switch	NoviSwitch 2116
Software	**Operating System**	Ubuntu 16.04
	Programming language	Python 3.7
	ML Library	Scikit-learn
	ML models	iForest, EiF, K-means, OC-SVM and Random Forest
	SDN controller	Faucet for proactive flows and Ryu for reactive flows

Reactive rules which match the IP address of the server (two rules per server: one matching source and one matching the destination IP address) are of the highest priority and get installed as a consequence of DGA queries detected by the DGA finder. To protect the SDN switch from TCAM exhaustion (scalable management of TCAM usage), reactive rules are automatically timed out after a period equals to the TTL obtained from their corresponding response. The reactive flow entries provide filtered packets (to/from potential C&C servers) to the packet processing engine on port 4 in step ④. Our packet processing engine (run on a generic server configured with Ubuntu version 16.04.4) analyzes suspicious traffic filtered and mirrored by the SDN switch. It constructs flow-level attributes that are fed, in step ⑤, to the machine learning (ML) models for prediction. The models are one-class classifiers (discussed in §IV-A) distinguishing, step ⑥, malicious flows from benign ones. We will describe in §V the performance of our prototype under real traffic of our campus network.

V. EVALUATION RESULTS

We have implemented a fully functional system (shown in Fig. 11), and operated it during a 50-day trial (3-Dec-2019 to 21-Jan-2020) under full campus network traffic. Table XI summarizes hardware and software specifications of the system components we developed for data collection, model training, testing, validation and performance evaluation. During this trial, our system automatically selected, mirrored, and

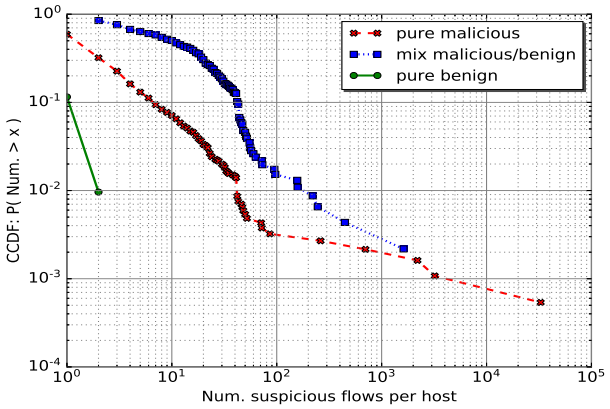


Fig. 12. CCDF of number of suspicious flows per host.

recorded suspicious DGA-based flows (subsequent to DGA-based DNS responses) in real-time at the line rate of up to 10 Gbps using reactive SDN rules. In addition to suspicious flows, the entire DNS traffic was recorded during this trial to correlate DGA queries with their corresponding suspicious flows. In this section, we evaluate the efficacy of our trained models by applying them (in offline mode) to recorded suspicious flows from the trial, diagnosing their health (malicious or benign?).

SDN-selected DGA Flows: We perform a correlation between SDN-selected suspicious flows and DNS responses of DGA families. Table X summarizes the distribution of suspicious flows (a total of 56,957) that are associated with (DNS queries of) five DGA families listed in rows. It is seen that ModPack family dominates with 63.10% of flows, followed by the Matsnu family with 36.53% of flows. Note that these flows are exchanged between enterprise hosts and a small number (a total of 20) of C&C servers on the Internet – the number of unique servers (IP addresses) is listed in the bracket in front of their respective DGA family. We observe that the activity of these servers varies across families. For example, 7 servers associated with the ModPack family handle about 36K flows while 8 Suppobox servers exchange 37 flows over the 50-day period of our experiment.

During our trial, we found that C&C servers of 14 DGA-enabled malware families were successfully resolved (476K DNS responses). However, the C&C servers of only 5 families were contacted by internal hosts, following their DNS resolution. We observe that those 9 malware families which generated no C&C communications, contribute to only 0.1% of the total resolved DGA queries. We note that the probability of communicating with an intended C&C server following a successful DNS resolution varies across families. For example, the top two families display a completely different pattern. ModPack, which dominates by making 35.9K flows, had 475K DNS queries resolved – apparently, most of these DNS queries (with an average TTL value of ≈5 minutes) are made purely to keep their local cache updated by the latest IP address of their intended server. On the other hand, Matsnu generated 20.8K flows with only 22 DNS responses (with much longer TTL values averaged at ≈5 hours) during our 50-day trial – in this case, DNS queries are only made when certain communications are desired. Of the remaining three families, Ramnit behaves similar to Matsnu by exchanging 169 C&C flows with 14 DNS responses (average TTL ≈4 hours), while Suppobox (37 flows, 97 DNS responses, average TTL ≈1.5 hours) and Bamital (4 flows, 11 DNS responses, average TTL ≈10 minutes) display a pattern like ModPack.

TABLE XII
RESULTS OF TESTING SUSPICIOUS FLOWS AGAINST THEIR CORRESPONDING EIF MODELS.

	HTTP	HTTPS	UDP	Aggregate
# suspicious flows	35645	19674	1638	56957
% malicious	99.94%	93.92%	92.71%	97.63%
% benign	0.06%	6.08%	7.29%	2.37%
# internal hosts making suspicious flows	262	2367	45	2488
# hosts with all flows malicious	239	1731	20	1818
# hosts with malicious and benign flows	17	533	25	567
# hosts with all flows benign	6	103	0	103

Diagnosing DGA Flows: Of the total of 56,957 suspicious flows (mirrored by our system during this trial period), 35645 are HTTP, 19674 are HTTPS, and 1638 are UDP. Recall from the previous section that we trained three EiF models (HTTP, HTTPS, and UDP) each with their respective malicious flows extracted from the CTU-13 dataset.

Table XII shows the testing results of suspicious flows against their corresponding EiF model. It can be seen that more than 90% of suspicious flows across the three types (HTTP, HTTPS, and UDP) have been classified as malicious. These high detection rates verify the efficacy of our trained models in diagnosing suspicious DGA flows.

In absence of ground-truth data whether suspicious flows are indeed malicious or not, we further analyzed these selected flows and their attributes. It is seen that 99.94% of suspicious HTTP flows are predicted as malicious. We found that a vast majority of suspicious HTTP flows (33.5K) consist of the only three-way handshake (initiated by internal hosts) followed by a TCP RST (reset) packet sent by the initiating host. Almost all of these 33.5K HTTP flows are classified as malicious – the CTU-13 dataset also had 810 flows (17% of total HTTP flows) of this kind. Excluding these specific 33.5K HTTP flows, again a vast majority (98.33%) of the remaining 2046 suspicious HTTP flows are predicted to be malicious, highlighting the fact that their traffic behavior conforms to the norms of known malware (*i.e.,* the CTU-13 dataset). Only 34 HTTP flows are classified as benign that carry a large number of packets (about 35 packets) compared to the malware norms (5 packets) – this corroborates to a great extent with our observations from misclassified flows during model validation, shown in Fig. 10(a). Similarly, we investigated the attributes of suspicious HTTPS and UDP flows that are classified as benign during trial evaluation, and found that benign-predicated: HTTPS flows contain an average of more than 200 packets, carrying relatively high volume (≈25 KB) of traffic (conforming to Fig. 10(b)); and UDP flows contain 320 to 390 packets, resulting in flow volume of average 350 KB (conforming to Fig. 10(c)). In summary, suspicious flows which are classified as benign are probably misclassified by the EiF models. This means that traffic flows subsequent to a DGA-based query are likely to be malicious.

Infected Hosts Initiating Malicious DGA Flows: We have so far identified malicious flows succeeding DGA queries, but

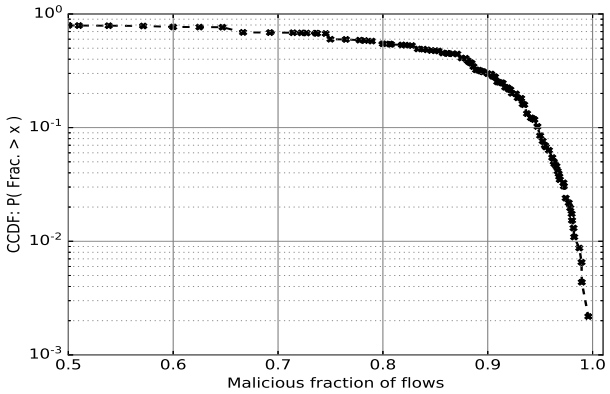


Fig. 13. CCDF of malicious fraction of flows per host in mix-malicious-benign category.

network operators are more interested in identifying hosts that are infected by malware. We mapped all of the suspicious flows (dynamically selected and mirrored by our system) to their corresponding hosts (inside the campus network) that initiated those flows. These hosts are in three categories, as shown by the bottom rows in Table XII: (a) hosts with all of their suspicious flows are predicted as malicious ("pure-malicious"), (b) hosts with some of their suspicious flows are predicted as malicious and some as benign ("mix-malicious-benign"), and (c) hosts with all of their suspicious flows are predicted as benign ("pure-benign").

It can be seen that at the aggregate level, shown in the last column of Table XII, the pure-malicious category dominates with 1818 hosts, followed by mix-malicious-benign and pure-benign, respectively with 567 and 103 hosts – across the campus network, there are a total of 2488 internal hosts that generate some suspicious flows (HTTP and/or HTTP and/or UDP). Note that 302 of these hosts are NAT gateways (each representing a number of wireless hosts), and the remaining 2186 are actual end hosts (clients or servers which are not NATed).

Focusing on the infected hosts, we found that they belong to 226 different subnets of size /24. Of these subnets, 34% have more than 10 infected hosts and 25% have more than 20 hosts infected. These insights help network operators who want to tighten the security posture of certain subnets those that have some degree of infection.

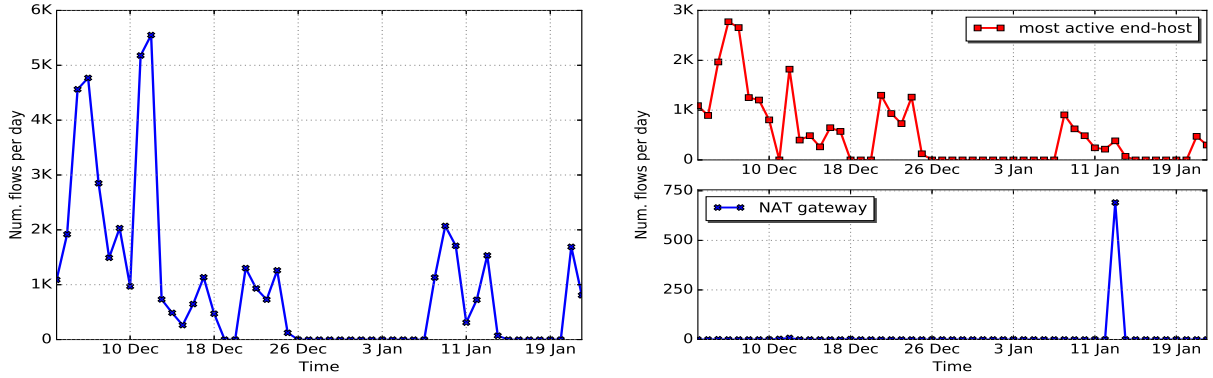
Next, we performed reverse lookups to infer the nature of infected hosts, and found that: 302 hosts are WiFi NAT gateways (from two dedicated subnets) with names as "`SSID-pat-pool-a-b-c-d.gw.univ-primay-domain`" where "`a.b.c.d`" is the public IP address of the NAT gateway, and SSID is the WiFi SSID for the University campus network; 34 hosts are the Mac devices (spread across 23 subnets) with names containing strings like "`mbp`" or "`imac`"; 616 hosts are returned with names including strings like '`desktop`" that they indicate are a user desktop machine (Windows/Linux); and the remaining 1433 hosts (spread across 186 subnets) are returned with no name – they are also typically regular end-hosts.

We plot in Fig. 12 the CCDF of suspicious flow count per host, within each of the three categories mentioned above. It can be seen that 99% of pure-benign hosts have at most two suspicious flows – rarely active. In the other two categories, instead, far more suspicious flows are observed per host (average of 22 flows) – several hosts in both of these very active categories have more than 1000 flows. Focusing on the mix-malicious-benign category, we see that 50% of hosts have more than 10 suspicious flows, represented by the tail of their CCDFs. Also, more than 80% of hosts in the mix category have more than half of their suspicious flows classified as malicious – the CCDF plot in Fig.13 particularly illustrates the distribution of the malicious fraction of flows across all hosts of this category. For these reasons, we deem the two active categories, consisting of 2385 hosts, to be likely "infected" by malware.

We plot in Fig. 14 the daily time trace of active malicious flows detected during our trial. Each data-point represents the number of active flows over a 24-hour window. Looking at Fig. 14(a), we observe that the activity of DGA-enabled C&C communications across the aggregate of all campus hosts was relatively high during the first half of December (peaking at a total of ≈5500 flows on 12th Dec), gradually fell reaching to a complete no-show during holiday shutdown periods (between 25-Dec-2019 and 8-Jan-2020), and afterward revived slowly. This pattern of malware activity correlates (to a great extent) with the number of active users on the network, suggesting infected regular hosts. Moving on to Fig. 14(b), we see malicious flow activity of two hosts. The top subplot corresponds to the most active infected end-host that generated a total of 24.8K malicious flows (all preceded by DGA responses of ModPack family). This host was active at the beginning of the trial (first three weeks in December) by making on average more than 1000 malicious flows per day, but its hyperactivity slowly diminished in the second half of our trial. The bottom subplot is a NAT gateway that displays a different pattern of malware activity. It made a total of 14 malicious flows (pertinent to Matsnu family) during December, went silent for three weeks, and then suddenly became heavily active in the middle of January by making about 700 malicious flows from the Suppobox family. It is expected to see a diversity of families in the NAT gateway since they make flows on behalf of a group of end-hosts.

We show in Fig. 15 the CCDF plot of delay between DGA responses (resolution of DGA query for C&C server) and the commencement of the first associated TCP/UDP flow. It can be seen that more than 90% flows start in less than 2 ms (very shortly) after their DNS resolution. Since flow arrival delays do not go beyond 100 ms, it is crucial for our detection system to immediately insert a corresponding reactive rule into the SDN switch, capturing and diagnosing the communication between the internal host and the external C&C server.

Next, we analyze the size of mirrored traffic flows which need to be analyzed in software – suggesting the computing cost. It can be seen in Fig. 16(a) that mirrored suspicious flows often carry a small number of packets – more than 86% of flows have less than 100 packets. Interestingly, these flows are relatively short – as shown in Fig. 16(b) more than 90% of flows last less than 2 minutes, hence getting timed out quickly from the switch TCAM table. This measure is important since TCAM is one of the precious resources in our system. Considering the time trace of reactive rules (during our



(a) Aggregate of all hosts.

(b) Most active infected end-host (top) and a NAT gateway (bottom).

Fig. 14. Time trace of active malicious flows (between infected internal hosts and malware servers) during the 50-day trial for: (a) aggregate of all campus hosts, and (b) most active infected end-host (top) and a NAT gateway (bottom).

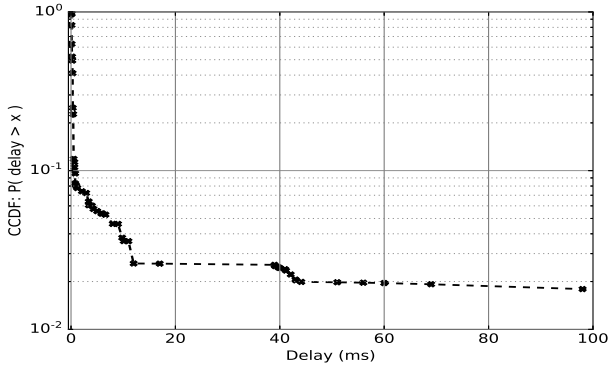
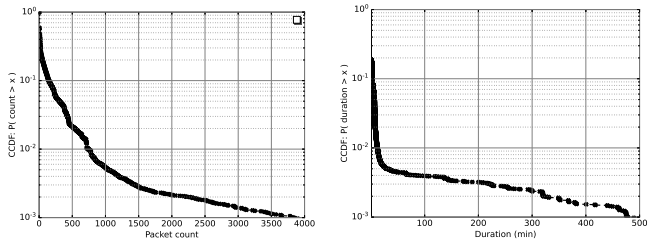


Fig. 15. Delay between DGA-based DNS responses and commencement of their subsequent TCP/UDP flow



(a) Packet count.

(b) Duration.

Fig. 16. CCDF of attributes: (a) packet count, and (b) duration, of suspicious flows.

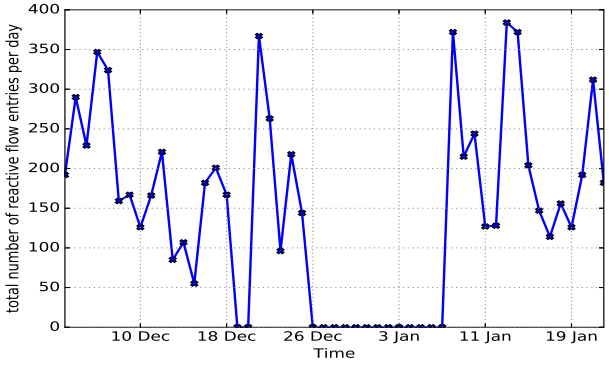


Fig. 17. Daily number of reactive flow entries installed by the SDN controller into the SDN switch, during the 50-day trial.

50-day trial) in Fig. 17, we see that no more than 400 entries per day are installed into the SDN switch by the controller. These metrics collectively serve as clear evidence of the cost-effectiveness of our solution.

Comparing Our Diagnosis Models with Zeek IDS: Lastly, we validate our results against logs of an open-source IDS Zeek (formerly Bro) – which is a powerful network analysis tool, widely used, and consistently maintained by the community for more than 20 years now [78]. Our main intention is to compare our ML-based flow-level approach with a decent rule-based packet-level method in detecting malicious traffic.

Zeek performs a packet-based analysis and raises alarms if a known malicious signature is found in a packet. We replayed the 50-day worth of selected traffic (of suspicious flows) onto Zeek to check how it flags malicious activities. Of a total of 5.5 M packets, Zeek raised alarms for 23.7 K (0.4%). In order to compare with our flow-level analysis, we aggregate those packets flagged by Zeek into flows. It turns out 14,455 flows (*i.e.,* 25.3% of suspicious flows), belonging to all of the five malware families (Table X), are detected as malicious by Zeek.

We found that all of Zeek flagged flows are a subset of malicious flows classified by our models. Starting from the HTTP flows, we found a small overlap (only 1087 flows, about 3%) in the outputs of our HTTP model and Zeek. This is mainly because those 33.5K HTTP flows (94% of total) that only carry three-way handshakes with a reset do not cause Zeek to raise any alarms when inspecting their individual packets. For HTTPS flows, instead, the overlap was far better – Zeek corroborates our HTTPS model by flagging 13368 flows ($\approx$ 68%) that we classified as malicious. Lastly, none of the UDP flows are flagged by Zeek, probably because individual UDP packets did not display any malicious pattern matching Zeek's known signatures. Zeek raises a number of different alert types for malicious HTTP and HTTPS packets – about half of malicious HTTP and HTTPS receive more than one alert type. The top four alerts are: "`above_hole_data_without_any_acks`", "`bad_TCP_checksum`", "`possible_split_routing`", and "`data_before_established`".

VI. Conclusion

Cyber-attacks on enterprise networks are increasingly becoming sophisticated. We have developed and validated a method for real-time selective mirroring of network flows for diagnosis by trained models in real-time. We analyzed 75 days' worth of DNS traffic (2.4B records), highlighted the prevalence of more than twenty DGA-enabled malware families across

internal hosts, and obtained insights into their behavioral patterns while communicating with their corresponding C&C server by analysis of a large PCAP collected during peak hour. We identified their traffic attributes and trained three one-class classifier models specialized in HTTP, HTTPS, and UDP protocols using public PCAP traces of known malware families. We then developed a system that continuously monitors DNS traffic, and automatically (using SDN reactive rules) select and mirror communications between internal hosts and malware C&C servers pertinent to DGA queries. We then evaluated the efficacy of our models by testing suspicious traffic flows against our trained models, identified infected hosts from suspicious flows. Finally, we compared our detection approach with software IDS Zeek.

REFERENCES

[1] Cybercrime Magazine. (2020) Global Cybercrime Damages Predicted To Reach $6 Trillion Annually By 2021. Accessed on 28.04.2020. [Online]. Available: https://bit.ly/2YpyEZZ

[2] Accenture Security. (2019) The Cost of Cybercrime. Accessed on 28.04.2020. [Online]. Available: https://accntu.re/2WgX3hx

[3] Security Magazine. (2020) As Cyber Attacks Become More Prevalent, Here?s Why Your Small Business is at Risk. Accessed on 28.04.2020. [Online]. Available: https://bit.ly/2YpyEZZ

[4] Accenture Security. (2018) Enhancing the Resilience of the Internet and Communications Ecosystem Against Botnets and Other Automated, Distributed Threats. Accessed on 20.04.2020. [Online]. Available: https://bit.ly/2VTd925

[5] Ars Technica. (2017) Failure to patch two-month-old bug led to massive Equifax breach. Accessed on 10.04.2020. [Online]. Available: https://bit.ly/3aSJgmO

[6] MIT Internet Policy Research Initiative. (2019) Why Botnets Persist: Designing Effective Technical and Policy Interventions. Accessed on 15.04.2020. [Online]. Available: https://internetpolicy.mit.edu/wp-content/uploads/2019/09/publications-ipri-2019-02.pdf

[7] M. Antonakakis *et al.*, "From throw-away traffic to bots: Detecting the rise of dga-based malware." in *Proc. USENIX Security*, Bellevue, WA, Aug. 2012, pp. 24–24.

[8] D. Plohmann *et al.*, "A comprehensive measurement study of domain generating malware," in *Proc. USENIX Security*, 2016, pp. 263–278.

[9] D. PLOHMANN. (2020) DGArchive. Accessed on 15.01.2020. [Online]. Available: https://dgarchive.caad.fkie.fraunhofer.de/

[10] S. Hao *et al.*, "An Internet-Wide View into DNS Lookup Patterns," VeriSign Labs, School of Computer Science, Georgia Tech, Tech. Rep., 2010.

[11] L. Bilge *et al.*, "EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis." in *Proc. USENIX NDSS*, San Diego, CA, USA, Feb 2011, pp. 1–17.

[12] M. Antonakakis *et al.*, "Building a Dynamic Reputation System for DNS," in *Proc. USENIX Security*, Washington, DC, USA, Aug 2010, pp. 18–18.

[13] S. Hao *et al.*, "Monitoring the Initial DNS Behavior of Malicious Domains," in *Proc. ACM SIGCOMM IMC*, Berlin, Germany, Oct 2011, pp. 269–278.

[14] J. Ahmed *et al.*, "Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks," in *Proc. IFIP/IEEE IM*, Washington, DC, USA, Apr 2019, pp. 649–653.

[15] M. Lyu *et al.*, "Mapping an Enterprise Network by Analyzing DNS Traffic," in *Proc. PAM*, Puerto Varas, Chile, Mar 2019, pp. 129–144.

[16] M. Antonakakis *et al.*, "Detecting malware domains at the upper dns hierarchy." in *Proc. USENIX Security*, San Francisco, CA, USA, Aug. 2011, pp. 27–27.

[17] T.-F. Yen *et al.*, "Traffic aggregation for malware detection," *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 1, no. 1, pp. 207–227, 2008.

[18] B. Anderson *et al.*, "Identifying encrypted malware traffic with contextual flow data," in *Proc. ACM AISec*, 2016, pp. 35–46.

[19] ——, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proc. ACM SIGKDD*, Halifax NS Canada, Aug 2017, pp. 1723–1732.

[20] F. Tegeler *et al.*, "Botfinder: Finding bots in network traffic without deep packet inspection," in *Proc. ENET*, 2012, pp. 349–360.

[21] S. Yadav *et al.*, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/ACM ToN*, vol. 20, no. 5, pp. 1663–1677, 2012.

[22] S. Talukder, "Tools and techniques for malware detection and analysis," *arXiv preprint arXiv:2002.06819*, 2020.

[23] S. Yadav *et al.*, "Detecting algorithmically generated malicious domain names," in *Proc. ACM SIGCOMM*, Melbourne, Australia, Nov 2010, pp. 48–61.

[24] S. Garcia *et al.*, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[25] S. Schüppen *et al.*, "FANCI: Feature-based Automated NXDomain Classification and Intelligence," in *Proc. USENIX Security*, 2018, pp. 1165–1181.

[26] H. S. Anderson *et al.*, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM SIGSAC*, Vienna, Austria, Oct. 2016, pp. 13–21.

[27] R. Vinayakumar *et al.*, "Detecting malicious domain names using deep learning approaches at scale," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1355–1367, 2018.

[28] G. Marín *et al.*, "Deepmal–deep learning models for malware traffic detection and classification," *arXiv preprint arXiv:2003.04079*, 2020.

[29] C. Choudhary *et al.*, "Algorithmically generated domain detection and malware family classification," *Security in Computing and Communications*, pp. 640–655, 2018.

[30] J. Spooren *et al.*, "Detection of algorithmically generated domain names used by botnets: a dual arms race," in *Proc. ACM/SIGAPP SAC*, Limassol Cyprus, 2019, pp. 1916–1923.

[31] G. Gu *et al.*, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *Proc. USENIX Security*, vol. 7, 2007, pp. 1–16.

[32] ——, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," in *Proc. USENIX Security*, San Jose, CA, USA, Jul-Aug 2008.

[33] ——, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Proc. USENIX NDSS*, San Diego, California, USA, Feb 2008.

[34] A. Gupta *et al.*, "Sonata: Query-driven streaming network telemetry," in *Proc. ACM Special Interest Group on Data Communication*, Budapest, Hungary, Aug 2018, pp. 357–371.

[35] Y. Afek *et al.*, "Network anti-spoofing with sdn data plane," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, 2017, pp. 1–9.

[36] S. K. Fayaz *et al.*, "Bohatei: Flexible and elastic ddos defense," in *Proc. USENIX Security*, Washington, D.C, USA, Aug 2015, pp. 817–832.

[37] M. Zhang *et al.*, "Poseidon: Mitigating volumetric ddos attacks with programmable switches," in *Proc. USENIX NDSS*, San Diego, California, USA, Feb. 2020, pp. 1–18.

[38] S. Schiavoni *et al.*, "Phoenix: Dga-based botnet tracking and intelligence," in *Proc. Springer DIMVA*, 2014, pp. 192–211.

[39] M. Lyu *et al.*, "Hierarchical Anomaly-Based Detection of Distributed DNS Attacks on Enterprise Networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1031–1048, 2021.

[40] C. J. Dietrich *et al.*, "On Botnets that use DNS for Command and Control," in *Proc. IEEE CND*, Gothenburg, Sweden, Sep 2011, pp. 9–16.

[41] M. Feily *et al.*, "A survey of botnet and botnet detection," in *Proc. ACM ESIST*, Athens, Glyfada, Greece, Jun 2009, pp. 268–273.

[42] J. Ahmed *et al.*, "Monitoring Enterprise DNS Queries for Detecting Data Exfiltration from Internal Hosts," *IEEE TNSM*, vol. 17, no. 1, pp. 265–279, 2019.

[43] ——, "A Tool to Detect and Visualize Malicious DNS Queries for Enterprise Networks," in *Proc. IFIP/IEEE IM*. IEEE, 2019, pp. 729–730.

[44] A. McDole *et al.*, "Analyzing cnn based behavioural malware detection techniques on cloud iaas," *arXiv preprint arXiv:2002.06383*, 2020.

[45] R. Sivaguru *et al.*, "Inline detection of dga domains using side information," *arXiv preprint arXiv:2003.05703*, 2020.

[46] J. M. Ceron *et al.*, "Mars: An sdn-based malware analysis solution," in *2016 IEEE ISCC*. IEEE, 2016, pp. 525–530.

[47] Sophos. (2020) Troj/Menti-Fam. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/38ttVrJ

[48] Microsoft. (2020) TrojanDownloader:Win32/Murlo.S. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/2xeYZyt

[49] T. Micro. (2020) WORM:NERIS.A. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/2wxU8YK

[50] Fortinet. (2020) NSIS.botnet. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/39uofPh

[51] F. Secure. (2020) Backdoor:W32/RBot. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/2VKtoPr
[52] Microsoft. (2020) Program:W32/Sogou. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/38spEEE
[53] F. Secure. (2020) Virus:Win32/Virut. Accessed on 25.02.2020. [Online]. Available: https://bit.ly/2TtmRqw
[54] S. Garcia. (2020) The CTU-13 Dataset. Accessed on 20.01.2020. [Online]. Available: https://www.stratosphereips.org/datasets-ctu13
[55] R. Perdisci *et al.*, "behavioral clustering of http-based malware and signature generation using malicious network traces."
[56] B. Biggio *et al.*, "Poisoning Behavioral Malware Clustering," in *Proc. ACM AISec*, Nov 2014.
[57] G. of Canada. (2020) Canadian Centre for Cyber Security. Accessed on 21.01.2020. [Online]. Available: https://cyber.gc.ca/en/
[58] T. I. Team. (2020) Andromeda under the microscope. Accessed on 21.01.2020. [Online]. Available: https://bit.ly/32VPcJp
[59] A. Griffin. (2020) Andromeda taken down. Accessed on 25.01.2020. [Online]. Available: https://bit.ly/32SGITa
[60] blueliv. (2020) Inside Tinba- DGA infection. Accessed on 9.02.2020. [Online]. Available: https://bit.ly/39uDHLg
[61] H. Security. (2020) Everything You Need to Know about the Notorious Zeus Gameover Malware. Accessed on 9.02.2020. [Online]. Available: https://bit.ly/39tmX7o
[62] D. Bonderud. (2020) Combating High-Risk, Low-Noise Threats. Accessed on 29.01.2020. [Online]. Available: https://ibm.co/38u8FBW
[63] B. AsSadhan *et al.*, "Periodic behavior in botnet command and control channels traffic," in *Proc. IEEE GLOBECOM*, 2009, pp. 1–6.
[64] V. Total. (2020) Virus Total. Accessed on 27.02.2020. [Online]. Available: https://bit.ly/32SskdI
[65] urlscan. (2020) X.co urlscan.io. Accessed on 27.02.2020. [Online]. Available: https://bit.ly/38ocE3b
[66] F. Allard *et al.*, "Tunneling activities detection using machine learning techniques," *Journal of Telecommunications and Information Technology*, pp. 37–42, 2011.
[67] A. L. Buczak *et al.*, "Detection of tunnels in PCAP data by random forests," in *Proc. ACM CISRC*, Oak Ridge, TN, USA, Apr 2016, pp. 1–4.
[68] D. A. Cieslak *et al.*, "Learning Decision Trees for Unbalanced Data," in *Proc. MLKDD*, Sep 2008, pp. 241–256.
[69] A. Sivanathan *et al.*, "Detecting Behavioral Change of IoT Devices using Clustering-Based Network Traffic Modeling," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295–7309, Aug 2020.
[70] F. T. Liu *et al.*, "Isolation forest," in *Proc. IEEE Data Mining*, Pisa, Italy, Dec 2008, pp. 413–422.
[71] S. Hariri *et al.*, "Extended Isolation Forest," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019.
[72] C. Ding *et al.*, "K-means clustering via principal component analysis," in *Proc. ICML*, Banff, Alberta, Canada, Jul 2004.
[73] B. Schölkopf *et al.*, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
[74] D. J. Ketchen *et al.*, "The application of cluster analysis in strategic management research: an analysis and critique," *Strategic management journal*, vol. 17, no. 6, pp. 441–458, 1996.
[75] NoviFlow. (2020) NoviSwitch 2122 High Performance OpenFlow Switch. Accessed on 01.02.2020. [Online]. Available: https://bit.ly/3cyhvSx
[76] R. Dev. (2020) Ryu SDN Framework. Accessed on 27.02.2020. [Online]. Available: https://osrg.github.io/ryu/
[77] T. M. Million. (2021) Top 1 million website in the world. [Online]. Available: http://downloads.majesticseo.com/majestic_million.csv
[78] Zeek. (2020) The Zeek Network Security Monitor. Accessed on 19.02.2020. [Online]. Available: https://zeek.org

Jawad Ahmed received his BS in Electrical (Telecom) Engineering from COMSATS Institute of Information Technology, Islamabad, Pakistan in 2011, his MS in Electrical Engineering from the National University of Sciences and Technology (NUST), Islamabad, Pakistan in 2016, and his Ph.D. in Electrical Engineering and Telecommunications from UNSW in Sydney, Australia in 2021. He worked as a lab engineer at NUST from 2012 to 2017. He is currently working as a Senior Research Associate at the University of New South Wales in Sydney, Australia. His major research interests include Software Defined Networking, Data Analytics, and Cybersecurity.

Hassan Habibi Gharakheili received his B.Sc. and M.Sc. degrees of Electrical Engineering from the Sharif University of Technology in Tehran, Iran in 2001 and 2004 respectively, and his Ph.D. in Electrical Engineering and Telecommunications from UNSW in Sydney, Australia in 2015. He is currently a Senior Lecturer at UNSW Sydney. His research interests include programmable networks, learning-based networked systems, and data analytics in computer systems.

Craig Russell received his Ph.D. in Applied Mathematics from Macquarie University, Sydney in 1997. He is currently Director of Engineering at Canopus Networks and Adjunct Senior Lecturer at UNSW, and has previously held commercial roles in the telecommunications and software industries. He has design, implementation and operational experience in a wide range of advanced telecommunications equipment and protocols as well as experience in developing software applications. His research interests are in software-defined networking and the application of machine learning techniques to solve problems in network security.

Vijay Sivaraman received his B. Tech. from the Indian Institute of Technology in Delhi, India, in 1994, his M.S. from North Carolina State University in 1996, and his Ph.D. from the University of California at Los Angeles in 2000. He has worked at Bell-Labs as a student Fellow, in a silicon valley start-up manufacturing optical switch-routers, and as a Senior Research Engineer at the CSIRO in Australia. He is now a Professor at the University of New South Wales in Sydney, Australia. His research interests include Software Defined Networking, network architectures, and cyber-security particularly for IoT networks.